DISEÑO DE BASE DE DATOS

José Rafael Capacho Portilla
Wilson Nieto Bernal

UNIVERSIDAD DEL NORTE
Editorial

Diseño de base de datos

Diseño de base de datos

José Rafael Capacho Portilla
Wilson Nieto Bernal

Área metropolitana
de Barranquilla (Colombia), 2017

UNIVERSIDAD DEL NORTE
Editorial

Capacho Portilla, José Rafael.
Diseño de base de datos / José Rafael Capacho Portilla, Wilson Nieto Bernal. -- Barranquilla, Colombia, Editorial Universidad del Norte, 2017.

292 p. : il. ; 24 cm.
Incluye referencias bibliográficas (p. 289-292).
ISBN 978-958-741-825-5 (PDF)

1. Diseño de bases de datos. I. Capacho Portilla, José Rafael. II. Nieto Bernal, Wilson.

(005.74 C236 ed.23) (CO-BrUNB)

Vigilada Mineducación
www.uninorte.edu.co
Km 5, vía a Puerto Colombia, A.A. 1569
Área metropolitana de Barranquilla (Colombia)

© Universidad del Norte, 2017
José Rafael Capacho Portilla y Wilson Nieto Bernal

Coordinación editorial
Zoila Sotomayor O.

Diseño y diagramación
Munir Kharfan de los Reyes

Diseño de portada
Andrés Racedo

Corrección de textos
Henry Stein

Impreso y hecho en Colombia
Printed and made in Colombia

© Reservados todos los derechos. Queda prohibida la reproducción total o parcial de esta obra por cualquier medio reprográfico, fónico o informático, así como su transmisión por cualquier medio mecánico o electrónico, fotocopias, microfilm, *offset*, mimeográfico u otros sin autorización previa y escrita de los titulares del *copyright*. La violación de dichos derechos constituye un delito contra la propiedad intelectual.

Los autores expresan sus agradecimientos a la Universidad del Norte,
en especial a los ingenieros Danilo Lizcano, funcionario de DTIC,
y Pedro Mario Wightman Rojas, director del Departamento
de Sistemas, y a la ingeniera Melissa Isabel Castro Sarmiento.

Los autores

José Rafael Capacho Portilla

Doctor en Procesos de Formación en Espacios Virtuales de la Universidad de Salamanca, USAL (España), con tesis laureada *Summa Cum Laude*. Máster en Educación de la Universidad Javeriana (Colombia). Ingeniero de Sistemas de la Universidad Industrial de Santander, UIS (Colombia). Líder de proyectos de acreditación nacional de alta calidad del PISC de la Universidad del Norte (Colombia), con el Ministerio de Educación Nacional, y del proyecto de acreditación internacional del PISC con la Accreditation Board for Engineering and Technology, ABET (2009). Profesor e investigador del Departamento de Sistemas de la Universidad del Norte y miembro del grupo de investigación en Redes e Ingeniería de Software del Departamento de Ingeniería de Sistemas de esta misma institución.

Wilson Nieto Bernal

Doctor en Tecnologías de la Información de la Universidad de Las Palmas de Gran Canaria, ULPGC (España), con tesis laureada y exaltación *Cum Laude*. Máster/experto en Gestión Tecnológica de la ULPGC. Magíster en Educación de la Universidad del Norte (Colombia). Ingeniero de Sistemas y especialista en Ingeniería de Software de la Universidad Industrial de Santander, UIS (Colombia). Con una amplia experiencia en ingeniería de sistemas de información organizacional y en procesos de I+D+i. Trabaja en las áreas de Tecnologías de la información y aplicaciones organizacionales, Gestión del conocimiento e innovación organizacional y Modelos de gestión organizacional. Es profesor de la Universidad del Norte en el Departamento de Ingeniería de Sistemas y Computación e Investigador Senior (Colciencias).

Contenido

INTRODUCCIÓN 1

JUSTIFICACIÓN 5

SÍNTESIS CONCEPTUAL 7

1. INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS 9

1.1 Introducción 9

1.2 Objetivos 10

1.3 Síntesis conceptual 11

1.4 Sistemas de información basados en archivos 11

1.4.1 Construcción de sistemas de información basados en archivos . . . 11

1.4.1.1 Técnica de construcción de sistemas de información basada en archivos 14

1.4.1.2 Desventajas y limitaciones de los sistemas de información basados en archivos 15

1.5 Sistemas de información construidos en bases de datos 18

1.5.1 Concepto de *bases de datos* 18

1.5.2 Entorno de operación de un Sistema de Gestión de Bases de Datos (SGBD) 21

1.5.3 Análisis de la independencia lógica/física de datos en un sistema de base de datos 24

1.5.3.1 Independencia lógica de datos 24

1.5.3.2 Independencia física de datos 29

1.5.4 Lenguajes de un SGBD 30

1.5.4.1 Lenguaje de definición de datos 30

1.5.4.2 Lenguaje de manipulación de datos 31

1.5.4.3 Lenguaje de control de datos 31

1.5.5 Integración de esquemas ANSI al flujo de control de datos en un SGBD 32

1.5.6 Usuarios de un sistema de bases de datos 35

1.5.7 Desarrollo histórico de los sistemas de gestión de bases de datos . 36

1.6 Ventajas de un sistema de gestión de bases de datos 39

1.7 Desventajas y limitaciones de los sistemas de bases de datos 45

1.8 Resumen 47

Ejercicios 49

2. DISEÑO DE BASES DE DATOS 51

2.1 Introducción. 51

2.2 Objetivos 52

2.3 Síntesis conceptual 53

2.4 Etapas de diseño de un sistema de bases de datos 53

2.5 Planificación del desarrollo del sistema informático en bases de datos . . 54

2.6 Diseño conceptual (lógico) de la base de datos 55

2.6.1 Diseño conceptual de la base de datos a través del modelo Entidad-Relación (M E-R) 56

2.6.1.1 Constructos básicos del modelo Entidad-Relación según la notación de P. Chen 57

2.6.2 Análisis del contexto de operación de la base de datos, soporte al sistema de información 70

2.6.3 Identificar las entidades del sistema 72

2.6.4 Asociar valores semánticos a las componentes del contexto 72

2.6.5 Agrupar las componentes del contexto 72

2.6.6 Diseñar los atributos de las Entidades y Relaciones 74

2.6.7 Seleccionar las claves primarias y foráneas de las entidades de la base de datos 75

2.7 Diseño físico de la base de datos 79
2.8 Carga de los datos a la base de datos 81
2.9 Operación de la base de datos 82
2.10 Mantenimiento de la base de datos 82
2.11 Diseño de bases de datos a partir de especificaciones de requerimientos 83
2.12 Resumen conceptual 88
Ejercicio 90

3. CREACIÓN DE DATOS DE LA BASE DE DATOS 93

3.1 Objetivos 93
3.2 Síntesis conceptual 93
3.3 Definición de *datos* 93
3.3.1 Tipos de datos SQL 94
3.3.1.1 Tipos de identificadores 95
3.3.1.2 Tipos de datos escalares (booleanos, caracteres, bit) 95
3.3.1.3 Datos numéricos (exactos, aproximados, fecha y hora, intervalo) 96
3.3.2 Control de integridad 99
3.3.2.1 Requerimiento de datos 99
3.3.2.2 Dominio de atributos 99
3.3.2.3 Integridad de entidades 102
3.3.2.4 Integridad referencial 105
3.3.2.5 Restricciones generales 107
3.3.3 Definición de datos de la base de datos 108
3.3.3.1 Creación de la base de datos 108
3.3.3.2 Creación del esquema de la base de datos 109
3.3.3.3 Creación de tablas 110
3.3.3.4 Edición de tablas 120
3.3.3.5 Eliminación de tablas 130
3.3.3.6 Creación de índices 130
3.3.3.7 Eliminación de índices 133

3.3.4 Vistas de la base de datos 133
3.3.4.1 Definición del concepto de vista de la base de datos 133
3.3.4.2 Creación de vistas 135
3.3.4.3 Creación de vistas con funciones de agrupamiento y combinación 142
3.3.4.4 Eliminación de vistas 144
3.3.4.5 Solución de vistas 146
3.3.4.6 Restricciones de las vistas 149
3.3.4.7 Actualización de vistas en una base de datos 155
3.3.4.8 Análisis de la utilización de vistas en una base de datos . . 158
3.4 Resumen 161
Ejercicios 162

4. MANEJO DE DATOS EN LA BASE DE DATOS 164

4.1 Objetivos 164
4.2 Síntesis conceptual 164
4.3 Manipulación de datos 165
4.4 Descripción de los atributos de las tablas 165
4.5 Consultas simples 173
4.5.1 Seleccionar todos atributos de todas las tuplas 175
4.5.2 Seleccionar algunos atributos de todas las tuplas 178
4.5.3 Uso de literales en la cláusula SELECT 180
4.5.4 Impresión de campos numéricos con formato predefinido 181
4.5.5 Selección de tuplas distintas 182
4.5.6 Generación de campos calculados 184
4.5.7 Búsqueda de tuplas basada en comparación 185
4.5.8 Búsqueda de tuplas basada en comparación compuesta 188
4.5.9 Selección de tuplas basada en rango 189
4.5.10 Selección de tuplas por pertenencia a un conjunto 191

4.5.11 Selección de tuplas utilizando patrones 196
4.5.12 Selección de tuplas utilizando con atributos nulos. 198
4.5.13 Generación de resultados ordenado. 201
4.5.13.1 Generación de tuplas con un atributos ordenado 201
4.5.13.2 Generación de tuplas con múltiples atributos ordenados 204
4.5.14 Generación de resultados con funciones de agregación. 207
4.5.14.1 Conteo de tuplas 207
4.5.14.2 Conteo de tuplas distintas 208
4.5.14.3 Utilización de las funciones de agregación Mínimo, Máximo y Promedio 209
4.5.15 Generación de resultados con funciones de agrupación 210
4.5.15.1 Selección de tuplas utilizando agrupación (GROUP BY) . . 210
4.5.15.2 Selección de tuplas utilizando totales de cuenta (HAVING COUNT) 212
4.6 Consultas compuestas 213
4.6.1 Selección de tuplas utilizando subconsultas. 213
4.6.1.1 Generación de subconsultas con el operador de igualdad. 213
4.6.2 Generación de subconsultas con funciones de agregación 215
4.6.2.1 Generación de subconsultas anidadas con la utilización de la cláusula IN 218
4.6.3 Selección de alguna o todas las tuplas. 219
4.6.3.1 Selección de algunas tuplas (ANY) 219
4.6.3.2 Selección de todas las tuplas (ALL). 220
4.6.4 Generación de tuplas a través de consultas multitablas 222
4.6.4.1 Combinación simple 222
4.6.4.2 Generación de tuplas ordenando una combinación 225
4.6.4.3 Generación de tuplas por combinación de *n* tablas 227
4.6.4.4 Generación de tuplas por múltiples columnas de agrupamiento. 228
4.6.5 Generación de tuplas por combinaciones 230
4.6.5.1 Selección de tuplas por combinación externa izquierda . 230
4.6.5.2 Selección de tuplas por combinación externa derecha . . . 233
4.6.5.3 Selección de tuplas por combinación externa completa. . 233

4.6.6 Selección de tuplas utilizando las cláusulas EXISTS y NOT EXISTS . 238
4.6.6.1 Generación de tuplas utilizando la cláusula EXISTS. 238
4.6.6.2 Generación de tuplas utilizando la cláusula NOT EXISTS. 239
4.6.7 Generación de tuplas por combinación de tablas de resultados . 240
4.6.7.1 Generación de tuplas por unión de tablas UNION. 241
4.6.7.2 Generación de tuplas por intersección de tablas INTERSECT. 243
4.6.7.3 Generación de tuplas por diferencia de tablas MINUS. . . . 244
4.7 Edición de la base de datos 248
4.7.1 Adición de tuplas a la base de datos INSERT 248
4.7.2 Modificación de tuplas de la base de datos UPDATE 251
4.7.3 Borrado de tuplas de la base de datos DELETE 253
4.8 Resumen 255
Ejercicios 256

5. DISEÑO DE BASES DE DATOS ORIENTADAS A OBJETOS 262

5.1 Síntesis conceptual 262
5.2 Objetivos 262
5.3 Introducción BDOO 263
5.4 Orientación a Objetos 264
5.5 El proceso de modelo de bases de datos orientado a Objetos 266
5.5.1 Identificación de las funcionalidades de los datos (reglas semánticas). 267
5.5.2 Las reglas semánticas 268
5.5.3 Identificación de los objetos de información 272
5.5.4 Elaboración del modelo de Dominio 273
5.5.5 Mapeo de datos 274
5.5.6 Identificación y establecimiento de la multiplicidad (fuente y destino) 275
5.5.7 Recomendaciones para modelar una base de datos orientado a objetos. 277

5.6 Paradigmas emergentes de modelo de datos NoSQL o base de datos NoSQL 278

5.7 Tipos de base de datos NoSQL 278

5.8 Porqué utilizar base de datos NoSQL 279

5.9 Recomendaciones prácticas para seleccionar sistemas de base de datos NoSQL 281

5.10 Algunos ejemplos de modelos de datos tipo NoSQL 282

5.10.1 Base de datos NoSQL –Apache Cassandra– 282

5.10.2 Modelo de datos en Cassandra 282

5.11 Conceptos de base de datos en Cassandra 283

5.12 Cassandra: una opción de BD NoSQL. 285

5.13 Resumen 285

Ejercicios 286

BIBLIOGRAFÍA 289

ÍNDICE DE FIGURAS

Figura 1.1 ANSI-SPARC Arquitectura de tres niveles 23
Figura 1.2 Atributos de la entidad PROVEEDORES 25
Figura 1.3 Cambio del esquema conceptual de la base de datos del Sistema de Inventarios en su entidad PROVEEDORES 26
Figura 1.4 Atributos de la entidad PROVEEDORES del Sistema de Inventarios . 26
Figura 1.5 Independencia lógica con relación a cambios en el esquema externo 27
Figura 1.6 Listado del contenido de la tabla de PROVEEDORES antes del cambio del esquema conceptual 28
Figura 1.7 Listado de la entidad PROVEEDORES después de realizar el cambio conceptual. 28
Figura 1.8 Independencia física de los datos 30
Figura 1.9 Flujo de datos en un SGBD 34
Figura 2.1 Contexto del Sistema (S) e identificación de las entidades (E_i) 71
Figura 2.2 Contexto del mundo real del SAE 77
Figura 2.3 Análisis de grupos aplicado al diseño lógico del SAE. 78
Figura 2.4 Modelo Entidad-Relación del Sistemas de Administración de Edificios (SAE) en sus dos entidades Edificios y Apartamentos . . 79
Figura 2.5 Diseño del M E-R de la oferta de cursos intersemestrales en una universidad 85
Figura 2.6 Diseño del M E-R de las especificaciones de requerimientos de la empresa importadora-exportadora 88
Figura 3.1 Instancia del esquema de la base de datos del SIH en su integridad referencial 107
Figura 3.2 Descripción de los atributos de la tabla de PRODUCTOS del SIM . . 111
Figura 3.3 Instancia de la relación PRODUCTOS del SIM 111
Figura 3.4 Creación de la relación PROFESORES en el SIAC. 115
Figura 3.5 Creación de la tabla de los elementos compuestos del SPOAPU . . . 119

Figura 3.6 Creación de la tabla de los elementos simples del SPOAPU 120
Figura 3.7 Tabla proveedor del SI 121
Figura 3.8 Tabla producto del SI 122
Figura 3.9 Alteración del nombre de la tabla proveedor del SI 123
Figura 3.10 Descripción de los atributos de la tabla PROVEEDORES_UNO del SI. 123
Figura 3.11 Adición de columnas a la tabla PROVEEDORES_UNO del SI 124
Figura 3.12 Tabla PROVEEDORES_UNO con la columna añadida 124
Figura 3.13 Eliminación de una columna de una tabla. 125
Figura 3.14 Alteración del nombre de la tabla PRODUCTO del SI 126
Figura 3.15 Adición de múltiples columnas a una tabla 126
Figura 3.16 Descripción de la tabla de PRODUCTOS_UNO del SI. 127
Figura 3.17 Alteración de las características de los campos de una tabla. 127
Figura 3.18 Resultado de la alteración de las características de los campos de una tabla 128
Figura 3.19 Asignación del campo de DEFAULT a la tabla PROVEEDORES_UNO del SI. 129
Figura 3.20 Adición de la última fecha de modificación a la tabla PROVEEDORES UNO del SI 129
Figura 3.21 Modificacion de un campo de DEFAULT de una tabla 129
Figura 3.22 Eliminación de la tabla de PROFESORS de SIAC 130
Figura 3.23 Listado de la tabla de ASIGNATURAS del SIAC 136
Figura 3.24 Creación de la vista de las materias del Departamento de Sistemas del SIAC. 137
Figura 3.25 Comprobación de la creación de la vista Asignaturas_Sistemas del SIAC 138
Figura 3.26 Descripción del contenido de la tabla ESTUDIANTES del SIAC 139
Figura 3.27 Contenido de la tabla de ESTUDIANTES del SIAC en sus atributos . 139

Figura 3.28 Creación de la vista de seguridad de los datos de los alumnos del SIAC 140

Figura 3.29 Contenido de la tabla de Alumnos_Sin_Nombre del SIAC 141

Figura 3.30 Creación de la vista Número de Asignaturas por Programa del SIAC 142

Figura 3.31 Contenido de la vista Número de Asignaturas por Programa 143

Figura 3.32 Creación de la vista de nemónicos de programas del SIAC 144

Figura 3.33 Vista parcial del contenido de los nemónicos de los programas . . . 145

Figura 3.34 Cancelación de la vista Nemónico_programas del SIAC. 146

Figura 3.35 Estructura de la consulta combinada de las tablas PENSUM y PROGRAMAS del SIAC para generar el número de asignaturas por programa de los programas con código inicial 6. 148

Figura 3.36 Resultados de la consulta combinada al generar del Sistema de Información Académico (SIAC) en función de las tablas base PENSUM y PROGRAMAS, el número de asignaturas de los programas con código inicial 6 148

Figura 3.37 Generación de tuplas del número de asignaturas por programa con base en la vista creada de AsignaturasPrograma para los programas cuyos códigos empiezan por 6 148

Figura 3.38 Resultados del número de asignaturas por programa para aquellos programas con código 6 generados a partir de la vista AsignaturasPrograma 149

Figura 3.39 Vista de número de empleados por proyecto del SIM 150

Figura 3.40 Descripción de la vista SIM_Empleados_Salarios 150

Figura 3.41 Contenido de tuplas de la vista SIM_Empleados_Salarios 151

Figura 3.42 Utilización de funciones de agregación en vistas generadas a partir de funciones de agrupación 151

Figura 3.43 Vista Costos_Transporte del SIM 152

Figura 3.44 Tuplas de la vista Costos_Transporte. 153

Figura 3.45 Vista de Costos_Seguros del SIM 153

Figura 3.46 Tuplas de la vista Costos_Seguros 154

Figura 3.47 Intersecto de vistas del SIM derivadas de funciones de agrupación 155
Figura 3.48 Descripción de la tabla de profesores de SIAC 156
Figura 3.49 Vista de ProfesoresSalarios del SIAC 156
Figura 3.50 Tuplas en la tabla de profesores 157
Figura 3.51 Tuplas contenidas en la vista ProfesoresSalarios del SIAC 157
Figura 3.52 Inserción de una tupla en la tabla de profesores 157
Figura 3.53 Comprobación de la actualización de la vista en función de la modificación de la tabla base 158
Figura 4.1 Tabla o relación de proyectos del SIM 165
Figura 4.2 Relación o tabla de empleados del SIM 166
Figura 4.3 Tabla o relación de productos 166
Figura 4.4 Tabla o relación de proveedores 167
Figura 4.5 Tabla o relación cuentas_por_pagar 167
Figura 4.6 Tabla o relación de cuentas_por_cobrar 168
Figura 4.7 Tabla o relación de compras 168
Figura 4.8 Tabla o relación cuenta 169
Figura 4.9 Tabla facturación 169
Figura 4.10 Tabla o relación ventas. 170
Figura 4.11 Tabla o relación producción 170
Figura 4.12 Tabla maquinaria 171
Figura 4.13 Relación o tabla mantenimiento 171
Figura 4.14 Relación transacción 171
Figura 4.15 Tabla transporte 172
Figura 4.16 Relación o tabla entrega 172
Figura 4.17 Tabla planta_producción 173
Figura 4.18 Atributos de la tabla mercadeo 173

Figura 4.19 Tuplas de la tabla de empleados enumerando cada uno de los atributos de la tabla 176
Figura 4.20 Utilización de * en la cláusula SELECT. 178
Figura 4.21 Selección de una serie de atributos de todas las tuplas de la tabla de proyectos 179
Figura 4.22 Uso de literales en las salidas de los atributos de las relaciones . . . 180
Figura 4.23 Salida de atributos con formatos numéricos 182
Figura 4.24 Listado de todas las ciudades donde se encuentran proyectos en ejecución. 183
Figura 4.25 Nombre de las ciudades distintas donde se están ejecutando proyectos 184
Figura 4.26 Campo calculado de SALDO a partir del Saldo_Anterior y el Movimiento 185
Figura 4.27 Proyectos del SIM superiores o iguales a 100 millones de pesos . . . 188
Figura 4.28 Proyecto del SIM ejecutados en Barranquilla con prioridad uno . . 189
Figura 4.29 Búsqueda por rangos de los productos del SIM con número de producción que están en el intervalo [10;25]. 190
Figura 4.30 Salarios de empleados del SIM no ubicados en el rango salarial entre 2 y 5 millones 191
Figura 4.31 Maquinaria que está en mantenimiento preventivo o pertenece al conjunto P 192
Figura 4.32 Proyectos nacionales con prioridades asociadas en [1,2] 194
Figura 4.33 Empleados del SIM no extranjeros y no técnicos ni ingenieros. . . . 195
Figura 4.34 Selección de tuplas con base en una secuencia de caracteres (% … %) del patrón 196
Figura 4.35 Búsqueda de patrones por cadena de caracteres de los cuales el primero es D, y se nota que en todas las tuplas está la palabra DANIEL 197
Figura 4.36 Selección de cualquier secuencia de caracteres de No_proyecto, siendo el último carácter igual a 1. 198
Figura 4.37 Utilización de NULL en la selección de tuplas 199

Figura 4.38 Utilización NOT NULL en la selección de tuplas de la base de datos del SIM 200
Figura 4.39 Tuplas totales de la tabla de mantenimiento del SIM 201
Figura 4.40 Utilización de la cláusula ORDER BY 203
Figura 4.41 Selección de tupla por ordenamiento de varias columnas o atributos de una tabla 205
Figura 4.42 Utilización de la cláusula ORDER BY con campos calculados 206
Figura 4.43 Utilización de la cláusula COUNT (*) 208
Figura 4.44 Utilización de COUNT (DISTINCT) 208
Figura 4.45 Utilización de MIN, MAX y AVG 209
Figura 4.46 Uso de la cláusula GROUP BY 211
Figura 4.47 Utilización de la cláusula HAVING COUNT 212
Figura 4.48 Anidamiento de consultas con el operador de igualdad 214
Figura 4.49 Anidamiento de cruce de las tablas de empleados, proyectos y plantas de producción del SIM 215
Figura 4.50 Utilización de subconsulta con la función de agregación promedio (AVG). 217
Figura 4.51 Utilización de subconsulta con la función de agregación MINIMO (MIN). 217
Figura 4.52 Subconsulta anidada utilizando la cláusula IN 219
Figura 4.53 Utilización de la cláusula ANY 220
Figura 4.54 Utilización de la cláusula ALL 221
Figura 4.55 Utilización de consultas multitablas en su combinación simple . . 223
Figura 4.56 Resultados de la utilización de la cláusula JOIN 224
Figura 4.57 Consulta multitabla utilizando el ordenamiento de una combinación 226
Figura 4.58 Combinación de n(n=4) tablas del SIM. 228
Figura 4.59 Cláusula SELECT con múltiples columnas de agrupamiento 229
Figura 4.60 Combinación externa izquierda LEFT JOIN. 231

Figura 4.61 Combinación externa izquierda LEFT JOIN entre la maquinaria asignada a las líneas de producción 232
Figura 4.62 Combinación externa derecha RIGHT JOIN 234
Figura 4.63 Combinación externa completa FULL JOIN. 236
Figura 4.64 Selección de tuplas utilizando la cláusula EXITS 239
Figura 4.65 Utilización de la cláusula NOT EXISTS 240
Figura 4.66 Empleo de la cláusula UNION para la unión de tablas 242
Figura 4.67 Utilización de la cláusula INTERSECT 243
Figura 4.68 Utilización de la cláusula MINUS. 244
Figura 4.69 Uso de la cláusula MINUS 246
Figura 4.70 Equivalencia de cláusulas MINUS y NOT EXISTS 247
Figura 4.71 Descripción de la definición de columnas de tabla EMPLEADOS de la base de datos del SIM 249
Figura 5.1 Modelo de dominio de un contexto universitario. 264
Figura 5.2 Conector de agregación del ejemplo del contexto de la Universidad. 268
Figura 5.3 Asociación entre universidad y estudiante de carácter binario 269
Figura 5.4 Visualización de una relación n-aria 270
Figura 5.5 Relación de generalización 271
Figura 5.6 Relación de dependencia. 271
Figura 5.7 Clase abstracta 272
Figura 5.8 Relaciones de agregación entre clases 273
Figura 5.9 Esquema detallado del modelo de la base de datos universitaria . . 275
Figura 5.10 Multiplicidad entre las clases “programa” y “profesor”. 276
Figura 5.11 Instanciación de la relación histórica de un estudiante con sus matrículas 277

ÍNDICE DE TABLAS

Tabla 1.1 Archivo de elementos simples 13
Tabla 1.2 Elementos compuestos del SPOAPU 13
Tabla 1.3 Instancia de la entidad ESTUDIANTE en un Sistema de Información Académico 20
Tabla 2.1 Definición de atributos de entidades de Sistemas de Información 59
Tabla 2.2 Simbología del modelo Entidad-Relación, de acuerdo con la notación de P. Chen 60
Tabla 2.3 Simbología del modelo Entidad-Relación de acuerdo con pie de cuervo 66
Tabla 2.4 Aplicación de la función semántica para asignación de los atributos de las entidades 73
Tabla 2.5 Tabla de justificación del valor de α 73
Tabla 2.6 Justificación del indicador de relación (IR_i) 74
Tabla 2.7 Redefinición de los atributos de las entidades del SAE 80
Tabla 2.8 Modelo Relacional (MR) de dos entidades de la base de datos del SAE 81
Tabla 4.1 Condiciones o predicados básicos de búsqueda de acurdo con ISO 186
Tabla 4.2 Operadores básicos para estructuras consultas basadas en comparación 186
Tabla 4.3 Funciones de agregación según el estándar ISO 207
Tabla 4.3 Precedencia de Operadores 187
Tabla 4.4 Sistemas Académico de Bases de Datos de Estudiantes 256
Tabla 4.5 Atributos de la tabla de PROGRAMAS 260

ÍNDICE DE EJEMPLOS

Ejemplo 1.1 Registros de un Sistema de Información 12

Ejemplo 1.2 Comprobación de la independencia lógica 25

Ejemplo 1.3 Cambios en el esquema externo con relación a la independencia lógica 27

Ejemplo 1.4 Cambio físico de la base de datos 29

Ejemplo 2.1 Diseño Conceptual de un Sistema de Administración de Edificios (SAE) 76

Ejemplo 2.2 Caso: Diseño del M E-R para una oferta de cursos intersemestrales en una Universidad 84

Ejemplo 2.3 Caso: Diseño de una base de datos de una empresa importadora-exportadora 86

Ejemplo 3.1 Restricción de dominio en atributos 100

Ejemplo 3.2 Definición de restricciones de dominio 101

Ejemplo 3.3 Integridad de entidades 103

Ejemplo 3.4 Definición de claves primarias compuestas 103

Ejemplo 3.5 Unicidad de claves 104

Ejemplo 3.6 Integridad referencial 106

Ejemplo 3.7 Chequeo de restricciones generales en una base de datos con la cláusula ASSERTION 108

Ejemplo 3.8 Crear el esquema de una base de datos identificado como MyData creado por la identificación de usuario Jamin 110

Ejemplo 3.9 Creación de la tabla de PROFESORES del SIAC. 114

Ejemplo 3.10 Creación de la tabla de los elementos compuestos del SPOAPU . 118

Ejemplo 3.11 Creación de tablas base en un Sistema de Inventarios (SI) 121

Ejemplo 3.12 Modificación del nombre de una tabla en un Sistema de Inventarios 122

Ejemplo 3.13 Modificar la tabla PROVEEDORES_UNO del SI 123

Ejemplo 3.14 Eliminación de una columna de una tabla 125
Ejemplo 3.15 Inserción de múltiples columnas a una tabla 125
Ejemplo 3.16 Modificación de los nombres de las columnas de las tablas 127
Ejemplo 3.17 Asignación de tablas con campos predeterminados. 128
Ejemplo 3.18 Modificación de campos de DEFAULT de columnas de tablas. . . 129
Ejemplo 3.19 Cancelación de tablas 130
Ejemplo 3.20 Creación de índices en una base de datos 132
Ejemplo 3.21 Creación de índices con bases en múltiples columnas de una tabla de la base de datos 132
Ejemplo 3.22 Cancelación de índices en una base de datos 133
Ejemplo 3.23 Creación de vistas horizontales 135
Ejemplo 3.24 Creación de vistas verticales. 139
Ejemplo 3.25 Creación de vistas con funciones de agrupación. 142
Ejemplo 3.26 Eliminación de vistas de la base de datos. 144
Ejemplo 3.27 Resolución de consultas con base en vistas creadas en una base de datos. 146
Ejemplo 3.28 Contraejemplo de cumplimiento de restricciones en el manejo de vistas. 149
Ejemplo 3.29 Contraejemplo de cumplimiento de combinación de vistas con funciones de agregación 152
Ejemplo 3.30 Actualización de vistas en función de tablas bases 156
Ejemplo 4.1 Selección con atributos de tablas del SIM. 175
Ejemplo 4.2 Selección utilizando el selector asterisco (*) 177
Ejemplo 4.3 Selección de algunos atributos de una tabla 178
Ejemplo 4.4 Utilización de AS en la selección de tuplas 180
Ejemplo 4.5 Utilización de formatos numéricos en la selección de tuplas . . . 181
Ejemplo 4.6 Utilización de selección de tuplas distintas SELECT DISTINCT . . 183
Ejemplo 4.7 Selección de tuplas con campos calculados 184

Ejemplo 4.8 Selección de tuplas con base en comparación simple. 187
Ejemplo 4.9 Selección con base en la utilización de comparaciones compuestas 188
Ejemplo 4.10 Selección de productos del SIM por rango 189
Ejemplo 4.11 Listado de empleados del SIM por rangos salariales. 190
Ejemplo 4.12 Selección de tuplas del SIM por pertenencia a un conjunto 192
Ejemplo 4.13 Selección de tuplas con pertenencia a múltiples conjuntos 193
Ejemplo 4.14 Listado de tuplas por la no pertenencia (NOT IN) a un conjunto 194
Ejemplo 4.15 Selección de tuplas utilizando las cláusulas LIKE y NOT LIKE. . . 196
Ejemplo 4.16 Producir un listado de empleados que tengan en sus nombres la letra D, seguido por cualquier conjunto de tres caracteres, así: D _ _ _ _ _ 197
Ejemplo 4.17 Listar los productos asociados a los proyectos cuyo número de proyecto termine en 1 197
Ejemplo 4.18 Seleccionar las máquinas del SIM que estando en mantenimiento no tienen comentarios asociados a sus números de órdenes de mantenimiento 198
Ejemplo 4.19 Seleccionar del SIM las órdenes de mantenimiento con comentarios asociados. 199
Ejemplo 4.20 Producir un listado de transacciones del SIM ordenadas ascendentemente por el campo o atributo de movimiento de la transacción. 202
Ejemplo 4.21 Generar un listado que muestre los movimientos de la tabla de transacciones del SIM ordenados ascendentemente, y dentro de cada intervalo de movimiento ordenar las tuplas por detalle de la transacción en orden ascendente 204
Ejemplo 4.22 Generar un listado de empleados del SIM ordenado ascendentemente por el cargo del funcionario, y dentro de los cargos ordenarlo ascendentemente por el salario del empleado, considerando un aumento del 5 por ciento. 206
Ejemplo 4.23 Contar el número de proyectos de la base de datos del SIM con alta prioridad (aquellos en los cuales la prioridad del proyecto es igual a uno). 207

Ejemplo 4.24 Contar el número de mantenimientos distintos que se utilizan en el Sistema Integrado de Manufactura 208

Ejemplo 4.25 Calcular los valores mínimos, máximos y promedios de los montos de los costos de inversión de los proyectos del SIM 209

Ejemplo 4.26 Producir un informe por números de proyectos del SIM ordenados ascendentemente por números productos en el que para cada proyecto se cuente el número de empleados asociados y la suma de sus salarios por proyecto 211

Ejemplo 4.27 Generar un informe agrupado por números de proyectos del SIM ordenados ascendentemente por números de proyectos, en el que para cada proyecto se cuente el número de empleados asociados y la suma de sus salarios por proyecto, para aquellos proyectos que reporten más de dos empleados asociados al proyecto 212

Ejemplo 4.28 Generar un listado de productos que pertenezcan a proyectos liderados por Linda Day. 213

Ejemplo 4.29 Generar un listado de las identificaciones, los nombres y los sueldos de los empleados que pertenecen a los proyectos ubicados en la planta de producción de la ciudad de Madrid (España). 214

Ejemplo 4.30 Generar un listado de transacciones del SIM mostrando el número de la transacción contable, su fecha, el movimiento asociado y la diferencia con el valor promedio total de las transacciones para aquellas transacciones que hayan registrado un movimiento superior al valor promedio de las transacciones. 215

Ejemplo 4.31 Construir un listado de órdenes de transporte, con sus rutas de distribución asociadas y costos de transporte, que sean mayores que el valor mínimo del mínimo costo de transporte . . 216

Ejemplo 4.32 Generar un listado de facturas en el que se muestren las órdenes de venta con las descripciones de los productos, las cantidades vendidas y NIT a los clientes situados en áreas residenciales de apartamentos 218

Ejemplo 4.33 Generar un listado de cuentas del SIM cuyo saldo actual sea mayor que el valor de la compra de las cuentas por pagar en dólares; o sea, aquellas cuentas que registran el valor de 'D' en el tipo de moneda. 219

Ejemplo 4.34 Seleccionar todas las transacciones contables del SIM cuyos movimientos sean menores que los movimientos de las cuentas del sistema contable que hayan registrado un valor de $5'000.000,oo. 220

Ejemplo 4.36 Producir un listado de empleados ordenado alfabéticamente por nombres en forma ascendente en el que se muestre el cargo, su sueldo y el número del proyecto en el cual está laborando 225

Ejemplo 4.37 Seleccionar del SIM los códigos de los máster de producción, identificando las máquinas que operan en dichas líneas de las plantas de producción que tengan en ejecución proyectos del Sistema Integrado de Manufactura 227

Ejemplo 4.38 Generar un listado agrupado por proyectos en el que se especifiquen los cargos de desempeño de cada empleado y cuente el número de empleado(s) por cargo perteneciente a cada proyecto del SIM, ordenado ascendentemente tanto por los números de los proyectos y dentro de cada proyecto por los cargos de los empleados 228

Ejemplo 4.39 Producir un listado ordenado por los nombres de los proveedores que tengan asociados los productos del SIM a los cuales proveen sus materias primas 230

Ejemplo 4.40 Listar la maquinaria asociada a las líneas de las planta de producción del SIM, mostrando tanto las máquinas asignadas como las no asignadas, y mostrar las máquinas por el número de la maquinaria. 230

Ejemplo 4.41 Generar un listado en el que se presenten las ventas de productos utilizando RIGHT JOIN con los productos producidos por el SIM en el que se muestren los productos que no han tenido compradores en la tabla de ventas. 233

Ejemplo 4.42 Construir un listado que muestre en su selección tanto las cuentas por pagar registradas en la tabla de cuentas del SIM como las cuentas del sistema contable del SIM que no están relacionadas con ninguna tupla de cuentas por pagar. 233

Ejemplo 4.43 Seleccionar del SIM los número de las órdenes de transporte con sus números de órdenes de venta asociadas, sus fechas de cargas y costes de seguros para aquellas órdenes de transporte que en sus rutas de distribución en sus entregas son atendidas por transportes aéreos; o para aquellas compañías en las que la identidad de la empresa de transporte tiene el token "AIR" 238

Ejemplo 4.44 Seleccionar del SIM las identificaciones de los productos distintos, con todas sus especificaciones, para los cuales no se haya presentado comprador en la tabla de ventas. 239

Ejemplo 4.45 UNION 241

Ejemplo 4.46 Seleccionar los códigos de las cuentas relacionadas con proveedores que hayan tenido a su vez un registro en la tabla de ventas 243

Ejemplo 4.47 Seleccionar los números de proyectos que habiendo sido asignados a los empleados no estén registrados en la tabla de proyectos 244

Ejemplo 4.48 MINUS con resultados de tuplas del SIM. 245

Ejemplo 4.49 Inserción de tuplas de una tabla de la base de datos 249

Ejemplo 4.50 Inserción de tuplas a la base de datos especificando solo algunas columnas de la tabla 250

Ejemplo 4.51 Inserción de tuplas en una tabla considerando campos nulos . . 251

Ejemplo 4.52 Actualización de columnas, de una tabla en todas sus filas 252

Ejemplo 4.53 Edición de tuplas o filas específicas de una tabla 252

Ejemplo 4.54 Actualización de varias columnas de una tabla 253

Ejemplo 4.55 Cancelación de una tabla de la base de datos 254

Ejemplo 4.56 Cancelación de tuplas de una tabla por cumplimiento de condición 254

SIGLAS Y ABREVIATURAS USADAS

ANSI	American National Standards Institute
BDOO	*Bases de Datos Orientadas por Objetos*
DBMS	Data Base Management Systems
M E-R	*Modelo Entidad – Relación*
MR	*Modelo Relacional*
NERSC	National Energy Research Scientific Computer
SGBD	Sistema de Gestión de Bases de Datos
SPARC	Standars Planning and Requirements Committee - SPARC

Introducción

La información en la sociedad red es un recurso fundamental de decisión que impacta las acciones en los contextos sociales, políticos y económicos de los países. La toma de decisiones acertadas en las organizaciones se fundamenta en informaciones almacenadas que cumplan las características de estar estandarizadas, actualizadas y con disponibilidad inmediata para los usuarios de la empresa. Las características mencionadas en las unidades de información de las empresas, en el marco de un paradigma informacional soportado por la red y al servicio de la sociedad del conocimiento, implican la organización de los datos en sistemas estructurados en bases de datos que operando en red permitan a las organizaciones ser competitivas en sus negocios con una visión global.

Los sistemas de información en bases de datos, a más de permitir el cumplimiento de las características mencionadas, cumplen con la escalabilidad y la portabilidad en las condiciones cambiantes que deben cumplir las organizaciones en su dinámica al tener que adaptarse a nuevos sistemas productivos, flujos de información e informes y reportes dinámicos derivados de la base de datos.

La importancia de la información de decisión para el logro de la productividad de las empresas se fundamenta en las siguientes hipótesis:

> **Hipótesis 1**: La información más precisa mejora las decisiones al reducir las pérdidas.
>
> **Hipótesis 2**: La información que reduce la aversión al riesgo aumenta la productividad cuando provoca acciones que se aproximan a los auténticos niveles de neutralidad del riesgo.
>
> **Hipótesis 3**: Las decisiones centralizadas fundamentan la consistencia de las propias decisiones y la perspectiva global, y evitan la duplicación innecesaria. Las decisiones

> descentralizadas promueven la búsqueda de la información, la distribución de incentivos y la adaptación. La productividad aumenta hasta que la distribución de control equilibra de forma óptima estos factores en función de su complementariedad e indispensabilidad.
>
> **Hipótesis 4**: Los incentivos absolutos favorecen que la información se comparta, lo cual estimula la productividad del grupo; los incentivos relativos desfavorecen que se comparta la información, pero estimulaban la productividad individual. La política óptima de incentivos en términos de productividad tiende hacia lo absoluto cuanto mayor es la independencia de las tareas.
>
> **Hipótesis 5**: El empuje de la información beneficia a los individuos y las organizaciones que controlan los valores infravalorados (los propietarios de estos incurren en pérdidas). La eficiencia aumenta cuando se reequilibra la reubicación de recursos tomando en cuenta los problemas y las oportunidades. [1]

Teniendo en cuenta que la totalidad de las hipótesis planteadas se basan en la información para el logro de la productividad, solo con sistemas de información diseñados y operando en teorías de bases de datos tendrán las empresas una alternativa real de aumentar su productividad apalancada por las bases de datos informacionales de la organización.

El texto que el lector tiene en sus manos desarrolla no solo la totalidad de los conceptos nucleares de bases de datos, sino que se proyecta en teorías relacionadas con las nuevas tecnologías de bases de datos.

Este libro presenta una introducción a los sistemas de bases de datos en su contenido inicial, a fin de exponer las características que diferencian los sistemas de bases de datos con los sistemas de información basados en archivos, e identificar las funciones y roles esenciales de un Sistema de Gestión de Bases de Datos: SGBD (DBMS*: Data Base Management Systems*). Con base en los conceptos introductorios se expone la totalidad del marco teórico y práctico para el diseño de bases de datos, y se presenta una metodología novedosa de diseño que privilegia la captura del contenido semántico en el modelo de datos de la organización. La fase de diseño cumple con el diseño lógico/conceptual de la base de datos, utilizando el Modelo Entidad-Relación (M ER) hasta llegar al modelo de datos físico utilizando la teoría de bases de datos relacionales, lo cual se concreta con el Modelo Relacional (MR). La manipulación de los datos se hace con la creación de los datos vía a poblar la base datos; comandos que siendo desarrollados con un gestor de bases de datos cubren los tópicos desde

los tipos de datos, su definición y control de integridad, hasta llegar al tópico de administración de vistas en la base de datos. El manejo de los datos de la base de datos se desarrolla utilizando lenguaje SQL; se debe tener en cuenta que el desarrollo tanto teórico como práctico de todas las sentencias SQL está referido a un Sistema Integrado de Manufactura (SIM), en cuyo modelo se tratan todas las áreas de la organización en su modelo de datos para crear y operar la base de datos.

Finalmente, el texto presenta una de las tecnologías emergentes en la construcción de *software* representada en la investigación por el diseño de bases de datos orientado por objetos.

Justificación

Los contextos sociales, políticos y económicos de la sociedad actual en la segunda década del siglo XXI (2010-2019) son guiados por el paradigma informacional, el cual está basado en las Tecnologías de la Información y las Comunicaciones (TIC). La Ciencia de la Computación y la Informática se han unido a través de la red Internet para procesar y distribuir grandes volúmenes de información al servicio de la generación *net*. Los volúmenes mencionados constituyen el acervo de conocimiento y documental que forman el registro histórico de la humanidad en cualquier dominio del saber humando. El conocimiento relacionado en su registro histórico y consulta se encuentra almacenado en las bases de datos, distribuidas en los servidores que soportan las unidades informacionales de la macrorred Internet.

Las bases de datos son repositorios que almacenan informaciones de carácter estructurado o no estructurado con sus contenidos tanto sintácticos como semánticos; contenidos cuya esencia constituyen las unidades de conocimiento de las cuales se sirven los países y los ciudadanos que pertenecen a una nación o Estado.

Por lo tanto, es de la mayor importancia el conocimiento relacionado con las fases de planificación, diseño, construcción, operación y mantenimiento de un sistema de bases de datos. El conocimiento mencionado según los lineamientos internacionales de la ACM/IEEE-CS corresponde al área del saber que en la Ciencia de la Computación recibe el nombre de *Information Management* o Administración de la Información. De acuerdo con el organismo relacionado,

> Information Management (IM) concierne principalmente con la captura, digitalización, representación, organización, transformación, y presentación de la información; algoritmos para un efectivo y eficiente acceso y actualización de la información almacenada, modelamiento y abstracción de datos, y técnicas de almacenamiento físico de archivos. El alumno necesita ser capaz de desarrollar modelos conceptuales y físicos de datos, determinar qué métodos de administración de la información y

> técnicas son apropiadas para un problema dado, y ser capaz de seleccionar e implementar una apropiada solución de Administración de la Información (IM) direccionada al diseño relevante concerniendo inclusive con la escalabilidad, accesibilidad y usabilidad (ACM/IEEE- CS. Computer Science Curricula 2013. The Joint Task Force on Computer Curriculum Assocciation for Computing Mechinery IEEE- Computer Society, p. 87).

Teniendo en cuenta la definición anterior, son áreas propias de IM [2] los conceptos de Administración de la información, los sistemas de bases de datos, dentro de los cuales se incluyen los sistemas de gestión de las bases de datos, el modelamiento de datos, la teoría de indexación de las bases de datos, el estudio de las bases de datos relacionales, la comprensión de los lenguajes de definición y consulta de las bases de datos, el procesamiento de transacciones en una base de datos, los sistemas de bases de datos distribuidas, el diseño físico de la base de datos, el estudio de las teorías de los almacenes y bodegas de datos relacionados con la minería de datos; y finalmente el almacenamiento y recuperación de la información en un sistema de bases de datos.

Por lo tanto, la cobertura y necesidad de comprensión de los temas desarrollados justifican la compilación de los conceptos esenciales para comprender los sistemas de bases de datos, en su estructuración conceptual lógica y física, y en sus lenguajes de definición y manipulación de los datos contenidos en la base de datos.

Síntesis conceptual

El desarrollo del contenido de este libro está estructurado en cinco capítulos:

El primer capítulo es una introducción a los sistemas de bases de datos. Inicia presentado los sistemas de información basados en archivos, como base para poder comparar estos sistemas con los sistemas de información construidos en bases de datos. El desarrollo de este tipo de sistemas inicia con la presentación conceptual de bases de datos, en la óptica de mostrar la estructura de un sistema de gestión de bases de datos. Se analiza la independencia lógico/física de los datos, y además se presentan los lenguajes de definición, manipulación y control de datos. Siguiendo el estándar ANSI se expone el flujo de control dentro de un gestor de bases de datos, identificando en dicho marco los tipos de usuarios del sistema de gestión, para finalmente presentar las ventajas de un sistema de gestión de bases de datos y las desventajas y limitaciones de los sistemas de bases de datos, punto en el cual el desarrollo conceptual introductorio del tema de base de datos queda en el nivel para desarrollar los tópicos relacionados con el diseño de base de datos.

Una vez fundamentados los conceptos de introducción a las bases de datos, el capítulo dos muestra el diseño de una base de datos, iniciando con las etapas de diseño del sistema, planteando el diseño conceptual de la base de datos utilizando el Modelo Entidad-Relación y sus constructos básicos según la notación de P. Chen, punto en el cual el lector está listo para aplicar la nueva metodología presentada en este texto para el diseño de bases de datos, la cual está relacionada con capturar la semántica del sistema de bases de datos. El nuevo enfoque presentado para el diseño de bases de datos inicia identificando el análisis del contexto al cual va a servir el sistema, para posteriormente asociar valores semánticos a las componentes identificadas en el contexto. Una vez asociados los valores semánticos, se agrupan las componentes del contexto como base para diseñar los atributos de las entidades y las relaciones, lo cual se complementa con la selección de las claves primarias y foráneas de las en-

tidades de la base de datos, con el fin de pasar a la fase de diseño físico de la base de datos. Concluida la fase de diseño, se expone la operación de la base de datos, su mantenimiento e introducción al diseño de bases de datos a partir de la especificación de requerimientos.

Con base en los desarrollos de los anteriores capítulos, el capítulo tres presenta la creación de la base de datos. Se inicia con la definición de datos, presentando los tipos de datos y el control de integridad de los datos, como bases para entrar a definir los datos en la base de datos en su creación del esquema de la base de datos y sus tablas, para finalmente mostrar el manejo de vistas en una base de datos.

Con base en los conceptos de bases de datos, el diseño de sistemas de información en bases de datos y la creación de la base de datos, el capítulo cuatro presenta el manejo de los datos, la manipulación de los datos y la descripción de los atributos de las tablas en una base de datos. Este capítulo desarrolla completamente el SQL utilizando consultas que se realizan contra una base de datos real de una empresa manufacturera. La manipulación de los datos utilizando el gestor comprueba el funcionamiento desde las consultas simples, pasando por las compuestas hasta llegar a la edición de la base de datos.

Finalmente, el capítulo cinco completa la investigación presentando una de las técnicas más actualizadas para el diseño y desarrollo de *software*, como es el caso del modelado de base de datos orientado por objetos (BDOO). La resolución de problemas del mundo real a través de las BDOO le permite al constructor de *software* la resolución de problemas a gran escala en su complejidad con relación a las bases de datos relacionales. El diseño de BDOO es útil en la estructuración de los datos heredados, agregados, derivados y embebidos con las aplicaciones.

Capítulo 1

Introducción a los sistemas de bases de datos

1.1 Introducción

En el marco de la sociedad informacional y global generadora de conocimientos, los sistemas de información impactan los contextos económicos, sociales y políticos de los países. Los sistemas informáticos en cualquier sector y a cualquier nivel de la economía de un país para responder a la competitividad en la globalidad de la información requieren el almacenamiento y la consulta de grandes volúmenes de datos. Estos volúmenes de datos, siendo de complejidad creciente tanto en número como en significado, requieren ser estructurados con alguna organización para poder ser consultados. Por lo tanto, la estructura, organización y consulta de amplios volúmenes de datos ha generado el concepto de sistemas de bases de datos.

Con base en lo anterior, la conceptualización de este tipo de sistemas requiere la diferenciación entre los sistemas de archivos *versus* los sistemas de bases de datos; la identificación de un Sistema de Gestión de Bases de Datos (SGBD); la identificación de estándares *American National Standards Institute* (ANSI)*; Standars Planning and Requirements Committee* (SPARC) para trabajar en un entorno de bases de datos; y finalmente, identificar tanto los usuarios como sus funciones relacionados con un entorno de bases de datos.

La conceptualización mencionada se convierte para el lector en un punto inicial para explorar acciones de aprendizaje y desarrollo de programas iniciales en bases de datos, con el fin de proyectarse en la acción de investigación relacionada con bases de datos, lo cual ha generado el surgimiento de una industria de servicios de información a nivel mundial. La industria del *software* en bases de datos ha generado a nivel mundial el posicionamiento de empresas tales comom SQL Server, DB2, Sybase, MySQL, PostgreSQL, Teradata, Informix, Ingres y Amazon´s SimpleDB, que son catalogadas como empresas líderes en la tecnología de bases de datos [3].

El desarrollo de los sistemas de bases de datos ha hecho posible el avance del almacenamiento y consulta de grandes volúmenes de información en las distintas áreas del saber humano racional, dentro de las cuales se destacan por su dimensión en tamaño las siguientes: la Librería del Congreso: Library of Congress (USA), la de la Agencia de Inteligencia Central —Central Intelligence Agency—: CIA (USA), Amazon, YouTube, Choice Point, Sprint, Google, AT&T, NERSC (National Energy Research Scientific Computer) y la correspondiente al, World Data Center for Climate [4].

Tanto las compañías de bases de datos como las empresas propietarias de las informaciones almacenadas en las grandes bases de datos mencionadas manejan sectores neurálgicos de la economía mundial, como son los casos de los relacionados con el clima, la educación, la defensa, las comunicaciones, la energía, entre otras. Por lo tanto, el área de bases de datos en su núcleo esencial de estructurar datos en una base de datos relacionados con cualquier área del saber y consultar dichos datos a través de un conjunto de programas de aplicación que interactúan con la base de datos, será en el futuro una de las áreas esenciales que demandarán talento humano preparado que satisfaga la necesidad de construir sistemas de bases de datos.

1.2 Objetivos

Al finalizar el estudio en este capítulo se pretende que el lector aprenda:

- El marco de conceptos de los sistemas de información basados en archivos.
- Las desventajas y limitaciones de los sistemas de archivos.
- El concepto de bases de datos y sus características definitorias.
- El entorno de un Sistema de Gestión de Bases de Datos (SGBD).
- El significado de los lenguajes de definición, manipulación y control de datos.
- La integración del esquema ANSI con el flujo de control de datos en un SGBD.
- Los tipos de usuarios en un sistema de bases de datos.
- Las ventajas, desventajas y limitaciones de un SGBD.

1.3 Síntesis conceptual

El capítulo introductorio a los sistemas de bases de datos inicia desarrollando los conceptos relacionados con los sistemas basados en archivos y analizando las ventajas y limitaciones de este tipo de sistemas. Reconociendo la importancia de los sistemas de información operando en archivo, por cuanto su desarrollo permitió, en mayor o en menor grado, el surgimiento de las bases de datos, el capítulo presenta en el desarrollo de los sistemas de información construidos en bases de datos los conceptos de bases de datos; el sistema de gestión de bases de datos; la independencia lógico/física de datos; y finalmente las ventajas, desventajas y limitaciones de los sistemas gestores de bases de datos como bases teóricas para el desarrollo de los conceptos relacionados con el diseño de bases de datos.

1.4 Sistemas de información basados en archivos

El procesamiento tradicional de sistemas de información ha basado su operación en la estructuración de archivos que soportan los datos almacenados, bases para el procesamiento y la generación de informes útiles al usuario derivados del sistema. El estudio de los sistemas de información basados en archivos se justifica por:

- La necesidad de conceptualizar la técnica de construcción de sistemas de información basada en archivos.
- La importancia de identificar las limitaciones de la técnica basada en archivos.
- La compresión de los problemas que conlleva el mantenimiento y la operación de los sistemas de información basados en archivos.
- La necesidad de migración de sistemas de información teniendo en cuenta la técnica de archivos a sistemas de información fundamentados en la tecnología de bases de datos.

1.4.1 Construcción de sistemas de información basados en archivos

Los sistemas de información de las décadas de los años 60 y 70 se soportaban en un conjunto de programas que administraban uno o varios archivos (o ficheros de datos).

La conceptualización del término "archivo" la definen los textos como

> Un archivo es una secuencia de registros. En muchos casos, todos los registros de un archivo son del mismo tipo. Si todos tienen exactamente el mismo tamaño (en bytes), se dice que el archivo se compone de registros de longitud fija. Si diferentes registros del archivo contienen tamaños distintos, se dice que el archivo está constituido por registros de longitud variable [5].

> Un archivo es simplemente una colección de registros que contienen datos lógicamente relacionados [6].

Con base en las definiciones anteriores, un archivo (A) se define como un conjunto de registros (Ri, $1 \leq i \leq n$) de naturaleza homogénea o heterogénea en su estructura. El conjunto de registros es *homogéneo* cuando la estructura de registro del archivo es de longitud fija; y se dice que un conjunto de registros es *heterogéneo* cuando los registros son de longitud variable. Cada uno de los registros del archivo está compuesto por un conjunto de k campos (Cj, $1 \leq j \leq k$), con una determinada estructura. Esta estructura de campos del archivo es *estática*, o sea, una vez que se ha creado el archivo en términos de los Cj campos que conforman cada registro, los campos no se pueden cambiar ni en su número ni en su definición de contenido (que sean numéricos, alfabéticos o alfanuméricos, etc.). La estructura es *dinámica* en términos de la capacidad de almacenamiento del conjunto de registros del archivo.

Ejemplo 1.1 Registros de un Sistema de Información

Un Sistema de Presupuesto de Obra y Análisis de Precios Unitarios (SPOAPU) se basa en dos archivos: i) El archivo de elementos simples y ii) El archivo de elementos compuestos. Suponga que el archivo de elementos simples está conformado por registros de longitud fija, los cuales están compuestos por los campos: Código del elemento simple, Nombre del elemento simple, cantidad, unidad de medida, valor unitario y código del proveedor. La tabla 1.1 muestra el archivo de elementos simples, en el cual cada registro tiene la misma longitud en bytes.

Tabla 1.1 Archivo de elementos simples

Código	Nombre	Cantidad	Unidad de medida	Valor unitario	Código proveedor	Registros Ris ↓
01 100 001	Cemento	50	Kg	$50 000	Diamante Beta	
01 100 002	Arena	100	Volteo	$25 000	No registra	
01 100 003	Ladrillo	1	Bloque	$20 000	Bloquera ALFA	
01 100 004	Hierro	1	Varilla	$75 000	Steel S. A.	
01 100 005	Agua	1	m^3	$5000,	Aguas Pradera	↑
01 100 006	Piedra	1	Volteo	$30 000	Cantera San José	Registros Homogéneos
01 100 007	Cal	25	Kg	$38 000	Cal el Sol	
01 100 008	Pintura	1	Galón	$80 000	Magic_Kolor	↓
01 100 009	Teja	1	Lámina	$38 000	Tejalit Luminic Dos	
…						
n-registros						
← Campos de los registros de longitud fija →						

Por su parte, suponiendo que el archivo de elementos compuestos está conformado por los campos Código del elemento compuesto, nombre del elemento, unidad de medida, valor unitario, la estructura de un conjunto de registros de los elementos compuestos, estos se muestran en la tabla 1.2.

Tabla 1.2 Elementos compuestos del SPOAPU

Código	Nombre	Cantidad	Unidad de medida	Valor unitario	Código proveedor	Registros Ris ↓
02 100 001	Concreto	50	M3	$50 000	Diamante Beta	
02 100 002	Arena	100	Volteo	$25 000	No registra	
02 100 003	Ladrillo	1	Bloque	$20 000	Bloquera ALFA	
02 100 004	Hierro	1	Varilla	$75 000	Steel S. A.	
02 100 005	Agua	1	m^3	$5000	Aguas Pradera	
02 100 006	Piedra	1	Volteo	$30 000	Cantera San José	↑
02 100 007	Cal	25	Kg	$38 000	Cal el Sol	Registros Homogéneos
02 100 008	Pintura	1	Galón	$80 000	Magic_Kolor	↓
02 100 009	Teja	1	Lámina	$38 000	Tejalit Luminic	
…						
k-registros						
← Campos de los registros de longitud fija →						

Los archivos de soporte al SPOAPU de elementos simples y compuestos son *dinámicos* en su número de registros *n* para los simples y *k* para los compuestos, pero una vez definidos los campos de los archivos en su identidad y sus contenidos, dichas definiciones son *estáticas* y se deben conservar tanto en los programas como en la estructura de archivos en el sistema de información; o sea, seis (6) campos para el archivo de elementos simples y seis (6) campos para el archivos de elementos compuestos. La estructura de registro de ambos archivo es *homogénea* en el sentido de que todos sus registros tienen la misma longitud en bytes.

1.4.1.1 Técnica de construcción de sistemas de información basada en archivos

Una vez conceptualizado el término "archivo" y sus componentes en registros y campos, los sistemas de información basados en archivos son un conjunto de programas de aplicación con la definición de los datos incorporados en la estructura de control de los programas, donde cada programa define y gestiona sus propios datos; y el conjunto de programas está al servicio de los requerimientos de información al usuario sin ninguna integración de datos ni de programas al servicio de la confiabilidad del sistema de información.

Con base en lo anterior, la técnica de construcción de sistemas de información basada en archivos sigue la siguiente metodología:

1. Identificación de los archivos que van a almacenar los datos del sistema de información.
2. Definición de la estructura de registro (fijo, variable) de cada uno de los archivos.
3. Definición de los campos de datos de cada uno de los registros.
4. Construcción de los programas de aplicación, incorporando en la mayoría de los casos la definición de los datos dentro de la lógica de control del programa.
5. Carga y mantenimiento de los archivos de datos, lo cual implica la gestión de datos a través de cada uno de los programas.
6. Operación del sistema de información con base en la estructura de archivos y programas de aplicación.

La metodología anteriormente expuesta tiene el siguiente análisis para la funcionalidad de un sistema de información:

- El diseño de los archivos está al servicio de sistemas de información de procesamiento en lotes o en *batch*, como es el caso de generación de los recibos de pagos de impuestos, servicios públicos o extractos bancarios.
- El conjunto de programas de aplicación que definen y gestionan los datos de un sistema de información basado en archivos son de propósito específico, en el sentido que están al servicio de un proceso dentro del sistema de información, como es el caso del proceso de facturación.
- Los datos en su carga, mantenimiento y confiabilidad son de directa responsabilidad de una oficina específica dentro de la organización, como es el caso de la oficina de facturación.
- Los programas de la aplicación en su estructura tienen incorporada tanto la lógica de control de programa como las instrucciones de definición de los datos de los archivos.
- Los programas para la gestión de datos manejan un archivo que se llama el "archivo maestro" y otro que es el "archivo de novedades". El primero tiene como función almacenar los cambios históricos que se realizan en los datos del archivo, y el segundo cumple con la función de registrar las novedades que van a afectar los datos del archivo maestro en sus operaciones de adición, cancelación o edición de los datos que contienen los registros de los archivos.

Teniendo en cuenta la metodología y el análisis de la técnica basada en archivos para soportar sistemas de información, a continuación se presenta las limitaciones de los sistemas que funcionan con base en archivos.

1.4.1.2 Desventajas y limitaciones de los sistemas de información basados en archivos

La operación de un sistema de información basado en archivos tiene las siguientes *desventajas*:

- El diseño del sistema no tiene en cuenta el contexto de operación del sistema de información para la organización; por lo tanto, los archivos que soportan el sistema de información representan solo una parte de la organización y no toda la organización.

- El no tener en cuenta el contexto del sistema de información se justifica por el hecho de que únicamente se diseñan los archivos, sus campos y contenidos sin tener en cuenta la sintaxis ni la semántica del sistema de información. La *sintaxis* del sistema de información son los átomos esenciales del sistema que hacen que el sistema funcione; en tal sentido, se ha definido el archivo de elementos simples del SPOAPU; pero ¿sirven los registros de los elementos simples para estimar un presupuesto de obra en ingeniería civil para una licitación? Por su parte, la *semántica* del sistema de información es el significado de cada uno de los átomos que componen en sistema; los átomos en este caso son los campos de los registros. Estos campos contienen solamente datos que a nivel semántico no tienen ningún significado.
- Con base en que no se tiene en cuenta el contexto del sistema de información, los datos no están interrelacionados; y si los datos no están interrelacionados no es posible reflejar la operación total de la organización a través de este tipo de sistemas de información.
- Si los datos no están interrelacionados y, consecuentemente, no reflejan la gestión de información de toda la organización, no es posible con este tipo de sistemas basado en archivos crear los indicadores de la organización, tales como indicadores operativos, de gestión-operativa, de gestión y de gestión-predicción en la empresa; simplemente porque la generación de estos índices depende de varias unidades organizacionales de la empresa.

Por su parte, las *limitaciones* de la técnica basada en archivos son las siguientes:

- *Generación de datos duplicados*: Dado que cada una de las unidades organizacionales de la empresa no comparten los datos, el proceso de creación y mantenimiento de los datos en la organización genera diferentes datos en varias dependencias y, consecuentemente, se pierde la integridad de la información. Tal es el caso cuando en el SPOAPU se tiene una obra en una ciudad base (Barranquilla) y otra obra civil en una ciudad alterna (Cartagena de Indias) y ambas obras utilizan diferente precios en el archivo de elementos simples. Por lo tanto, la duplicidad de datos genera inconsistencia en los datos.
- *Dependencia lógico/física entre los programas y los datos*: Dado que la definición de los datos está incorporada en los programas, cada vez que se vayan a cambiar los datos se tienen que cambiar los programas. Por consiguiente, esta dependencia lógico/física genera dos tipos de errores: i) Los *errores lógicos* en el sistema de información, lo cual implica que si hay varios programas mane-

jando uno o varios archivos, todos los programas que afectan los archivos se deben cambiar; ello implica que el sistema de información se sale del control en su lógica de operación. ii) Los *errores físicos*, por cuanto varios programas están afectando a varios archivos, pero esos varios archivos no son consistentes entre sí; debido a lo cual se genera la pérdida de control en la integridad de los datos de los archivos.

- *Generación de formatos incompatibles en los archivos*: Con base en el hecho de que la definición de los datos está incrustada en los programas de aplicación, entonces las estructuras de los archivos dependen tanto del lenguaje de programación que se utilice como de la lógica de control de los programas. Por lo tanto, se tiene como resultado un agregado de archivos desestructurado, por cuanto los formatos de los archivos no son compatibles entre sí.
- *Generación de islas de datos*: Con base en que en este tipo de sistemas no se tiene en cuenta el contexto de operación del sistema y la totalidad de la organización, cada unidad organizacional genera sus propios archivos y datos; luego, si los datos están en archivos separados, no es fácil que estén disponibles. Por lo tanto, la no disponibilidad de datos genera inconsistencia y falta de integridad de los datos.
- *Consultas fijas a los archivos*: La gestión sobre los datos en términos de mantenimiento y recuperación de informaciones de los archivos solo se puede hacer con programas de aplicación cuyas lógicas de control sirvan para responder únicamente consultas de propósito específico. Ello implica que cada consulta a los archivos de datos debe ser solucionada necesariamente por un programa, y en el caso de que cambie la consulta se debe cambiar también el programa. Luego son consultas que pueden ser solucionadas en la mayoría de sus casos por lenguajes procedimentales; por lo tanto, los sistemas de información basado en archivos tienen la limitación estructural de no permitir consultas generales sobre los datos o consultas *ad hoc*, o sea, consultas que no estén predefinidas sobre los datos.
- *Aumento de programas de aplicación*: El costo del mantenimiento de este tipo de sistemas aumenta; porque como cada consulta al sistema de información necesita un programa de propósito específico, entonces cualquier consulta, por sencilla que sea, requiere de la construcción de un programa.
- *Son estáticos en la gestión de datos*: Dado que desde su misma génesis de la creación de los archivos esta no considera el contexto del sistema en sus ca-

> racterísticas sintáctico-semánticas, menos puede considerar la dinámica cambiante del sistema de información que está modelando la organización en términos de necesidades de información. Por lo tanto, los sistemas de información con base en archivos tienen la limitación de que no son capaces de resolver los cambios del entorno al cual sirven en términos de necesidades de información.

En conclusión, teniendo en cuenta las desventajas y limitaciones de la técnica basada en archivos como apoyo a la construcción de sistemas de información, se requiere: i) Independizar los programas de aplicación de los datos. ii) Tener un control sobre la gestión de los datos que no esté incorporada en la lógica de control de los programas. iii) Explorar otras alternativas de tipos de lenguajes de programación que puedan atender la solución de consultas predefinidas o de propósito específico en los sistemas de información. iv) Una metodología y arquitectura para la construcción de sistemas de información que puedan absorber las características dinámicas de la organización en términos de necesidades de información. Por lo tanto, para cumplir con los requerimientos mencionados es necesario explorar los conceptos relacionados con la tecnología de bases de datos como apoyo a la fabricación de sistemas de información.

1.5 Sistemas de información construidos en bases de datos

1.5.1 Concepto de *bases de datos*

El término "base de datos" conceptualmente se define como

> La colección de datos, normalmente denominada base de datos, que contiene información relevante para una empresa [7].

> Una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización [6].

> Una base de datos es un conjunto de datos relacionados entre sí. Por datos entendemos hechos conocidos que pueden registrarse y que tienen un significado implícito [5].

> La base de datos es una colección de ocurrencias de múltiples tipos de registro, pero incluye además las relaciones que existen entre registros, entre agregados de datos y entre ítem de datos [8].

Con base en las definiciones anteriores, el término "bases de datos" en este libro se entiende como la representación a nivel integrado de una colección estructurada de datos que contienen físicamente el diseño lógico de un conjunto de entidades, instancias de las diferentes entidades del sistema de información que se está modelando en una organización y las interrelaciones de las entidades; representación que necesita de una gestión de datos a fin de ser utilizados de una forma compartida por todos los usuarios de una organización en la resolución de sus necesidades de información.

En la definición anterior se destacan varios conceptos, los cuales se exponen en orden de complejidad creciente:

- *Dato*: es el átomo nominado más pequeño de información de las base de datos, el cual está compuesto por cualquier número de bytes o bits.
- *Datos estructurados*: es una colección de ítems de datos con un significado sintáctico y semántico para la organización o empresa cuyo agrupamiento conforma agregados de datos. En un sistema contable, un código de cuenta está conformado por los dígitos 01010101001, en el que la sintaxis está compuesta por el conjunto de dígitos; y su semántica de izquierda a derecha para la organización significa lo siguiente: 01=Activo, 01=Activo Corriente, 01= Bancos, 01= Bancos Internacionales, y 001= *Sea Ligth Bank*.
- *Entidad*: es un objeto único que pertenece a una organización, el cual va a ser modelado en su sintaxis y su semántica a través de un diseño lógico para ser representado físicamente en una base de datos. Son objetos singulares o únicos a nivel genérico: las personas, las cosas, los espacios, los tiempos, las normas, los acontecimiento, entre otros. Son entidades singulares concretas: un estudiante, en un sistema de información académico; un artículo, en sistema comercial de inventarios; un cliente, en un sistema bancario; y un mapa es un sistema de posicionamiento geográfico, entre otros casos.
- *Instancia*: es una ocurrencia concreta de una entidad. Son instancias en un sistema de información académico pertinentes a la entidad de ESTUDIANTE, los contenidos de información de las alumnas de la unidad académica —Programa de Sistemas—, supuestos los atributos de la entidad ESTUDIAN-

TE, conformados por: Código del alumno, Nombre del estudiante, sexo del estudiante, nivel y código del Programa. Concretamente, la instancia parcial presentada en la tabla 1.3, con las características semánticas de que: i) Los cuatro primeros dígitos del código (2013) es el año de ingreso de la alumna; los dos siguientes es el semestre de ingreso (01, primer semestre), los dos siguientes (13) el código del programa; y finalmente, los tres últimos dígitos son el número asignado al alumno. ii) Son alumnas por cuanto el sexo es (F) femenino. iii) Los niveles (n) en el instanciamiento están en el rango [1;10], niveles o semestres. iv) El código del programa de Sistemas es el 13.

Tabla 1.3 Instancia de la entidad ESTUDIANTE en un Sistema de Información Académico

CÓDIGO	NOMBRE_ALUMNO	SEXO	NIVEL	PROGRAMA
20130113010	ALVEAR DIANA	F	1	13
20130113011	BECERRA MARTHA	F	2	13
20130113012	CARDENAS OLGA	F	3	13
20130113013	CAMACHO JANETH	F	4	13
20130113014	DE ARCO JULIANA	F	4	13
20130113015	ESQIVEL ANDREA	F	5	13
20130113018	FANDIÑO ANA LUISA	F	6	13
20130113019	GUTIERREZ TERESA	F	7	13
20130113022	HARKER MARGARETH	F	1	13
20130113035	KEMPE YOLANDA	F	9	13

- *Interrelación entre entidades*: es el grado de asociación entre las entidades. Una entidad "estudiante" se asocia a una entidad "profesor" en la semántica del proceso educativo enseñanza-aprendizaje. Una entidad "cliente" se asocia a una entidad "préstamo" en el proceso financiero-contable del portafolio de servicios de una entidad bancaria al otorgar préstamos a sus clientes.
- *Diseño lógico*: refiere al conjunto de entidades y relaciones que modelan la representación de un sistema de información para una organización. Uno de los modelos de diseño de la lógica de un sistema es el Modelo Entidad-Relación (M E-R).

- *Diseño físico*: es la representación en estructuras de datos del diseño lógico de un sistema de información utilizando la tecnología de bases de datos. Las estructuras de datos que físicamente representan en la máquina un diseño lógico basado en un M E-R son las tablas; punto en el cual físicamente la organización a través del sistema de información se modela a través del Modelo Relacional (MR).
- *Gestión de datos*: son las operaciones de carga, edición y consulta de los datos almacenados en el Sistema de Base de Datos.
- *Usuario*: son los actores que construyen y se sirven de los datos contenidos y procesados de la base de datos. Los usuarios se suelen clasificar en dos tipologías: i) Usuarios no informáticos de la base de datos: son los que realizan operaciones de consulta a la base de datos utilizando sistemas de menús o aplicaciones soportadas por programas de propósito específico. Corresponde a estos niveles: Un estudiante de Medicina en un sistema académico o un cliente en un sistema bancario. ii) Usuarios informáticos: son los expertos conocedores en arquitecturas de *hardware-software* que permiten la creación y gestión de la base de datos. Pertenecen a este nivel diseñadores, programadores y administradores del sistema de base de datos.
- *Organización*: unidad empresarial de cualquier sector económico cuyas decisiones van a ser soportadas por sistemas de información construidos en bases de datos.
- *Modelado de una organización*: es la representación lógica y física de los flujos de información de la organización. Representación lógica que se puede modelar gráficamente utilizando el M E-R; y representación física que se puede modelar en términos de estructuras de datos físicas en el computador a través del MR.

La concepción del concepto de bases de datos requiere en la carga de los datos, el mantenimiento de los mismos, y su procesamiento, el concepto de un Sistema de Gestión de Bases de Datos.

1.5.2 Entorno de operación de un Sistema de Gestión de Bases de Datos (SGBD)

Un Sistema de Gestión de Bases de Datos (SGBD) es un sistema construido en *software* que permite crear, editar y controlar el acceso de la base de datos.

Las componentes de un SGBD son: i) El *hardware* (contenedor de datos). ii) El *software* (programas de aplicación). iii) Los datos (átomos de la base de datos). iv) Los procedimientos (reglas que gobiernan la base de datos). v) Los usuarios (personas que interactúan con la base de datos); entorno, para cuyo entendimiento se necesita el desarrollo. Luego, los datos como átomos sobre los cuales el SGBD hace la función esencial de gestionarlos, entonces el SGBD es realmente una interface (puente) de comunicación entre los usuarios (externos o de visión lógica de la base de datos) y los componentes de *hardware-Software* del computador (internos o de visión física de la base de datos).

Las organizaciones *American National Standard Institute*: ANSI, y *Standard Planning and Requirements Committee*: SPARC (ANSI-SPARC), que propenden por la estandarización de teorías y prácticas, en este caso en el área de bases de datos, proponen una arquitectura de tres (3) niveles de abstracción en la arquitectura de un SGBD. Niveles para cuyo entendimiento es necesario desarrollar el concepto de "esquema".

El esquema de la base de datos es un elemento fundamental en la comprensión de la arquitectura de un SGBD, el cual tiene como función independizar el gestor de la base de datos o el SGBD de la base de datos propiamente dicho; logrando de esta forma independizar la gestión de la base de datos a través del gestor de los datos almacenados físicamente.

Luego, la arquitectura de tres (3) niveles propuesta por la ANSI-SPARC que a través de los esquemas de la base de datos permite la descripción global de la base de datos está compuesta por:

1. *El nivel o esquema externo (Teoría)*: Es el conjunto de visiones o enfoques lógicos que los usuarios tienen de la base de datos. Está compuesto por los procesos, los programas de aplicación y los usuarios que tienen la concepción de las diferentes visiones, enfoque o vistas (siendo la vista un subconjunto particular de la base de datos) de las base de datos. Corresponde también este esquema al soporte teórico del contexto de funcionamiento de la base de datos en la lógica de operación. En tal sentido, a nivel lógico la vista de un cliente en un Sistema Bancario es únicamente su cuenta bancaria personal; pero la vista lógica de un asesor del banco son el conjunto de procesos y aplicaciones del banco que soportan el portafolio de servicios a fin de asesorar a los usuarios.

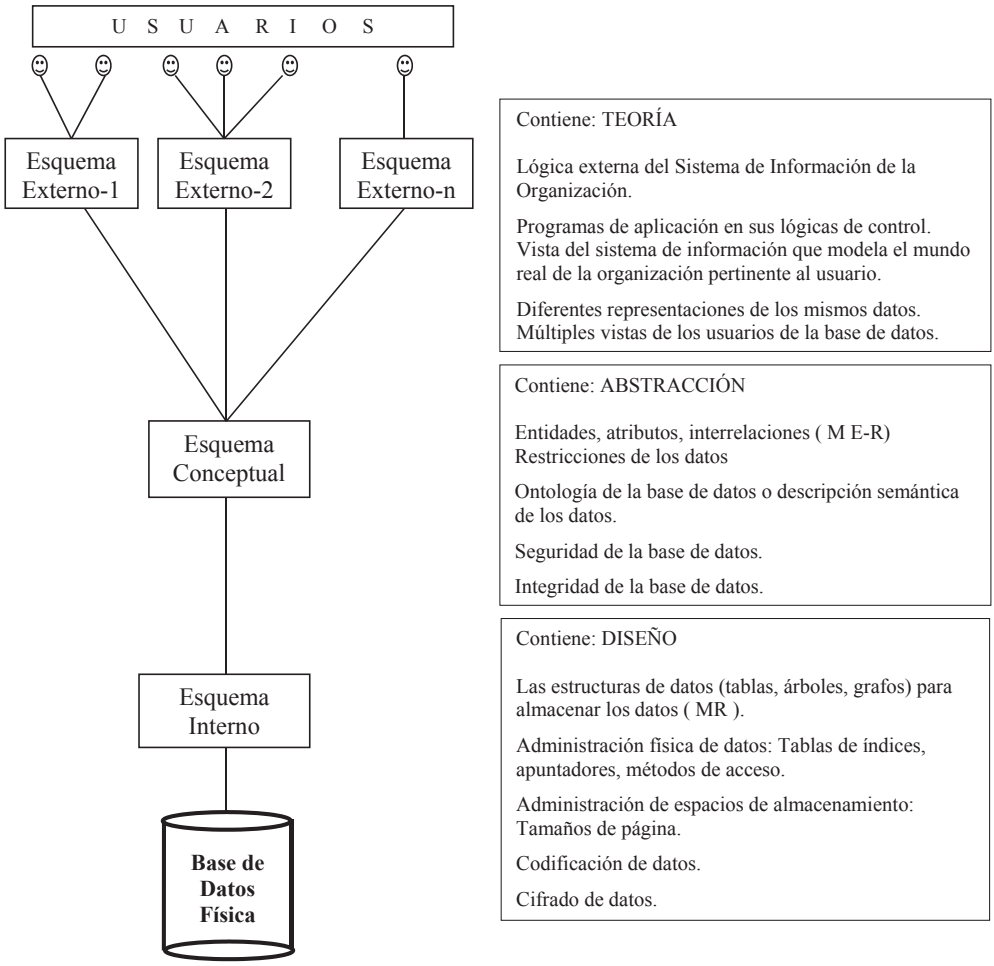


Figura 1.1 ANSI-SPARC Arquitectura de tres niveles

2. *El nivel conceptual o esquema conceptual (Abstracción)*: Es la descripción lógica esencial, única y global de la base de datos; este esquema describe a nivel conceptual el almacenamiento de los datos en la base de datos y las interrelaciones existentes entre los datos. El esquema corresponde en sus conceptos a la abstracción que se hace del contexto de operación del sistema de información de la organización al ser modelado por la tecnología de bases de datos.

3. *El nivel interno o esquema interno (Diseño)*: Es la descripción física de la base de datos contenida en la computadora. Este esquema corresponde al diseño físico de la base de datos o lo que es equivalente, al conjunto de estructuras de datos físicas donde se encuentran almacenados los datos o átomos esenciales de la base de datos.

Los niveles expuestos anteriormente según la arquitectura ANSI-SPARC se muestran en la figura 1.1, en la cual se identifican tanto los esquemas como sus niveles correspondientes y el paralelo de los procesos: 1) Teoría, paralelo al nivel externo, el cual es el fundamento teórico del sistema de información que se está modelando con la tecnología de bases de datos. 2) Abstracción, paralelo al nivel conceptual o el modelado abstracto de sistema de información en bases de datos. 3) Diseño, en su paralelismo al nivel interno y corresponde al almacenamiento físico en su soporte a través de las estructuras de datos.

1.5.3 Análisis de la independencia lógica/física de datos en un sistema de base de datos

La importancia de la arquitectura ANSI-SPARC desarrollada en la sección anterior es base para el logro de la independencia de los datos; lo cual implica que los niveles conceptual y externo no se ven afectados por los cambios en el nivel interno que se haga en la base de datos física. Adicionalmente, es de la mayor importancia resaltar que la definición de los datos no va incorporada dentro de la lógica de control de los programas de aplicación que constituyen el sistema de información. La independencia se puede analizar desde dos puntos de vista: independencia lógica e independencia física.

1.5.3.1 **Independencia lógica de datos**

La independencia lógica, con base en la arquitectura ANSI-SPARC, se presenta cuando los usuarios del nivel o esquema externo no se ven afectados por los cambios que se presenten en el nivel conceptual, ni tampoco por los cambios que se presenten en el mismo nivel externo.

Los cambios en el esquema conceptual, con base en el Modelo Entidad-Relación (M E-R), implican que se pueden efectuar cambios de entidades, atributos e interrelaciones en el M E-R que al ser reflejados a nivel conceptual en la base de datos no im-

plican ni la modificación de los esquemas existentes de la base de datos ni el cambio de los programas de aplicación existentes que soportan el sistema de base de datos. Adicionalmente, los cambios en el esquema externo solo deben afectar a los usuarios implicados en el cambio, no a los demás usuarios del esquema externo.

Ejemplo 1.2 Comprobación de la independencia lógica

Suponga que un Sistemas de Inventarios tiene la entidad PROVEEDORES con los campos se que muestran en la figura 1.2.

Name	Null?	Type
NIT_PROVEEDOR	NOT NULL	VARCHAR(50)
NOMBRE_PROVEEDOR	NOT NULL	VARCHAR(50)
DIRECCION_ PROVEEDOR	NOT NULL	VARCHAR(50)
CIUDAD_PROVEEDOR		VARCHAR(50)
PAIS_PROVEEDOR		VARCHAR(50)
EMAIL_PROVEEDOR	NOT NULL	VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 1.2 Atributos de la entidad PROVEEDORES

Suponga que se desea un listado de PROVEEDORES del Sitema de Inventarios, el cual se presenta en la figura 1.6; es de la mayor importancia resaltar que la instrucción que realiza la impresión del listado es el SELECT presentado en la figura en mención.

Se hace necesario adicionar a la entidad PROVEEDORES el campo del teléfono del proveedor, identificado con TEL_PROVEEDOR de características numéricas de longitud 10, lo cual se presente en la figura 1.3.

```
ALTER TABLE proveedores
ADD tel_proveedor NUMERIC(10)
```

Figura 1.3 Cambio del esquema conceptual de la base de datos del Sistema de Inventarios en su entidad PROVEEDORES

La descripción de la tabla alterada en sus contenidos de atributos se muestra en la figura 1.4, donde se observa que el número de atributos de la entidad cambió, por cuanto se le adicionó el campo de los teléfonos de proveedores.

```
DESCRIBE proveedores;
```

Name	Null?	Type
NIT_PROVEEDOR	NOT NULL	VARCHAR(50)
NOMBRE_PROVEEDOR	NOT NULL	VARCHAR(50)
DIRECCION_PROVEEDOR	NOT NULL	VARCHAR(50)
CIUDAD_PROVEEDOR		VARCHAR(50)
PAIS_PROVEEDOR		VARCHAR(50)
EMAIL_PROVEEDOR	NOT NULL	VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
COD_CUENTA	NOT NULL	VARCHAR(50)
TEL_PROVEEDOR		**NUMBER(10)**

Figura 1.4 Atributos de la entidad PROVEEDORES del Sistema de Inventarios

Finalmente, se comprueba que la instrucción base para imprimir el listado de PROVEEDORES es la misma, como se presenta en la figura 1.7, lo cual implica que no es necesario reescribir la instrucción en su lógica de control para producir el nuevo listado de PROVEEDORES, redefinido con el esquema conceptual variado.

Ejemplo 1.3 Cambios en el esquema externo con relación a la independencia lógica

Suponga que un Sistema de Bases de Datos de una empresa constructora tiene en su esquema externo las vistas de los usuarios: Propietarios, Contadores e Ingenieros. Los propietarios son los dueños de los proyectos de obras arquitectónicos y civiles. Los contadores cumplen las funciones financiero-contables de las obras. Finalmente, los ingenieros son los responsable de la construcción de obras civiles.

Para la constructora es necesaria la creación de un nuevo esquema externo correspondiente a la vista de los Asesores, que son las personas encargadas de la promoción y ventas de los resultados de las obras civiles de la constructora.

La figura 1.5 muestra que a pesar de haber variado el esquema externo, lógicamente el nuevo esquema en la vista creada de asesores se soporta en los esquemas conceptual e interno de la base de datos.

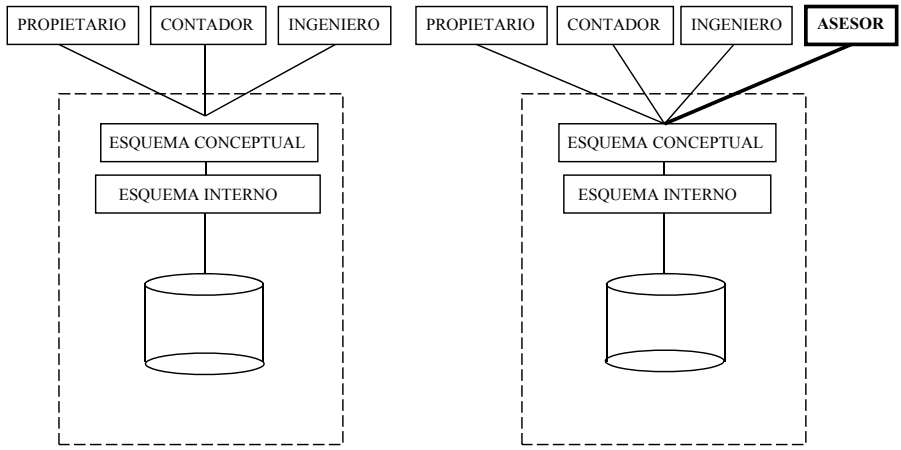


Figura 1.5 Independencia lógica con relación a cambios en el esquema externo

```
/* Listado de proveedores del Sistema de Inventarios */
/* Antes del cambio del esquema CONCEPTUAL */
SELECT * FROM proveedores;
```

NIT_ PROVEEDOR	NOMBRE_ PROVEEDOR	DIRECCION_ PROVEEDOR	CIUDAD_ PROVEEDOR	PAIS_ PROVEEDOR	EMAIL_PROVEEDOR	COD_ PRODUCTO	COD_CUENTA
72435228001	DELL CORPORATION	HIGH STREET 101 No. 14-28	SEATLE	USA	main_100dell@del.usa.com.co	100-100-101-002	10-001-001-001-004
72435228002	TOSHIBA COMPANY	Tokyo-to Chuo-ku Yaesu 1-Chome 5-ban 3-go	TOKIO	JAPAN	main_001TOSHIBAI@TOSHIBA.com.co	100-100-101-003	10-001-001-001-005
72435228003	IBM	Carrera 53 No. 100 - 25. Bogotá Colombia	BOGOTÁ	COLOMBIA	main_010ibm@ibm.com.co	100-100-101-004	10-001-001-011-000

Figura 1.6 Listado del contenido de la tabla de PROVEEDORES antes del cambio del esquema conceptual

```
/* Listado de proveedores del Sistema de Inventarios */
/* Después del cambio del esquema CONCEPTUAL */

SELECT * FROM proveedores;

/* Note que LÓGICAMENTE la instruccion de control del listado de
proveedores no cambia, aunque el */
/* Esquema conceptual del base datos de soporte al Sistema de
Inventarios en su entidad PROVEEDORES */
```

NIT_ PROVEEDOR	NOMBRE_ PROVEEDOR	DIRECCION_ PROVEEDOR	CIUDAD_ PROVEEDOR	PAIS_ PROVEEDOR	EMAIL_PROVEEDOR	COD_ PRODUCTO	COD_CUENTA	TEL_ PROVEEDOR
72435228001	DELL CORPORATION	HIGH STREET 101 No. 14-28	SEATLE	USA	main_100dell@del.usa.com.co	100-100-101-002	10-001-001-001-004	
72435228002	TOSHIBA COMPANY	Tokyo-to Chuo-ku Yaesu 1-Chome 5-ban 3-go	TOKIO	JAPAN	main_001TOSHIBAI@TOSHIBA.com.co	100-100-101-003	10-001-001-001-005	
72435228003	IBM	Carrera 53 No. 100 - 25. Bogotá Colombia	BOGOTÁ	COLOMBIA	main_010ibm@ibm.com.co	100-100-101-004	10-001-001-011-000	

Figura 1.7 Listado de la entidad PROVEEDORES después de realizar el cambio conceptual

1.5.3.2 **Independencia física de datos**

Conceptualizar la independencia física en el marco de la arquitectura ANSI-SPARC implica que cualquier cambio que se presente en el esquema interno de la base de datos no debe afectar los esquemas conceptual y externo de la base de datos.

El esquema interno de la arquitectura mencionada hace referencia al diseño físico concreto de la base de datos en sus estructuras de datos de soporte, para que el motor o gestor de la base de datos funcione en su gestión a la base de datos. Luego, un cambio en el esquema interno debe necesariamente cubrir el proceso de rehacer la base de datos física.

El cambio de la base de datos implica: la generación de nuevos métodos de acceso a los datos; la reorganización de índices de la base de datos; las variaciones en la ubicación de los datos (registros físicos) de la base de datos; las alteraciones en los métodos de compresión de los datos; y las modificaciones en los métodos de ciframiento de la información, entre otros casos. Luego, si se cumple la independencia física de la base de datos, estos cambios no deben afectar los programas de aplicación de la base de datos, por cuanto no están variando los esquemas conceptual ni externo de la base de datos.

Ejemplo 1.4 Cambio físico de la base de datos

Suponga que se ha analizado el desempeño de una base de soporte a los sistemas de información en una organización. El análisis ha dado como conclusión que debe rediseñar el esquema interno de la base de datos, crear un esquema nuevo y observar su desempeño en términos de tiempo de respuesta y espacios de almacenamiento de la base de datos.

La figura 1.8 presenta el cambio del esquema interno de la base de datos anterior a un esquema nuevo de la base de datos; y se resalta que los esquemas externo y conceptual no tienen ninguna afectación al desechar el esquema anterior de la base de datos.

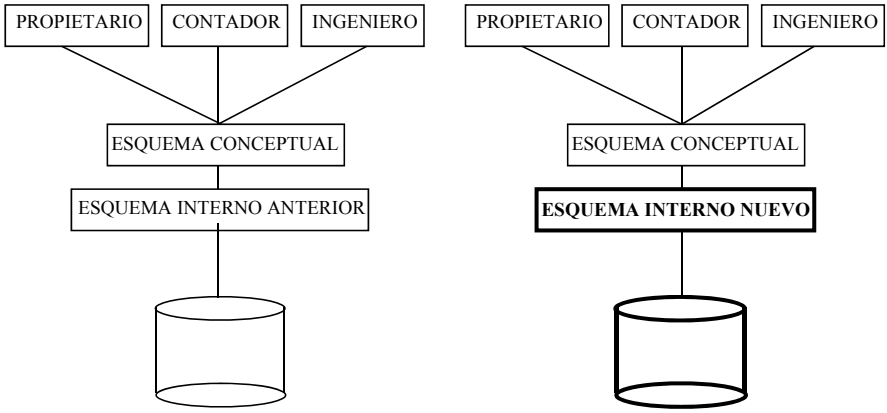


Figura 1.8 Independencia física de los datos

1.5.4 Lenguajes de un SGBD

La construcción del sistema de información basado en la tecnología de base de datos utiliza el Modelo Entidad-Relación y el Modelo Relacional. El Modelo Relacional fue propuesto por E. F. Codd en 1970, y para el acceso a los datos le fue asociado a este modelo el cálculo de predicados. Luego, con base en el modelo y el cálculo mencionado, la empresa IBM construyó el lenguaje SEQUEL, que por sus siglas en inglés significa "Structured English Query Languge"; lenguaje que fue implementado para los Sistemas de Gestión de Bases de Datos [9].

Teniendo en cuenta el desarrollo del lenguaje SEQUEL, el lenguaje SQL está compuesto por tres sublenguajes para la administración de los datos contenidos en la base de datos: i) El DDL o *Data Definition Languaje*; ii) el DML o *Data Manipulation Languaje* y iii) el DCL o *Data Control Languaje*[10].

1.5.4.1 Lenguaje de definición de datos

El lenguaje de definición de datos o DDL (en inglés *Data Definition Languje*) permite definir los datos a la base de datos en sus componentes de entidades, atributos, relaciones para que los programas de aplicación puedan funcionar a través del SGBD;

definición que permite asociar tanto las reglas de integridad como la normas de seguridad de los componentes de la base de datos.

Los comando DDL permiten crear (CREATE), alterar (ALTER), cancelar (DROP) y truncar (TRUNCATE) tablas de la base de datos; siendo el resultado de la compilación de las instrucciones DDL un conjunto de tablas que conforman el catálogo del sistema. Catálogo que está compuesto por el directorio de datos y el diccionario de datos.

1.5.4.2 **Lenguaje de manipulación de datos**

El lenguaje de manipulación de datos (DML o *Data Manipulation Languje*) permite realizar las operaciones de creación (INSERT), modificación (UPDATE), consulta (SELECT) y cancelación (DELETE) de los datos de la base de datos.

Luego, el DDL permite construir instrucciones que afecten el contenido de los datos a través del SGBD; lo cual se puede hacer en dos niveles: i) A nivel externo, proporcionando el lenguajes de comandos sencillo que permite la interacción eficiente entre el usuario y la base de datos y, consecuentemente, una alta productividad con bajos tiempos de programación en la fase de producción del sistema. ii) A nivel interno, permite interactuar a bajo nivel con los datos vía a lograr la optimización de los tiempos de respuesta en la gestión de los datos.

1.5.4.3 **Lenguaje de control de datos**

El lenguaje de control de datos (DCL o *Data Control Languaje*) pemite que el SGBD controle el acceso a los datos de la base de datos.

En el control de acceso a los datos se puede: i) otorgar (GRANT) permisos a los usuarios para hacer las acciones de conexión (CONNECT), selección (SELECT), inserción (INSERT), modificación (UPDATE), cancelación (DELETE) y cambiar el contexto de operación de la base de datos (USE); ii) cancelar los permisos de acción sobre la base de datos (REVOKE).

1.5.5 Integración de esquemas ANSI al flujo de control de datos en un SGBD

Esta sección intenta integrar los niveles de la arquitectura ANSI-SPARC con el flujo de control de datos del Sistema de Gestión de Base de Datos (SGBD), en el interior del cual se encuentra el *software* que hace la función del Gestor de la Base de Datos (GBD) propiamente dicho.

La estructura de integración mencionada se presenta en la figura 1.9. Con base en la arquitectura ANSI-SPARC, y los actores de base de datos, la integración entre la arquitectura y los actores se justitifica con el siguiente análisis. Los ususarios de la base de datos utilizan los esquemas externo y conceptual de la base de datos; en tal sentido, son usuarios cuya función principal es consultar informaciones de la base de datos. Los programadores se fundamentan en el esquema externo y conceptual con el fin de construir los programas de aplicación (aunque se aclara que para casos complejos de resolución de problemas de programación les pueden llegar a través de los aplicativos por consultar el esquema interno o físico de la base de datos). Finalmente, el administrador de la base de datos es el directo responsable del esquema interno, el cual contiene el diseño físico de la base de datos.

Los programadores al construir su programa de aplicación se comunican con el SGBD cumpliendo los siguientes pasos: i) El código en el respectivo de lenguaje de programación dentro del SGBD pasa por un preprocesador DML, el cual se encarga de convertir la instrucciones del DML encapsuladas en el programa a funciones estándar en el lenguajes host. ii) El DML interactúa en este punto con el procesador de consultas con el fin de generar el código apropiado. iii) Las dos acciones anteriores (i) y ii)) dan como resultado el código de máquina o en lenguaje objeto del programador. iv) El código pasa al Gestor de Bases de Datos, GBD. v) En el GBD (figura 1.9), lo primero que se chequea son los controles de autorización contra la base de datos. v) Verificadas las autorizaciones, el programa pasa al procesador de comandos, y en este paso interactúa con el comprobador de integridad y el optimizador de consultas, lo cual da como resultado una transacción. vi) La transacción se comunica con el Gestor de Datos. vii) El Gestor de Datos se comunica con el gestor de archivos. viii) Es el gestor de archivos quien activa los métodos de acceso a los datos. ix) Los métodos de acceso a los datos interactúan con los búferes del sistema. x) Finalmente, son los búferes los que en últimas interactúan tanto con los datos físicos la base de datos física como con la metadata en la cual se encuentra el catálogo del

sistema de bases de datos. Luego, con base en lo anterior, es el gestor de archivos con la operación del método de acceso quien extrae la información de los búferes del sistema y al pasar por el Gestor de Datos utilizando el gestor de recuperación de datos hace interactuar el Gestor de la Base de datos (SBD) interno al Sistema de Gestión de Bases (SGBD); y este en últimas le responde a través de la ejecución del programa al programador de la base de datos.

El flujo de control y datos de una consulta del usuario que está relacionado con los niveles externo y conceptual efectúa las siguientes acciones: i) El usuario construye una consulta a la base de datos. ii) La consulta entra al SGBD e interactúa con el preprocesador de consultas, el cual se encarga de convertir la consulta a instrucciones en lenguaje de máquina, las cuales pasan al Gestor de la Base de datos (GBD). iii) El GBD comprueba la autorización de la consulta a la base de datos. iv) Autorizada la consulta, se comprueba la integridad de la consulta a través del procesador de comandos. v) Este procesador pasa los resultados al gestor de transacciones, y este a su vez al Gestor de Datos. vi) Es el gestor de datos quien ordena al gestor de archivos la operación de la transacción. vii) El gestor de archivos activa el método de acceso. viii) El método de acceso se comunica con los búferes del sistema. ix) Los búferes del sistema interactúan directamente con los datos físicos del sistema y la metadata, la cual representa el catálogo del sistema. La recuperación de la información para solucionar la consulta del usuario en un sentido inverso inicia en la base de datos física, pasa al sistema de búferes, y a través del método de acceso en su ejecución se comunica con el gestor de archivos; este *software* entrega los resultados al gestor de datos (GD). El GD a través del gestor de recuperación entrega los resultados de la consulta al SGBD, y este a su vez da los resultados de la consulta al usuario.

Finalmente, las operaciones que efectúa el administrador de la base de datos son: i) La acción del administrador de la base de datos pasa al SGBD, y la recibe el compilador DDL; la función de este compilador es convertir las instrucciones en DDL en un conjunto de tablas que contienen la metada (son los datos acerca de los datos almacenados físicamente). ii) El compilador DDL entrega sus resultados al gestor de catálogo (aclarando que el catálogo del sistema es el que se utiliza en la gran parte de los SGBD). iii) Es el gestor del catálogo el que se comunica con el gestor de archivos. iv) El gestor de archivos opera el método de acceso. v) El método de acceso se comunica con los búferes del sistema y vi) Estos búferes finalmente interactúan con el catálogo del sistema. Luego, el catálogo del sistema y los datos físicos del sistema devuelven los resultados de los búferes al gestor de archivos, y este los pasa al gestor

del diccionario de datos, el cual retorna la acción ejecutada al administrador de la base de datos.

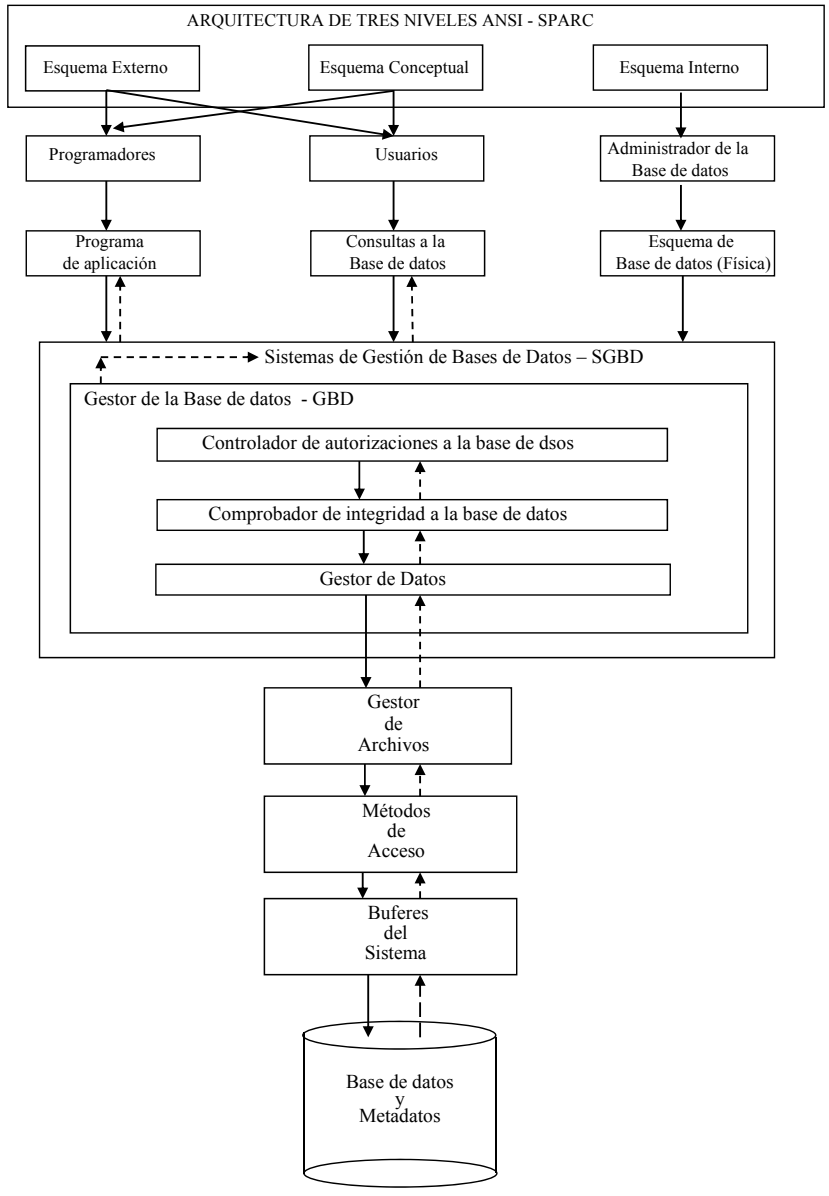


Figura 1.9 Flujo de datos en un SGBD

1.5.6 Usuarios de un sistema de bases de datos

El funcionamiento de un sistema de base de datos dentro de una organización demanda la participación de varias personas; personas que cumplen diferentes roles, entre los cuales, sin hacer una enumeración exhaustiva, se encuentran los de gestión, seguridad, diseño, programación, y utilización de la base de datos. Los roles mencionados pueden tener perfiles de conocimiento informático, o más cercanos al manejo de los lenguajes de programación que se utilizan para hacer el mantenimiento a la base de datos; o usuarios no informáticos, los que en su perfil son no conocedores de las herramientas de *hardware* y *software* y comunicaciones que se manejan en la fase de producción de un sistema de base de datos.

Con base en lo anterior, la tecnología de base de datos ha generado diferentes roles y, consecuentemente, diferentes perfiles a nivel de profesiones en el marco de la Ingeniería de Software, dentro de los cuales algunos de los más relevantes son:

- Administrador de datos (*Data Administrator*): Es la persona que gestiona los datos de la base de datos. La gestión de datos implica planificación, almacenamiento, proyección, mantenimiento, estándares y procedimientos en la operación de la base de datos.
- Propietario de la base de datos (*Owner Database*): Es la unidad organizacional o persona en la empresa dueña de la base de datos; lleva a cabo las acciones de configuración, construcción lógica, mantenimiento, permisos y borrado de la base de datos propietaria.
- Administrador de la base de datos (*Database Aministrator* (DBA): Responde por la administración física de la base de datos y las acciones de seguridad, integridad, mantenimiento y confiabilidad del sistema y de las aplicaciones de la base de datos.
- Diseñador de la base de datos (*Database Designer*): Es la persona o conjunto de personas de la organización (en la mayoría de los casos son ingenieros que interactúa con personas no informáticas con el fin de conocer los requerimientos de información de la organización) que se encargan del análisis sintáctico-semántico del contexto de operación de la base de datos, el diseño lógico/físico de base de datos, la identificación de las necesidades de información derivadas de la base de datos para la organización, en términos de infor-

mes, reportes, listados, o consultas no preestablecidas que se puedan realizar a la base de datos.

- Administrador de la Seguridad de la base de datos (*Security Manager Database*): Debido a las necesidades de seguridad de la información, sobre todo en las grandes organizaciones, se estila el rol de un ingeniero dedicado solo a las funciones de ciframiento, deciframiento y seguridad de la base de datos. Es el encargado por la organización, de común acuerdo con el DBA, de modificar los roles de membresía y los permisos de administración de las diferentes vistas de usuarios en la base de datos.
- Programadores de la base de datos (*Programmers Database*): Son los encargados de construir los programas de aplicación que interactúan con el SGDB a fin de dar los resultados de los requerimientos de información de la base de datos.
- Usuarios finales de la base de datos (*End-user of the Database*): Los usuarios que requieren servicios de información de la base de datos se suelen dividir en dos grupos: i) los inexpertos y los ii) los avanzados. i) los usuarios inexpertos son aquellos no conocedores de las acciones de diseño, programación y mantenimiento de la base de datos; desconocen por lo regular la existencia de un gestor de bases de datos en la organización, y se comunican con la base de datos a través de programas de aplicación que hacen los usuarios ubicados en el nivel por seguir. ii) los usuarios avanzados son aquellos cuyo nivel de conocimiento en base de datos les permite un control casi autónomo a través del SGBD; corresponde a este grupo aquellas personas que utilizan el lenguaje SQL para hacer consultas a la base datos; y dependiendo de su nivel de experiencia en programación en bases datos, pueden llegar a programar sus propias aplicaciones.

1.5.7 Desarrollo histórico de los sistemas de gestión de bases de datos

El desarrollo histórico de los Sistemas de Gestión de Bases de Datos (SGBD) en el marco de la Ingeniería de Software es paralelo al de los Sistemas de Información y los lenguajes de bases de datos en el marco de la Informática.

A continuación se presenta una síntesis histórica de los SGBD, sin pretender hacer un desarrollo histórico exhaustivo del nacimiento, desarrollo y evolución de los sis-

temas generadores de información apoyados por máquinas de procesamiento de datos.

La síntesis relacionada comprende:

1961: Diseño del almacén de datos integrados (en sus siglas en inglés Integrated Data Store): IDS, a cargo de Charles Bachmann, General Electric (USA).

1964: Creación del sistema jerárquico de archivos por la NAA (North American Aviation), identificado como GUAM (Generalized Update Access Method) dentro del proyecto Apolo de lanzamientos espaciales de la NASA [11].

1966: Primera base de datos Jerárquica IBM-NAA, identificada como Information Managemente System: IMS [11].

1967: Creación de normas para el modelo de bases de datos de red a través del grupo Database Task Group (DBTG) en la reunión Conference on Data Systems Languages (CODASYL).

1970: E. F. Ted Codd desarrolló los conceptos del modelo relacional en el documento de la ACM denominado "Un modelo relacional de datos para grandes bancos de datos compartidos".

1971: Informe de resultados del grupo de trabajo sobre bases de datos del DBTG de la Conference on Data Systems Languages.

1974: Donald Chamberlin en el laboratorio de San José de California (USA) en sus trabajos con la empresa IBM definió el lenguaje definido en inglés como Stuctured Query Language (SQL), el cual operacionaliza con sus especificaciones las características de las bases de datos que siguen el Modelo Relacional.

1974-1978: Desarrollo del prototipo de bases de datos relacional identificado como System R, liderado por Frank King de la IBM. Conversión del System R en SQL/DS y posteriormente en DB2 a cargo de IBM [11].

1975: Organización de la primera conferencia internacional sobre grandes bases de datos, identificada como Very Large Database Foundation (VLDF).

1976: Creación del Modelo Entidad-Relación (M E-R) de Peter Chen, e IBM desarrolló un protortipo de SGBD, el cual recibió el nombre de SEQUEL/2.

1977: Fin del desarrollo del Ingres DBMS a cargo de alumnos de la Universidad de California en Berkeley. Concretamente, en este mismo año, en Silicon Valley (USA), los ingenieros Larry Elison, Bob Miner y Ed Qates fundaron una empresa consultora identificada como Software Development Laboratories (SDL) [12].

1980: Difusión en el mercado de los sistemas de base de datos relacionales.

1982: Inicio del proyecto Ingres en la Universidad de Berkeley, liderado por Michael Stonebraker; en este año se data el inicio de PostgreSQL.

1984: Mark Hoffman y Bob Epstein fundaron Sybase en Berkeley, California (USA).

1985: Inicia el proyecto denominado Post-Ingres bajo la dirección de Michael Stonebraker, con la experiencia de Ingres, y se produce el gestor relacional SGBD con las características de ser orientado a objetos y de libre utilización.

1986: Estandarización del lenguaje SQL, lo cual produjo una mayor utilización de los SGBD relacionales.

1988: Sybase fue la primera empresa que mercadeó la tipología cliente/servidor para bases de datos relacionales [13].

1989: Sale el SQL Server 1.0 para el sistema operacional OS/2 de IBM.

1990: Aparición de los conceptos de Bases de Datos Distribuidas y Sistemas de Bases de Datos Orientados por Objetos (Informix y DB2).

1999: Microsoft lanza al mercado la versión de SQL Server 7.0, versión que tiene incorporada la herramienta OLAP (On line Analytical Processing); "solución que es utilizada en la llamada inteligencia empresarial o *Business Intelligence,* donde el objetivo es agilizar la consulta de grandes cantidades de datos" [14].

2000: Adaptación de los sistemas de bases de datos a sistemas de bases de datos multimedia, consolidación de las bases de datos orientadas por objetos, bases de datos bajo la Web.

2010: Consolidación de motores de bases de datos a través de las compañías que han investigado y creado sistemas de bases datos tales como Microsoft, IBM.

2013: Avance de las nuevas tendencias en bases de datos relacionales y las herramientas de minería de datos (data mining) basada en cubos OLAP.

2014: Bases de datos deductiva utilizando lenguajes de Programación Lógica [15].

2015: Análisis y diseño de sistemas de información orientados por objetos utilizando UML [16].

1.6 Ventajas de un sistema de gestión de bases de datos

El análisis del balance con relación a las ventajas y desventajas de los sistemas de bases de datos como soporte a la construcción de sistemas de información da como resultado una alta favorabilidad de decidir por la utilización de estos, y este hecho se debe a la alta productividad en el diseño de sistemas de informáticos en tiempos relativamente cortos; la alta capacidad de almacenamiento y gestión para el manejo de grandes volúmenes de datos propios de la tecnología de bases de datos en sus conceptos ampliados de bodegas de datos (*Data Warehouse*) y Minería de Datos (*Data Mining*).

Lo anterior se reafirma con el hecho de que lenguajes para el manejo de bases de datos, tales como el SQL, forman parte de la arquitectura de aplicación identificada como *System Application Architecture* (SAA) de la compañía IBM, e igualmente es un estándar de la *Federal Information Processing Standard*: FIPS [17], con la característica que como estándar lo hace obligatorio para las ventas de los gestores de bases de datos del Gobierno de Norteamérica. Adicionalmente, SQL se utiliza e influye en otros estándares: sin hacer una enumeración exhaustiva, entre ellos se encuentran los propuestos por la American National Standard en cuanto a las especificaciones del *software* para computadores pertinentes al diccionario para recursos e información, identificado por sus siglas en inglés como *Information Resource Dictionary Systems* (IRDS) de la *International Standard Organization:* ISO [18].

Con base en lo anterior, las principales ventajas [19] de la utilización de un SGBD de soporte al proceso de construcción y producción de un sistema de información son:

- *Utilización de estándares*: Los sistemas de información del mundo real manejan tanto grandes volúmenes de datos como altos niveles de complejidad creciente para representar los fenómenos cambiantes del mundo real. Ahora bien, las grandes oportunidades para la construcción de sistemas de información están en el interior de proyectos gubernamentales de los países (USA, Australia o Brasil) o uniones (Unión Europea). Se han relacionado en el análisis las organizaciones ISO, FIPS, ANSI-SPARC, entre otras; luego la utilización de estándares, a más de facilitar el proceso de mantenimiento de los sistemas de información, permite la utilización de normas en procesos de construcción, documentación, actualización, procedimientos, reglas de acceso al sistema de información; utilización que es lograble mediante los SGBD.
- *Aumento de la productividad*: El diseño y construcción de un sistema de información soportado en base de datos es más rápido en sus tiempos de desarrollo. Realmente, una vez realizado el Modelo Entidad-Relación (M E-R) del sistema (o diseño lógico), convertido el M E-R al Modelo Relacional (MR), el SGBD ofrece funciones estándar al programador para la creación de las tablas, sus relaciones y atributos representativos de la base de datos. Construidos los esquemas de las base de datos, y cargados los datos, ya en este punto el sistema entra en la fase de producción y mantenimiento. Luego, se debe tener en cuenta que el M E-R, el MR y el SGBD permiten disminuir los tiempos de diseño y programación de los aplicativos de la base de datos y, consecuentemente, aumentar la productividad.
- *Disminución de la redundancia de datos*: El modelamiento de la organización en su totalidad en bases de datos hace que en el M E-R se identifiquen los requerimientos de información de las unidades organizacionales de la empresa. Luego, se tiene un diseño de datos pertinente a cada una de las dependencias de la empresa (compras, inventarios, ventas, contabilidad, presupuestos, etc.), pero a su vez también pertinente a las necesidades de la empresa como un todo en sus flujos de información. Se aclara en este punto que la redundancia se disminuye en un sistema de base de datos *versus* un sistema de archivos no se elimina completamente; hay, de todos modos, duplicidad de elementos de datos en el modelado de las relaciones; pero dicha duplicidad se puede con-

trolar y es evidentemente menor que al formar islas de datos completamente duplicadas en la empresa.

– *Coherencia de datos*: Teniendo en cuenta la disminución de la redundancia de los datos, el SGBD en sus operaciones de adición, edición y cancelación de datos en la base de datos se realizan una sola vez, y efectuadas se reflejan para toda la empresa. Es de esperarse que la coherencia de datos implique la aplicación de políticas adecuadas y estándares en términos de las copias de la base de datos; pero, en síntesis, la coherencia de datos es indudablemente mayor en un sistema de base de datos que cuando se utiliza un sistema de archivos.

– *Producción de información genérica*: Los sistemas de bases de datos permiten generar una mayor cantidad de información a partir de los mismos datos almacenados. Luego, los informes, reportes, listados derivados de los datos son para la organización mayores tanto en número como en contenido. Se aclara en este punto que se tienen tres categorías de unidades de información: i) Los *informes*: son contenidos de información generados de la base de datos que representan la integración de los resultados de los flujos de información de varias unidades de la organización durante períodos mensuales, semestrales o anuales; y contienen los indicadores de gestión, decisión y prospectiva de la organización. Tal es el caso del balance contable de la empresa. ii) Los *reportes*: son piezas de información que representan el estado de operación de una dependencia de la empresa durante períodos cortos de tiempo; estas unidades representan las acciones de las operaciones diarias de las dependencias de la empresa y forman los indicadores de la gestión-operativa de la dependencia, que alimentan los índices de gestión de la empresa, como es el caso del reporte diario de personal. iii) Finalmente, los *listados*: son vaciados de informaciones de la base de datos filtrados con algunas características que generan grupos de datos en la mayoría de los casos ordenados por uno o varios atributos de las relaciones; informaciones que permiten construir los indicadores de operación de la empresa; el listado de paradas de máquina en una línea de producción refleja lo estable o no de la fase operativa del departamento de producción de una empresa. Se debe tener en cuenta que la construcción de los índices mencionados y sus análisis requieren en el accionar de la empresa la producción de informaciones de varios tipos en cualquier dependencia y a cualquier nivel de la empresa. Ello se logra a través de un sistema de bases de datos.

- *Globalización de datos organizacionales*: La dinámica del mundo real de las compañías hace que no sean funcionales los sistemas de almacenamiento y edición de datos propietarios de una dependencia de la empresa. La propiedad de los datos en base de datos no son de una o varias dependencias sino de la organización; lo anterior implica que toda la organización comparte los datos; lo que se llamaría una utilidad de datos globalizada. Este término implicaría una estructura global de datos integrados en la base de datos, pero una utilización localizada en los requerimientos de información de cada dependencia de la empresa. En tal sentido, los SGBD ofrecen comandos de autorización, definición, manipulación y control de datos que hacen más seguro utilizar la globalidad de los datos de la empresa y, consecuentemente, la posibilidad de compartir los datos contenidos en la base de datos.
- *Integridad de datos*: La integridad de datos se define como la fiabilidad y coherencia que deben tener los datos en sus contenidos sintácticos y semánticos. En tal sentido, no hay sueldos alfabéticos porque son numéricos, ni nombres numéricos porque son alfabéticos a nivel sintáctico. Igualmente, no hay sueldos negativos en un sistema de nómina, ni puede haber un stock de artículos negativos en un sistema de inventarios en un sentido semántico. Luego la coherencia de datos el DBA de la base de datos la asegura a través del SGBD con la aplicación de un conjunto de reglas que hacen que los contenidos de las base de datos no sean violados. El aseguramiento de la coherencia de los datos implica la confiabilidad de las informaciones generadas por la base de datos a través de los aplicativos de la base de datos.
- *Seguridad de datos*: La importancia de la utilización de los sistemas de bases de datos radica en el análisis de su seguridad, debido a que la seguridad de la base de datos es centralizada, mientras que la consulta de los datos es distribuida; es decir, permite el sistema de base de datos la distribución de los datos con el cumplimiento de condiciones de seguridad. La seguridad está centralizada en el Administrador de la Base de Datos (DBA); y es precisamente el SGBD el *software* que permite al DBA imponer las reglas de seguridad para las operaciones en la base de datos.
- *Menor costo*: Los costos de programación y mantenimiento de sistemas de información basados en archivos son altos porque tanto aplicaciones como datos son propiedad de dependencias específicas de la empresa; ello induce la asignación de presupuestos para cada dependencia al menos en los procesos de construcción de *software*. Los sistemas de información en base de datos,

al permitir una administración centralizada de los datos operacionales de la empresa y un desarrollo de aplicaciones que consulten una única unidad de datos, hacen que se disminuyan los costos tanto de la administración de los datos como de la construcción de programas. Luego, el presupuesto asignado sería el necesario para operar los computadores en red que acezarían al server que contenga la base de datos, porque los costes de gestión de datos y construcción de aplicaciones son compartidos por toda la empresa.

– *Accesibilidad de datos*: La producción de informaciones con base en un sistema de archivos necesariamente requiere de un programador que construya la aplicación que consulte los archivos y genere los resultados. Aplicación o programa de computadora que necesariamente es estático en su lógica de control; o lo que es equivalente, una vez que el programador ha construido el conjunto de instrucciones del programa que cumplen con los requerimientos de información del usuario, dicho programa necesariamente se debe cambiar en el caso de que cambien los requerimientos. En los sistemas de bases de datos, el acceso a los datos para las dependencias de la empresa es distribuido para cada dependencia pero global en términos de la consulta a toda la base datos; se aclara que se tienen las restricciones de las vistas de usuarios a la base de datos impuestas por el DBA. Luego, si se puede consultar la base de datos en su globalidad, aumenta el acceso de los datos para todas las dependencias. Esta accesibilidad se ve potencializada porque los SGBD ofrecen un conjunto de comandos que permiten la resolución de preguntas no preestablecidas o *ad hoc* contra la base de datos. Luego, por un lado, la dependencia se libera completamente de las consultas preestablecidas y estáticas; y pasa a un sistema de consultas a la base de datos que correlacionan con la dinámica de la empresa; incluso, los comandos que ofrecen los SGBD hacen que dichas consultas puedan ser solucionadas por usuarios sin tener un conocimiento experto de programación.

– *Mantenimiento simplificado de aplicaciones*: El mantenimiento de un sistema de archivos implica la creación de programas en los que la lógica del programa está integrada a la definición de los datos; la dependencia lógico/física en los programas de aplicación da como resultado que cualquier cambio en la definición de los datos implica necesariamente el cambio en los programas de aplicación. En los sistemas de bases de datos, el concepto anteriormente desarrollado de independencia de datos hace que cuando cambie la estructura de los datos en la base de datos no se tengan que cambiar los programas;

porque precisamente el SGBD cumple con la función de separar la descripción de los datos de las aplicaciones; luego las aplicaciones en este sentido no cambian cuando cambian la estructura de los datos. Por lo tanto, los tiempos y el esfuerzo para mantener las aplicaciones no solo se simplifican sino que disminuyen considerablemente.

- *Mayor capacidad de repuesta*: Con base en la aplicación de estándares en los sistemas de bases de datos, y teniendo en cuenta la coherencia, integración seguridad, y las posibilidades de compartir los datos, estando ellos almacenados de una forma centralizada, los sistemas en bases de datos ofrecen en la construcción de sistema de información una mayor capacidad de respuesta justificada porque el SGBD permite: i) La administración centralizada de los datos. ii) La simultaneidad en el acceso de los datos a través de diferentes usuarios. iii) La rápida absorción de la dinámica de la empresa u organización en términos de requerimientos de información, en el mantenimiento de los datos y la construcción de los programas de aplicación. iv) La escalabilidad de los datos en términos de pasar la base de datos a bodegas de datos o minerías de datos; bajo el concepto de repositorios de datos.
- *Acceso Concurrrente*: Los SGBD ofrecen un acceso concurrente a los datos (simultaneidad en el acceso), lo cual significa que dos o más usuarios tienen la posibilidad de accesar simultáneamente la base de datos, lo cual asegura las operaciones de gestión (adición, modificación y retiros de la base de datos) a la base de datos, lo cual garantiza tanto la coherencia como la integridad de los datos.
- *Facilidad en definición de requerimientos*: La construcción de un sistema de información en bases de datos necesariamente implica como uno de sus primeros pasos el análisis del contexto de operación del sistema de información con relación al mundo real para el cual la base de datos va a generar información. Luego, el diseño del modelo de la base de datos para la organización debe tener en cuenta de una forma integrada toda la organización; en tal sentido, la filosofía de base de datos es considerar la organización como un todo; el DBA a través del SGBD define, en conjunto con los creadores de la base de datos, las acciones tanto de diseño como de uso operativo de la base de datos en términos de prioridades, tiempos de respuestas y generación de requerimientos de información. Lo anterior da como resultado la atención del sistema de base de datos focalizado al cumplimiento de la misión crítica de la organización en términos de generación de información.

1.7 Desventajas y limitaciones de los sistemas de bases de datos

El análisis de las ventajas de la creación de sistemas de información en bases de datos hace pensar al lector que una vez solucionadas las dificultades de los sistemas de archivos con las bases de datos, estas aseguran la mejor inversión en costo, almacenamiento y funcionalidad del sistema informático. Sin embargo, las bases de datos tienen desventajas; algunas de las cuales, específicas de los SGBD, se presentan a continuación.

- *Costos de inversión en software*: Dependiendo del estado de desarrollo informático de la organización (sistemas soportados en archivos o empresa no sistematizada), la inversión en un SGBD del tipo monousuario no es tan alta; sin embargo, el costo de un SGBD del tipo multiusuario operando en computadores del tipo mainframe es costosa. A pesar de que a nivel académico hay SGBD libres, los autores no recomiendan este tipo de gestores, debido a que están en construcción, no han sido probados exhaustivamente y pueden generar fallas irrecuperables en los datos de la organización, con el problema adicional de que el servicio de mantenimiento, garantía y asesoría sobre el SGBD por parte de una empresa informática no existe, pues son de carácter libre.
- *Costos de inversión en hardware*: Los requerimientos de *hardware* tanto para el almacenamiento del *software* del SGBD como para el almacenamiento de la base de datos pueden ser altos; en la mayoría de los casos, la organización necesita aumentar los espacios de almacenamiento secundario; y si el SGBD requiere de una máquina dedicada, se requiere la inversión en *hardware*.
- *Costos de personal*: El diseño y operación de sistemas de información para bases de datos involucra un grupo de personas especializadas en informática. Se ha expuesto la importancia del Administrador de la Base de Datos (DBA), los diseñadores de la base de datos y los programadores de aplicaciones como mínimo para iniciar la construcción de este tipo de sistemas de información (a menos que una persona asuma los roles mencionados, lo cual no es fácil). Luego, la inversión en personal de ingenieros es uno de los rubros que la organización debe considerar si desea implantar sus sistemas en bases de datos.
- *Complejidad de los sistemas*: Es lógico pensar en el hecho de que si los sistemas de bases de datos pueden absorber la dinámica de cambios de la organización,

el sistema es de complejidad creciente en su dinámica, integración, consistencia, control y seguridad de datos. Luego, la complejidad disminuye asegurando tanto un SGBD apropiado para la organización como un recurso de talento humando de programadores en bases de datos, diseñadores de bases de datos administradores de datos, y usuarios que comprendan de una forma correcta la creación y operación de un sistema de bases de datos.

- *Impacto en la organización por fallos*: Teniendo en cuenta que los datos de la organización están centralizados en la base de datos, y todas las dependencias de la empresa se suplen de la misma estructura de los datos almacenados de la base de datos, entonces cualquier falla del SGBD impacta a la organización estructuralmente en términos de atención a las necesidades de información. La fallas por causas de *hardware, software* o humanas, si bien es cierto se presentan en todos los tipos de sistemas, se resalta el alto impacto alcanzado en la empresa al utilizar bases de datos, por la cobertura y la dificultad de los procesos de recuperación de fallas en este tipo de sistemas.
- *Costos de Actualización del Sistemas de Información*: El cambio de un sistema de información basado en archivo a un sistema de bases de datos implica en la mayoría de casos cambios en: i) *Hardware*, porque el sistema ya no está soportado por una estructura de archivos sino por un SGBD. ii) Conversión de aplicaciones manejadoras de archivos a programas que permitan manejar los datos de la base de datos a través del SGBD. iii) Cambios de personal, ya mencionado, porque el personal de ingenieros, analistas y programadores debe actualizarse a un grupo de personas conocedoras de los conceptos de base de datos. iv) Cambios en la estructura de procesamiento de la organización, por cuanto ya no se van a manejar islas de información con computadores autónomos en cada dependencia, sino necesariamente se tiene que tener una infraestructura de computadores en red, mediante la cual las dependencias de la empresa se comuniquen con el servidor donde está alojado el SGBD. v) Cambios en la estructura organizacional y de gestión de la empresa; este cambio es quizás uno de los aspectos más difíciles y de mayor impacto en la compañía, porque se entra a trabajar bajo la filosofía de acceso compartido de todas las dependencias a un conjunto de datos, donde la seguridad, integridad y coherencia de los datos no solo dependen del DBA sino también de los usuarios directos que afectan los datos fuentes.
- *Baja en la calidad de servicio*: En un sistema de archivos, los usuarios están acostumbrados a ser servidos por aplicaciones de propósito específico muy

especializadas; donde los datos fuentes de la dependencia propietaria de la información son manejados por los programas con un alto nivel de detalle; por otro lado, un sistema soportado en base de datos se caracteriza por generar informes, reportes o listados muy generales para las dependencia, en el entendido de que los datos para la organización están estructurado de una forma general y al servicio de todas las dependencia, sin altos niveles de detalle y con niveles no tan especializados en las lógicas de control de los programas de cada dependencia en cuanto al nivel de especialización de la información generada por el SGBD; lógicamente, se aclara en este punto que sistemas en bases de datos OLAP prestan un servicio con un buen grado de especialización en cuanto a los requerimientos de información de los usuarios.

1.8 Resumen

Un **archivo** es un conjunto de registros compuestos por un conjunto de campos que tienen valores de información, sobre los cuales se pueden hacer operaciones de adición, cancelación y edición de registros y campos con relación a la estructura de organización del archivo.

Las principales limitaciones de la técnica basada en archivos son: la duplicidad de datos, la dependencia lógico/física entre los programas y los datos, la incompatibilidad de formatos entre los archivos, la generación de islas de información en el sistema de archivos dentro de la organización, las consultas predestinadas o fijas del sistema, el excesivo aumento de programas de aplicación y finalmente lo estático de la gestión de los datos.

Una **base de datos** es una estructura integrada de datos que contiene físicamente la representación lógica de un conjunto de entidades y relaciones que representan el modelo de las necesidades de información de una organización. El diseño lógico de un sistema de bases de datos es la representación de las entidades y relaciones que modelan el sistema, y el diseño físico es la representación estructurada de los datos utilizando una tecnología de bases de datos.

Un **Sistema de Gestión de Bases de Datos** (SGBD) es un *software* que permite crear, editar y controlar el acceso de la base de datos. Las organizaciones ANSI-SPARC en sus estándares proponen una arquitectura en tres niveles de abstracción para una arquitectura de un SGBD. La función de un esquema en bases de datos es independizar

el gestor de la base de datos de los datos físicos almacenados en la base de datos. Los niveles mencionados son: i) El nivel externo o teoría: es la vista lógica de la base de datos que ven los usuarios. ii) El nivel conceptual o abstracción: es la descripción lógica de la base de datos a través del esquema. iii) El nivel interno o diseño: es el contenido de los datos en el computador.

La independencia lógica de los datos ocurre cuando los usuarios de la base de datos no se afectan por lo cambios conceptuales e internos de la base de datos; la independencia física se presenta cuando los cambios en el nivel físico de la base de datos no afectan los niveles conceptual y externos de la misma.

Los lenguajes que componen el SQL son tres: el lenguajes de definición de datos (*Data Definition Languaje*: DDL), el lenguaje de manipulación de datos (*Data Manipulation Languaje*: DML) y el lenguaje de control de datos (*Data Control Languaje*). Estos lenguajes dan capacidades al SGBD para realizar las operaciones en la base de datos.

Las principales ventajas de un SGBD son: i) La posibilidad de utilizar estándares en la creación del sistema. ii) El aumento de productividad en el diseño y la operación del sistema de información. iii) La disminución de la redundancia de datos. iv) La coherencia de los datos. v) La producción de información genérica para la organización. vi) La globalización de los datos organizacionales. vii) La integridad de los datos. viii) La seguridad de los datos. ix) El menor costo en el tratamiento de datos. x) La mejor accesibilidad a los datos. xi) El mantenimiento simplificado de las aplicaciones. xii) La mayor capacidad de respuesta del sistema. xiii) El acceso concurrente a los datos. xiv) La facilidad en la definición de los requerimientos de información del sistema.

Ejercicios

1. Una organización dedicada a la comercialización y distribución de gas natural requiere mantener actualizada la información de su sistema de quejas, peticiones y reclamos de sus clientes (PQR); para ello se requiere contar con la información del cliente, la factura, el inmueble, las quejas, los reclamos, las peticiones y, naturalmente, las resoluciones. Dentro de este contexto explique por qué en las organizaciones los conceptos de los sistemas de información basados en archivos son relevantes.
2. Elabore una tabla comparativa donde describa las desventajas y limitaciones de los sistemas de archivos que describen datos en las organizaciones.
3. En el contexto del sistema PQR identifique cuáles deben ser los datos del cliente, los datos estructurados de algún elemento de la factura, dos entidades principales de información, una instancia de una queja dentro del sistema de información.
4. Dentro del mismo contexto del sistema PQR cuál es la diferencia conceptual entre el diseño lógico y el diseño físico del sistema de información de base de datos.
5. Elabore un diagrama general de bloques donde se puedan visualizar los diferentes componentes de un entorno general de un Sistema de Gestión de Bases de Datos (SGBD) en el cual se visualice la independencia de los diferentes componentes.
6. Siguiendo con el contexto del sistema PQR, cuál sería la función de cada uno de los componentes del sistema gestor de base de datos en cuanto al uso del lenguaje de definición (DDL), manipulación (DML) y control de datos (DCL).
7. Si en el Sistema de Información PQR aparecen nuevos requerimientos, como por ejemplo, llevar la prestación del servicio técnico por parte de empleados de la empresa, lo cual puede conllevar a la modificación de la base de datos ¿cuáles serían las herramientas para abordar la actualización del sistema desde el componentes de base de datos?

8. ¿Qué tipo de usuarios internos y externos se pueden identificar dentro del contexto del sistema PQR anteriormente descrito, incluso con la modificación del punto 7, asociados con el sistema de bases de datos?
9. Utilizando el anterior contexto del sistema PQR elabore una tabla comparativa donde se visualicen las ventajas, desventajas y limitaciones de un SGBD.
10. Consulte en la web tres productos diferentes del mercado que permitan construir el SGBD para el sistema PQR en mención, describa sus características tecnológicas, el tipo de licenciamiento y los requerimientos tecnológicos para su instalación, configuración y administración.

Capítulo 2

Diseño de bases de datos

2.1 Introducción

Los sistemas de información presente y futuros tienden a manejar datos en los dominios del conocimiento humano del tacto (o la escritura), la visión (o la imagen), el olfato (o el olor), el gusto (o el sabor), entre otros. Los dominios en los cuales se maneja los datos tienen dos problemas estructurales: el primero, el aumento en complejidad creciente del número de datos que en tiempo se deben almacenar en una base de datos; y el segundo, los datos en sus componentes sintácticas (números o caracteres, entre otros) y semánticas (relativo al significado) no están estructurados. Para el primer problema, las redes sociales (Facebook, Twitter, Linkedin, entre otras) manejaron rangos estimados entre [250.000,00 ; 750.000.000] de visitas en un mes [20] en 2013, lo cual significa que el almacenamiento del *tracking* permanente y sin pérdidas de información de una cuenta en una red social para una empresa de seguridad estatal no es sencillo. Para el segundo problema, casos como el censado ubicuo implica el almacenamiento y consultas de datos heterogéneos, tanto en su sintaxis como en su significado.

Con base en lo anterior, y teniendo como soporte los conceptos de los sistemas de bases de datos, sus gestores de datos o Sistemas de Gestión de Bases de datos (SGBD) y los estándares de bases de datos, se hace necesario estructurar una metodología que a través de un conjunto de pasos asegure el éxito del diseño de una base de datos desde su concepción hasta su creación y operación al servir al usuario.

Este capítulo expone un método de diseño de bases de datos cuyo fundamento es captar la semántica del sistema de información que se va a modelar en bases de datos. Reconociendo que existen varios modelos para la realización del diseño lógico de una base de datos, el texto optó por la utilización del Modelo Entidad-Relación en la fase de diseño a fin de capturar la semántica del sistema de información que se va

a construir; semántica que se complementa con el análisis del contexto de operación en el cual va a funcionar el sistema de información operando en bases de datos. El método expuesto es útil tanto para el diseño de una base de datos dentro de una empresa que no tenga sus sistemas de información estructurados en bases de datos como para una organización que pretenda rediseñar su sistema de bases de datos.

Finalmente, el capítulo cierra con la presentación de un conjunto de casos en los que se muestran los diseños de bases de datos a partir de especificaciones de requerimientos, en el entendido de que el lector adquiera experiencia inicial en la estimación del costo y el tiempo de la estructuración de un sistema de bases de datos conocido; pero cuando el sistema que se va a diseñar en bases de datos no es conocido y es muy improbable estimar con certeza las variables de tiempo y costo del sistemas, el análisis del contexto disminuye la incertidumbre porque selecciona la información pertinente del sistema por construir.

2.2 Objetivos

Teniendo en cuenta lo expuesto en el primer capítulo, este capítulo permitirá al lector aprender:

- Las etapas de diseño de un sistema de información fundamentado en bases de datos.
- El diseño conceptual de un sistema de base datos.
- Los constructos básicos del Modelo Entidad-Relación (M E-R).
- El análisis de contexto de un sistema de información en bases de datos.
- La identificación de las componentes esenciales de un sistema de bases de datos.
- El diseño físicos de la base de datos.
- El mantenimiento de la base de datos.
- El diseño de base de datos bajo especificaciones de requerimientos.

2.3 Síntesis conceptual

La construcción de un sistema de información en bases de datos implica tanto la apropiación de los conceptos de bases de datos como el análisis del contexto y los requerimientos de información que necesita el usuario del sistema. El éxito de un sistema en base de datos depende necesariamente de las etapas de diseño de un sistema de información. Las etapas del diseño del sistema deben tener en cuenta una planificación del desarrollo del sistema de información. Con base en la planificación, el diseño del sistema necesita del análisis del contexto donde va a operar el sistema, lo cual necesariamente implica la apropiación conceptual de los modelos diseño de bases de datos; el texto para la fase de diseño tiene en cuenta los constructos del Modelo Entidad-Relación para la definición de las entidades, relaciones y atributos de la fase de diseño de la base de datos, pero a través del análisis del contexto complementa la semántica del sistema que debe contener una base de datos. Una vez realizado el diseño conceptual, se introduce el desarrollo del diseño físico de la base de datos como base para el aprendizaje continuado del manejo de datos a través de un gestor de bases de datos.

2.4 Etapas de diseño de un sistema de bases de datos

La creación de una base de datos de soporte a la operación de un sistema informático está estructurada en los siguientes pasos metodológicos:

- Planificación del desarrollo del sistema informático.
- Diseño Conceptual (Lógico) de la base de datos.
  i) Analizar del contexto del sistema en bases de datos.
  ii) Identificar las entidades del sistema.
  iii) Asociar valores semánticos a las componentes del contexto.
  iv) Agrupar las componentes del contexto.
  v) Diseñar los atributos de las Entidades y Relaciones.
  vi) Seleccionar las claves primarias y foráneas de las entidades/relaciones de la base de datos.

- Diseño Físico de la base de datos.
- Carga de los datos a la base de datos.
- Operación de la base de datos.
- Mantenimiento de la base de datos.

Los pasos mencionados están enmarcados en el ciclo de vida de desarrollo del *software*, el cual comprende las fases de planificar, analizar, diseñar, implementar, operar y mantener un sistema informático, las cuales, sin ser de carácter exhaustivo —por cuanto la fase de diseño se suele interpretar en las subfases de diseño lógico-conceptual y físico del sistema—, tienen en cada una de sus etapas la estructura de ser un sistema retroalimentado de ciclo cerrado. En el entendido de que en cada una de las fases el constructor del sistema debe fundamentarse en las fases anteriores, es decir que una vez implementada la base de datos, en el caso de no ser útil, debe necesariamente volver a las fases predecesoras, y en este sentido analizar el diseño físico y el diseño lógico-conceptual de soporte a los esquemas de la base de datos, o inclusive regresar a la parte inicial de planificación de la construcción del sistema informático.

2.5 Planificación del desarrollo del sistema informático en bases de datos

La planificación del sistema informático soportado en base de datos implica el análisis y evaluación de las alternativas del *hardware, software, comware* (comunicaciones) y *humanware* (o recursos humanos) que se encargarán de la construcción del sistema. Conlleva concretamente: i) Definir el tipo de sistema informático (centralizado, distribuido, bajo Web, inalámbrico, etc.). ii) Identificar la plataforma de *hardware* que soportará la operación del sistema en base de datos. iii) Identificar el gestor de base de datos que se va a utilizar, los lenguajes de programación que han de interactuar con el gestor de la base de datos para lograr la operación del sistema. iv) La identificación de los recursos de comunicaciones necesarios para la operación del sistema, y v) La identificación del recurso de talento humano encargado de la construcción del sistema.

La planificación del análisis, desarrollo y construcción de un sistema de información en bases de datos requiere por parte de la empresa que lo va a implementar, el conocimiento de las ventajas y posibles desventajas en términos de recursos computacionales y de personal preparado en bases de datos para asegurar el éxito del siste-

ma; bondades y obstáculos presentados en el primer capítulo sin hacer un análisis exhaustivo.

Una vez planificado el desarrollo de sistemas de información en su tiempo de ejecución y costo de inversión, el paso siguiente es construir el diseño conceptual de la base de datos; para lo cual se analizarán inicialmente los modelos que apoyan el diseño lógico-conceptual de la base de datos.

2.6 Diseño conceptual (lógico) de la base de datos

El diseño lógico-conceptual de una base de datos corresponde a la estructura de la base de datos. Esta estructura está determinada por un modelo de base de datos. Luego, el modelo de base de datos es un tipo de modelo de datos que determina, en últimas, la estructura lógica de una base de datos. Esta es la forma como los datos están organizados y, consecuentemente, el modelo de datos determina la forma como: i) Los datos son almacenados en la base de datos. ii) La estructura de organización de los datos, en términos de sus estructuras de datos de soporte. iii) La forma como los datos pueden ser gestionados o manipulados a través del SGBD. iv) Implícitamente, el modelo en mención también determina las reglas o políticas de seguridad de los datos suministradas por el gestor de la base de datos.

Existen varios modelos que apoyan el diseño lógico de una base de datos. Entre estos modelos se encuentran:

- *Modelo de bases de datos jerárquico*: soportado por la teoría de árboles, en el que los datos son estructurados en una organización de orden familiar: Padre, Hijo, Nieto [21], [8], [6].
- *Modelo de base de datos en red o tipo plex*: en este modelo, los datos son organizados en función de un conjunto nodos y arcos que conectan los nodos mencionados, dentro de los cuales se encuentran las unidades de datos primarias, formando de esta manera una estructura de grafos [8], [22].
- *Modelo Entidad-Relación (*M E-R*)*: en este modelo, los datos son modelados a través de un conjunto de entidades, sus atributos y relaciones, obtenidos con base en un análisis de contexto [23], [24], [25].
- *Modelo orientado por objetos*: modelo en el que las unidades de información en la base de datos son representadas en forma de objetos, y el sistema de ges-

tión de la base de datos utiliza la programación orientada por objetos [24], [6], [26].

- *Modelo de datos documental*: es un modelo para la organización de información documental no estructurada o semiestructurada; se le conoce como sistemas de bases de datos orientados a la documentación [27].
- *Modelo del tipo entidad-atributo-valor*: este modelo conserva uno de los núcleos principales respecto al modelo entidad relación, el concepto de entidad; con la característica de que para este caso la cantidad de atributos en sus especificidades de identidad del atributo, definición de atributo (numérico o no numérico, entre otros), sus parámetros y valores son potencialmente grandes en su número; pero el número de atributos que participan en un proceso de gestión de la información para una entidad dada no es tan grande.
- *Modelo de esquema en estrella*: es el conjunto de una o más tablas de hechos que contienen informaciones relativas a las medidas, métricas o datos representativos de las acciones de un proceso de negocios en la cuales se toman se toman decisiones inteligentes con fundamento en el esquema [28].

Con base en los diferentes tipos de modelos que soportan el diseño lógico de la base se profundizará en el Modelo Entidad-Relación, por su importancia y utilización en el diseño lógico-conceptual de sistemas de bases de datos; y también por la facilidad para transformar el diseño de la base de datos del M E-R al Modelo Relacional.

2.6.1 Diseño conceptual de la base de datos a través del modelo Entidad-Relación (M E-R)

Teniendo en cuenta los niveles de la arquitectura ANSI-SPARC, el nivel externo representa la parte lógica del sistema de base de datos en sus vistas a los usuarios finales de la base de datos, sean estos con perfiles de conocimiento informático o no informático. Por su parte, el segundo nivel de la arquitectura mencionada a nivel conceptual se puede interpretar como un puente de comunicación entre el nivel externo (la lógica del sistema) y el nivel interno (la física del sistema).

Luego para el diseño del nivel conceptual de la base de datos se aplicará el Modelo Entidad-Relación; conceptualización que es aplicable a cualquier tipo de base de datos, sea esta de naturaleza del tipo jerárquica, red o relacional, etc. Lo anterior justificado por el hecho de que en la etapa de diseño conceptual de la base de datos aún no se tiene definido concretamente el gestor de la base de datos que se va a utilizar.

La justificación de utilizar el M E-R en el diseño conceptual de base de datos es la siguiente:

- Este modelo interactúa de una forma lógica con la primera etapa del diseño de la base de datos, la de análisis de contexto de la base de datos.
- El M E-R es simple y legible en la conceptualización de bases de datos, por su facilidad para representar las componentes del modelo a través de sencillos gráficos entendibles al usuario.
- Por su simplicidad y legibilidad, este modelo es fácil de aprender y utilizar rápidamente por los diseñadores de la base de datos, con el valor agregado adicional en productividad que se emplea en el diseño de la mayoría de las aplicaciones.
- Existen en el mercado herramientas informáticas tanto públicas como propietarias que facilitan la utilización del M E-R, como es el caso del producto DIA *(Diagram Editor)*, que es libre, o las herramientas CASE, que utilizan variantes del M E-R para hacer el diseño conceptual del modelado de datos.
- La notación gráfica del modelado de datos utilizando el M E-R es base para el modelamiento orientado por objetos, identificado con UML por sus siglas en inglés *(Unified Modeling Language)* de Booch [29].
- El M E-R permite reflejar en el diseño tanto la estructura de los datos como su integridad; como este modelo no refleja la manipulación de los datos al convertir el diseño conceptual diagramático al Modelo Relacional, esto permite una mejor aproximación a los procesos de manipulación de los datos en cuanto a su manipulación.

2.6.1.1 Constructos básicos del modelo Entidad-Relación según la notación de P. Chen

El modelado conceptual de una base de datos a través del Modelo Entidad-Relación tiene tres constructos principales:

- Las entidades.
- Los atributos de las entidades.
- Las relaciones.

Una **entidad** es un objeto identificable y diferenciable de los demás de su especie; en tal sentido se catalogan como objetos: organizaciones, personas, conceptos, categorías, lugares o situaciones, los cuales se identifican en el contexto de un sistema de información como una entidad tipo o un tipo de entidad. Formalmente, el tipo de entidad es *Un grupo de objetos con las mismas propiedades que la empresa identifica como poseedores de una existencia independiente* [6]. Tipos de entidades que se pueden instanciar, siendo la instancia de una entidad *Un objeto identificable de forma unívoca dentro de un tipo de entidad* [6]. Se identifican como entidades: un alumno en un Sistema de Información Académico; un artículo en un Sistema de Inventarios; una cuenta en un Sistema Bancario.

Es de la mayor importancia resaltar que el diseño de entidades de un sistema de información en el marco del M E-R no se hace de forma intuitiva, sino que requiere de una metodología formal que asegure la correcta identificación y selección de las entidades que cumplen con responder a los requerimientos de información del usuario; metodología que será desarrollada en este capítulo.

Los **atributos de una entidad** son las propiedades sintáctico-semánticas que como datos específicos y pertinentes a la entidad la identifican y definen como elementos que son almacenados en la base de datos. Las propiedades enunciadas son base para los sistemas basados en conocimiento [30]. Con relación a los sistemas antes mencionados, los atributos propios de las entidades ejemplificadas se muestran en la tabla 2.1.

Tabla 2.1 Definición de atributos de entidades de Sistemas de Información

Sistema de Información Académicos		Sistema de Inventarios		Sistema Bancario	
Entidad	**Atributos**	**Entidad**	**Atributos**	**Entidad**	**Atributos**
ALUMNO	Código	ARTÍCULO	Referencia	CUENTA	Número de cuenta
	Nombre		Nombre		Propietario
	Dirección		Valor de compra		Saldo anterior
	Teléfono		Proveedor		Movimiento
	E_mail		Stock		Saldo actual
	Sexo				
	Promedio				

Por su parte, una **relación** puede ser definida como el grado de asociatividad entre una o varias entidades que se pueden instanciar como objetos dentro de una base de datos, y son significativas sintáctica y semánticamente para el contexto de operación del sistema que soporta la base de datos. Conceptualmente, la instancia de una relación se define como *Una asociación identificable de forma unívoca que incluye una instancia de cada uno de los tipos de entidad participantes* [6].

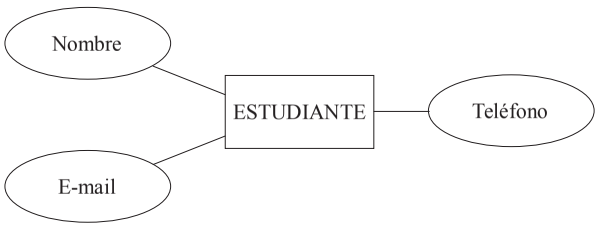
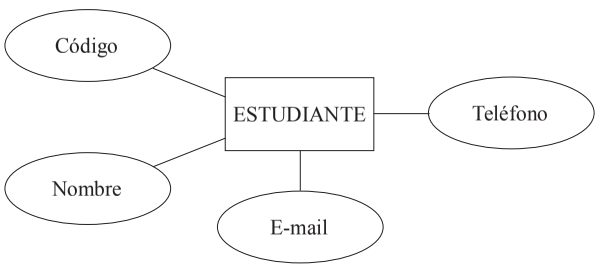
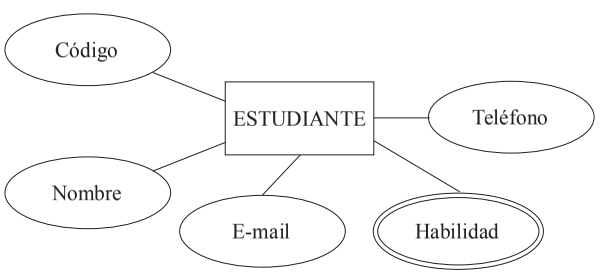
La utilización del M E-R en el diseño de bases de datos requiere de una simbología gráfica, la cual es base para el diseñador en sus símbolos de apoyo para capturar el modelamiento del sistema de información a través del modelo de bases de datos mencionados. Es de la mayor importancia aclarar que existen simbologías o notaciones alternas del M E-R; entre ellas están la notación de P. Chen o la de pie de cuervo, entre otras.

El modelado gráfico Entidad-Relación según la notación de Peter Chen se presenta en la tabla 2.2. La simbología del Modelo Entidad-Relación, de acuerdo con pie de cuervo, se presenta en la tabla 2.3.

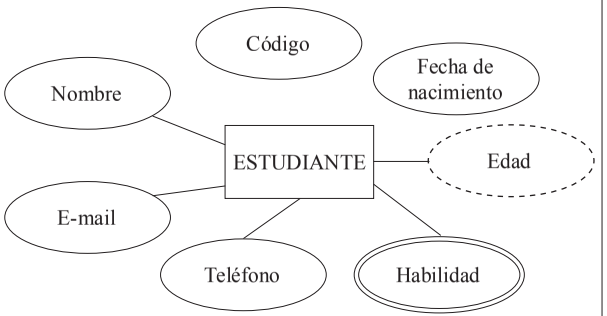
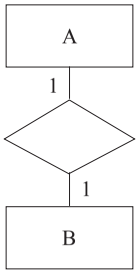
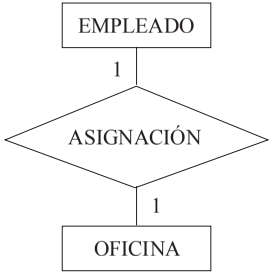
Tabla 2.2 Simbología del modelo Entidad-Relación, de acuerdo con la notación de P. Chen

Símbolo	Significado	Concepto	Ejemplo	Simbología
Nombre de la Entidad	Entidad Fuerte	Tienen atributos que las identifican completamente a través de sus claves primarias.	ESTUDIANTE es una entidad en un Sistema Académico.	ESTUDIANTE
Nombre de la Entidad Débil	Entidad Débil	Aquella cuyos atributos no la identifican completamente.	La UBICACIÓN de la entidad OFICINA depende de la entidad EDIFICIO para su ubicación en un Sistema de Planta Física de una Universidad.	EDIFICIO — UBICA — OFICINA
	Relación	Asocia una o varias entidades.	La entidad ESTUDIANTE está relacionada con la entidad ASIGNATURA en su matrícula en un Sistema Académico.	ESTUDIANTE — MATRICULA — ASIGNATURA
	Relación asociada con una entidad débil.		La UBICACIÓN de la entidad SALÓN depende de la entidad EDIFICIO para su ubicación en un Sistema de Planta Física de una Universidad.	EDIFICIO — UBICA — SALON
Relación Rol Rol Nombre de la Entidad	Relación Recursiva en la cual se indica el nombre del rol de la entidad en la relación.	Es una relación en la que alguna entidad está asociada más de una vez.	En la relación TRIPULACIÓN, la entidad PERSONA se asocia más de una vez con los roles Piloto y Azafata.	PILOTO TRIPULACIÓN — PERSONA AZAFATA

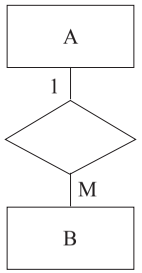
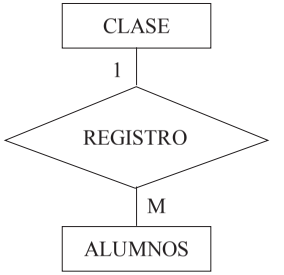
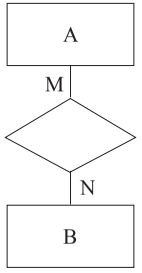
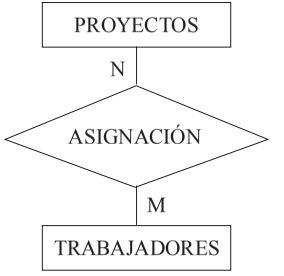
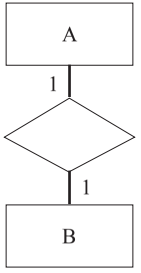
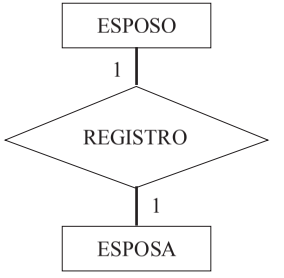
Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
Nombre atributo	Atributo	Se utiliza para identificar un atributo de una entidad.	La entidad ESTUDIANTE en un Sistema Académico tiene como atributos: (Nombre, E_mail,Teléfono)	
Nombre atributo	Atributo de clave principal	Se utiliza para identificar la clave principal de una entidad.	La entidad ESTUDIANTE en un Sistema Académico tiene como atributos: (Código,Nombre, E_ mail,Teléfono) y se identifica completamente con la llave primaria Código en un Sistema Académico.	
Nombre atributo	Atributo multivaluado	Es aquel que tiene más de una ocurrencia para un determinado valor de una clave.	A la entidad ESTUDIANTE en un Sistemas Académico se le puede agregar el atributo “habilidad”. (Código,Nombre, E_ mail,Teléfono, habilidad), Donde “habilidad” puede tomar los valores de lectura, de escritura, de razonamiento abstracto.	

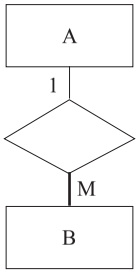
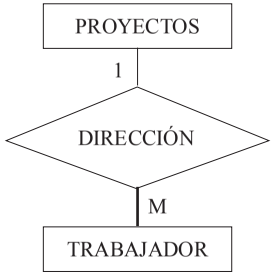
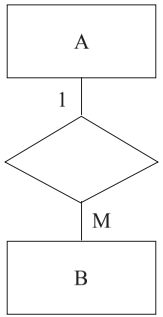
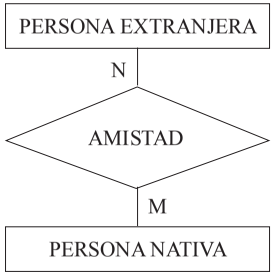
Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
Nombre atributo	Atributo derivado	Es aquel atributo que se puede derivar de otros atributos o entidades relacionadas.	A la entidad ESTUDIANTE en un Sistema Académico, con los atributos mencionados en el ejemplo anterior es posible agregarle los atributos de fecha_nacimiento y edad; donde "edad" se deriva de la fecha_actual – fecha_ nacimiento. (Código,Nombre, E_ mail,Teléfono, habilidad, fecha_nacimiento), Donde "habilidad" puede tomar los valores de lectura, de escritura, de razonamiento abstracto.	
	Relación uno a uno (1:1)	Es la relación que existe entre dos entidades, A y B, cuyo número de ocurrencias es solo una al estar relacionadas entre sí.	En un sistema de asignación de planta física de una organización hay una relación entre el empleado y la oficina, la cual se asigna al trabajador para la atención al usuario. Luego, 1 empleado puede atender en 1 y solo una oficina.	

Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
	Relación de uno a muchos (1:M)	Es la relación que existe entre la entidad A y la entidad B, con la característica de que en la A hay solo una (1) ocurrencia y en la B hay varias (M) ocurrencias.	En un Sistema de Registro Académico, una (1) clase tiene varios (M) alumnos. Luego, la relación de registro a la clase es de 1 a varios, o de 1 a M.	
	Relación de muchos a muchos (M:N)	Relación existente entre dos entidades, A y B, donde el número de ocurrencias de ambas pueden ser muchos.	En una organización que trabaje por proyectos existen N proyectos en curso, los cuales son atendidos por M trabajadores. Por lo tanto, la relación de asignación de proyectos a trabajadores es N:M, y de trabajadores a proyectos es M:N.	
	Relación (1:M) con participación obligatoria de ambas entidades	Es la relación que existe entre la entidad A y la entidad B, con la característica de que para una ocurrencia de la entidad A necesariamente debe existir una ocurrencia de la entidad B.	En un sistema de registros matrimoniales entre parejas de diferente sexo necesariamente debe existir el rol de esposo (hombre), diferente del rol de esposa (mujer). Luego, en este caso es una relación (1:1), en la cual necesariamente debe existir una ocurrencia de las dos entidades.	

Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
	Relación (1:M) con participación opcional para la entidad A	Se define como la relación entre las entidades A y B, en la cual la participación de una de ellas es opcional y la de la otra es obligatoria.	En una organización que trabaje por proyectos, si existe 1 proyecto en curso, el cual es atendido por M trabajadores, si se analiza la relación dirección de los proyectos, entonces la entidad "trabajador" es obligatoria, mientras que la entidad "proyecto" es opcional. Luego, no se puede tener un proyecto sin director; pero la entidad "proyecto" es opcional en la relación "dirección".	
	Relación (1:M) con participación opcional de ambas entidades	Es la relación entre las entidades A y B, en la cual es opcional la participación de ambas entidades.	En el análisis de la relación amistad entre amigos en las redes sociales no todas las personas de un país extranjero son amigas de todas las personas de un país nativo. Luego, algunas personas extrajeras son amigas de algunas personas nativas; luego se tiene la opcionalidad de la participación de las dos entidades.	

Continúa...

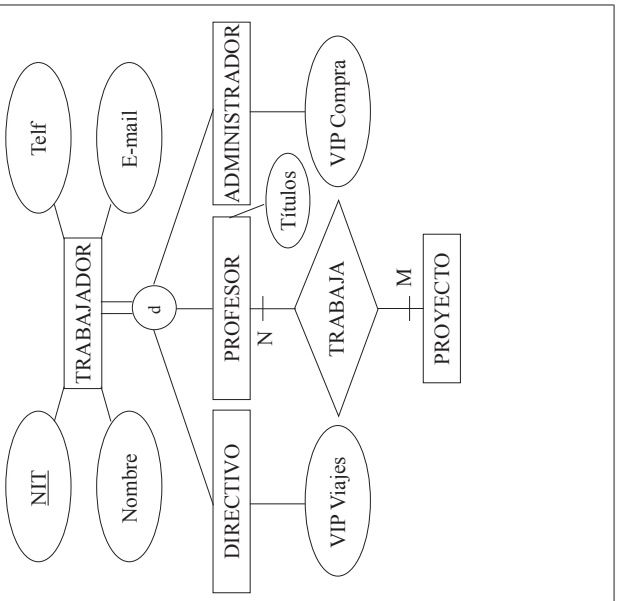
Símbolo	Significado	Concepto	Ejemplo	Simbología
	Generalización / Especialización (G/E)	La G/E permite modelar una entidad Superclase, la cual se puede especializar en subclases. Superclase: modela lo genérico o los atributos comunes de la entidad. Subclase: modela las características propias o especializadas de la entidad. d: define una relación disyunta. o: define una relación no disyunta o total. La doble línea es una participación obligatoria, mientras que una línea simple es una participación opcional.	Suponga un Sistema Universitario de trabajadores, dividido en tres áreas organizacionales: Directivos, Profesores y Administradores. La superclase TRABAJADOR tiene los atributos: Nit, Nombre, Tel., E_mail. Las subclases son Directivos, Profesores y Administradores. El directivo tiene asignada una tarjeta de viajes VIP. El profesor debe tener títulos de especialización, maestría y doctorado; y en la relación trabaja tiene que trabajar simultáneamente en varios proyectos académicos, de investigación y de consultaría. El administrativo tiene asignada una tarjeta VIP pero para compra.	

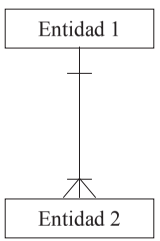
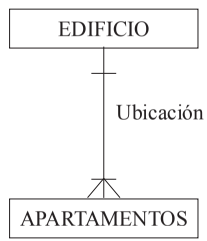
Tabla 2.3 Simbología del modelo Entidad-Relación de acuerdo con pie de cuervo

Símbolo	Significado	Concepto	Ejemplo	Simbología
Nombre de la Entidad	Entidad	Es un objeto singular identificable y diferenciable que pertenece a un sistema de información.	SUCURSAL es una entidad identificable y diferenciable en un Sistema Bancario.	SUCURSAL
Nombre de la relación	Relación	Es el grado de asociación entre un conjunto de entidades.	A una (1) SUCURSAL pertenecen varios (n) clientes en un Sistema Bancario.	SUCURSAL — Pertenecen — CLIENTES
Rol Rol Nombre de la Entidad	Relación Recursiva	Es una relación en la que alguna entidad se asocia más de una vez. En la relación se especifica el nombre del rol, con el fin de identificar los roles que participan en la entidad a través de la relación.	En la relación MATRIMONIO, la entidad PERSONA tiene dos roles: el ESPOSO y la ESPOSA.	Matrimonio; Esposo; Esposa; PERSONA

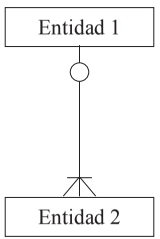
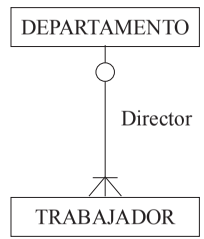
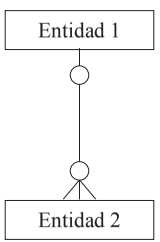
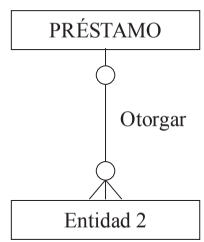
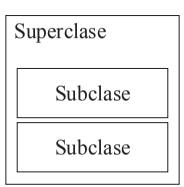
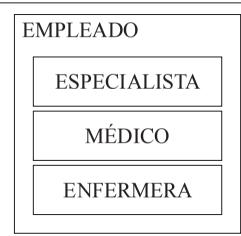
Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
Nombre de la Entidad Nombre atributos Atributo 1 Atributo 2 Atributo 3 … Atributo n	Atributos	Son las características sintácticas y semánticas que identifican a una entidad. Los atributos se denotan a nivel numerado en la sección inferior del símbolo de la entidad. La clave primaria se subraya, y si los atributos son multivaluados, se incluyen entre llaves { }.	La entidad CUENTA en un sistema contable tiene como atributos: (Código_Cuenta, Saldo_Anterior, Movimiento, y Saldo_Actual).	CUENTA Código Cuenta Saldo_Anterior Movimiento Saldo_Actuar
———	Relación uno a uno	Relación en la cual el grado de interrelación entre las entidades en su cardinalidad es 1 a 1.	En la relación de identidad de las entidades PERSONA y PASAPORTE, una persona es propietaria de un pasaporte, mientras que un pasaporte pertenece a una persona.	PERSONA — Identidad — PASAPORTE
——<	Relación de uno a muchos	Es una relación en que el grado de cardinalidad entre las dos relaciones es de uno a muchos. La relación del lado de muchos se presenta con varios bifurques, y de ahí el nombre de notación pie de cuervo.	En una empresa de producción cuyo modelo trabaje por PROYECTOS, la relación de ejecución de un PROYECTO está a cargo de varios EMPLEADOS.	PROYECTO — Ejecución —< EMPLEADOS

Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
	Relación de varios a varios	Es aquella relación en la que una entidad está interrelacionada con varios elementos de una segunda entidad; pero, a su vez, los varios elementos de la segunda entidad están interrelacionados con varios elementos de la primera entidad.	En la relación Asesoría en un sistema universitario de seguimiento de estudiantes varios profesores atienden a varios estudiantes, y a su vez, varios estudiantes son atendidos por varios profesores.	
	Relación de uno a muchos con participación obligatoria de ambas entidades	Es una relación entre dos entidades con cardinalidad varios, en la que necesariamente ambas entidades deben estar presentes en la relación.	Es un sistema de organización urbana de condominios compuesto por varios edificios, un (1) EDIFICIO como entidad de un condominio tiene varios (n) APARTAMENTOS en la relación de ubicación de los apartamentos que pertenecen a los edificios del condominio. Luego, si no existe el edificio no hay apartamentos; pero, consecuentemente, los apartamentos necesariamente deben pertenecer a un edificio.	

Continúa...

Símbolo	Significado	Concepto	Ejemplo	Simbología
	Relación uno a muchos, en la que la primera entidad es opcional y la segunda es obligatoria.	Es una relación en la que la primera entidad puede existir si como mínimo existe una segunda entidad. En este caso se dice que esta segunda entidad es obligatoria, mientras que la primera es opcional.	En un sistema de producción departamentalizado existe la entidad DEPARTAMENTO y a cada uno de los departamentos está adscrito un grupo de entidades que son los TRABAJADORES. En la relación Dirección no hay un departamento sin DIRECTOR; luego la entidad DEPARTAMENTO es opcional.	
	Relación de uno a muchos con participación opcional de ambas entidades	Relación en la que tanto la primera como la segunda entidad son opcionales, en la relación uno referida a la primera y muchos con relación a la segunda.	En un Sistema Universitario con existencia de una cooperativa que ofrece préstamos a sus empleados, en la relación OTORGAR un PRÉSTAMO, a los EMPLEADOS que pueden haber solicitado un préstamo; pero no todos los empleados tienen préstamo. Luego ambas entidades son opcionales en la relación 1 a n.	
	Generalización/ Especialización	Es la representación encapsulada , para denotar la generalización/ especialización en una base de datos.	Un sistema hospitalario tiene la entidad EMPLEADO como superclase, y los empleados están categorizados en subclases Especialistas, Médicos y Enfermeras.	

Una vez desarrollada la simbología del Modelo Entidad-Relación según la notación de P. Chen y la notación en pie de cuervo, el paso siguiente es realizar el desarrollo del análisis del contexto de operación del sistema dentro de la metodología desarrollada.

2.6.2 Análisis del contexto de operación de la base de datos, soporte al sistema de información

Los sistemas de información son bases para la toma de decisiones en los contextos sociales, políticos, económicos y ambientales de una organización o un país. La importancia de analizar el contexto de operación de un sistema de información radica en el hecho de que los sistemas de soporte a la toma de decisiones (DSS: *Decision Support Systems*) no solamente se basan en los contenidos de datos sintácticos, sino que se fundamentan también en los contenidos de datos semánticos.

Luego, es lógico pensar en las diferencias estructurales entre un Sistema Integrado de Manufactura (SIM), un Sistema Académico Universitario (SAU) o un Sistema Informático Bancario (SIB), entre otros. La maquinaria es a la línea de producción en el SIM como el profesor es a la clase en el SAU, o como el cliente es a la línea de atención en el SIB. Sin embargo, la máquina se puede rotular por cuanto pertenece a un sistema de inventarios en el proceso de manufactura; pero el docente o cliente en los sistemas SAU y SIB no se pueden rotular con una simple identificación sintáctica (entiéndase, asignación de un código: NIT, Cédula de Ciudadanía (C. de C.), Cédula de Extranjería (C. de E.) o Pasaporte), por cuanto hay características tanto en el profesor como en el cliente que no se pueden representar con un simple dato sintáctico.

Con base en lo anterior, el análisis del contexto exige la captura de las características tanto sintácticas como semánticas de los datos en su estructuración para ser definidos en los esquemas de la base de datos. Luego, es de la mayor importancia identificar, con base en el contexto, las entidades y los atributos de la base de datos[1]. Lo anterior aparentemente es sencillo, se hace de una forma intuitiva; pero la intuición no siempre asegura la calidad de las informaciones en sus procesos y salidas derivadas de la operación de un sistema informático. En un SIB son claramente identificables tan-

[1] La conceptualización formal de una entidad y sus características se presentan en la sección 2.6.1.1 de este texto al desarrollar el Modelo Entidad-Relación.

to la entidad —cliente— y la entidad —cuenta—; como es clara la diferenciación en un SAU entre la entidad —estudiante— y la entidad —asignatura—; sin embargo, no es tan clara la diferenciación en un SIM entre las entidades: indicadores de operación, de gestión-operativa, de gestión o de predicción en el marco de un DSS.

Ahora, el valor agregado del análisis del contexto al presentar una teoría formal para el diseño de las entidades que participan en una base de datos consiste en que el diseño de las entidades no es intuitivo; por el contrario, se pueden utilizar métricas para el diseño de las entidades; métricas que aseguran la diferencia conceptual al menos en su proceso de discriminación entre dos entidades participantes en una base de datos, de lo cual se puede concluir con seguridad que en un SIB es muy diferente el cliente del banco como entidad que utiliza el portafolio de servicios del banco que la cuenta bancaria como entidad de la cual es propietario el cliente del banco. La formalización del análisis del contexto que se desarrollará en el marco de la presentación del Modelo Entidad-Relación.

i) Sea S un sistema de información soportado en Bases de Datos. El contexto semántico del sistema S está compuesto por un conjunto de características (C) tales que $S = (C_1, C_2, C_3, \ldots C_i, \ldots, C_n)$. Cada $C_i \in S$ tiene las siguientes propiedades: i) C_i es un dato significativo semánticamente para S. ii) Dadas dos características $(C_i, C_j) \Rightarrow (C_i \cap C_j) = \emptyset$ a nivel semántico. iii) Cada C_i es un dato fuente o calculado que describe la organización de S en la base de datos. Luego, cada $C_i \subset S$ se entiende como un conjunto disgregado de características del sistema, tal y como se presenta en la figura 2.1.

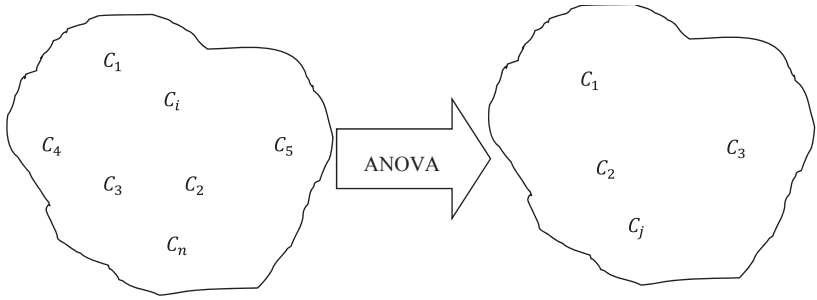


Figura 2.1 Contexto del Sistema (S) e identificación de las entidades (E_i)

2.6.3 Identificar las entidades del sistema

ii) El diseño de las E_i entidades del sistema S se realiza con el siguiente procedimiento: primero, con base en las C_i características del contexto semántico se hace una primera selección de las posibles entidades denotadas por $S = (C_1, C_2, C_3, ..., C_j)$, siendo $j << n$, porque es válido el supuesto de que no todas las características semánticas del sistema se deben seleccionar como entidades. Segundo, a cada una de las C_j características se le asigna a su vez X_k variables de discriminación. Tercero, las C_j características son evaluadas a través de las X_k variables, lo cual genera una tabla de análisis de varianza (ANOVA) [31], [32]. El análisis de varianza de la tabla mencionada se realiza con hipótesis generalizada. Cuarto, con base en los resultados de la ANOVA, y utilizando la distribución F de Fisher, suponga que H_0 = *No se encuentran diferencias significativas entre las C_j características* y H_1 = *Hay diferencias significativas entre las C_j*, con un $\alpha = 0{,}05$ si el valor del F crítico (F_c) es igual a $F_c = 2{,}265$ y el valor del F calculado (F_d) es igual a $F_d = 2{,}598$, luego, como $F_d = > F_c$, entonces se rechaza la hipótesis nula; y por lo tanto hay diferencias significativas entre los C_j seleccionados. Finalmente, como hay diferencias significativas entre los C_j por la métrica ANOVA utilizada, las C_j pasan a ser las E_i entidades semánticas de S. Las entidades de S no solo son semánticamente diferentes entre ellas sino que son pertinentes al contexto de operación en bases de datos del sistema S.

2.6.4 Asociar valores semánticos a las componentes del contexto

iii) A cada $C_i \in S$ se le asocia una función semántica con relación al diseño de los atributos de la entidad de una entidad E_i de la base de datos, así: $f(C_i) = \alpha$, / $\alpha \in [0: \beta]$, siendo β el valor umbral de discriminación semántica. Por lo tanto, si $Lim_{ci \to 0} f(C_i) = 0$, implica un valor de asociación semántico bajo con relación a la entidad E_i; pero si $Lim_{ci \to \beta} f(C_i) = \beta$, se entiende como un valor de asociación semántico alto respecto a E_i.

2.6.5 Agrupar las componentes del contexto

iv) La matriz de agrupamiento de las componentes del contexto, con relación a una entidad E_i, define los atributos que componen una entidad E_i, de la siguiente forma: la primera línea de la matriz contiene las $C_i \in S$ relacionadas con los posibles atributos de E_i. La segunda línea de la matriz contiene la aplicación de la función semántica para cada C_i relacionada con E_i. La tercera línea contiene la agrupación

de los atributos $\alpha_i \in E_i$. La matriz que agrupa los atributos de la entidad se presenta en la tabla 2.4.

Tabla 2.4 Aplicación de la función semántica para asignación de los atributos de las entidades

E_i							
C_i	C_1	C_2	C_3	…	C_i	…	C_n
$f(C_i)$	$f(C_1)$	$f(C_1)$	$f(C_1)$		$f(C_i)$		$f(C_n)$
α =	0	β	0.9 β		0.9 β		0.7 β
$a_i \in E_i$	-	a_2	a_3	…	a_1	…	a_{m_i}

La asignación de alfa (α) es un resultado que representa el promedio de los usuarios participantes en el diseño de la base de datos. Los usuarios sugeridos son: los encargados de la escritura de los esquemas de la base de datos, o relacionados con el lenguaje Data Definition Language (DDL); los relacionados con las consultas a la base de datos o encargados del Data Management Language (DML); los usuarios no conocedores de informática que vayan a utilizar el sistema de información *S*.

La justificación de los valores α se presenta en la tabla 2.5, supuesto un valor de $\beta = 1$.

Tabla 2.5 Tabla de justificación del valor de α

N° Juez	C_1	C_2	C_3	…	C_i	…	C_n
1	0	1	1	…	0.8	…	0.5
2	0	1	1	…	1	…	0.6
3	0	1	0.8	…	0.7	…	0.7
4	0	1	1	…	0.9	…	0.6
5	0	1	1	…	1	…	1
6	0	1	0.9	…	0.8	…	0.5
7	0	1	0.9	…	0.8	…	0.7
8	0	1	0.7	…	0.7	…	0.6
9	0	1	0.9	…	0.7	…	1
10	0	1	0.8	…	0.6	…	0.8
Prom.	**0**	**1**	**0.9**	**…**	**0.8**	**…**	**0.7**

2.6.6 Diseñar los atributos de las Entidades y Relaciones

v) Con base en la matriz que agrupa las características de las entidades se diseñan los atributos (α_{ji}) de las n entidades del sistema S, al derivarse de la matriz los atributos de las entidades representadas por

$$E_1 = [(a_{11}), (a_{12}), (a_{13}), ..., (a_{1i}), ..., (a_{1\,m_1})];$$

$$E_2 = [(a_{21}), (a_{22}), (a_{23}), ..., (a_{2i}), ..., (a_{2\,m_2})];$$

$$E_3 = [(a_{31}), (a_{32}), (a_{33}), ..., (a_{3i}), ..., (a_{3\,m_3})];$$

...

$$E_j = [(a_{j1}), (a_{j2}), (a_{j3}), ..., (a_{ji}), ..., (a_{j\,m_j})];$$

...

$$E_n = [(a_{n1}), (a_{n2}), (a_{n3}), ..., (a_{ni}), ..., (a_{n\,m_n})].$$

Sean n entidades pertenecientes a S, cada una de ellas con sus atributos asociados; el indicador de relación $IR_{i,j}$ para $1 \leq i, j \leq n$ es el grado de relación entre las entidades E_i , E_j para valores de $i \neq j$. El IR representa el atributo de la entidad j con el mayor umbral de relación con la entidad i en el intervalo $[1.\delta]$. La construcción del indicador de relación se presenta en la tabla 2.6.

Tabla 2.6 Justificación del indicador de relación (IR_i)

E₁							
E_1	a_{11}	a_{12}		a_{1i}		$a_{1_{m1}}$	$\boldsymbol{a_{31}}$ δ
E_2	a_{21}	a_{22}		a_{2i}		$a_{2_{m2}}$	
E_3	$\boldsymbol{a_{31}}$ δ	a_{32} 0.1δ		a_{3i} 0.2δ		$a_{3_{m3}}$ 0.1δ	
...							
E_j	a_{j1}	a_{j2}		a_{ji}		$a_{j_{mj}}$	
...							
E_n	a_{n1}	a_{n2}		a_{ni}		$a_{n_{mn}}$	

E_j							
E_1	a_{11}	a_{12}		a_{1i}		$a_{1_{m1}}$	
E_2	a_{21}	a_{22}		a_{2i}		$a_{2_{m2}}$	
E_3	a_{31}	a_{32}		a_{3i}		$a_{3_{m3}}$	
...							
E_j	a_{j1}	a_{j2}		a_{ji}		$a_{j_{mj}}$	$\boldsymbol{a_{n1}}$ δ
...							
E_n	$\boldsymbol{a_{n1}}$ δ	a_{n2} 0.4δ		a_{ni} 0.1δ		$a_{n_{mn}}$ 0.3δ	

E_n							
E_1	a_{11}	a_{12}		a_{1i}		$a_{1_{m1}}$	
E_2	$\boldsymbol{a_{21}}$ δ	a_{22} 0.1δ		a_{2i} 0.2δ		$a_{2_{m2}}$ 0.1δ	
E_3	a_{31}	a_{32}		a_{3i}		$a_{3_{m3}}$	
...							
E_j	a_{j1}	a_{j2}		a_{ji}		$a_{j_{mj}}$	
...							
E_n	a_{n1}	a_{n2}		a_{ni}		$a_{n_{mn}}$	$\boldsymbol{a_{21}}$ δ

La justificación del cálculo del indicador de relación $IR_{i,j}$ se hace de la misma forma que la justificación del valor α, mostrado en la tabla 2.4.

2.6.7 Seleccionar las claves primarias y foráneas de las entidades de la base de datos

vi) La selección de la clave primaria de cada una de las entidades E_i, $1 \leq i \leq n$ resulta del proceso de agrupamiento semántico. La clave primaria es el máximo valor de β asociado con cada entidad E_i. La clave foránea corresponde al valor máximo de β que corresponde al atributo de la entidad E_j la cual se desea relacionar con la entidad E_i, para $1 \leq i,j \leq n$, siendo $i \neq j$. El indicador de relación $IR_{i,j}$ permite seleccionar la(s) clave(s) foráneas. Por lo tanto, la(s) clave(s) foráneas son los máximos valores del umbral de relación δ.

Las entidades en las base de datos requieren una identidad. La identificación o identidad de una entidad requiere el modelamiento de las características propias de la entidad. Identificar las características propias de una entidad se concreta, como se expresó anteriormente, en la definición de los atributos de la entidad. Luego, son precisamente los atributos de una entidad los que realmente la definen y la distinguen dentro de la base de datos de las demás entidades. La identificación de una entidad requiere la definición concreta de una clave única dentro de los atributos de la entidad.

Con base en lo anterior, la definición de una clave genera el concepto de "clave primaria", o lo que es equivalente, la *primary key*. Para la selección de una clave primaria del conjunto de características que definen la entidad se debe desarrollar el concepto de "clave candidata". Las claves candidatas se seleccionan de cada de una de las características de la entidad; lo anterior implica que todos los atributos o características que definen la entidad son factible de ser claves candidatas; pero la importancia de una clave es que diferencie los datos que componen una entidad.

Formalmente, Sea S un sistema de información soportado en Bases de Datos, compuesto por un conjunto de entidades (E) tales que el sistema $S = (E_1, E_2, E_3, \ldots E_i, \ldots E_n)$. Cada una de las E_i entidades son identificadas a su vez por un conjunto de características tales que para una entidad específica E_i se presenta que $E_i = (C_1, C_2, C_3, \ldots C_i, \ldots C_n)$. Entonces las C_n características son factibles de ser claves can-

didatas. Suponga que seleccionamos de las n características las características C_i, C_j tal que $i \neq j$. Suponga que los contenidos concretos de los datos que contiene la característica C_i = (1456, 1789, 1777, 8967, ...); mientras que los contenidos de los datos de la característica C_j = (*Kim Hauser, Kimberly Hann, Kim Hauser, Lynda Hauser*). Lo anterior implica que los contenidos de C_i son completamente diferenciables, o lo que es equivalente, la característica no contiene en sus datos dos atributos que sean iguales; al no ser iguales discrimina los datos al menos en su contenido sintáctico; pero en C_j se presentan los nombres de dos personas iguales sin ninguna diferenciación. Luego, en el diseño la alternativa es seleccionar la característica que discrimina los datos; en este caso la entidad E_i tendrá como clave primaria el atributo o la característica entidad C_i que se está diseñando en la base de datos.

Luego, se utiliza el *término* ***clave primaria*** *para denotar la clave candidata elegida por el diseñador de la base de datos como elemento principal de identificación de las entidades pertenecientes a un conjunto de entidades*[7].

La interrelación entre las E_i entidades que componen en la base de datos requiere del concepto de "clave externa" o **clave foránea** (***foreing key***) de una entidad. Sean dos entidades, E_i y E_j; sea R una relación que tiene como objetivo relacionar ($\leftrightarrow$) la entidad i con la entidad j; entonces dicha interrelación requiere de una clave externa que se puede incluir en la entidad E_i; clave que refencia a la clave primaria de la entidad E_j.

Ejemplo 2.1 Diseño Conceptual de un Sistema de Administración de Edificios (SAE)

Utilizando el Modelo Entidad-Relación diseñar dos entidades del SAE y una relación y realizar el respectivo análisis del contexto de operación del sistema de información.

R/. Análisis del Contexto: El contexto de operación del SAE pertenece al área financiero-contable, donde un conjunto residencial (o condominio) tiene casas o apartamentos de propiedad horizontal. Los propietarios de los bienes raíces (casas o apartamentos) pagan una cuota mensual por el valor de la administración del bien. Luego, el objetivo del SAE es llevar el estado de la contabilidad y las finanzas del edificio, haciendo el balance financiero del condominio con cortes de períodos mensual, semestral y anual.

Inicialmente, al analizar el contexto de operación del SAE, el mundo real tiene un conjunto de características, las cuales se deben analizar con relación a su pertinencia al SAE, lo cual se representa en la figura 2.2.

Se debe tener en cuenta que el contexto de operación de un sistema de información es el mundo real. Entonces, si el contexto de acción de un sistema es el mundo real, todas las características que contenga dicho mundo son potencialmente factibles de participar como entidades, atributos o relacionadas en el diseño a través del M E-R.

En el análisis de la caracterización del contexto definir entidades para un SAE, tales como *día, sol, mar o locomotora* no tienen mucho sentido a nivel conceptual como elementos que necesariamente deben pertenecer al sistema que se está diseñando. Sin embargo, elementos como el nombre del propietario, el NIT o el nombre del edificio son elementos que conceptualmente deben participar estar presentes en el diseño de la base de datos.

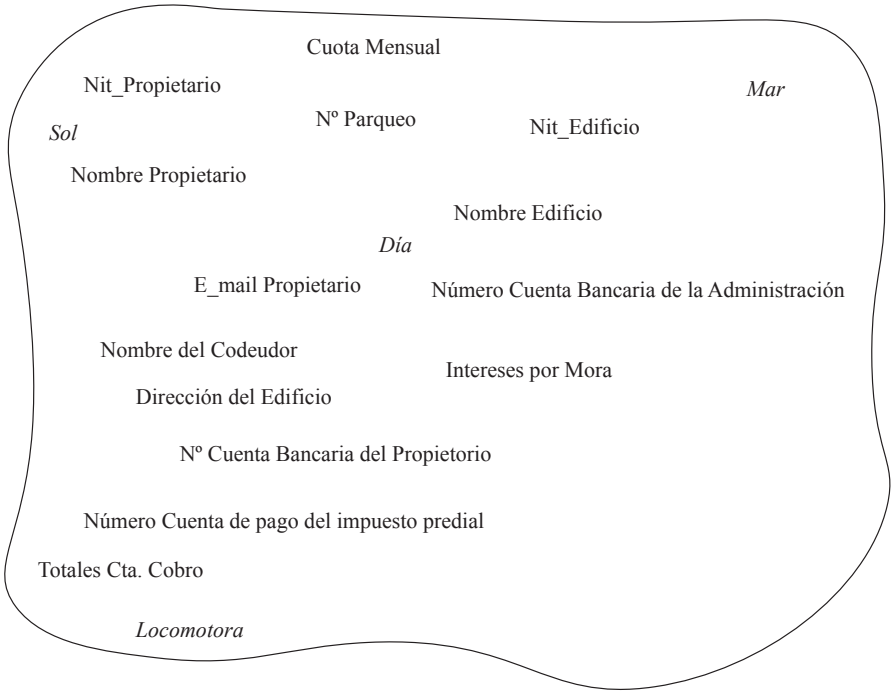


Figura 2.2 Contexto del mundo real del SAE

Con base en el contexto del mundo real se diseñan las entidades E_i del SAE y sus atributos. Las entidades que se van a diseñar son E_1 = *Edificios*, E_2 = *Apartamentos.* Identificadas las entidades, el paso siguiente es diseñar las características o atributos de cada una. Los atributos se diseñarán utilizando el Análisis de Grupos. Es decir, agrupar a cada entidad aquellas características del mundo real que más la identifiquen y definan. Luego, los atributos de la entidad E_1 = *Edificios* se agruparán en negrilla; mientras que las características de la E_2 = *Apartamentos* se agruparán en letra cursiva, según la figura 2.3.

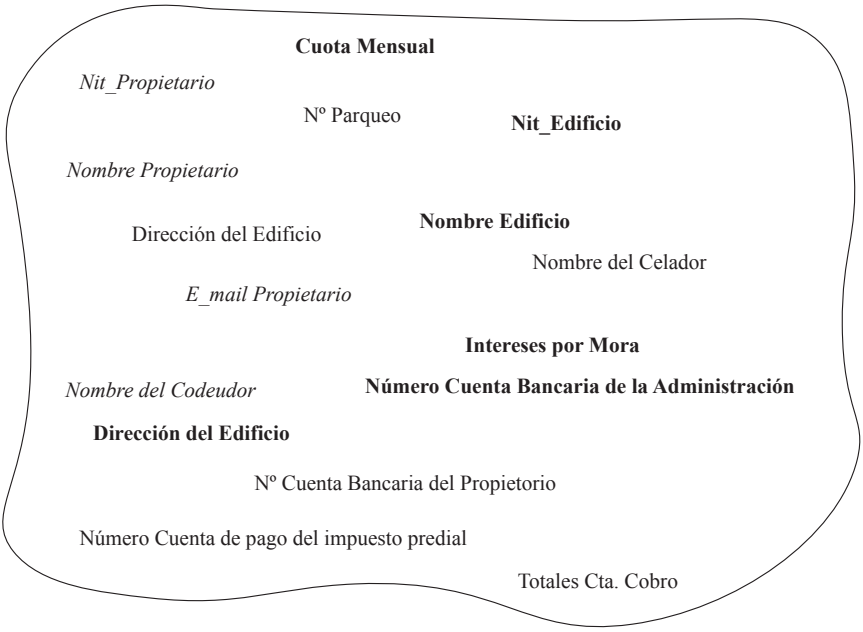


Figura 2.3 Análisis de grupos aplicado al diseño lógico del SAE

Teniendo en cuenta las características que definen cada uno de los grupos se construye el diseño lógico utilizando el Modelo Entidad-Relación. El M E-R permite la modelación gráfica de las entidades, los atributos de las entidades y las relaciones que participan en un sistema de información. Las entidades se grafican con rectángulos, los atributos con elipses, las relaciones con rombos [23] y las claves primarias se subrayan. El diseño del M E-R para el SAE en el diseño de dos entidades: E_1 = *Edificios* y E_2 = *Apartamentos*, se presenta en la figura 2.4.

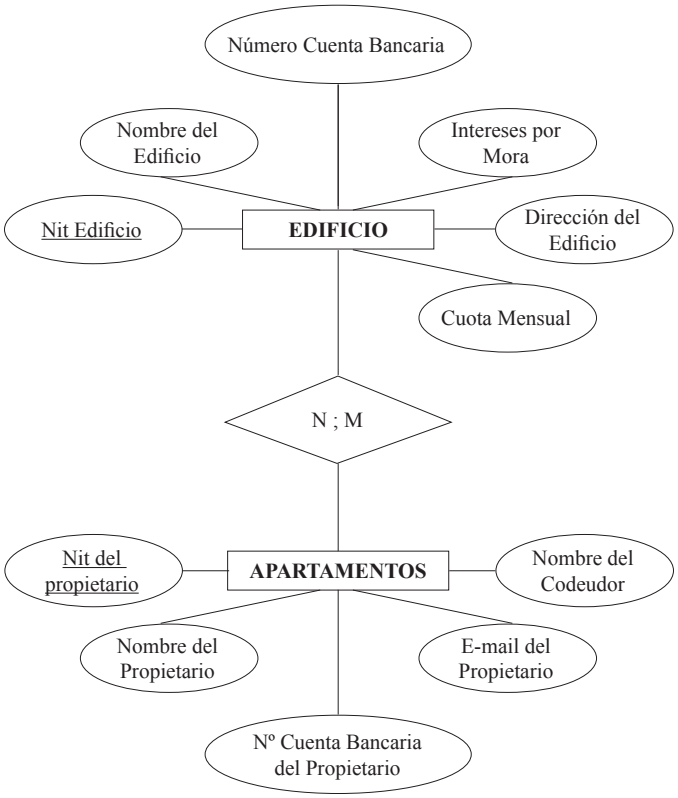


Figura 2.4 Modelo Entidad-Relación del Sistemas de Administración de Edificios (SAE) en sus dos entidades Edificios y Apartamentos

2.7 Diseño físico de la base de datos

Con base en los pasos anteriores, los resultados de la fase de diseño lógico de la base derivados del M E-R son: las entidades, los atributos de las entidades, las claves primarias de las entidades, las relaciones y las claves foráneas en su diseño gráfico.

El M E-R proporciona el diseño lógico de la base de datos. Luego, el paso del diseño lógico al diseño físico de la base de dato implica convertir el M E-R a una estructura de bases de datos, que en el caso de que el gestor de la base de datos sea relacional, entonces las estructuras de datos que se maneja en la organización física interna de los datos son tablas relacionales. Para el paso del M E-R de carácter gráfico a las es-

tructuras de tablas se utiliza el Modelo Relacional (MR). El MR [33] representa la estructura de la base de datos *como una colección de relaciones*[5]. Las relaciones en la base de datos contienen del M E-R tanto las entidades con sus atributos como las relaciones representadas a través de tablas . Luego, el contenido de una tabla relacional es una estructura de n filas, llamadas tuplas, por m columnas. Las columnas identifican los atributos de las entidades, y las tuplas o filas almacenan los datos concretos de la tabla relacional. Modelo Relacional que se utiliza en proyectos multidisciplinarios que contienen grandes cargas de datos [34].

El diseño físico del Modelo Relacional que representa las dos entidades de la base de datos del SAE con su estructura de tablas se muestra en la tabla 2.8, para lo cual se tiene en cuenta la redefinición de los atributos de las tablas en sus descriptores. Una instancia de datos concreta del Modelo Relacional del sistema se muestra en la tabla 2.7.

Tabla 2.7 Redefinición de los atributos de las entidades del SAE

$E_1 = Edificios$	
Nombre del atributo inicial	Nombre del descriptor en la columna de la tabla
Nit Edificio	Nit_E
Nombre del Edificio	Nom_E
Número de la Cuenta Bancaria de la Administración	Cta_Ad
Interés por Mora	I_m
Dirección del Edificio	D_e
Cuota Mensual	C_m
$E_2 = Apartamentos$	
Nombre del atributo inicial	Nombre del descriptor en la columna de la tabla
Nit del Propietario	Nit_Pr
Nombre del Propietario	Nom_Pr
Nombre del Codeudor	Nom_Co
E_mail del Propietario	Email_Pr
Número de la Cuenta Bancaria del Propietario	Cta_Pr

Tabla 2.8 Modelo Relacional (MR) de dos entidades de la base de datos del SAE

Nit-E	Nom_E	Cta_Ad	I_m	D_e	C_m
80-1415-24	Dubai Inn				170000
80-2240-35	Mar 789				200000
80-1424-32	Navas				120000
80-2430-12	Puerto Sol				300000
80-1514-18	Danys				450000
80-3245-26	Rinko				350000
80-5689-12	Piaget84				150000
80-1548-24	Humbolt				190000
80-1565-12	Abraham78				200000

Nit_Pr	Nom_Pr	Nom_Co	Email_Pr	Cta_Pr	
72545980	Becerra Ángel				80-1548-24
79458963	Cárdenas Jorge				80-1548-24
78956233	Cárcamo Luis				80-1548-24
77896536	Duarte Miguel				80-1548-24
79856456	Durán Ana				80-1424-32
76895145	Mora Eugenio				80-1424-32
78963556	Montes Luisa				80-1424-32
79568145	Visbal Marta				80-1424-32
78956456	Wright Sandy				80-1424-32

2.8 Carga de los datos a la base de datos

La carga de los datos a la base de datos es la definición concreta de la base de datos (física) utilizando el gestor de la base de datos. Poblar una base de datos requiere de la definición de los esquemas, dominios, tablas y vistas de la base de datos. Definiciones que requieren necesariamente del cumplimiento de las características que aseguran la integridad de la base de datos con el fin de tener unos datos fidedignos que puedan aportan a las decisiones de los usuarios con base en las consultas que se le hagan a la base de datos.

Luego, es necesario que la carga de datos tanto en sus definiciones como en la acción de poblar la base de datos tenga en cuenta las reglas de integridad de la base de datos

en sus diferentes restricciones de datos, entidades, dominios o generales para el aseguramiento de la calidad de los datos.

La carga de los datos en la base de datos a través del gestor de datos se desarrolla en el capítulo tres.

2.9 Operación de la base de datos

Los sistemas de información en su desarrollo no son sistemas estáticos sino dinámicos. La dinámica del mundo real representativa del contexto de un sistema informático se debe reflejar tanto en la estructura como en la operación de la base de datos. Teniendo en cuenta la dinámica del contexto del sistema, el sistema se debe analizar permanente en su operación en fase de producción; y por lo tanto es dicho análisis el que permite incorporar la dinámica del contexto en los diseños tanto lógicos como físicos de la base de datos.

El análisis de la operación del sistema no es fácil. Como mínimo la operación de la base de datos de soporte al sistema de información debe necesariamente ser efectiva. La efectividad implica el cumplimiento de dos características, la eficacia y la eficiencia.

La *eficacia* se cumple cuando la base de datos al ser operada a través del gestor de la base de datos << hace lo que debe hacer >>; es decir, funciona; nunca se cae la base datos o siempre responde los requerimientos del usuario. Por su parte, la *eficiencia* ha de cumplir con la respuesta rápida al usuario. Transportar grandes volúmenes de datos requiere de un tiempo de operación, para ejecutar la consulta o el "quering" contra la base de datos.

2.10 Mantenimiento de la base de datos

Cumplir con la dinámica del sistema de información requiere, por lo tanto, cumplir con el mantenimiento de los datos. Este mantenimiento se hace con el rediseño del M E-R y, por lo tanto, el rediseño del MR que soporta la base de datos. Por consiguiente, en la base de datos es necesario: crear nuevas tablas, alterar tablas, crear nuevas tablas de índices, rediseñar las restricciones de la integridad de la base de datos, entre otras acciones.

Habiendo desarrollado la síntesis de los pasos de para el desarrollo de un sistema informático soportado en base de datos, el paso siguiente es definir la base de datos utilizando las sentencias SQL pertinentes a la creación de la base de datos.

2.11 Diseño de bases de datos a partir de especificaciones de requerimientos

Los estados en los que se puede encontrar una organización con relación al diseño de sistemas en bases de datos son tres: i) La empresa va a estructurar un sistema de bases de datos sin tener experiencia en el manejo de sistemas de información. ii) La empresa está interesada en construir un sistema en bases a partir de un sistema de información basado en archivos. iii) La empresa va a rediseñar su sistema en bases de datos existente.

En cualquiera de los tres estados antes mencionados, la empresa debe empezar la construcción del sistema de bases de datos con base en la especificación de requerimientos de información del sistema. La especificación de requerimientos en el desarrollo de un sistema de información pertenece al área de Diseño y Desarrollo de Software en el marco de la Ingeniería de Software. La especificación de requerimientos son las necesidades de información que debe cumplir el sistema de información en su operación para todos los usuarios (internos y externos) al sistema.

Formalmente

> ... Los requerimientos del usuario describen los objetivos o tareas que el usuario debe ser capaz de llevar a cabo con el producto. Las opciones para representar los requerimientos del usuario incluyen casos de uso, descripción del escenario, tablas de eventos-respuestas. Los requerimientos del usuario describen, por lo tanto, qué el usuario será capaz de hacer con el sistema [35].

Con base en lo anterior, una profundización de Ingeniería de Requisitos está por fuera del alcance de este libro. El concepto de la especificación de requerimientos que se empleará en este texto es una descripción del escenario del sistema de información. Luego, en el caso de que se tenga la descripción de los requerimientos del sistema, es más fácil realizar tanto el análisis del contexto de operación del sistema como el diseño del Modelo Entidad-Relación, y consecuentemente, su paso al Modelo Relacional.

El método de ejemplificación del diseño del M E-R para el desarrollo de esta sección es el método de casos, como se ejemplifica a continuación:

Ejemplo 2.2 Caso: Diseño del M E-R para una oferta de cursos intersemestrales en una Universidad

Diseñar el Modelo Entidad-Relación (M E-R) de soporte a un sistema universitario de oferta de cursos vacacionales (intersemestrales) ofrecidos por una Universidad, estructurada en varias sedes distribuidas en las ciudades capitales de varios departamentos, que tiene los siguientes requerimientos de información:

a) Teniendo en cuenta que la Universidad tiene sedes en varias ciudades, cada sede tiene un código, un nombre, la ciudad, el departamento de ubicación, la oferta de cursos intersemestrales (Idiomas, Artes, **Música**, Deportes, etc.), de los cuales se requiere el código del curso, su nombre, el número de créditos, el nivel del curso (especificando 100=Infantil, 200=Primaria, 300=Bachillerato, 400=Pregrado, 500=Postgrado), el número límite de alumnos por curso.

b) Cada una de las seccionales de la Universidad estructura autónomamente la programación de cursos intersemestrales; pero debe ofertar como mínimo un curso vacacional, lo cual indica que el mismo curso vacacional puede ser ofrecido por varias sedes; pero el registro y la liquidación del curso en la base de datos es responsabilidad directa de la respectiva sede de la institución.

c) Los cursos intersemestrales se ofrecen en cada sede con alojamientos en casas, apartamentos o en el campus universitario de cada sede.

d) Teniendo en cuenta los niveles de oferta de los cursos intersemestrales, y que son ofrecidos para nacionales y extranjeros, para el registro de alumnos se requiere tener el código del alumno (Código de la Universidad (CU),Tarjeta de Identidad (TI), Cédula de Ciudadanía (CC), Pasaporte (PA)), tipo de documento de identidad (CU,TI,CC,PA), apellidos y nombres del estudiante, e_mail alumno, e_mail del responsable, teléfono del alumno y teléfono del responsable.

e) La estimación de la demanda a los cursos intersemestrales se basa en la cobertura poblacional de la ciudad donde está ubicada la sede, para lo cual se requiere el número de habitantes de cada ciudad y su superficie.

f) La Universidad necesita el control de la operación de la oferta de cursos vacacionales, para lo cual es necesario tener la liquidación para cada estudiante y por cursos; en el entendido de que los cursos con mayor demanda son aquellos que registran un mayor nivel de calidad y son los cursos candidatos para reabrirse en la próxima temporada de vacacionales. La liquidación por créditos se hace tanto en pesos como en dólares, por lo cual se requiere tener en cuenta el tipo de moneda, y una vez el alumno seleccione el curso vacacional se le debe enviar al *e_mail* personal y al *e_mail* al responsable el volante de pago de la liquidación de los cursos seleccionados para el vacacional.

g) Finalmente, debido a las necesidades de los estudiantes al aplicar a dichos intersemestrales de tiempos libres para conocer las ciudades y los sitios turísticos donde están ubicadas las sedes, se requiere que cada curso tenga una oferta variada de programación de horarios diurnos y nocturnos, enfatizando el menor número de cursos ofertados durante los jueves y viernes, y una programación de cursos únicamente de 10 a 12 del día y de 2 a 4 de la tarde.

La solución del M E-R de la oferta de cursos intersemestrales en una universidad se muestra en la figura 2.5.

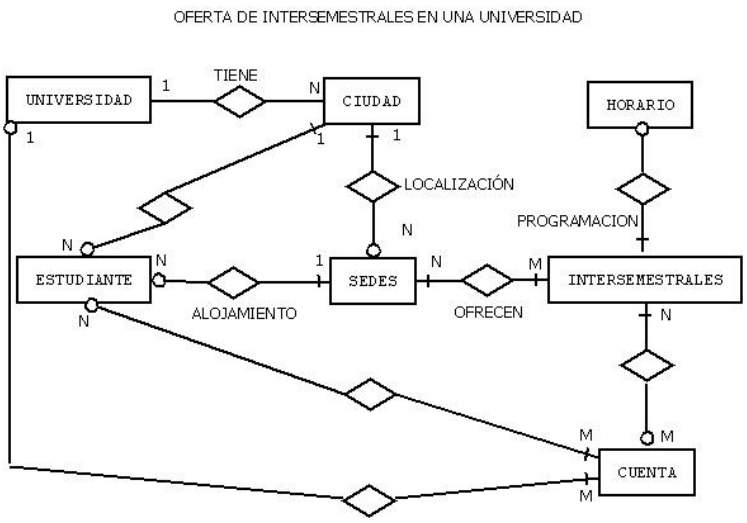


Figura 2.5 Diseño del M E-R de la oferta de cursos intersemestrales en una universidad

Ejemplo 2.3 Caso: Diseño de una base de datos de una empresa importadora-exportadora

La especificación de requerimientos de la empresa importadora-exportadora tiene la siguiente descripción.

En el marco del Tratado de Libre Comercio (TLC) entre continentes, una compañía multinacional requiere el diseño conceptual de una base datos utilizando el Modelo Entidad-Relación (M E-R), con los siguientes requisitos:

a) La multinacional es una empresa comercializadora de productos, cuyo contexto de operación son los continentes de América, Europa y Asia; por lo tanto, se requiere considerar en las transacciones comerciales los tipos de monedas estándares de comercialización asociadas a cada continente.

b) La comercializadora estratifica sus trabajadores en tres grupos: gerentes, administradores y vendedores. De los gerentes es importante conocer su nivel de formación y experiencia en el manejo de empresas comerciales; de los administradores, su nivel de estudios y tipos de certificación en el manejo de proyectos comerciales; y en el caso de los vendedores, su nivel de estudio y el número y el nivel de especialización de los idiomas que manejan.

c) La comercializadora tiene en los países de los cuales importa y a los cuales exporta sus mercancías gestores del negocio (un grupo gestor por ciudad) en las ciudades bases de producción de las mercancías y de entrega de productos. Los grupos de gestión tienen una identificación internacional, un nombre al cual está asociada una única dirección electrónica y una página web de gestión. Un grupo de gestión de negocio está integrado por un gerente de zona, quien lidera a varios administradores de países y de los cuales dependen varios vendedores.

d) Cada uno de los trabajadores de la empresa está asignado a un grupo de gestión; el empleado puede cambiar de grupo gestor (inclusive puede volver al mismo grupo), y de ello se requiere un registro histórico. Trabajadores que se identifican por un código internacional de la compañía (chip de ubicación), el pasaporte, el nombre y el número del móvil; empleados de los cuales es necesario también conocer el año de nacimien-

to, el país de origen, el número del seguro médico, y la identidad de la persona de contacto con el trabajador, la cual se identifica a través de un Nit, nombre y un número celular, con la característica que pueden haber varios contactos por trabajador.

e) La actividad de la empresa consiste en comprar productos en el exterior y vender dichos productos de acuerdo con los pedidos de los clientes. Los clientes en la empresa son identificados mediante un código de cliente, un índice de prioridad, el nombre del cliente, sus direcciones electrónicas (principal y opcional) y sus celulares de contacto (2).

Se debe tener en cuenta que el índice de prioridad está en función de dos indicadores, que son: el primero, representado por número de pedidos ejecutados/número de pedidos ordenados por el cliente; el segundo, calculado por número de pedidos pagados/número de pedidos ejecutados.

f) Para poder llevar a cabo la gestión de compra y ventas de mercancías, la empresa dispone de una red logística compuesta por los tipos de transporte terrestre, marítimo y aéreo, los cuales son solicitados por subcontratos a los compañías transportadoras; pero la empresa requiere tener la constancia de la matrícula del transporte, el tipo de transporte, el costo del transporte de la mercancía y el costo del seguro de transporte. Adicionalmente, de los aviones se requiere el precio del combustible, de los barcos su capacidad, y de los camiones su máxima capacidad de contendor en pies.

g) Los viajes de compra o distribución de productos son organizados por un grupo gestor del negocio, y se identifican con un código del viaje, el cual es interno a cada grupo gestor; en tal sentido, el código del viaje es posible que se repita en diferentes grupos de gestión. En cada uno de los viajes es necesario conocer: i) Qué tipo de transporte se ha utilizado. ii) Qué vendedor ha realizado la gestión del viaje del producto y si el producto es de importación (compra) o de exportación (distribución). iii) El recorrido del viaje, representado por la las ciudades de salida y llegada de la mercancía, como también la fecha de salida, la fecha de llegada de la mercancía.

Teniendo en cuenta las especificaciones de requerimientos de información del sistema, diseñar el Modelo Entidad-Relación representativo de la base de datos.

El diseño del M E-R se presenta en la figura 2.6.

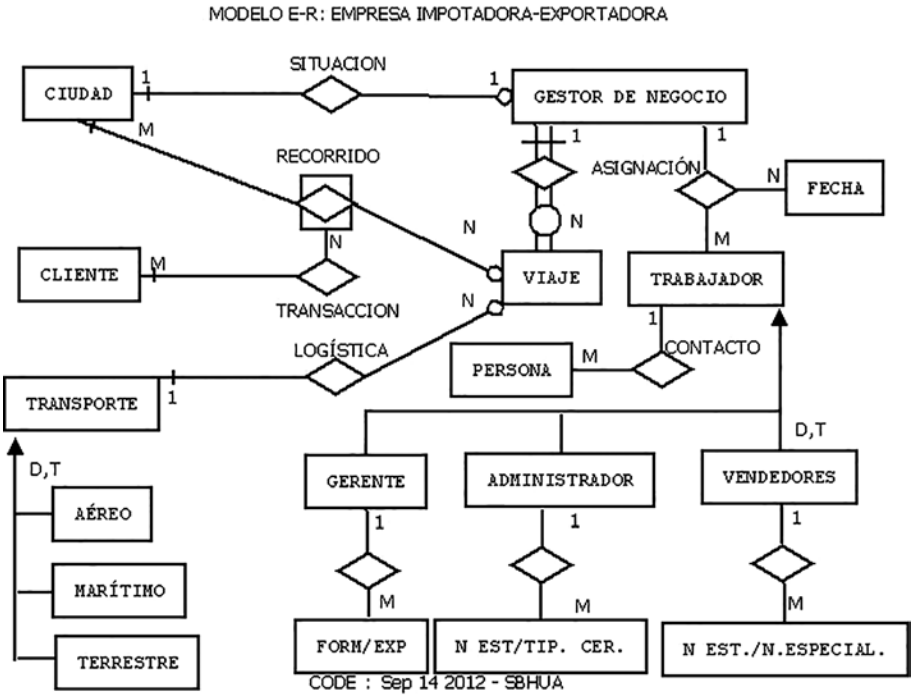


Figura 2.6 Diseño del M E-R de las especificaciones de requerimientos de la empresa importadora-exportadora

2.12 Resumen conceptual

La **fase de diseño de una base de datos** está compuesta por los siguientes pasos: Planificación del desarrollo del sistema. Diseño conceptual del sistema de bases de datos, el cual comprende: analizar el contexto, identificar las entidades del sistema, asociar los valores semánticos a las componentes del contexto, agrupar las componentes, diseñar los atributos de las entidades y las relaciones y seleccionar las claves primarias y foráneas de las entidades o relaciones de la base de datos. Diseño físico de la base de datos en el computador utilizando el gestor de la base de datos. Operar la base de datos para identificar su desempeño y finalmente mantener la base de datos.

Los modelos que soportan el diseño lógico de una base de datos sin hacer una enumeración exhaustiva son: el jerárquico; el de red o plex; el de entidad-relación: el orientado por objetos; el documental; del tipo entidad-atributo-valor y el de esquema en estrella. El texto propone para esta fase el Modelo Entidad-Relación.

Una **entidad** es un objeto que se identifica de los demás de su especie a través de sus atributos; una **relación** es el grado de asociatividad entre entidades.

El M E-R da como resultado el diseño lógico de la base de datos, mientras que el Modelo Relacional permite pasar a tablas relacionales el diseño mencionado de la base de datos.

Una **tabla relacional** es una estructura que se almacena físicamente en la base de datos compuesta por un conjunto de columnas y un conjunto de líneas o tuplas; columnas que contienen las claves primarias y foráneas de la relación.

Una vez desarrolladas las fases de análisis y diseño de una base de datos utilizando los modelos Entidad-Relación y Relacional, el capítulo siguiente desarrolla la creación de la base de la datos a través de la utilización de un gestor de bases de datos.

Ejercicio

A continuación se describe la siguiente especificación de un problema de un sistema de información, la cual es importante leerla detenidamente para resolver los ejercicios de este capítulo.

COMPAÑÍA DE SEGUROS ABC

Una compañía de seguros está interesada en implementar un sistema de información (ERP) que le permita gestionar las operaciones de seguros que se explicitan a continuación: La compañía ofrece seguros divididos en tres modalidades: de vida, educativo y automotor, los cuales son contratados por los cliente, que a su vez se dividen en dos categorías: clientes corporativos y personales. Un cliente puede comprar o contratar más de un seguro, incluyendo cualquiera de los tres que se ofrecen; cuando un cliente compra un seguro debe incluir un beneficiario, cuyos datos pueden tomarse de la misma entidad de cliente.

Los clientes son atendidos por un agente de seguros (vendedor de seguros); este puede atender a varios clientes sin importar su categoría; a la compañía le interesa llevar la información de la facturación que se origina cuando un cliente compra o contrata un seguro. Igualmente es necesario registrar el valor facturado por cada agente de seguro.

- Del cliente interesa almacenar: datos personales y corporativos.
- Del vendedor (empleado), sus datos personales y salario=básico + comisiones.
- La factura: es importante registrar el valor, la fecha, su identificador; una factura, por supuesto, puede cobijar la compra de uno o más seguros.
- Del seguro en general: identificador, una descripción y costo total.
- Seguro de vida: tiene un identificado, costo, el beneficiario, una vigencia y deducciones.
- Seguro Soat: contiene identificador, costo, la placa vehículo, vigencia.
- Seguro Educativo: identificados, costo, vigencia y un beneficiario.

Desarrolle:

1. Tomando como referencia la descripción del sistema ERP anterior, elabore un diagrama general que representa las diferentes etapas del diseño de un sistema de información fundamentado en bases de datos.
2. Utilizando la descripción del ejercicio citado ERP, elabore el diseño conceptual de un sistema de base datos utilizando un Modelo de Entidad-Relación; solamente describa las entidades, relaciones y sus atributos principales. Utilice la simbología de Peter Chen.
3. De acuerdo con el diseño elaborado en el punto 2, describa los constructos básicos del Modelo Entidad-Relación.
4. Utilizando el modelo inicial elaborado en el punto 2 y 3 realizar un análisis de contexto de un sistema de información en bases de datos, que permita identificar llaves principales y foráneas de cada una de las entidades.
5. Cuáles son los componentes esenciales para este sistema de bases de datos.
6. De acuerdo con el sistema ERP compañía de seguros ABC, describa un segmento de la base de datos que muestre el diseño físicos de la misma; se puede tomar como muestra al menos 3 entidades de información para la representación.
7. De acuerdo con el sistema ERP compañía de seguros ABC, describa un segmento de la base de datos que muestre el diseño físico de la misma, escriba el diseño de la base de datos física que represente la relación lógica y física de las entidades de información que describen los datos de las diferentes clases de seguros y sus clientes.
8. Si se hace necesario una modificación del sistema de información relacionada con la información de la tipología de clientes (personales, corporativos), como se vería afectado el modelo lógico y físico del sistema.
9. De acuerdo con el ejercicio planteado, sistema ERP compañía de seguros ABC, describa las especificaciones de requerimientos que indican la selección de un sistema de base de datos para la implementación física del sistema.

10. Utilizando la descripción del ejercicio citado ERP, elabore el diseño conceptual detallado para el sistema de base datos utilizando la técnica de Modelo de Entidad-Relación, describa las entidades, relaciones y sus atributos principales. Y utilice la simbología de Peter Chen.

Capítulo 3

Creación de datos de la base de datos

3.1 Objetivos

Al término de la lectura de este capítulo y con base en el aprendizaje de los capítulos anteriormente expuestos, se pretende que el lector aprenda:

- La definición de datos para poblar la base de datos, en términos de creación de la base de datos, los esquemas, las tablas y los índices.
- Los tipos de datos que soporta el estándar del lenguaje SQL.
- La definición de controles de integridad a la base de datos mediante la utilización del SQL.
- La administración de vistas de la base de datos.

3.2 Síntesis conceptual

El éxito de un sistema de información radica en la fidelidad de los datos que están almacenados en el sistema con el propósito de generar información confiable al usuario. Aceptando que las etapas sugeridas para diseñar la base de datos han sido realizadas correctamente, poblar la base de datos requiere el conocimiento del tipo de datos soportados por el estándar SQL y la utilización de funciones para cumplir con las restricciones de integridad en la base de datos. La creación de vistas de la base de datos y su administración no solo contribuye a implementar un cierto nivel de seguridad de la base de datos, sino que sirven para tener un control independiente de la misma, razón por la cual es importante su creación y administración.

3.3 Definición de *datos*

Se recuerda al lector, como se explicó en los capítulos anteriores, que la construcción de un sistema de información (S) soportado en base de datos tiene como base los

siguientes pasos: i) *Identidad del sistema que se va a construir*: especifica el área de conocimiento y sector económico en el cual va a operar S. ii) *Análisis de contexto del Sistema*: identifica el contenido sintáctico y semántico de las componentes del Sistema. iii) *Diseño de las componentes del sistema de la base de datos*: define las componentes del esquema[1] de la base de datos, donde se identifican las entidades, atributos y relaciones de la misma, paso que corresponde al diseño lógico-conceptual de la base de datos. iv) *Construcción de la base de datos*: comprende tanto la definición de la base de datos como el poblar esta utilizando un gestor de bases de datos (o Diseño Físico de la base de datos a través del Gestor); y iv) *Sintonización de la base de datos*: es el análisis de la operación del sistema de información (S) soportado en bases de datos para identificar las mejoras en el espacio de almacenamiento y el tiempo de respuesta en la operación de la base de datos, con el fin de dar una mejor respuesta al usuario del sistema informático.

Luego, la construcción de la base de datos requiere la definición de los datos; y esta definición necesita el conocimiento de los tipos de datos que se van a definir en la base de datos. Tipos de datos que en este texto se presentan con relación al estándar SQL de la ISO.

3.3.1 Tipos de datos SQL

La estructura sintáctico-semántica del esquema que define la base de datos requiere manipular los valores de los datos de la misma. Por lo tanto, cada valor manipulado se define en la base de datos a través de un tipo de dato; el tipo de dato sirve para diferenciar los contenidos de la base de datos en su especificidad digital (booleano), numérica (numérico exacto o aproximado), carácter, u objetos de tipo carácter definidos de gran tamaño [37].

Con base en lo anterior, al definir el esquema de la base de datos en su construcción física, o lo que es equivalente, al crear una tabla, sus atributos y relaciones deben tener un identificador válido para definir el nombre de la tabla y, a su vez, especificar los tipos de datos de cada uno de los atributos (o columnas de la tabla), con el fin de que la integración de los atributos que componen todos los atributos (columnas) de

[1] El concepto de esquema de la base de datos puede ser definido como esquemas de la misma, al ser integrados en una ERP donde las diferentes estructuras de los esquemas de las bases de datos soportan los módulos de la ERP.

la entidad integren de una forma significativa los contenidos identificados como las tuplas (o filas) de la tabla o conjuntos de tablas que componen el esquema de la base de datos. Los tipos de datos cumplen la función de definir el rango de cada valor de la tabla o el argumento que ha de tener el contenido del atributo.

A su vez, el tipo de datos se puede clasificar como escalar o no escalar. El tipo de datos *escalar* se caracteriza por tener un solo valor, el cual recibe también el nombre de "valor atómico", como es el caso de almacenar la cadena de dígitos binarios de longitud fija así: '01101', con la instrucción *bitstring* BIT (5). Por su parte, el tipo de datos *no escalar* es aquel que contiene un conjunto de valores y recibe el nombre de "colección de datos", tal es el caso de un objeto en la base de datos.

3.3.1.1 **Tipos de identificadores**

La función principal de un identificador SQL es concretar la identidad o el nombre de un objeto dentro de la base de datos. Los caracteres que pueden conformar un indicador son: i) letras mayúsculas (A_Z); ii) letras minúsculas (a_z); iii) números digitales (0_9); y finalmente, el carácter de subrayado (_). Por su parte, en la construcción del identificador se deben cumplir las siguientes reglas: i) el identificador debe comenzar con una letra; ii) el nombre del identificador no puede contener espacios; y iii) la longitud máxima del identificador puede ser solo de 128 caracteres.

3.3.1.2 **Tipos de datos escalares (booleanos, caracteres, bit)**

Datos booleanos

Los datos de Boole corresponden a los valores de verdad comparables y asignables TRUE (verdadero) o FALSE (falso), pero también soportan la condición UNKNOWN (o desconocido) como valor NULL.

Datos tipo carácter

El almacenamiento y la manipulación de los datos alfanuméricos corresponden al tipo de dato carácter. Estos datos están compuestos por frases de texto libre o palabras. Este tipo de datos tiene pocas restricciones en su operación; por cuanto una columna representativa de un atributo de una tabla definida como entidad en la base de datos que esté definida como carácter puede almacenar tanto números como

letras o, en síntesis, caracteres alfanuméricos. Lo anterior contrasta con los datos definidos como del tipo numérico, por cuanto en este caso solo pueden almacenar números (con contenidos digitales); entendible la restricción por la realización de cálculos aritméticos (o lógicos) con este tipo de datos.

La notación de este tipo de datos se representa por

```
CHAR|CHARACTER [VARYING] [longitud]
```

Los datos tipo CHAR sirven para definir sartas de longitud fija; y el gestor de la base de datos asegura que los datos almacenados en una tabla que contenga una columna CHAR tienen el tamaño especificado por la longitud. Por ejemplo, la definición de un código de longitud fijo para el Sistema Integrado de Manufactura (SIM), correspondiente al atributo de identificación de la tabla de proyectos:

```
No_proyecto CHAR(10).
```

3.3.1.3 **Datos numéricos (exactos, aproximados, fecha y hora, intervalo)**

Los tipos de datos numéricos son aquellos que almacenan valores positivos, negativos o cero y son empleados en la definición de cantidades numéricas exactas o aproximadas.

Datos tipo BIT

La definición de sartas de bits o secuencias de dígitos binarios (0,1) es similar a la especificación del tipo de dato carácter, el cual tiene como formato

BIT [VARYING] [longitud]

El dato tipo BIT representa una sarta de bits de amplitud predeterminada (fija) en la definición. Luego para almacenar una sarta binaria de longitud fija específica '10101010' se tiene que declarar una columna tipo sarta, tal como

```
BitString BIT (8)
```

Definición de datos numéricos exactos

Para definir números con una representación exacta se utiliza el tipo de datos numéricos. Cada número definido a través de este tipo está compuesto por dígitos (0,9), una coma decimal y signo, estos dos últimos (, +, -) de naturaleza opcional. Este tipo de datos consta de dos partes: precisión (p) y escala (e). La precisión representa el número total de dígitos significativos, y la escala es el número total de posiciones decimales.

El formato para la definición de datos numéricos exactos es [6]:

```
NUMERIC[precisión|escala]
DECIMAL [precisión|escala]
INTEGER|INT
SMALLINT|INT
```

NUMERIC/DECIMAL

La definición de datos del tipo numérico y decimal se utiliza para representar números en formato de notación decimal.

El formato de la notación DECIMAL se denota como DECIMAL (p,e), donde p es la precisión y e la escala; teniendo en cuenta que las características de precisión están sujetas a la implementación cuando son predeterminadas.

Con base en lo anterior, si se define una columna de una tabla como

MOVIMIENTO DECIMAL(7,1) corresponde a un número que tiene 6 dígitos antes del punto decimal y un solo dígito después del punto.

A nivel complementario, si en una tabla se define la columna

```
Total_a_pagar DECIMAL(8,2)
```

la definición permite contener valores que alcanzan el rango de 999.999,99 .

INTEGER

Un entero es un tipo de dato ANSI-SQL, el cual refiere a valores numéricos, los cuales tienen solamente parte entera y no tienen punto flotante o punto decimal. En este caso, una definición del tipo INTEGER solamente almacenará números enteros; ello implica que si un número es definido como entero y tiene parte decimal se truncará en su almacenamiento la parte decimal del número. Luego, al almacenar el número 2341.345, realmente el número quedará almacenado como 2341. Se debe tener en cuenta que al utilizar SMALLINT el espacio utilizado para almacenar los datos es menor.

Definición de datos numéricos aproximados

Los números que no tienen una representación exacta, a diferencia de los números naturales (1, 2, 3, 4, 5, 6,...), son los números reales. Un número real en su notación de parte entera y parte fraccionaria (o decimal) tiene la característica de que el punto decimal puede estar en cualquier parte del número, o inclusive puede no tener parte decimal. Suponga la cantidad 3415.14; se puede escribir de la forma 341.514E10 (341.514×10^1), 34.1514E-2 o 0.341514E4; notaciones todas que son equivalentes. Luego, los datos numéricos aproximados reciben también el nombre de "datos en coma" o "punto flotante", y en este caso son similares a la notación científica.

El formato de definición de este tipo de datos es el siguiente [6]:

```
FLOAT [precisión]
REAL
DOUBLE PRECISION
FLOAT [precisión] , REAL
```

En la notación de los números punto flotante son importantes los conceptos de i) la mantisa y ii) el exponente. i) La *matisa* es el número el cual puede tener o no el punto decimal. ii) Por su parte, el *exponente* es el valor por el cual se multiplica la mantisa para dar la variabilidad necesaria al punto decimal, para lo cual se utiliza una potencia multiplicada por 10.

En la definición de este tipo de datos se debe tener en cuenta que la precisión de los datos del tipo REAL y DOUBLEPRECISIÓN dependen de la implementación de la versión del gestor de la base de datos.

Datos del tipo ***FECHA*** *y* ***HORA***

Los datos que indican las fechas del calendario en sus días y los tiempos horarios en sus horas, minutos y segundos son definidos de acuerdo con los formatos palabras reservadas de DATE, TIME y TIMESTAMP. El formato para definir las fechas y horas se presenta a continuación:

```
DATE
TIME [precisiónTemporal] [WITH TIME ZONE]
TIMESTAMP [precisiónTemporal] [WITH TIME ZONE]
DATE [6]
```

La función DATE tienen como objetivo almacenar las fechas calendario mediante la utilización de los campos AÑO, MES, DÍA (YEAR,MONTH,DAY). La especificidad de la función DATE en sus valores se puede manejar como una sarta de caracteres, en cuyo caso es una literal del tipo string. La especificación de una literal con el estándar ANSI para la función DATE es

```
DATE '2013-11-24'
```

Se debe tener en cuenta que la literal ANSI DATE no contiene espacio para el tiempo; y el usuario debe especificar la función en el formato 'YYYY-MM-DD', lo cual corresponde a YYYY = Año, MM = Mes y DD = Día.

3.3.2 Control de integridad

3.3.2.1 Requerimiento de datos

3.3.2.2 Dominio de atributos

Los contenidos de los atributos de una tabla en una base de datos como unidades de información tienen una sintaxis y una semántica asociada. La *sintaxis* tiene que ver con el tipo de dato que va a contener el atributo es sus especificidades numéricas, alfabéticas o alfa numéricas, entre otras. Por su parte, la *semántica* (una vez verificada

la sintaxis de un dato) es el correcto contenido que debe tener un dato como unidad de información en su significado. En tal sentido, no hay unidades temporales que superen los 12 meses del año, ni hay rotaciones de la Tierra diferentes de día (1) o noche (0). Luego, tanto la sintaxis como la semántica de los datos en la base de datos tienen que ver con su dominio en la base de datos; o el rango de valores sintácticos y semánticos que pueden ser contenidos dentro de los atributos en una base de datos.

El estándar ISO en las instrucciones SQL de la creación (CREATE) y alteración (ALTER) de tablas proporciona acciones para especificar los dominios de los atributos, y así cumplir con las restricciones de su control de integridad sintáctico-semántica.

El cumplimiento de la integridad de los datos al aplicar restricciones de dominio se puede realizar con las cláusulas CHECK y CREATE DOMAIN.

CHECK

La cláusula CHECK permite la definición de restricciones ya sea sobre una tabla completa o para un atributo de la base de datos. La sintaxis de la cláusula es la siguiente:

```
CHEK (CondiciónBúsqueda)
```

Ejemplo 3.1 Restricción de dominio en atributos

Suponga que un Sistema de Registro de Matrimonios tiene la entidad CÓNYUGE y sus atributos asociados son Nit, nombres y apellidos, sexo, estado civil, fecha de nacimiento y dirección electrónica. Crear la tabla de conyuges aplicando restricción de domino a los atributos sexo que sea masculino (M) o femenino únicamente y estado civil que sea casado (C) o soltero (S).

```
CREATE TABLE conyuge
(Nit                  VARCHAR(10),
Nombres_Apellidos     VARCHAR(30),
Sexo                  CHAR NOT NULL CHECK(Sexo IN ('M','F')),
Estado_Civil          CHAR NOT NULL CHECK(Estado_Civil IN
                      ('S','C')),
Fecha                 DATE
E_MAIL                VARCHAR(15));
```

CREATE DOMAIN

La sintaxis de la instrucción que permite definir la restricción de dominio es:

```
CREATE DOMAIN NombreDominio AS [AS] TipoDatos
[DEFAULT opcionPredeterminada]
[CHECK (CondicionBúsqueda)]
```

La instrucción CREATE DOMAIN permite la definición explícita de los dominios de un conjunto de datos que va a ser definidos en la base de datos; se puede utilizar opcionalmente en combinación con la cláusula DEFAULT, en cuyo caso el dominio toma como valor la opción que se define como predeterminada; y en el caso de que se utilice el CHECK, entonces chequea que el dato pertenezca a la condición de búsqueda definida en el chequeo.

Ejemplo 3.2 Definición de restricciones de dominio

En el Sistema de Información Hospitalario (SIH), la entidad EMPLEADO está compuesta por los siguientes atributos: Nit, Nombre, Cargo, Email, salario y Código del hospital al cual pertenece el empleado. Suponer que el hospital tiene los cargos definidos por estratos, así: Auxiliar (A) , Enfermera (E), Médico General (M) y Doctor Especialista (E). Crear la tabla de la entidad EMPLEADO del hospital definiendo la restricción de dominio según los estratos definidos y considerando que el valor por *default* es el médico general (M).

La creación del dominio se hace de la siguiente forma:

```
CREATE DOMAIN TipoCargo AS CHAR  // Dominio definido como carácter
DEFAULT 'M'                      // El default del dominio es M
CHECK (VALUE IN ('A´,'E','M','D')),   // Chequeo de valor del dominio
```

Con base en la creación del dominio, la creación de la tabla de los empleados del hospital es:

```
CREATE TABLE empleado
(Nit                VARCHAR(10),
Nombre_Empleado     VARCHAR(30),
Cargo TipoCargo     NOT NULL      // Nótese que al definir el cargo en
                                  la tabla se utiliza
E_mail              VARCHAR(15), // el nombre del dominio en lugar del
                                  tipo de dato
Salario             DECIMAL (10,2),   // CHAR
Codigo_Hospital     VARCHAR(10));
```

Una vez se ha definido el dominio asociado a un atributo o a una tabla en la base de la base de datos, el dominio se puede eliminar utilizando la instrucción

DROP DOMAIN NombreDominio [**RESTRICT** | **CASCADE**]

En el caso de utilizar RESTRICT, el dominio no puede ser borrado de la base de datos en el caso de que dicho dominio se esté utilizando en la creación de una tabla, vista o aserción. Al utilizar CASCADE, los atributos de las tablas que estén utilizando el dominio se modificarán automáticamente.

Las restrincciones de dominio permiten controlar la sintaxis y la semántica de los contenidos de los atributos de una tabla en la base de datos, como una de las acciones iniciales para mantener la integridad de la base de datos; acciones que se complementan con la integridad de las entidades y la integridad referencial.

3.3.2.3 **Integridad de entidades**

Una vez definidas las entidades en una base de datos, estas se deben poder identificar como únicas para la base de datos. Luego, la identidad sin ambigüedad de una entidad genera el concepto de restricción de integridad de las entidades. Esta restricción implica que ninguna clave primaria de las tuplas de una entidad puede ser nula, y adicionalmente, sus valores deben ser unívocos. La justificación de esta restricción es la siguiente: i) La clave primaria debe servir al gestor de la base de datos para identificar las tuplas de una forma individual en una tabla o relación. ii) En el caso de que la clave primaria sea nula, no se podrían identificar algunas tuplas de la base de datos. iii) La nulidad de la clave primaria en un conjunto de tuplas de una entidad implica la ambigüedad de las tuplas porque no se podrían distinguir debido a la falta de su identidad específica. iv) Si los valores de las claves primarias de las tuplas

no son unívocos, entonces se genera ambigüedad en la especificación de las tuplas, debido a que el gestor no sabría a qué tupla se está refiriendo.

> El estándar ISO soporta las restricciones de integridad de entidades mediante la cláusula PRIMARY KEY en las instrucciones CREATE y ALTER TABLE [6].

Se aclara al lector que la cláusula de creación de tablas en su contenido de sintaxis y significado se desarrollará concretamente en este mismo capítulo en la sección 3.3.3.3.

Ejemplo 3.3 Integridad de entidades

Un Sistema de Información Hospitalario (SIH) tiene como una de sus entidades definidas en la base de datos la tabla PACIENTE, la cual tiene como atributos asociados: Nit, Nombres_Apellidos, Sexo, Fecha_Nacimiento, Dirección, E_mail. Definir la clave primaria de la entidad PACIENTE como no nula al crear la tabla.

```
CREATE TABLE paciente
(Nit                  VARCHAR(10) NOT NULL, // Nit no nullo
 Nombres_Apellidos    VARCHAR(50),
 Sexo                 CHAR NOT NULL CHECK(Sexo IN('M', 'F')),
 Fecha_Nacimiento     DATE,
 Direccion            VARCHAR(50),
 E_mail               VARCHAR(20),
 PRIMARY KEY (Nit));  // Definición de la clave primaria.
```

En la definición de las claves primarias de las entidades, los sistemas de información pueden requerir la definición de claves primarias compuestas. Para la definición de claves principales o primarias compuestas, en la cláusula PRIMARY KEY se colocan cada uno de los atributos que componen la clave separados por comas.

Ejemplo 3.4 Definición de claves primarias compuestas

El Sistema de Información Hospitalario (SIH) requiere la definición de una entidad llamada SALA, a la cual se llevan los pacientes para su curación por tipo de especialidades: Cardiología, Neumología, Psiquiatría... Definir una llave primaria compues-

ta por el Nit del hospital y el código de la sala; los atributos adicionales son el nombre de la sala nombre del coordinador de sala y el número de la cama del paciente.

```
CREATE TABLE sala
(Codigo_Hospital     VARCHAR(5)       NOT NULL,
 Codigo_Sala         VARCHAR(5)       NOT NULL,
 Nombre_Sala         VARCHAR(30),
 Coordinador         VARCHAR(30),
 No_cama             SMALLINT
 PRIMARY KEY (Codigo_Hospital,Codigo_Sala));
```

El cumplimiento de la restricción de la unicidad de cualquiera de las claves se garantiza utilizando la palabra clave UNIQUE. Luego, cada atributo de una tabla definida con UNIQUE debe declararse como NOT NULL; y en este caso, el SQL rechazaría cualquier posibilidad de crear valores duplicados en las claves definidas con dichas especificaciones cuando se realicen las operaciones de inserción INSERT o modificación UPDATE a la base de datos.

Ejemplo 3.5 Unicidad de claves

Para el SIH crear la entidad HOSPITAL con los atributos Nit del hospital, Nombre del centro hospitalario, Dirección, E_mail y teléfono del hospital, con la característica de que la clave primaria es única.

```
CREATE TABLE hospital
(Codigo_Hospital        VARCHAR(10)   NOT NULL,
 Nombre                 VARCHAR(30),
 Dirección              VARCHAR(30),
 E_mail                 VARCHAR(20),
 Telefono               VARCHAR(10),
 PRIMARY KEY (Codigo_Hospital),
 UNIQUE (Codigo_Hospital));
```

Con base en lo anterior, la *clave primaria* es una clave única que define las características de cada una de las tuplas que pertenecen a una tabla en la base de datos. La

correcta definición de las claves primarias en una tabla en su no nulidad y unicidad permite el cumplimiento de la integridad referencial en una base de datos.

3.3.2.4 **Integridad referencial**

Se debe tener en cuenta que la utilización de la restricción de integridad de entidades solo se utiliza en las relaciones individuales o tablas. Dado que los esquemas de las bases de datos están compuestos por varias relaciones (tablas), se requiere definir el concepto de "integridad referencial".

La *integridad referencial* es la restricción colocada entre dos relaciones de una base de datos, la cual permite mantener la consistencia entre las tuplas de las relaciones mencionadas. Suponga que un esquema de la base de datos tiene definidas dos entidades: la entidad "padre", identificada como PADRE, y la entidad "hijo", identificada como HIJO. Adicionalmente, tenga en cuenta que la entidad HIJO tiene asociadas tanto la clave primaria (PRIMARY KEY) como una clave o conjunto de claves externas (FOREIGN KEY). La clave externa en una tabla tiene la funcionalidad de enlazar cada tupla de la tabla HIJO, que contiene la clave externa, con cada tupla o fila de la tabla PADRE, que contiene el valor correspondiente de la clave candidata en mención.

El estándar ISO en las cláusulas utilizadas para la creación (CREATE) y modificación (ALTER) de tablas soporta la integridad referencial mediante la sentencia FOREIGN KEY, cuya sintaxis es la siguiente:

```
FOREIGN KEY (NombreCampo) REFERENCES NombredelaTabla
(NombredelCampo)
```

Donde el Nombre de la Tabla padre a la cual hace referencia la clave foránea en la tabla hija.

La creación de las tablas representativas de la entidad HOSPITAL y de la entidad EMPLEADO son bases para entender la integridad referencial del SIH en el siguiente sentido. La tabla EMPLEADO representa la entidad HIJA; porque es precisamente el empleado quien depende de la entidad PADRE, en este caso el HOSPITAL, por cuanto es el hospital la institución que contrata los servicios de sus empleados. Luego, la creación de la tabla hija EMPLEADO se puede completar para que haga referencia con su clave externa a la tabla padre u HOSPITAL mediante la utilización de la clave externa de la tabla "empleado".

Ejemplo 3.6 Integridad referencial

Redefinir la creación de la tabla de EMPLEADO del SIH, para que como tabla "hija" hacer referencia a la tabla padre HOSPITAL.

Se transcribe la creación de la tabla "empleado" para facilidad del lector:

```
CREATE TABLE empleado
   (Nit                 VARCHAR(10),
    Nombre_Empleado     VARCHAR(30),
    Cargo TipoCargo     NOT NULL        // Notese que al
                        definir el cargo en la tabla se utiliza
    E_mail              VARCHAR(15),    // el nombre del
                        dominio en lugar del tipo de dato
    Salario             DECIMAL (10,2), // CHAR
    Codigo_Hospital     VARCHAR(10));
```

La redefinición de la tabla EMPLEADO es la siguiente:

```
CREATE TABLE empleado   // Redefinición de empleado como tabla
                        HIJA
   (Nit                 VARCHAR(10),
    Nombre_Empleado     VARCHAR(30),
    Cargo TipoCargo     NOT NULL
    E_mail              VARCHAR(15),
    Salario             DECIMAL (10,2),
    Codigo_Hospital     VARCHAR(10)
    PRIMARY KEY         (Nit),
    FOREIGN KEY         (Codigo_Hospital)
    REFERENCES          hospital(Codigo_Hospital));

CREATE TABLE hospital   // Tabla PADRE
   (Codigo_Hospital     VARCHAR(10)     NOT NULL,
    Nombre              VARCHAR(30),
    Direcccion          VARCHAR(30),
    E_mail              VARCHAR(20),
    Telefono            VARCHAR(10),
    PRIMARY KEY (Codigo_Hospital),
    UNIQUE (Codigo_Hospital));
```

La integridad referencial en el ejemplo desarrollado implica que si la clave externa en la tabla "hija" (en este caso el Codigo_Hospital) tiene un valor, dicho valor necesa-

riamente debe existir en la tabla "padre" y, consecuentemente, debe hacer referencia a una tupla válida y existente de la tabla "padre".

En la figura 3.1 se presenta una instancia de interrelación en el esquema del SIH para las tablas HOSPITAL y EMPLEADO.

HOSPITAL

CÓDIGO_ HOSPITAL	NOMBRE	DIRECCIÓN	E_MAIL	TELÉFONO
10	Departamental	...	...	...
20	San Juan	...	...	...
40	San José	...	...	...
50	Universitario	...	...	...
...	...	...	...	...

EMPLEADO

NIT	NOMBRE_EMPLEADO	CARGO	E_MAIL	SALARIO	CODIGO_HOSPITAL
13...	Arteaga Luis	D	...	...	50
18...	Becerra María	A	...	...	40
72...	Camargo Andrés	D	...	...	20
77...	Ural Samuel	E	...	...	10
...	...	...	...	...	...

Figura 3.1 Instancia del esquema de la base de datos del SIH en su integridad referencial

3.3.2.5 **Restricciones generales**

La utilización de las restricciones desarrolladas anteriormente a fin de asegurar la integridad de la base de datos se complementa con el cumplimiento de la definición de las restricciones generales que tiene una base de datos con relación al funcionamiento real de la empresa. En este sentido, aceptando que es ilógico en el Sistema de Información Hospitalario (SIH) que haya un empleado que no pertenezca a un hospital, o un salario definido alfabéticamente para una liquidación, hay condiciones restrictivas generales que ha de cumplir la base de datos. Tal es el caso cuando un doctor está encargado de muchos pacientes al mismo tiempo por su capacidad en cobertura para poder atenderlos. El estándar ISO permite la definición de este tipo

de restricciones para la base de datos utilizando la cláusula CREATE ASSERTION, cuya sintaxis se presenta a continuación:

```
CREATE ASSERTION NombredelaAserción
CHECK (CondiciondeBúsqueda)
```

Una *aserción* es una cláusula del SQL que hace posible satisfacer una condición mediante la utilización del CHECK. Las acciones que se pueden activar en la base de datos con la cláusula ASSERTION pueden hacer detener la operación de una tabla o inclusive toda la base de datos.

Ejemplo 3.7 Chequeo de restricciones generales en una base de datos con la cláusula ASSERTION

Definir la restricción general para el SIH de un hospital que no pueda tener más de 500 empleados, con base en las tablas de empleados y hospitales dadas previamente en el SIH.

```
CREATE ASSERTION HospitalSobrecargado
    CHECK
    (NOT EXIST
                (SELECT Nit
                FROM empleado
                GROUP BY Codigo_Hospital
                HAVING CONUNT (*) > 500 ));
```

En las secciones siguientes de este capítulo se profundizará sobre la creación y alteración de tablas a través del desarrollo en concreto de la creación de tablas (CREATE TABLE) y modificación de tablas (ALTER TABLE).

3.3.3 Definición de datos de la base de datos

3.3.3.1 Creación de la base de datos

Los gestores de bases proporcionan acceso a la base de datos a través del Sistema de Gestión de la Base de datos (SGBD). El SGBD facilita dos lenguajes, el DDL y el DML.

El DDL (en inglés *Data Definition Language* o lenguaje de definición de datos) permite definir la base de datos.

El DML (en inglés *Data Manipulation Laguage*) es un lenguaje de manipulación de datos, el cual permite a los usuarios mantener la base de datos en sus acciones de inserción, edición consulta y borrado de los datos creados en la base de datos.

El DDL, por lo tanto, permite la creación y el borrado de objetos de la base de datos. Los objetos de la base de datos, sin hacer una enumeración exhaustiva, son: los esquemas, los dominios, las tablas, las vistas y los índices de la base de datos.

3.3.3.2 **Creación del esquema de la base de datos**

La creación de la base de datos depende del gestor que se esté manejando. En los sistemas de bases de datos complejos y al servicio de grandes organizaciones (compañías financieras o empresas petroleras multinacionales) al utilizar estos sistemas multiusuarios la gestión para crear una base de datos está centralizada en el DBA (*Data Base Administrator*) o el Administrador de la Base de Datos. Por su parte, en sistemas de base de datos del tipo monousuario o instaladas en computadores personales o al servicios de pequeñas organizaciones, en la mayoría de los casos el gestor de la base de datos en el momento de la instalación permite crear una base de datos predeterminada; dándole posteriormente al usuario la posibilidad de generar la creación de otras bases de datos.

De acuerdo con el estándar ISO de lo complejo a lo simple con relación a la tecnología de bases de datos existen inicialmente en la creación de la base de datos tres categorías: los entornos, los catálogos y los esquemas.

Un *entorno* de base de datos es un conjunto de datos organizado compuesto por las relaciones y otros objetos que componen la base de datos.

El sistema de organización de la base de datos dentro del entorno de la misma necesita de uno o más *catálogos*, los cuales almacenan la descripción de los elementos de datos que componen la base de datos.

Los catálogos contienen un conjunto de *esquemas*. Formalmente, ... *un esquema es una colección nominada de objetos de base de datos que están relacionados entre sí de alguna manera (los objetos de la base de datos están descritos en un esquema u otro)* [6].

La sintaxis para la creación de un esquema de la base de datos tiene el siguiente formato:

```
CREATE SCHEMA [Nombre] | AUTHORIZATION IdentificacióndelCreador]
```

Ejemplo 3.8 Crear el esquema de una base de datos identificado como MyData creado por la identificación de usuario Jamin

```
R/. CREATE SCHEMA MyData AUTHORIZATION Jamin;
```

El mantenimiento del entorno y los catálogos de la base de datos en la administración del espacio de almacenamiento en el servidor (base de datos multiusuario) o en el disco duro (base de datos monousuario), dada la creación de varios esquemas de bases de datos requiere un comando inverso al de creación de la base de datos; luego, una vez creado un esquema en la base de datos, este se puede cancelar utilizando la función DROP, que en SQL tiene la siguiente sintaxis:

```
DROP SCHEMA Nombre [RESTRICT | CASCADE]
```

Al ejecutar el comando SQL DROP para cancelar un esquema de base de datos, la opción que opera de una forma predeterminada es RESTRICT; ello implica que el contenido del esquema de la base de datos debe estar vacío, porque de lo contrario el comando causará un error. En el caso de utilizar la opción CASCADE, se eliminarán de la base de datos cada uno de los objetos en el orden en que fueron predeterminados.

3.3.3.3 Creación de tablas

Teniendo como base la fase del diseño lógico y físico de la base de datos, el M E-R permitió diseñar gráficamente la lógica de las componentes sintáctico-semánticas de la base de datos. Por su parte, el Modelo Relacional permite identificar los conceptos de relación, atributo y dominio.

Luego, teniendo en cuenta que el Modelo Relacional se basa en el concepto matemático de relación, la relación se representa físicamente en la base de datos en forma de una tabla. Las tablas en una base de datos son entonces estructuras de datos relacionales.

Formalmente, una **relación** en una base de datos es una estructura en forma de tabla bidimensional compuesta por columnas y filas. Las columnas de la tabla en la base de datos representan los **atributos** nominados en una relación; y las filas o líneas de la tabla es el contenido de informaciones de la relación, la cual recibe el nombre de **tupla**. La ejemplificación de los conceptos de relación para un Sistema Integrado de Manufactura (SIM) se muestra a continuación.

La descripción de los atributos de la tabla de PRODUCTOS del SIM se presenta en la figura 3.2.

```
/* Descripción de la relación productos en la base de datos
del Sistema Integrado de Manufactura - SIM */
DESCRIBE productos;
```

Name	Null	Type
COD_PRODUCTO	NOT NULL	VARCHAR(50)
NOMBRE_PRODUCTO	NOT NULL	VARCHAR(50)
TIPO_PRODUCTO	NOT NULL	CHAR(2)
COSTO_PRODUCTO	NOT NULL	NUMBER(15,2)
CANTIDAD_PRODUCTOS	NOT NULL	NUMBER(8,2)
NO_PROYECTO	NOT NULL	VARCHAR(50)

Figura 3.2 Descripción de los atributos de la tabla de PRODUCTOS del SIM

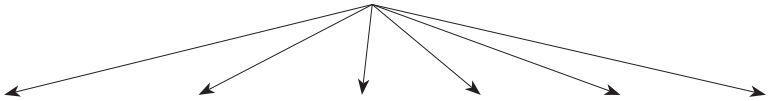
La instanciación de la tabla de PRODUCTOS se presenta en la figura 3.3.

```
/* Contenido de las tuplas de la tabla de PRODUCTOS del SIM */
SELECT * FROM productos;
```

Figura 3.3 Instancia de la relación PRODUCTOS del SIM

Relación o tabla de Productos:

ATRIBUTOS DE LA RELACIÓN



COD_PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
100-100-101-000	PORTATIL NGP	T	800000	20	P001BA0001
100-100-101-001	PROCESSOR XTS	P	400000	10	P002NY0002
100-100-101-002	INTERFACES MULTIPROPÓSITO	T	200000	3	P003MI0003
100-100-101-003	PHOTONIC CONTOLLED ONE SIM	T	20000	1	P004SE0004
100-100-101-004	DRIVE CONTROLLERS SIM	P	25000	2	P005BA0005
100-100-101-005	CONCECTOR RXL NGN	T	1800000	1	P006SE0007
100-100-101-006	TWO BRAIN MOTOR	T	3000000	1	P006NY0008
100-100-101-007	MOTOR DEDUCTIVE	T	1000000	20	P005BA0009
100-100-101-008	WEB 10.4 NGWEB	T	1800000	20	P005BA0010
100-100-101-009	MOTOR AIRPL. XT CONTROL TAB.	T	1000000	20	P005BA0011
100-100-101-010	MOTOR WEM DB 2.0	P	500000	4	P005BA0012
100-100-101-011	OPS - 2018	T	175000	8	P006BO0013
100-100-101-012	ROUT NO.1435	T	30000	9	P007TA0014
100-100-101-013	BODY TRACKING XC-XB-XD	T	180000	12	P008TA0015
100-100-101-014	OPTICAL COMPUTER	T	570000	5	P009TA0014
100-100-101-015	CHIP RSA	MT	40000	100	P010BA0015
100-100-101-016	LPS - NEW GENERATION	MT	80000	57	P020BA0016
100-100-101-017	CODY SYS. UVXA	MT	50000	30	P011BO0017
100-100-101-018	OPTICAL CONTROL FBA-12	T	100000	10	P012BO0222
100-100-101-019	CAD GRAPHIC SYSTEMS	P	5000	1	PA11TA1111

Clave Primaria

Clave Secundaria

Teniendo en cuenta que la relación en la base de datos es una tabla, la cual está compuesta de atributos, el concepto de "dominio" en el marco del Modelo Relacional es muy importante. El **dominio** es el conjunto de valores que pueden contener los atributos de una relación. La definición del dominio concreta el tipo de datos que va a contener el atributo dentro de la relación; y consecuentemente, especifica los contenidos de la tuplas de la tabla o relación a través de la especificación de cada uno de los atributos de la tabla.

Los dominios de los atributos de la tabla relacional PRODUCTOS se muestran en la tabla 3.1.

N°	Atributo	Nombre del dominio	Significado	Definición del Dominio
1	COD_PRODUCTO	Código del Producto	Conjunto de códigos que pueden asumir los productos.	Tipo carácter variable de longitud 50.
2	NOMBRE_PRODUCTO	Nombre del Producto	Conjunto de todos los nombres de los proyectos.	Tipo carácter variable de amplitud 50.
3	TIPO_PRODUCTO	Tipo del producto	Tipo del producto en el proceso de producción P= en proceso, T= Terminado, …	Carácter de longitud constante igual a dos.
4	COSTO_PRODUCTO	Costo de producto	Costo de fabricación del producto.	Numérica de longitud 15 con dos decimales.
5	CANTIDAD_PRODUCTOS	Cantidad de producto	Conjunto que contiene la cantidad de productos según el tipo.	Numérica de longitud 8 con dos decimales.
6	NO_PROYECTO	Número del proyecto	Número del proyecto al cual pertenece el producto en su proceso de fabricación.	El tipo es carácter de longitud 50.

El grado de una relación corresponde al número de atributos de la relación; en tal sentido, el grado de la relación PRODUCTOS es igual a seis.

La **cardinalidad** de una relación es el número de tuplas que contiene la tabla. Por lo tanto, la cardinalidad de la relación es igual a 20.

En la sección anterior se desarrolló la creación de un esquema de la base de datos, o lo que es equivalente, la estructura de la base de datos. Con base en el Modelo Rela-

cional se deben crear las estructuras de las tablas con el fin de formar las relaciones que se necesiten construir en la base de datos.

La creación de relaciones en la base de datos se hace con el comando SQL CREATE TABLE, el cual tiene en su sintaxis en siguiente formato específico:

```
CREATE TABLE Nombre_de_la_Tabla:
```

Ejemplo 3.9 Creación de la tabla de PROFESORES del SIAC

Suponer un Sistema de Información de Información Académico (SIAC) compuesto por un conjunto de entidades: ESTUDIANTES, PROFESORES, PROGRAMAS, MATRÍCULA, ASIGNATURAS y PÉNSUM.

La semántica o significado del conjunto de entidades en el SIAC es:

- *Estudiantes*: conjunto de personas que aplican a un proceso formativo en una institución educativa.
- *Profesores*: conjunto de personas a cuyo cargo está el proceso de enseñanza de unos saberes pertenecientes a un área del conocimiento.
- *Programas*: conjunto de áreas de conocimiento cuyas enseñanzas están organizadas en una estructura curricular, por lo general estructuradas en asignaturas, problemas o proyectos.
- *Matrícula*: proceso de registro de un conjunto de asignaturas de un período lectivo de un SIAC.
- *Asignaturas*: es el conjunto de enseñanzas específicas a un área del conocimiento, a través de cuya organización se cumple el proceso de enseñanza-aprendizaje dentro de un programa académico.
- *Pénsum*: es la estructura curricular que soporta el programa académico, la cual está en la mayoría de los casos organizada por semestres.

Crear la relación PROFESORES del SIAC.

La creación de la tabla que representa la relación PROFESORES en la base de datos SIAC se muestra en la figura 3.4.

```
CREATE TABLE profesores
   (
      Nit                  NUMBER (10),
      Primer_Apellido      VARCHAR(20),
      Segundo_Apellido     VARCHAR(20),
      Primer_Nombre        VARCHAR(20),
      Segundo_Nombre       VARCHAR(20),
      Salario              DECIMAL(10,2)
   );
```

Figura 3.4 Creación de la relación PROFESORES en el SIAC

Con relación al ejemplo anterior, si bien es cierto que cumple con la creación de la tabla a través del gestor de bases de datos SQL, no tiene en su estructura la definición de las claves primarias, las clave foráneas, ni menos las condiciones de chequeo de los atributos de los campos que se pueden implementar. Un formato más completo de la instrucción CREATE se presenta a continuación, el cual se muestra en [6]:

```
CREATE TABLE Nombre_de_la_Tabla
   ({Nombre_de_la_Columna Tipo_de_Dato [NOT NULL] [UNIQUE]
   [DEFAULT opciónPredeterminada][CHECK (condiciónBúsqueda)] […]}
   [PRIMARY KEY (listaColumnas),]
   {UNIQUE (listaColumnas)][…]}
   {FOREIGN KEY (listaColumnasClaveExterna)
   REFERENCES nombreTablaPadre [(listaColumnasClavesCandidatas)]
      [MATCH {PARTIAL|FULL}
      [ON UPDATE acciónReferencial]
      [ON DELETE acciónReferencial]][…]}
   {[CHECK (condicionBusqueda)] […]});
```

CREATE: La instrucción CREATE TABLE permite la creación de una tabla en la base de datos identificada con el Nombre_de_la_Tabla, el cual debe cumplir con la reglas para definir identificadores en SQL.

Nombre_de_la_Columna: Una vez se identifica la creación de la tabla, se especifican los atributos de la relación, definiendo la identidad de las columnas de la tabla, las cuales deben cumplir con la escritura de los identificadores SQL.

Tipo_de_Dato: Definida la identidad del atributo (o el Nombre_de_la_Columna), se debe definir el tipo de datos que contendrá la columna, para lo cual se utiliza la sección de Definición de Datos de este texto.

NOT NULL: Se utiliza para aquellas columnas que necesariamente tienen que tener datos que no pueden quedar en nulos, como es el caso de la columnas que contienen claves externas.

UNIQUE: Se utiliza para garantizar la unicidad de las claves alternativas; y se debe tener en cuenta que una vez se define la cláusula UNIQUE en los atributos de la tabla, entonces esta se debe necesariamente acompañar de la cláusula NOT NULL.

DEFAULT: Es útil para especificar un valor predeterminado de una columna.

CHECK: Permite definir una restrincción sobre una columna o una tabla sujeta a una condición de búsqueda.

PRIMARY KEY: Esta opción de la cláusula CREATE TABLE es de la mayor importancia, pues permite definir la columna o conjunto de columnas que constituyen la clave principal de la una tabla. A nivel predeterminado, el gestor de la base de datos asume la opción NOT NULL en las columnas de la clave principal en el caso de que no se especifiquen. La definición de la clave primaria es única; por lo tanto, solo se permite definir una sola cláusula PRIMARY KEY en la construcción de cada tabla de la base de datos. Los gestores de bases de datos garantizan la unicidad de la clave primaria de una tabla, asegurando en este sentido un solo valor (unívoco) y la no nulidad a fin de realizar las operaciones de inserción y modificación de las tablas, permitiendo la integridad de las entidades.

FOREIGN KEY: Es útil para la especificación de una clave externa. Esta clave se debe concretar en la tabla "hija" de una tabla "padre". La cláusula sirve para dar cumplimiento a la integridad referencial de la base de datos. Se cumple con la integridad referencial cuando una clave externa en la tabla hija hace referencia a una clave primaria en la tabla padre; y por lo tanto, dicha llave externa hace referencia necesariamente a una tupla o fila que existe y es válida dentro de la tabla padre.

La especificación de la clave foránea en sus componentes se denota como:

- *Lista de Columnas de la Clave Externa*: Es la columna o el conjunto de columnas de la tabla que componen la clave externa.
- *REFERENCES*: En la referencia se indica la tabla padre y la clave primaria de la tabla padre que especifica la tupla en el padre con la cual está relacionada la tabla hija. En el caso de que se obvien la lista de las columnas candidatas, entonces se supone que la clave externa corresponde a la clave principal en la tabla padre. Es lógico suponer que para que se cumpla la integridad en la referenciación, la tabla padre debe estar creada y, consecuentemente, en ella debe estar definida una clave primaria.
- *MATCH*: Sirve para definir las restricciones de valores nulos en la tabla externa. La tipologías de estas restricciones pueden ser tres: i) MATCH FULL: significa que los contenidos de los valores de los datos de la clave externa deben tener todos ellos valores que no sean nulos o, en su defecto, ser todos nulos. ii) MATCH PARTIAL: se utiliza en el caso de que los componentes de la clave externa deban ser todos nulos o cuando la existencia de al menos una fila en la tabla padre pudiera *satisfacer si se asignaran los valores correctos a los restantes valores nulos* [6].
- *ON UPDATE*: Se utiliza para especificar las acciones que el gestor de la base de datos debe realizar cuando haya una actualización de alguna de las claves candidatas de la tabla padre correspondientes con la clave foránea o externa de la tabla denominada como hija.
- *ON DELETE*: Permita especificar las acciones que se deben seguir en el caso de borrar una fila de la tabla padre que contenga una llave candidata que corresponda con una clave foránea de la tabla hija. Las acciones que se pueden tomar en la integridad referencial son las siguientes: i) CASCADE: en este caso se borra la tupla o fila de la tabla padre y ejecuta el borrado automático de las tuplas en la tabla hija; la acción CASCADE es recurrente de tabla a tabla; se hace la misma acción en el caso de que la tupla borrada esté relacionada con otra tabla, entonces se borran automáticamente las tuplas de la tabla "nieta". ii) SET NULL: en este caso se borra la tupla de la tabla padre, asignándoles valores que son NULL (nulos) a los valores de la clave foránea de la tabla hija. iii) SET DEFAULT: el gestor de la base de datos borra la tupla de la tabla padre; y en el caso de que se haya especificado un valor de *default*, entonces se asigna a cada una de las componentes de la clave externa el valor que se haya predeterminado. iv) NO ACTION: el gestor en este caso rechaza la operación

de borrado de la tabla padre, y se debe tener en cuenta que esta es la opción predeterminada en el caso de no especificar ninguna acción.

– *CONSTRAINT*: Al igual que la cláusula CHECK, permite la definición de restricciones adicionales en el momento de la creación de la tabla. En el caso de que dicha creación se utilice con restricción de columna, entonces el CHECK solo puede hacer referencia a la columna que está siendo definida. La comprobación del cumplimiento del CONSTRAINT se puede evaluar después de la ejecución de cada proposición SQL, o cuando se termine la transacción en la cual esté encapsulada la instrucción SQL.

Creación de tablas con claves primarias y foráneas

La creación de tablas con claves primarias se hace utilizando PRIMARY KEY, a la cual se le asocia el atributo de tabla que se define con las características de ser la llave primaria. Las tablas que en la base de datos son definidas con claves primarias son las que pueden ser las denominadas tablas "padres" de otras tablas en la base de datos.

La cláusula FOREIGN KEY se utiliza para especificar el atributo de la tabla con la característica de ser la clave externa en una tabla del tipo "hija/hijo". Clave externa que es un atributo o columna de la tabla que debe necesariamente referenciar la clave principal de la tabla "padre".

Ejemplo 3.10 Creación de la tabla de los elementos compuestos del SPOAPU

Un Sistema de Presupuestos de Obra y Análisis de Precios Unitarios (SPOAPU) está compuesto por un conjunto de entidades, que sin hacer una enumeración exhaustiva son: PROYECTO, ELEMENTOS SIMPLES, ELEMENTOS COMPUESTOS y CONSTRUCTORA.

El significado del conjunto de entidades del SPOAPU es el siguiente:

– *Proyecto*: es el plan de actividades de construcción de una obra civil, consistente en un edificio, puente, represa o carretera en las áreas de Arquitectura o Ingeniería Civil.

- *Elementos simples*: Es el conjunto de elementos primarios o esenciales que componen un proyecto de ingeniería de obra civil. Son elementos simples: la arena, el cemento y el agua.
- *Elementos compuestos*: Es la agrupación de elementos simples para formar los elementos compuestos en una obra de civil. Son elementos compuestos las columnas, las placas o pisos de un edificio; y estos, a su vez, están compuestos de mortero y hierro; donde el mortero está compuesto de la integración de los elementos simples la arena, el cemento y el agua.

Con base en la definición del SPOAPU, construir las relaciones de los elementos compuestos y los elementos simples del sistema, definiendo las llaves primarias de ambas relaciones; y considerando que la relación de los elementos compuestos es el padre de la relación de los elementos simples que es la relación hijo del sistema.

La creación de la tabla de elementos compuestos se muestra en la figura 3.5, en la cual se observa la ejemplificación de la creación de la clave primaria de la tabla.

```
/* Creación de la tabla elementos compuestos del Sistema de
Presupuestos de Obra y Análisis de Precios Unitarios */

/* con ejemplificación de la creación de llaves primarias */

CREATE TABLE compuestos
   (  codigo_elemento      numeric(10)     not null,
      nombre_elemento      VARCHAR(50)     not null,
      unidad_medida        VARCHAR(5)      not null,
      cantidad             numeric(10)     not null,
      valor                decimal(10,2)   not null,
      PRIMARY KEY (codigo_elemento) /* Especificación de la
                                    llave primaria */
   ); /* TABLA PADRE */
```

Figura 3.5 Creación de la tabla de los elementos compuestos del SPOAPU

La construcción de la tabla hija del SPOAPU se presenta en la figura 3.6, en la cual se especifica: la llave primaria de la tabla de elementos simples, su llave foránea y la referencia a la tabla padre que contiene la clave foránea; o sea, la pertenencia del elemento simple al compuesto.

```
/* Creación de la tabla elementos simples del Sistema de
Presupuestos de Obra y Análisis de Precios Unitarios con
ejemplificación de la creación de llaves primarias */

/* Tabla HIJA */

CREATE TABLE simples
   ( codigo_elemento_simple   numeric(10)    not null,
     nombre_elemento          VARCHAR(50)    not null,
     unidad_medida            VARCHAR(5)     not null,
     valor_unitario           decimal (10,2) not null,
     codigo_elemento          numeric(10)    not null,
     PRIMARY KEY (codigo_elemento_simple),
     FOREIGN KEY (codigo_elemento)           /*
   Especificación de la llave foránea de la tabla */
     REFERENCES compuestos (codigo_elemento) /* Referencia
   a la tabla de elementos compuestos */
   );
```

Figura 3.6 Creación de la tabla de los elementos simples del SPOAPU

3.3.3.4 Edición de tablas

La edición de tablas, de acuerdo con el estándar ISO, se realiza utilizando la cláusula SQL ALTER TABLE, la cual permite la modificación de una tabla creada en sus características de definición. La estructura de la sintaxis de esta cláusula se presenta a continuación [6]:

```
ALTER TABLE Nombre_de_Tabla
[RENAME TO Nombre_Nueva_tabla]
[ADD|COLUMN] Nombre_de_Columna Tipo_de_Dato [NOT NULL][UNIQUE]
[DEFAULT Opción_Predeterminada][CHECK (Condición_de_Búsqueda)]]
[DROP [COLUMN] Nombre_de_la_Columna [RESTRICT|CASCADE]]
[ADD [CONSTRAINT [Nombre_de_la_Restrincción]]
                definiciónRestrincciónTabla]
[DROP CONSTRAINT NombreRestrincción [RESTRICT|CASCADE]]
[ALTER|COLUMN ] SET DEFAULT opciónPredeterminada]
[ALTER|COLUMN] DROP DEFAULT]
```

Ejemplo 3.11 Creación de tablas base en un Sistema de Inventarios (SI)

Construir un Sistema de Inventarios (SI) que está soportado por un conjunto de entidades. De las entidades en mención se deben crear las tablas relacionales correspondientes a la entidad PROVEEDOR y la entidad PRODUCTO, con sus respectivas claves primarias. Siendo la entidad PROVEEDOR la tabla "padre" y la entidad PRODUCTO la tabla "hija".

La creación de la tabla PROVEEDOR se presenta en la figura 3.7.

```
/* Creación de la tabla PROVEEDOR del Sistema de
Inventarios - SI */
CREATE TABLE proveedor
   (
      Codigo_Proveedor        NUMERIC(10)       NOT NULL,
      Identidad_Empresa       VARCHAR(30)       NOT NULL,
      Nombre_Representante    VARCHAR(30)       NOT NULL,
      Email_Proveedor         VARCHAR(20)       NOT NULL,
      Email_Representante     VARCHAR(20)       NOT NULL,
      Tel_Proveedor           NUMERIC(10)       NOT NULL,
      PRIMARY KEY(Codigo_Proveedor)
   );
```

Figura 3.7 Tabla proveedor del SI

De la creación de la tabla PROVEEDOR se debe notar que i) La clave primaria es de naturaleza numérica, y todos los campos de la tabla están definidos como no nulos. ii) Por otra parte, la tabla no tiene referencia a una tabla "padre", o sea que no es una tabla "hija".

La creación de la tabla PRODUCTO del SI se presenta en la figura 3.8.

```
/* Creación de la tabla PRODUCTO del Sistema de Inventarios - SI */
CREATE TABLE producto
   (
      Codigo_Producto         NUMERIC(10)        NOT NULL,
      Nombre_Producto         VARCHAR(50)        NOT NULL,
      Tipo_Producto           VARCHAR(5)         NOT NULL,
      Precio_Compra           DECIMAL(10,2)      NOT NULL,
      Precio_Venta            DECIMAL(10,2)      NOT NULL,
      Codigo_Proveedor        NUMERIC(10)        NOT NULL,
      PRIMARY KEY (Codigo_Producto),
      FOREIGN KEY (Codigo_Proveedor)
      REFERENCES proveedor (Codigo_Proveedor)
   );
```

Figura 3.8 Tabla producto del SI

De la tabla del PRODUCTO SI se debe tener en cuenta que i) Es una tabla "hija" por cuanto tiene una llave foránea que referencia a la tabla PROVEEDOR en su campo de llave primaria, la cual es Codigo_Proveedor. ii) Al igual que la tabla anterior, todos los campos son definidos como no nulos.

Alteración del nombre de la tabla

La cláusula RENAME TO en SQL da la opción al usuario de modificar el nombre de una tabla creada.

Ejemplo 3.12 Modificación del nombre de una tabla en un Sistema de Inventarios

Modificar el nombre de la tabla PROVEEDOR a PROVEEDORES_UNO del Sistema de Inventarios.

El nombre modificado de la tabla se presenta en la figura 3.9.

```
/* Modicación del nombre de la tabla PROVEEDOR del SI */
ALTER TABLE proveedor
RENAME TO PROVEEDORES_UNO;
```

Figura 3.9 Alteración del nombre de la tabla proveedor del SI

Alteración del número de columnas de una tabla creada

La cláusula ADD permite la modificación de una tabla en su número de columnas, o añadir una nueva columna a una tabla previamente creada.

Ejemplo 3.13 Modificar la tabla PROVEEDORES_UNO del SI

Modificar la tabla de PROVEEDORES_UNO del Sistema de Inventarios, agregándole una columna a la tabla llamada Coeficiente de cumplimiento del Proveedor, del tipo numérico de longitud 3, para que tome valores en el intervalo [0.100].

La descripción inicial de la tabla de PROVEEDORES_UNO se muestra en la figura 3.10.

```
/* Descripción de los atributos iniciales de la tabla
PROVEEDORES_UNO del SI */
DESCRIBE proveedores_uno;
```

Name	Null?	Type
CODIGO_PROVEEDOR	NOT NULL	NUMBER(10)
IDENTIDAD_EMPRESA	NOT NULL	VARCHAR(30)
NOMBRE_REPRESENTANTE	NOT NULL	VARCHAR(30)
EMAIL_PROVEEDOR	NOT NULL	VARCHAR(20)
EMAIL_REPRESENTANTE	NOT NULL	VARCHAR(20)
TEL_PROVEEDOR	NOT NULL	NUMBER(10)

Figura 3.10 Descripción de los atributos de la tabla PROVEEDORES_UNO del SI

La ejecución de alteración de la tabla PROVEEDORES_UNO está en la figura 3.11.

```
/* Alteración del número de columnas de la tabla
PROVEEDORES_UNO del SI*/

/* Se le agrega a la tabla la columna de Coeficiente de
Cumplimiento del Proveedor identificada como Coeficiente_
Proveedor */

ALTER TABLE proveedores_uno

ADD Coeficiente_Proveedor NUMERIC (3);
```

Figura 3.11 Adición de columnas a la tabla PROVEEDORES_UNO del SI

La descripción del número de atributos alterados de la tabla PROVEEDORES_UNO del SI se presenta en la figura 3.12.

```
/* Descripción de la tabla alterada de proveedores_uno del SI */

DESCRIBE proveedores_uno;
```

Name	Null?	Type
CODIGO_PROVEEDOR	NOT NULL	NUMBER(10)
IDENTIDAD_EMPRESA	NOT NULL	VARCHAR(30)
NOMBRE_REPRESENTANTE	NOT NULL	VARCHAR(30)
EMAIL_PROVEEDOR	NOT NULL	VARCHAR(20)
EMAIL_REPRESENTANTE	NOT NULL	VARCHAR(20)
TEL_PROVEEDOR	NOT NULL	NUMBER(10)
COEFICIENTE_PROVEEDOR		**NUMBER(3)**

Figura 3.12 Tabla PROVEEDORES_UNO con la columna añadida

Cancelación del número de columnas de una tabla

La eliminación de las columnas de una tabla se realiza mediante la instrucción DROP COLUMN.

Ejemplo 3.14 Eliminación de una columna de una tabla

Eliminar el Coeficiente de cumplimiento del Proveedor de la tabla PROVEEDORES_UNO del SI.

La acción de cancelar la columna de la tabla PROVEEDORES_UNO se muestra en la figura 3.13.

```
/* Eliminación del coeficiente de cumplimiento del
proveedor de la tabla de PROVEEDORES_UNO del SI */
ALTER TABLE proveedores_uno
DROP COLUMN coeficiente_proveedor;
```

Figura 3.13 Eliminación de una columna de una tabla

Alteración de múltiples columnas de una tabla

La alternación de múltiples columnas de una tabla se puede realizar con la sintasis SQL:

```
ALTER TABLE Nombre_de_la_Tabla
   ADD
   (
      Nombre_Columna_1      Definición_Tipo_Columna_1,
      Nombre_Columna_2      Definición_Tipo_Columna_2,
      …
      Nombre_Columna_1      Definición_Tipo_Columna_1
   );
```

Ejemplo 3.15 Inserción de múltiples columnas a una tabla

Renombrar la tabla PRODUCTO del SI con el nombre PRODUCTOS_UNO, y agregar los campos de stock de inventario y detalle del producto.

La alteración de la tabla renombrada se presenta en la figura 3.14; y la inserción de las columnas del nivel de los inventarios y el campo de detalles de los productos se muestran en la figura 3.15.

```
ALTER TABLE producto
RENAME TO productos_uno;
```

Figura 3.14 Alteración del nombre de la tabla PRODUCTO del SI

```
ALTER TABLE productos_uno
ADD
(
   stock              NUMERIC(5),
   detalle            VARCHAR(100)
);
```

Figura 3.15 Adición de múltiples columnas a una tabla

Modificar características de columnas de tablas

La modificación de los nombres de las columnas de una tabla creada se hace con la instrucción MODIFY, de acuerdo con la siguiente sintaxis:

```
ALTER TABLE Nombre_de_la_Tabla
   MODIFY
   (
      Nombre_Columna_1      Definición_Tipo_Columna_1,
      Nombre_Columna_2      Definición_Tipo_Columna_2,
      …
      Nombre_Columna_1      Definición_Tipo_Columna_1
   );
```

Ejemplo 3.16 Modificación de los nombres de las columnas de las tablas

Con base en los atributos contenidos en la tabla PRODUCTOS_UNO del SI, la cual en su estado actual se muestra en la figura 3.16, modificar los nombres de los campos de la tabla: i) Código del producto a tipo numérico de longitud 12, ii) stock a numérico de longitud 6, y iii) el campo de detalle por detalle del producto de longitud 50.

Name	Null?	Type
CODIGO_PRODUCTO	NOT NULL	NUMBER(10)
NOMBRE_PRODUCTO	NOT NULL	VARCHAR(50)
TIPO_PRODUCTO	NOT NULL	VARCHAR(5)
PRECIO_COMPRA	NOT NULL	NUMBER(10,2)
PRECIO_VENTA	NOT NULL	NUMBER(10,2)
CODIGO_PROVEEDOR	NOT NULL	NUMBER(10)
STOCK		NUMBER(5)
DETALLE		VARCHAR(100)

Figura 3.16 Descripción de la tabla de PRODUCTOS_UNO del SI

Las acciones de modificación de las características de los campos de la tabla PRODUCTOS_UNO se presentan en la figura 3.17.

```
/* Alteración de los nombres de la tabla PRODUCTOS_UNO del SI */
ALTER TABLE productos_uno
MODIFY
(
   Codigo_Producto  NUMBER(12), /* El campo del código de
                                longitud 10 cambió a 12 */
   Stock            NUMBER(6),  /* El atribuito stock varió
                                de longitu 5 a 6 */
   Detalle          VARCHAR(50) /* El atributo de detalle
                                varió de 100 a 50 en su
                                longitud */
);
```

Figura 3.17 Alteración de las características de los campos de una tabla

El contenido de cada uno de los campos modificados de PRODUCTOS_UNO se presenta en la figura 3.18.

```
/* Descripción de la tabla PRODUCTOS_UNO alterada en sus
características de longitudes de algunos campos */

DESCRIBE productos_uno;
```

Name	Null?	Type
CODIGO_PRODUCTO	NOT NULL	NUMBER(12)
NOMBRE_PRODUCTO	NOT NULL	VARCHAR(50)
TIPO_PRODUCTO	NOT NULL	VARCHAR(5)
PRECIO_COMPRA	NOT NULL	NUMBER(10,2)
PRECIO_VENTA	NOT NULL	NUMBER(10,2)
CODIGO_PROVEEDOR	NOT NULL	NUMBER(10)
STOCK		**NUMBER(6)**
DETALLE		**VARCHAR(50)**

Figura 3.18 Resultado de la alteración de las características de los campos de una tabla

Asignación de campos de tablas con default

Para la asignación de campos preestablecidos en una tabla se utiliza SET DEFAULT, la cual tiene como finalidad asignar datos predefinidos a las columnas de aquellas tuplas que no tengan definidos sus datos de entrada.

Ejemplo 3.17 Asignación de tablas con campos predeterminados

Alterar la tabla PROVEEDORES_UNO del SI adicionando un campo identificado como tipo_proveedor, al cual se le ha asignado por *default* que el tipo de proveedor es internacional con el carácter 'I'.

La asignación del campo preestablecido o por DEFAULT al campo tipo_proveedor se presenta en la figura 3.19.

```
ALTER TABLE proveedores_uno
ADD tipo_proveedor CHAR(1) DEFAULT 'I';
```

Figura 3.19 Asignación del campo de DEFAULT a la tabla PROVEEDORES_UNO del SI

Modificación de campos default de columnas de una tabla

Los campos DEFAULT de una tabla se pueden modificar con la opción MODIFY, la cual cambia el valor preestablecido (o por *default*) de una columna de una tabla.

Ejemplo 3.18 Modificación de campos de DEFAULT de columnas de tablas

Suponer que se ha modificado la tabla PROVEEDORES_UNO del SI, a la cual se le ha adicionado una columna identificada como última fecha de modificación, asignar por DEFAULT la fecha del sistema (SYSDATE) , tal y como se presenta en la figura 3.20.

Modificar el campo identificado en la tabla PROVEEDORES_UNO del SI, de tal manera que la nueva fecha de la modificación sea la fecha del sistema menos uno.

El resultado de la tabla alterada por la modificación del DEFAULT del campo Ultima_modificación se muestra en la figura 3.21.

```
ALTER TABLE proveedores_uno
ADD Ultima_Modificacion DATE DEFAULT SYSDATE;
```

Figura 3.20 Adición de la última fecha de modificación a la tabla PROVEEDORES UNO del SI

```
ALTER TABLE proveedores_uno
MODIFY Ultima_Modificacion DATE DEFAULT SYSDATE-1;
```

Figura 3.21 Modificacion de un campo de DEFAULT de una tabla

3.3.3.5 **Eliminación de tablas**

La eliminación de las tablas de la base de datos de puede hacer utilizando la cláusula DROP, según el siguiente formato:

```
DROP TABLE Nombre_de_la_Tabla [RESTRICT|CASCADE]
```

Se debe tener en cuenta que el comando DROP TABLE no solo elimina la entrada de la tabla de la base de datos que se está eliminando, sino también cancela cada una de las tuplas que contenga la tabla eliminada.

La opción de borrado de la tabla en su acción de propagación opera de la siguiente forma: i) En el caso de utilizar la opción RESTRICT al operar el comando DROP, este es rechazado en el caso de que existan objetos que dependan de la tabla que está eliminando. ii) La opción CASCADE hace que al ejecutar el comando DROP cancele automáticamente los objetos que dependen de la tabla eliminada, y a su vez se cancelen los objetos que dependan de cada una de las tablas "hijas" que se están eliminando.

Ejemplo 3.19 Cancelación de tablas

Con base en las tablas creadas en el Sistema de Información Académico (SIAC) eliminar la tabla de profesores de dicho sistema.

La acción de borrado de la tabla de profesores del sistema SIAC se presenta en la figura 3.22.

```
/* Borrado de la tabla de profesores del SIAC */
DROP TABLE profesores;
```

Figura 3.22 Eliminación de la tabla de PROFESORS de SIAC

3.3.3.6 **Creación de índices**

El desempeño de una base de datos es la capacidad que tiene el gestor de la base de datos para extraer rápidamente los datos de la base de datos.

Los autores Thomas M. Conolly y Carolyn E. Begg remiten la definición del término "índice" a la teoría de archivos utilizando las estructuras de datos y lo definen como

> Una estructura de datos que permite al SGBD localizar registros concretos dentro de un archivo más rápidamente, acelerando así la respuesta a las consultas de los usuarios [6].

Teniendo en cuenta el concepto anterior, una vez que los datos son almacenados en la base de datos, se reconoce que la esencia de su almacenamiento son los archivos, los cuales pueden ser accesados por métodos de búsqueda del tipo secuencial, hashing, binaria o árboles (árboles B^+), entre otros métodos. El hecho de que un archivo esté indexado por cualquier llave, si bien es cierto implica una sobrecarga en índices a nivel de memoria en la capacidad de almacenamiento de la base de datos, esta inversión en el diseño físico de la base de datos se justifica por la velocidad del tiempo de respuesta que tiene la base de datos a través del gestor de datos; pues es claro que no es lo mismo accesar un archivo secuencial con un método secuencial que consultar un archivo secuencial que esté indexado, por cuanto estos índices se pueden accesar por el método de acceso binario, lo cual mejora considerablemente el desempeño en el tiempo de respuesta de la base de datos.

Reconociendo que las cláusulas para la creación y el borrado de índices no son una versión estándar del SQL, hay dialectos que soportan este tipo de cláusula, cuya sintaxis es la siguiente:

```
CREATE [UNIQUE] INDEX NombreIndice
ON NombreTabla (NombreColumna [ASC | DESC ] [ . . . .])
```

La identificación de cada una los componente de la creación de índices se presenta a continuación:

- *NombreIndice*: es la identidad que se le va a dar al índice.
- *NombreTabla*: es la tabla de la cual se va a crear el índice, con la restricción de que solo se pueden crear índices de tablas bases; luego, no es permitido la creación de índices a partir de vistas.
- La utilización de la cláusula UNIQUE implica la unicidad de la columna o combinación de columnas que se usan en el proceso de generar los índices.

- *NombreColumna*: es el nombre de columna o las columnas de la tabla de las cuales se requieren generar los índices.
- *ASC*: indica que los índices se van a generar en orden ascendente, y
- *DESC*: se aplica cuando los índices se requieren ser generados en orden descendente.

Ejemplo 3.20 Creación de índices en una base de datos

Teniendo como base el conjunto de tablas del Sistema Integrado de Manufactura (SIM), crear una tabla índice en orden ascendente identificada como TablaIndiceProductos, con base en la tabla de PRODUCTOS, y utilizando como único campo de indexación el número del proyecto.

```
/* Crea con base en la tabla PRODUCTOS, se crea la tabla índice
de productos en orden ascendente, la cual contiene los números
de los proyectos, utilizando como único campo de indexación del
número del proyecto */

   CREATE UNIQUE INDEX TablaIndiceProductos

   ON Productos (No_proyecto) ASC;
```

Ejemplo 3.21 Creación de índices con bases en múltiples columnas de una tabla de la base de datos

Crear una tabla índice identificada como IndexCiudadesMontosProyecto con base en la tabla de PROYECTOS del SIM, ordenada ascendentemente por las ciudades donde operan los proyectos y por los montos de los proyectos.

```
/* La siguiente instrucción SQL crea una tabla con identidad
IndexCiudadesMontosProyecto, teniendo en cuenta los datos
almacenados de la tabla PROYECTOS, ordenando ascendentemente
las ciudades; y dentro de las ciudades entregando los costos de
los proyectos, también ordenados ascendentemente. */

   CREATE INDEX IndexCiudadesMontosProyecto

   ON Proyectos (ciudad_ejecución, costo_proyecto);
```

3.3.3.7 Eliminación de índices

La eliminación de un índice de la base de datos se realiza mediante la cláusula DROP INDEX, la cual tiene la siguiente sintaxis:

```
DROP INDEX NombreIndice;
```

Donde el NombreIndice es el nombre que tiene el índice en la base de datos, con la característica requisito de que la tabla de índices ha debido ser creada con anterioridad a la cancelación.

Ejemplo 3.22 Cancelación de índices en una base de datos

Para el SIM, cancelar el índice identificado como IndexCiudadesMontosProyecto.

```
/* La siguiente instrucción elimina la tabla de índices
identificada como IndexCiudadesMontosProyecto de la base de
datos del SIM. */

    DROP INDEX IndexCiudadesMontosProyecto;
```

3.3.4 Vistas de la base de datos

3.3.4.1 Definición del concepto de vista de la base de datos

La teoría de bases de dados según la arquitectura ANSI-SPARC [38] define el concepto de vista de la siguiente forma:

i) Nivel Externo: Es la vista que los usuarios tienen de la base de datos; o lo que es equivalente, es la información relevante que los usuarios logran ver de las base de datos de acuerdo con su nivel de autorización para consultar los diferentes esquemas de la base de datos. Vista que corresponde también a la conceptualización en el nivel de TEORÍA que se tiene de la base de datos, en su operación a un contexto en el marco de un sistema de información.

ii) Nivel Conceptual: Es la vista que corresponde a la ABSTRACCIÓN de la base de datos, y contiene la descripción de los datos que contiene la base de datos a través de los esquemas de la base de datos; ello implica que en este nivel se muestran las tablas representativas de los esquemas de la base de datos y sus relaciones.

iii) Nivel Interno: Es la vista que corresponde a la representación física de la base de datos en la computadora; este nivel equivale a la vista propiamente física de la base de datos con sus estructuras de datos, índices, métodos de acceso, entre otros; vista que también se denomina el nivel de DISEÑO físico de la base de datos.

Ahora, el concepto de vista que se maneja de acuerdo con la clasificación de los niveles ANSI-SPARC es totalmente diferente del concepto referido en el marco del Modelo Relacional, al manejar vistas o secciones de la base de datos concretando el concepto que se debe seguir.

Teóricamente, en el Modelo Relacional una vista es

> El resultado dinámico de una o varias operaciones relacionales que operan sobre las relaciones bases para producir otra relación. Una vista es una relación virtual que no tiene por qué existir necesariamente en la base de datos, sino que puede producirse cuando se solicite por parte de un usuario concreto, generándosela en el momento de la solicitud [6].

Se debe tener en cuenta que las relaciones convertidas físicamente a tablas por el gestor de la base de datos al aplicar el MR son las **tablas bases** que contienen los datos en la base de datos. De acuerdo con el concepto de vista del MR se tienen: i) Una vista no es una tabla base en la base de datos. ii) Al no ser una tabla base, la vista puede o no puede existir físicamente en la base de datos. iii) Al ser una tabla, la vista tiene las mismas características de una tabla relacional en su disposición de columnas y tuplas. iv) La generación de una vista en su contenido de datos requiere necesariamente de la resolución **de la vista**; o lo que es equivalente, la vista tiene que ejecutar las proposiciones SQL para lograr generar la vista. v) La vista requiere la materialización física de su contenido; o sea, la generación de la tabla temporal que contiene los datos de la vista.

3.3.4.2 **Creación de vistas**

La sintaxis de la cláusula que permite la creación de vistas en una base de datos es la siguiente:

```
CREATE VIEW Nombre_de_la_Vista [(Nuevo_Nombre_Columna [,…])]
AS sub_selección [WITH | CASCADED | LOCAL] CHECK OPTION]
```

Creación de vistas horizontales

La creación de vistas horizontales tiene como finalidad restringir el acceso a los usuarios de la base de datos a un conjunto de tuplas seleccionadas de una o más tablas bases contenidas en los esquemas de la base de datos.

Ejemplo 3.23 Creación de vistas horizontales

Teniendo como base el Sistema de Información de Académico (SIAC): i) Generar un listado que muestre la impresión parcial de las tuplas de la tabla ASIGNATURAS del SIAC. ii) Crear una vista horizontal de la tabla de asignaturas que muestre solamente las materias que pertenecen al Departamento de Sistemas; o aquellas asignaturas que comienzan en el código de la asignatura con el número 29.

i) La generación del listado de ASIGNATURAS del SIAC se muestra en la figura 3.23, donde se puede observar en la impresión parcial del listado que hay asignaturas diferentes de las identificadas en sus primeros dos dígitos con el código 29.

```
/* Impresión parcial de la tabla de ASIGNATURAS del SIAC * /
SELECT * FROM asignaturas;
```

CODASG	NOMASG	NHTASG	NHPASG	CREASG
1310	INTR. A LA ECONOMIA	3	0	3
1311	MICROECONOMIA	5	0	4
1312	MACROECONOMIA	4	0	4
1321	TEORIA ECONOMICA	3	0	2
1323	ECONOMIA COLOMBIANA	2	0	2
1324	DESARROLLO ECONOMICO	2	0	2
1325	SEM. ECONOMIA COLOMBIANA	2	0	2
1327	ECONOMIA POLITICA	2	0	2
1350	ESTADISTICA	3	0	3
1351	ESTADISTICA EDUCACIONAL	3	2	3
1352	ANALISIS ESTADISTICO	4	0	3
1355	ESTADISTICA I	3	0	3
1356	ESTADISTICA I	4	0	4
1357	ESTADISTICA II	3	0	3
1359	ESTADISTICA II	4	0	4
1364	COMERCIO EXTERIOR	4	0	3
1377	FUNDAMENTOS DE MERCADEO	3	0	3
1400	ELECTIVA PROFESIONAL I	2	0	2
1401	ELECTIVA PROFESIONAL II	2	0	2
1402	ELECTIVA PROFESIONAL III	2	0	2
1410	ADMON EDUCATIVA I	3	0	3
1411	ELEMENTOS DE ADMON.	3	0	3
1412	FUNDAMENTOS DE ADMON.	4	0	4
1414	ADMINISTRACION DE SEGURO	2	0	2

Figura 3.23 Listado de la tabla de ASIGNATURAS del SIAC

ii) En la figura 3.24, se muestra la creación de la vista identificada como Asignaturas_Sistemas.

```
CREATE VIEW Asignaturas_Sistemas
   AS
   SELECT * FROM asignaturas
   WHERE codasg LIKE '29%'
```

Figura 3.24 Creación de la vista de las materias del Departamento de Sistemas del SIAC

La comprobación de que la relación o tabla identificada como Asignaturas_Sistemas del SIAC realmente ha sido creada se presenta en la figura 3.25, donde se puede observar que todos los códigos de las asignaturas empiezan en sus dos primeros dígitos con 29, y el contenido en la identidad del nombre de las asignaturas corresponde a áreas de conocimiento de Ingeniería de Sistemas, tales como Estructuras de Datos, Sistemas Operacionales, entre otras.

```
/* Comprobación de la creación de la vista de Asignaturas_
Sistemas del SIAC */

   SELECT * FROM Asignaturas_Sistemas;
```

CODASG	NOMASG	NHTASG	NHPASG	CREASG
2900	INFORMATICA EDUCATIVA	1	3	1
2901	ELECTIVA NO PROFESIONAL	2	0	2
2902	ELECTIVA PROFESIONAL I	2	2	2
2903	ELECTIVA PROFESIONAL II	2	2	2
2911	INT.A LA ING.DE SISTEMAS	2	0	1
2913	INFORMATICA JURIDICA	1	1	2
2914	INFORMATICA PARA ADMON.	0	2	0
2915	INFORMATICA I	3	2	3
2917	INFORMATICA II	3	2	3
2918	INFORMATICA BASICA	1	1	2
2919	INFORMATICA II	2	1	2
2922	INTR SISTEMA COMPUTACION	3	1	3
2924	ANALISIS NUMERICO	4	2	3
2926	TEORIA DE SISTEMAS	3	0	3
2928	ANALISIS NUMERICO	3	0	3
2930	ESTRUCTURA DE DATOS I	3	1	3
2931	INFORMATICA I	1	1	1
2932	INFORMATICA II	1	1	1
2934	INT.TEO.COMPILADORES	3	1	3
2936	MODELOS DE INGENIERIA I	3	1	3
2938	CONMUTACION	3	1	3
2940	ESTRUCTURA DE DATOS II	3	1	3
2941	TEC.ANALISIS DE ALGORIT.	3	2	3
2942	SISTEMAS OPERACIONALES	3	1	3

Figura 3.25 Comprobación de la creación de la vista Asignaturas_Sistemas del SIAC

Creación de vistas verticales

La creación de vistas verticales cumple la función de restringir a los usuarios de la base de datos una o más columnas que son seleccionadas de una o más tablas.

Ejemplo 3.24 Creación de vistas verticales

Suponer que en el SIAC se asume como información sensible los nombres de los alumnos y el programa al cual pertenecen contenidos en la tabla de ESTUDIANTES, la cual en su descripción se presenta en las figuras 3.26 y 3.27; construir una vista identificada como Alumnos_Sin_Nombres, la cual contenga solo los campos de código del estudiante, sexo y nivel de ubicación del alumno.

```
/* Descripción del contenido de la tabla de estudiantes del
SIAC */

DESCRIBE ESTUDIANTES;
```

Figura 3.26 Descripción del contenido de la tabla ESTUDIANTES del SIAC

Se debe notar que los campos del código del programa al cual pertenece el alumno (codprg) y el nombre del estudiante (nomest) están contenidos como atributos en la tabla ESTUDIANTES, tal y como se presentan en la figura 3.27.

Name	Null?	Type
CODEST	NOT NULL	NUMBER(9)
CODPRG	**NOT NULL**	**NUMBER**
NOMEST	**NOT NULL**	**VARCHAR(50)**
NIVEST	NOT NULL	NUMBER
SEXEST	NOT NULL	CHAR(1)

Figura 3.27 Contenido de la tabla de ESTUDIANTES del SIAC en sus atributos

La creación de la vista identificada como Alumnos_Sin_Nombres creadas a partir de la tabla base de ESTUDIANTES del SIAC se muestra en la figura 3.28.

```
/* Creación de la vista de seguridad de la tabla de
ESTUDIANTES */
   CREATE VIEW Alumnos_Sin_Nombres
   AS
   SELECT codest,sexest,nivest FROM estudiantes;
```

Figura 3.28 Creación de la vista de seguridad de los datos de los alumnos del SIAC

La comprobación de que la vista en su contenido ha sido creada en las base de datos se muestra en la figura 3.29, donde se puede observar la no aparición de los datos de los nombres de los estudiantes y los nombres de los programas al cual pertenecen los alumnos.

```
SELECT * FROM Alumnos_Sin_Nombres;
```

CODEST	SEX	NIVEST
912101080	F	9
912110010	M	7
912110060	M	8
912110100	M	10
912110170	M	7
912110190	M	7
912110370	F	8
912110450	M	8
912110480	M	10
912110530	M	7
912110630	M	8
912110910	M	8
912120060	M	9
912120110	M	10
912120210	M	10
912120300	M	8
912120320	M	9
912120510	M	9
912130040	F	8
912130050	M	6
912130100	M	9
912130130	F	8
912130160	M	8
912130170	F	10

Figura 3.29 Contenido de la tabla de Alumnos_Sin_Nombre del SIAC

3.3.4.3 **Creación de vistas con funciones de agrupamiento y combinación**

Las funciones de agregación y agrupación pueden ser utilizadas en la construcción de vistas a las cuales se les da una identidad, para posterior consulta o gestión del contenido de la vista creada en la base de datos.

Ejemplo 3.25 Creación de vistas con funciones de agrupación

Teniendo en cuenta las tablas base del Sistema de Información Académico (SIAC), crear una vista de Números de Asignaturas por programa, identificada como AsignaturasPrograma, que contenga los códigos de las asignaturas y el número de asignaturas por programa.

La figura 3.30 presenta el resultado de la creación de la vista. La salida de la creación de la vista se presenta en la figura 3.31, y se muestra el contenido de la vista de la base de datos.

```
/* Producir un listado del número de asignaturas por
programas, ordenado ascendentemente por el código del
programa */
CREATE VIEW AsignaturasPrograma (codprg,cont)
   AS
      SELECT s.codprg, COUNT(s.codasg) AS "No. ASIG. /PROG"
      FROM pensum s, programas p
      WHERE s.codprg = p.codprg
      GROUP BY s.codprg
      ORDER BY s.codprg ASC
```

Figura 3.30 Creación de la vista Número de Asignaturas por Programa del SIA

```
SELECT * FROM AsignaturasPrograma
```

CODPRG	CONT
10	139
11	149
12	131
13	123
14	165
23	72
24	52
30	112
32	107
37	22
38	81
39	76
40	124
50	107
60	133
61	145
62	132
63	118
64	162
65	108
95	37

Figura 3.31 Contenido de la vista Número de Asignaturas por Programa

3.3.4.4 **Eliminación de vistas**

La instrucción DROP VIEW se utiliza para la eliminación de vistas y su sintaxis se presenta a continuación:

```
DROP VIEW Nombre_de_la_Vista [RESTRICT|CASCADE]
```

Al utilizar el borrado de vistas de una base de datos, el usuario tiene dos opciones: i) Al emplear RESTRICT, el comando no se ejecuta si se presenta un objeto cuya existencia depende de que continúe existiendo la vista. ii) En el caso de que se utilice la opción CASCADE, entonces DROP VIEW en su operación borra todos los objetos que hagan referencia a la vista.

Ejemplo 3.26 Eliminación de vistas de la base de datos

Con base en las tablas base del SIAC: i) Crear en función de la tabla PROGRAMAS una nueva vista de la base de datos que contenga solo los códigos y los nemónicos de las unidades académicas. ii) Imprimir el listado parcial de la vista, identificada como Nemónicos_programas. iii) Cancelar la vista creada.

La creación de la vista se presenta en la figura 3.32.

```
CREATE VIEW Nemonicos_Programas
AS
   SELECT codprg,mneprg
   FROM programas;
```

Figura 3.32 Creación de la vista de nemónicos de programas del SIAC

El contenido parcial de la vista del listado de nemónicos de los programa del SIAC se presenta en la figura 3.33.

```
SELECT * FROM Nemonicos_programas;
```

CODPRG	MNEPRG
10	II
11	IM
12	IC
13	IS
14	IE
20	MD
21	EN
22	EF
23	EN
24	MD
30	DR
32	PS
35	EP
36	RE
37	LI
38	LI
39	CS
40	DR
50	AD
60	II
61	IM
62	IC
63	IS
64	IE

Figura 3.33 Vista parcial del contenido de los nemónicos de los programas

La cancelación de la vista creada se presenta en la figura 3.34.

```
DROP VIEW Nemonicos_programas;
```

Figura 3.34 Cancelación de la vista Nemónico_programas del SIAC

3.3.4.5 **Solución de vistas**

Una vez creada una vista, se pueden realizar consultas sobre las vistas creadas, lo cual conlleva a identificar un proceso de gestión de vistas o solución de vistas de la base de datos.

Ejemplo 3.27 Resolución de consultas con base en vistas creadas en una base de datos

Con base en la vista creada del número de asignaturas por programa identificada como AsignaturasPrograma: generar un listado que muestre el número de asignaturas por programa para aquellos programas cuyos códigos de programa inicien con el código numérico 6.

Al analizar la consulta, su resolución está basada en la vista creada del número de asignaturas por programa, la cual se transcribe en su creación como se muestra seguidamente:

```
/* Producir un listado del número de asignaturas por programas,
ordenado ascendentemente por el código del programa */
   CREATE VIEW AsignaturasPrograma (codprg,cont)
   AS
      SELECT s.codprg, COUNT(s.codasg) AS "No. ASIG. /PROG"
      FROM pensum s, programas p
      WHERE s.codprg = p.codprg
      GROUP BY s.codprg
      ORDER BY s.codprg ASC
```

El proceso de gestión de la consulta con base en la vista se realiza de la siguiente forma:

i) Los nombres de las columnas de la vista contenidas en la tabla Asignaturas-Programa son generadas por la cláusula SELECT y se traducen a sus correspondientes nombres de columnas de definición identificados como los códigos de los programas (codprg) y los códigos de las asignaturas (codasg), de lo cual se obtiene

```
SELECT s.codprg, COUNT(s.codasg) AS "No. ASIG. /PROG"
```

ii) Los nombres de vista al ejecutar la cláusula FROM son sustituidos por las correspondientes listas FROM de la consulta realizada, lo cual se ejecuta con el comando

FROM pensum s, programas p

iii) La cláusula WHERE de la consulta del usuario se combina con la cláusula WHERE de la consulta, lo cual se hace utilizando la operación lógica AND, con lo cual quedaría

```
WHERE s.codprg = p.codprg AND s.codprg LIKE '6%'
```

iv) La cláusula GROUP BY es copiada de la columna de definición, lo cual se opera con

```
GROUP BY s.codprg
```

v) La cláusula ORDER BY es copiada de la consulta del usuario, traduciendo el nombre correspondiente que aparece en la vista al nombre de la columna de la consulta de definición, lo cual se realiza a través de

```
ORDER BY s.codprg ASC
```

vi) Finalmente, la integración de la consulta al ser combinada se expresa de la siguiente forma:

```
SELECT s.codprg, COUNT(s.codasg) AS "No. ASIG. /PROG"
FROM pensum s, programas p
WHERE s.codprg = p.codprg AND s.codprg LIKE '6%'
GROUP BY s.codprg
ORDER BY s.codprg ASC
```

La ejecución de la consulta combinada del número de asignaturas por programas cuyos códigos de asignaturas comienzan con el número seis (6) se muestran en la figura 3.35, y los resultados en la figura 3.36.

```
SELECT s.codprg, COUNT(s.codasg) AS "No. ASIG. /PROG"
FROM pensum s, programas p
WHERE s.codprg = p.codprg AND s.codprg LIKE '6%'
GROUP BY s.codprg
ORDER BY s.codprg ASC
```

Figura 3.35 Estructura de la consulta combinada de las tablas PENSUM y PROGRAMAS del SIAC para generar el número de asignaturas por programa de los programas con código inicial 6

CODPRG	No. ASIG. /PROG
60	133
61	145
62	132
63	118
64	162
65	108

Figura 3.36 Resultados de la consulta combinada al generar del Sistema de Información Académico (SIAC) en función de las tablas base PENSUM y PROGRAMAS, el número de asignaturas de los programas con código inicial 6

La comprobación del funcionamiento de la consulta gestionada anteriormente, al correr la consulta contra la vista identificada como AsignaturasPrograma, se presenta en la figura 3.37, con lo cual se comprueba en la figura 3.38 que el número de tuplas son iguales (6), como se observa en la figura 3. 36.

```
SELECT * FROM AsignaturasPrograma
WHERE codprg LIKE '6%'
```

Figura 3.37 Generación de tuplas del número de asignaturas por programa con base en la vista creada de AsignaturasPrograma para los programas cuyos códigos empiezan por 6

CODPRG	CONT
60	133
61	145
62	132
63	118
64	162
65	108

Figura 3.38 Resultados del número de asignaturas por programa para aquellos programas con código 6 generados a partir de la vista AsignaturasPrograma

3.3.4.6 **Restricciones de las vistas**

La variabilidad de dialectos SQL existentes hace que algunos de ellos tengan restricciones en el manejo de las vistas; se aclara en este caso que la restricción depende del dialecto que se esté manejando.

- La primera restricción del manejo de vistas tiene que ver con el hecho de que si una columna de vista utiliza funciones de agregación, entonces dicha columna falla al ser utilizada para una función de agregación o en combinación con la cláusula WHERE dentro de un SELECT.

El siguiente representa un contraejemplo, que indica que dicha restricción no se cumple en un dialecto del SQL.

Ejemplo 3.28 Contraejemplo de cumplimiento de restricciones en el manejo de vistas

---

Teniendo como base el Sistema Integrado de Manufactura (SIM), realizar los siguientes comandos SQL: i) Crear una vista con base en la tabla de empleados, identificada como SIM_Empleados_Salarios utilizando GROUP BY, que agrupe el número de empleados por proyecto. ii) Describir la estructura de la tabla creada. iii) Seleccionar el contenido de las tuplas de la vista creada. iv) Contar el número de empleados contenidos en la vista SIM_Empleados_Salarios.

i) La cláusula que crea la vista agrupando el número de empleados en cada proyecto se presenta en la figura 3.39.

```
CREATE VIEW SIM_Empleados_Salarios
   AS
      SELECT No_proyecto,COUNT(Id_empleado) AS "EMPLEADOS",
      SUM (sueldo) AS "SALARIOS"
      FROM empleados
      GROUP BY No_proyecto
      ORDER BY No_proyecto
```

Figura 3.39 Vista de número de empleados por proyecto del SIM

ii) La estructura de la tabla creada representativa de la vista SIM_Empleados_Salarios se presenta en la figura 3.40.

```
DESCRIBE SIM_Empleados_Salarios
```

Nombre	¿Nulo?	Tipo
NO_PROYECTO	NOT NULL	VARCHAR(50)
EMPLEADOS		NUMBER
SALARIOS		NUMBER

Figura 3.40 Descripción de la vista SIM_Empleados_Salarios

iii) El contenido de las tuplas de vista SIM_Empleados_Salarios (figura 3.41).

```
SELECT * FROM SIM_Empleados_Salarios
```

NO_PROYECTO	EMPLEADOS	SALARIOS
PA11TA1111	1	980000
P001BA0001	4	3680000
P002NY0002	2	5300000
P003MI0003	3	8700000
P004SE0004	3	6500000
P005BA0005	2	5300000
P005BA0009	1	3000000
P005BA0012	1	800000
P006NY0008	2	5900000
P006SE0007	1	5000000

Figura 3.41 Contenido de tuplas de la vista SIM_Empleados_Salarios

iv)La consulta sobre la vista SIM_Empleados_Salarios, cuya creación ha utilizado GROUP BY, supuestamente no debe funcionar al realizar una consulta con una función agregación, como es el caso de contar el número de empleados de la vista, asociados a cada proyecto; pero sus resultados se presentan en la figura 3.42.

```
SELECT SUM(EMPLEADOS)
from SIM_Empleados_Salarios
```

SUM(EMPLEADOS)
20

Figura 3.42 Utilización de funciones de agregación en vistas generadas a partir de funciones de agrupación

– La segunda restricción se relaciona con el caso de que una vista generada a partir de una función de agrupamiento no es posible que se combine con otra vista u otra tabla base. Esta restricción también se demostrará por contraejemplo, validando que dependiendo del dialecto del SQL la restricción funciona.

Ejemplo 3.29 Contraejemplo de cumplimiento de combinación de vistas con funciones de agregación

Teniendo como base la tabla de TRANSPORTE del SIM: i) Crear una primera vista identificada como Costos_Transporte, agrupada por los números de ruta de distribución, donde se muestre la suma de los costos de las órdenes de transporte. ii) Imprimir el contenido de la primera tabla creada identificada como Costos_Transporte. iii) Crear una segunda vista identificada como Costos_Seguro, agrupados por los números de ruta de distribución, donde se muestre la suma de los costos de los seguros. iv) Imprimir el contenido de la vista de los Costos de Seguro. v) Combinar las dos vistas previamente creadas, donde se muestre el intersecto de los números de rutas de distribución y se compruebe que todos tienen asociados tanto los costos del transporte como los costos del seguro.

i) La creación de vista identificada como Costos_Transporte se presenta en la figura 3.43.

```
CREATE VIEW Costos_Transporte
   AS
      SELECT No_ruta_distribucion,COUNT(No_orden_transporte)
      AS "ORDENES", SUM (coste_transpote) AS "SUMA
      ORDENES"
      FROM transporte
      GROUP BY No_ruta_distribucion
      ORDER BY No_ruta_distribucion
```

Figura 3.43 Vista Costos_Transporte del SIM

ii) El contenido de la vista Costos_Transporte se muestra en la figura 3.44.

```
SELECT * FROM Costos_Transporte
```

NO_RUTA_DISTRIBUCION	ORDENES	SUMA ORDENES
RD-BAQ-CO-BOG-CO-001	1	5000
RD-BAQ-CO-BOG-CO-011	1	5000
RD-BAQ-CO-CAR-CO-003	1	5000
RD-BAQ-CO-CAR-VN-002	1	5000
RD-BAQ-CO-LA-USA-008	1	40000
RD-BAQ-CO-MI-USA-006	1	30000
RD-BAQ-CO-NY-USA-007	1	5000
RD-BAQ-CO-NY-USA-013	1	5000
RD-BAQ-CO-PAN-PAN-009	1	5000
RD-BAQ-CO-SPA-BA-004	1	70000
RD-BOG-CO-CA-CO-010	1	5000
RD-BOG-CO-CA-CO-014	1	19999
RD-SAM-CO-NY-USA-005	1	5000
RD-SAM-CO-NY-USA-012	1	50000
RD-SAM-CO-NY-USA-014	1	5000
RD-SAM-CO-NY-USA-015	5	870000

Figura 3.44 Tuplas de la vista Costos_Transporte

iii) La creación de vista identificada como Costos_Seguros se presenta en la figura 3.45.

```
CREATE VIEW Costos_Seguros
AS
   SELECT No_ruta_distribucion,COUNT(Tipo_tranporte) AS
   "ORDENES",
   SUM (coste_seguro) AS "SEGUROS"
   FROM transporte
   GROUP BY No_ruta_distribucion
```

Figura 3.45 Vista de Costos_Seguros del SIM

iv) El contenido de la vista de Costos_Seguros se presenta en la figura 3.46.

```
SELECT * FROM Costos_Seguros
```

NO_RUTA_DISTRIBUCION	ORDENES	SEGUROS
RD-BAQ-CO-BOG-CO-001	1	1000
RD-BAQ-CO-BOG-CO-011	1	5000
RD-BAQ-CO-CAR-CO-003	1	5000
RD-BAQ-CO-CAR-VN-002	1	1200
RD-BAQ-CO-LA-USA-008	1	5000
RD-BAQ-CO-MI-USA-006	1	5000
RD-BAQ-CO-NY-USA-007	1	3452
RD-BAQ-CO-NY-USA-013	1	5000
RD-BAQ-CO-PAN-PAN-009	1	2300
RD-BAQ-CO-SPA-BA-004	1	5000
RD-BOG-CO-CA-CO-010	1	3300
RD-BOG-CO-CA-CO-014	1	5000
RD-SAM-CO-NY-USA-005	1	20000
RD-SAM-CO-NY-USA-012	1	5600
RD-SAM-CO-NY-USA-014	1	700000
RD-SAM-CO-NY-USA-015	5	33965

Figura 3.46 Tuplas de la vista Costos_Seguros

v) La prueba a nivel de contraejemplo que la combinación de tablas de vistas funciona se presenta en la figura 3.47.

```
SELECT No_Ruta_Distribucion FROM Costos_Seguros
INTERSECT
SELECT No_Ruta_Distribucion FROM Costos_Transporte
```

NO_RUTA_DISTRIBUCION
RD-BAQ-CO-BOG-CO-001
RD-BAQ-CO-BOG-CO-011
RD-BAQ-CO-CAR-CO-003
RD-BAQ-CO-CAR-VN-002
RD-BAQ-CO-LA-USA-008
RD-BAQ-CO-MI-USA-006
RD-BAQ-CO-NY-USA-007
RD-BAQ-CO-NY-USA-013
RD-BAQ-CO-PAN-PAN-009
RD-BAQ-CO-SPA-BA-004
RD-BOG-CO-CA-CO-010
RD-BOG-CO-CA-CO-014
RD-SAM-CO-NY-USA-005
RD-SAM-CO-NY-USA-012
RD-SAM-CO-NY-USA-014
RD-SAM-CO-NY-USA-015

Figura 3.47 Intersecto de vistas del SIM derivadas de funciones de agrupación

3.3.4.7 **Actualización de vistas en una base de datos**

Teniendo en cuenta que las vistas se soportan en las tablas bases o fuentes de una base de datos, entonces toda actualización sobre dichas tablas se ve necesariamente reflejada en las vistas creadas en la base datos.

Ejemplo 3.30 Actualización de vistas en función de tablas bases

Suponer que un Sistema de Información Académica (SIAC) tiene una tabla de profesores como se presenta en la figura 3.48. i) Crear una vista del SIAC, identificada como ProfesoresSalarios, donde únicamente se presente el nit del docente y su salario asociado. ii) Mostrar todas las tuplas contenidas en la tabla base de profesores. iii) Mostrar todas las tuplas contenidas en la vista anteriormente creada. iii) Mostrar todas las tuplas de la tabla base de los profesores del SIAC. iv) Insertar una nueva tupla en la tabla base. v) Comprobar que la inserción en la tabla base de los profesores del SIAC se ve reflejada en vista ProfesoresSalarios previamente creada.

```
DESCRIBE profesores;
```

Nombre	¿Nulo?	Tipo
NIT		NUMBER(10)
PRIMER_APELLIDO		VARCHAR(20)
SEGUNDO_APELLIDO		VARCHAR(20)
PRIMER_NOMBRE		VARCHAR(20)
SEGUNDO_NOMBRE		VARCHAR(20)
SALARIO		NUMBER(10,2)

Figura 3.48 Descripción de la tabla de profesores de SIAC

i) La vista creada en la base de datos SIAC identificada como ProfesoresSalarios se presenta en la figura 3.49.

```
CREATE VIEW ProfesoresSalarios
AS SELECT Nit,salario
FROM profesores;
```

Figura 3.49 Vista de ProfesoresSalarios del SIAC

ii) Las tuplas contenidas en las tablas base de profesores del SIAC se presentan en la figura 3.50.

```
SELECT * FROM profesores
```

NIT	PRIMER_ APELLIDO	SEGUNDO_ APELLIDO	PRIMER_ NOMBRE	SEGUNDO_ NOMBRE	SALARIO
72456268	AVENDAÑO	RUIZ	DANIEL	ALEJANDRO	50000000
73456269	BECERRA	LÓPEX	ALEJANDRA	SOFIA	10000000
74456270	CONTRERAS	NAVAS	LUIS	CARLOS	90000000
75456277	DIAZ	DIAZ	BETTY	IRENE	80000000

Figura 3.50 Tuplas en la tabla de profesores

iii) Tuplas contenidas en la vista de ProfesoresSalarios se presentan en la figura 3.51.

```
SELECT * FROM ProfesoresSalarios;
```

NIT	SALARIO
72456268	50000000
73456269	10000000
74456270	90000000
75456277	80000000

Figura 3.51 Tuplas contenidas en la vista ProfesoresSalarios del SIAC

iv) La inserción de una tupla en la tabla base de profesores del SIAC se presenta en la figura 3.52.

```
INSERT INTO profesores
VALUES(78456214,'PADILLA','ROSADO','DANIEL','JESÚS',85000000)
```

Figura 3.52 Inserción de una tupla en la tabla de profesores

v) La comprobación de que en la vista previamente creada se refleja la inserción efectuada en la tabla base se presenta en la figura 3.53.

```
SELECT * FROM ProfesoresSalarios;
```

NIT	SALARIO
72456268	50000000
73456269	10000000
74456270	90000000
75456277	80000000
78456214	85000000

Figura 3.53 Comprobación de la actualización de la vista en función de la modificación de la tabla base

3.3.4.8 **Análisis de la utilización de vistas en una base de datos**

La dinámica de los sistemas de información hace que la carga de datos y las consultas de los usuarios a la base de datos vuelvan complejo el sistema de bases de datos. Complejidad que está relacionada tanto con el aumento de la capacidad de almacenamiento de los datos como con la eficiencia en el tiempo de respuesta de la base de datos a los usuarios.

La utilización de vistas en una base de datos es una acción que se debe analizar con relación a los espacios de almacenamiento y los tiempos de respuesta del sistema para la resolución de la vista. El análisis anterior, en la operación de la base de datos, debe confirmar las ventajas en la utilización de vistas, que sin hacer una enumeración exhaustiva son las siguientes:

– *Aumento de la seguridad en la base de datos*: Es lógico pensar en el hecho de que una empresa en su estructura organizacional tiene varios estratos de empleados en su organigrama. Estratos que dependen de las funciones que el trabajador debe efectuar para la empresa. Genéricamente, estos estratos por lo regular son clasificados en niveles de operación, gestión operativa, gestión propiamente dicha, gestión-decisión y prospectiva de la organización. Cada uno de los estratos relacionados tiene una visión de la empresa según sus fun-

ciones, y no todos necesitan la totalidad de los datos de la base de datos para realizar sus funciones. Luego, si la vista de la base de datos que el trabajador puede operar se corresponde altamente con las funciones del trabajador, aumenta la seguridad de la base de datos; porque no todos pueden ver la totalidad de los datos almacenados en la base de datos. En tal sentido, una decisión de aumento de salario corresponde a funciones en un estrato de gestión-decisión de la empresa; decisión salarial que no es necesario que la gestione el nivel operativo de la empresa; lo uno, porque no son sus funciones, y lo otro, porque por confidencialidad no es lógico que todos los obreros conozcan los salarios de todos los empleados de la empresa. Se concluye, por lo tanto, que las vistas permiten el aumento en la seguridad de la base de datos.

- *El logro de la independencia de los datos*: Los estratos o niveles en los cuales se estructura la organización a nivel de funciones significa que las funciones de los empleados, si bien han de estar relacionadas, son independientes. La independencia implica apalancada por la seguridad que si todos tienen acceso a las tablas base de la base de datos, no solo la seguridad de los datos disminuye sino los datos que ven los usuarios no son pertinentes a la operación de las funciones. Con base en lo anterior, el logro de la independencia de datos en su pertinencia de datos se alcanza con la utilización de vistas. Luego, la vista, al resolverse por el gestor de la base de datos, presenta una imagen fiel de la base de datos en sus datos; pero se debe tener en cuenta que dicha imagen es estática de la base de datos; en tal sentido, la utilización de vistas en procesos de eliminación, modificación, reestructuración o división de las tablas bases en la base de datos implica mucho cuidado.
- *Pertinencia de la información para el usuario*: Con base en los estratos de la organización, dichos estratos implican la pertinencia de los datos que el usuario debe ver. La pertinencia de datos significa que los datos presentados de la base de datos hacen lo que deben hacer para el desempeño de las funciones del usuario, y lo hacen con el menor número de datos presentados de las tablas base. Por lo tanto, la pertinencia de los datos implica que a los usuarios no se les cargue con un número mayor de datos de los que necesita; pero tampoco se les impida la vista de aquellos datos que no faciliten el buen desempeño de sus funciones. Luego las vistas contribuyen a la mayor pertinencia de los datos presentados a los usuarios, y de resultas, la menor complejidad de datos y mayor comodidad en el procesamiento de los mismos.

- *Integridad de los datos*: Las ventajas de seguridad, independencia y pertinencia de datos se complementan al utilizar vistas de la base de datos logrando el aseguramiento de su integridad. Esta se consigue al utilizar la cláusula SLQ CREATE VIEW WITCH CHECK OPTION, porque hace cumplir las restrincciones que tengan las tablas bases de la base de datos, asegurando de esta forma la integridad de la base de datos.
- *Aumento de seguridad en la base de datos*: Las vistas generadas de las tablas base en una base de datos solo se pueden operar si el usuario está autorizado o tiene permiso de accesar una sección de la base de datos. Por lo tanto, es el cumplimiento de los controles de autorización lo que permite el aumento de la seguridad de los datos, por cuanto una vista autorizada solo permite ver los datos específicos que el usuario está autorizado ver o manipular como unidades de información que no afecten la seguridad de la base de datos.
- *Aumento de la simplicidad en la consulta a la base de datos*: La pertinencia de datos para los usuarios implica disminución de la complejidad en el manejo de los datos. Con base en lo anterior, la utilización de vistas dan al usuario una imagen personalizada de los datos pertinentes que este necesita ver de la base de datos, lo cual disminuye, consecuentemente, la complejidad en el manejo de los datos.
- *Fiabilidad de los datos*: Los datos mostrados en la vistas de una base de datos son los contenidos fidedignos de la imagen de las tablas base de la base de datos; por lo tanto, los cambios efectuados a una o varias tablas de la base de datos se reflejan de inmediato en el contenido de la vista del usuario, mostrando de esta forma unos valores actualizados de la base de datos.

Si bien el análisis anterior presenta un conjunto de ventajas al utilizar vistas en una base de datos, el desempeño de la base de datos, sus restricciones de actualización y de estructura se convierten en desventajas cuando se utilizan vistas en un sistema de base de datos. El análisis de dichas desventajas se presenta a continuación.

- *Disminución del desempeño de la base de datos*: La resolución de una vista implica necesariamente la consulta de las tablas base a través del gestor de la base de datos. Luego, resolver una vista con consulta multitablas demanda que para procesar la vista, el gestor debe combinar varias tablas cada una de las veces que se acceda a la vista; lo anterior implica la desmejora en el tiempo

de respuesta de la base de datos, por la demora en tiempo que requiere el procesamiento de la vista.

- *Restricciones de actualización*: Teniendo en cuenta que los datos contenidos en las vistas dependen de las tablas bases de la base de datos, para la generación de una vista, el gestor debe traducirla a una actualización contenida en las tablas bases de las cuales se deriva la vista. En tal sentido, la actualización de una vista sencilla se logra a través del gestor; pero no es lo mismo con la actualización de una vista compleja, pues en el caso de no estar actualizada no sería útil al usuario.
- *Restricciones de estructura de la vista*: Las característica en número y tipo de columnas de una tabla vista en cuanto a su estructura se concretan en el momento de su creación en función de una tabla base. En el caso de que la estructura de la tabla cambie, la estructura de la tabla vista no contendrá los cambios operados en la tabla base; por lo tanto, la vista no sería completa en términos del contenido actualizado de la tabla base.

3.4 Resumen

La creación de la base de datos define los tipos de datos SQL en sus especificados de datos: identificadores, escalares y numéricos.

El control de integridad desarrolla el dominio de atributos, la integridad de las entidades, la integridad referencial y las restricciones generales. Para cumplir con la integridad de los datos en el dominio de atributos se utilizan las cláusulas CHECK y CREATE DOMAIN. La integridad de las entidades se cumple cuando ninguna clave primaria de sus tuplas es nula y todos sus valore son unívocos. La integridad referencial se cumple cuando la clave foránea de una tabla relacional "hija" existe en una tabla "padre" y su existencia implica una tupla válida en la base de datos.

La definición de los datos de la base de datos se desarrolla creando la base de datos, su esquema, la construcción y eliminación de tablas, la creación y cancelación de índices, para terminar con el manejo de las vistas de la base de datos.

Ejercicios

Explicar conceptualmente el porqué es necesaria la definición de los datos de la base de datos, con base en los pasos sugeridos por el texto para construir un Sistema de Información.

Dado el siguiente esquema relacional, referido a un Sistema de Información Contable de un conjunto de Proyectos (SICOP), para el cual son sugeridas las **claves primarias**:

Cuenta (**Nro. Cuenta**, Nro. Plan de Cuentas, Nombre de la cuenta, Saldo anterior, Movimiento, Saldo actual).

Transacción (**Nro. Transacción**, Nro. Cuenta, Fecha de Transacción, Tipo Transacción, Valor de la Transacción, Detalle de la transacción).

Proyecto (**Nro. Proyecto**, Nro. Plan de Cuentas, Monto del Proyecto, Líder del Proyecto, Nro. Cuenta, Nro. Transacción).

Responda las siguientes preguntas:

1. Crear las tablas del sistema de bases de datos (SICOP) teniendo en cuenta las características de integridad referencial del lenguaje SQL.
2. Con base en la tabla de la entidad Transacción de SICOP cree una nueva tabla, representativa de la copia de la transacciones mensuales de la empresa, clasificadas por años y meses desde una año inicial hasta el actual, como soporte de las transacciones en caso de contingencia (pérdidas de informaciones) de la empresa.
3. Modifique la estructura de SICOP y para la entidad PROYECTO cree una tabla de la base de datos que tenga en cuenta el tipo de monedas con el monto del proyecto (3) en dólares, euros o pesos y sus respectivos montos asociados para rangos de valores de [1;150] millones de tipos de monedas.

4. Modifique la tabla de la entidad Proyecto anteriormente creada, de tal forma que considere dos fechas:Fecha Inicial y Fecha Final del proyecto, teniendo en cuenta que ambas fechas deben tener una definición en días, semanas y años a la fecha actual.
5. Cree una vista de SICOP que solo contenga los números de las cuentas, el nombre de la cuenta, de un Nro. de Plan de Cuentas dado correspondiente a un Proyecto de la Empresa.
6. Construya una tabla diferente de la entidad Proyecto que contenga los proyectos con acta de cierre, o sea, terminados por la empresa, y utilizando la cláusula SQL INSERT copie en la tabla construida los proyectos cerrados, para lo cual se requiere que la nueva tabla tenga el número del acta de cierre del proyecto, la fecha de cierre y el nombre y código de la empresa interventora de proyectos.
7. Construya dos vista de SICOP: la primera, que contenga los números, nombres de las cuentas y números de plan de cuentas de la entidad Cuenta que no tuvieron movimiento en el último año; la segunda, que contenga los números de proyectos y sus respectivas numeraciones de planes de cuentas, con el fin de hacer un "merge" (mezcla) en una nueva tabla representativa de los proyectos sin actividad laboral durante el último año para la empresa.
8. Construir una tabla de índices de la bases de datos de SICOP con base en la entidad Cuenta, identificada como TablaIndiceCuenta, ordenada de forma ascendente, utilizando como campo único de indexación el número de la cuenta (**Nro. Cuenta**,).
9. Teniendo como base la TablaIndiceCuenta para un año dado de la base de datos de SICOP, genere las tablas índices de la entidad Transacción, seleccionando un índice que le permita para cada mes del año seleccionado comprobar la integridad referencial en la base de datos, en el caso de que la base del sistema SICOP esté bien diseñada.

Capítulo 4

Manejo de datos en la base de datos

4.1 Objetivos

Con el estudio de este capítulo se pretende que el lector aprenda:

- El concepto e importancia de conocer el Lenguaje Estructurado de Consulta (LEC), en sus siglas en inglés SQL (*Structured Query Language*).
- La sintaxis para escribir comandos SQL.
- La descripción de los atributos de las tablas en una base de datos.
- La resolución de consultas simples a través de un gestor de bases de datos utilizando la cláusula SELECT, en combinación con las cláusulas WHERE, ORDER BY, GROUP BY, HAVING COUNT y funciones de agregación.
- La resolución de consultas compuestas en SQL utilizando la cláusula SELECT en subconsultas; combinaciones simples y de n-tablas; en conjunto con las cláusulas EXISTS y NOT EXISTS; hasta llegar a generar tuplas por combinación de tablas de resultados utilizando los operadores de conjuntos unión (UNION), intersecto (INTERSECT), y diferencia (MINUS) de tablas.
- La edición de la base de datos en sus acciones de inserción (INSERT), modificación (UPDATE) y cancelación (DELETE).

4.2 Síntesis conceptual

El manejo de una base datos a través de un Sistemas de Gestión de Bases de Datos (SGBD) implica el conocimiento de un lenguaje que permita tanto la consulta como la manipulación de la base de datos, la consulta para extraer información de la misma y la manipulación para editarla. El lenguaje en mención se denomina Lenguaje de Manipulación de Datos (LMD), que por sus siglas en inglés se especifica como DML o *Data Manipulación Language*. Luego el DML, a través del gestor de la base de datos, permite el manejo de los datos en sus acciones de consulta y edición de datos contenidos en la base de datos.

4.3 Manipulación de datos

Las instrucciones del DML que sirven para la manipulación de los datos son frases SQL que permiten: i) La consulta o extracción de los datos de la base de datos utilizando la cláusula SELECT. ii) La inserción de datos a la base de datos, la cual se realiza con la cláusula INSERT, que permite poblar las tablas de un esquema de la base de datos insertando las tuplas en las tablas. iii) La edición o actualización de un conjunto de datos en la base de datos a través de la cláusula UPDATE. iv) La cancelación de tuplas de una base de datos, o borrado de los datos de una tabla, lo cual se hace utilizando la cláusula DELETE.

4.4 Descripción de los atributos de las tablas

Los datos en el lenguaje SQL se describen a través de la cláusula

```
DESCRIBE NombreTabla
```

La base de datos del Sistema Integrado de Manufactura (SIM) se describe a continuación:

La relación PROYECTOS se muestra en la figura 4.1.

```
DESCRIBE proyectos;
```

Name	Null?	Type
NO_PROYECTO	NOT NULL	VARCHAR(50)
NOMBRE_PROYECTO	NOT NULL	VARCHAR(50)
COSTO_PROYECTO	NOT NULL	NUMBER(15,2)
CIUDAD_EJECUCION		VARCHAR(50)
LIDER_PROYECTO	NOT NULL	VARCHAR(50)
FECHA_INICIAL	NOT NULL	VARCHAR(50)
FECHA_FINAL	NOT NULL	VARCHAR(50)
PRIORIDAD_PROYECTO	NOT NULL	CHAR(1)
COMENTARIOS	NOT NULL	VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)

Figura 4.1 Tabla o relación de proyectos del SIM

La relación "empleados" se presenta en la figura 4.2.

```
DESCRIBE empleados;
```

Name	Null?	Type
ID_EMPLEADO	NOT NULL	VARCHAR(10)
NOMBRE_EMPLEADO	NOT NULL	VARCHAR(50)
DIRECCION_FIS		VARCHAR(50)
EMAIL_EMPLEADO	NOT NULL	VARCHAR(50)
TELEFONO_EMPLEADO	NOT NULL	NUMBER(12,2)
CIUDAD_NAC		VARCHAR(50)
FECHA_NAC		VARCHAR(50)
SEXO_EMPLEADO	NOT NULL	CHAR(1)
CARGO_EMPLEADO	NOT NULL	VARCHAR(50)
FECHA_INGRESO	NOT NULL	VARCHAR(50)
SUELDO	NOT NULL	NUMBER(15,2)
NO_PROYECTO	NOT NULL	VARCHAR(50)

Figura 4.2 Relación o tabla de empleados del SIM

La relación "productos" se describe en la figura 4.3.

```
DESCRIBE productos
```

Name	Null?	Type
COD_PRODUCTO	NOT NULL	VARCHAR(50)
NOMBRE_PRODUCTO	NOT NULL	VARCHAR(50)
TIPO_PRODUCTO	NOT NULL	CHAR(2)
COSTO_PRODUCTO	NOT NULL	NUMBER(15,2)
CANTIDAD_PRODUCTOS	NOT NULL	NUMBER(8,2)
NO_PROYECTO	NOT NULL	VARCHAR(50)

Figura 4.3 Tabla o relación de productos

Los atributos que componen la tabla o relación de "proveedores" se muestran en la figura 4.4.

```
DESCRIBE proveedores;
```

Name	Null?	Type
NIT_PROVEEDOR	NOT NULL	VARCHAR(50)
NOMBRE_PROVEEDOR	NOT NULL	VARCHAR(50)
DIRECCION_PROVEEDOR	NOT NULL	VARCHAR(50)
CIUDAD_PROVEEDOR		VARCHAR(50)
PAIS_PROVEEDOR		VARCHAR(50)
EMAIL_PROVEEDOR	NOT NULL	VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 4.4 Tabla o relación de proveedores

La descripción de cada uno de los campos o atributos de la tabla cuentas_por_pagar se muestra en la figura 4.5.

```
DESCRIBE cuentas_por_pagar;
```

Name	Null?	Type
NIT_PROVEEDOR	NOT NULL	VARCHAR(50)
NO_ORDEN_COMPRA	NOT NULL	VARCHAR(50)
VALOR_COMPRA	NOT NULL	NUMBER(20,2)
TIPO_MONEDA		CHAR(1)
MONTO_30		NUMBER(20,2)
MONTO_60		NUMBER(20,2)
MONTO_90		NUMBER(20,2)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 4.5 Tabla o relación cuentas_por_pagar

La descripción de la tabla de cuentas_por_cobrar se muestra en la figura 4.6.

```
DESCRIBE cuentas_por_cobrar
```

Name	Null?	Type
NIT_COMPRADOR	NOT NULL	NUMBER(8,2)
VALOR_CUENTA_COBRAR	NOT NULL	NUMBER(8,2)
CARTERA_30		VARCHAR(50)
CARTERA_60		NUMBER(8,2)
CARTERA_90		NUMBER(8,2)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 4.6 Tabla o relación de cuentas_por_cobrar

Cuando una tabla se va a referenciar en su descripción a otro usuario, se antepone la descripción del usuario al nombre de la tabla, de la siguiente forma:

DESCRIBE Nombr_usuario.NombreTabla. Si el nombre del usuario es esa, la descripción de la tabla "compras" se presenta en la figura 4.7.

```
DESCRIBE esa.compras;
```

Name	Null?	Type
NO_ORDEN_COMPRA	NOT NULL	VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
NOMNBRE_PRODUCTO	NOT NULL	VARCHAR(50)
VALOR_UNITARIO	NOT NULL	NUMBER(10,2)
FECHA	NOT NULL	VARCHAR(50)
TIPO_COMPRA	NOT NULL	CHAR(1)
TIPO_MONEDA		CHAR(1)
DETALLE		VARCHAR(50)
ID_PROVEEDOR	NOT NULL	VARCHAR(50)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 4.7 Tabla o relación de compras

La tabla "cuenta" se presenta en la figura 4.8.

```
DESCRIBE cuenta;
```

Name	Null?	Type
COD_CUENTA	NOT NULL	NUMBER(8)
NOMBRE_CUENTA		VARCHAR(50)
SALDO_ANTERIOR		NUMBER(8)
MOVIMIENTO		NUMBER(8)
SALDO_ACTUAL		NUMBER(8)
TIPO_MONEDA		VARCHAR(50)

Figura 4.8 Tabla o relación cuenta

Los atributos de la tabla "facturación" se presentan en la figura 4.9.

```
DESCRIBE facturación;
```

Name	Null?	Type
NO_ORDEN_VENTA	NOT NULL	VARCHAR(50)
DESCRIPCION		VARCHAR(50)
CANTIDAD_VENDIDA	NOT NULL	NUMBER(8,2)
VALOR_UNITARIO_VENTA		NUMBER(10,2)
DESCUENTO		NUMBER(10,2)
VALOR_TOTAL		NUMBER(10,2)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
NIT_COMPRADOR	NOT NULL	VARCHAR(50)

Figura 4.9 Tabla facturación

Los atributos que componen la tabla "ventas" se muestran en la figura 4.10.

```
DESCRIBE ventas;
```

Name	Null?	Type
NIT_COMPRADOR	NOT NULL	VARCHAR(50)
NOMBRE_COMPRADOR	NOT NULL	VARCHAR(50)
DIRECCION_COMPRADOR	NOT NULL	VARCHAR(50)
TELEFONO_COMPRADOR	NOT NULL	VARCHAR(50)
TIPO_VENTA	NOT NULL	CHAR(1)
NUMERO_ORDEN		VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 4.10 Tabla o relación ventas

Los campos que componen la relación "producción" en el SIM se presentan en la figura 4.11.

```
DESCRIBE producción;
```

Name	Null?	Type
COD_MASTER_PRODUCCION	NOT NULL	VARCHAR(50)
FECHA_INICIAL	NOT NULL	VARCHAR(50)
FECHA_FINAL	NOT NULL	VARCHAR(50)
ESTADO_PROCESO_PRODUCCION	NOT NULL	CHAR(3)
TARGET_PRODUCCION		NUMBER(8,2)
NO_MAQUINA	NOT NULL	VARCHAR(50)
COD_PRODUCTO	NOT NULL	VARCHAR(50)
NO_PROYECTO	NOT NULL	VARCHAR(50)

Figura 4.11 Tabla o relación producción

Los campos de la relación "maquinaria" se presentan en la figura 4.12.

```
DESCRIBE maquinaria;
```

Name	Null?	Type
NO_MAQUINA	NOT NULL	VARCHAR(50)
NOMBRE_MAQUINA	NOT NULL	VARCHAR(50)
ESTADO_MAQUINA	NOT NULL	CHAR(1)

Figura 4.12 Tabla maquinaria

La relación "mantenimiento" se presenta en la figura 4.13.

```
DESCRIBE mantenimiento;
```

Name	Null?	Type
COD_MANTENIMIENTO	NOT NULL	VARCHAR(50)
FECHA_INICIAL	NOT NULL	VARCHAR(50)
FECHA_FINAL	NOT NULL	VARCHAR(50)
TIPO_MANTENIMIENTO	NOT NULL	CHAR(1)
COMENTARIOS		VARCHAR(50)
NO_MAQUINA	NOT NULL	VARCHAR(50)

Figura 4.13 Relación o tabla mantenimiento

La tabla "transacción" se muestra en la figura 4.14.

```
DESCRIBE transacción;
```

Name	Null?	Type
NO_TRANSACCION_CONTABLE	NOT NULL	VARCHAR(50)
FECHA_TRANSACCION	NOT NULL	VARCHAR(50)
MOVIMIENTO	NOT NULL	NUMBER(15,2)
TIPO_TRANSACCION	NOT NULL	CHAR(1)
DETALLE_TRANSACCION		VARCHAR(50)
COD_CUENTA	NOT NULL	VARCHAR(50)

Figura 4.14 Relación transacción

§ Descripción de los atributos de las tablas

La relación "transporte" se muestra en la figura 4.15.

```
DESCRIBE transporte;
```

Name	Null?	Type
NO_ORDEN_TRANSPORTE	NOT NULL	VARCHAR(50)
NO_ORDEN_VENTA	NOT NULL	VARCHAR(50)
FECHA_CARGA	NOT NULL	VARCHAR(50)
TIPO_TRANPORTE	NOT NULL	CHAR(1)
COSTE_SEGURO	NOT NULL	NUMBER(10,2)
COSTE_TRANSPOTE	NOT NULL	NUMBER(10,2)
COD_CUENTA	NOT NULL	VARCHAR(50)
NO_RUTA_DISTRIBUCION	NOT NULL	VARCHAR(50)

Figura 4.15 Tabla transporte

Los campos que componen la relación "entrega" se presentan en la figura 4.16.

```
DESCRIBE entrega;
```

Name	Null?	Type
NO_RUTA_DISTRIBUCION	NOT NULL	VARCHAR(50)
EMPRESA_TRANSPORTADORA	NOT NULL	VARCHAR(50)
FECHA_ENVIO	NOT NULL	VARCHAR(50)
FECHA_ENTREGA	NOT NULL	VARCHAR(50)

Figura 4.16 Relación o tabla entrega

La descripción de la tabla "planta_producción" se muestra en la figura 4.17.

```
DESCRIBE planta_producción
```

Name	Null?	Type
NO_LINEA_PRODUCCION	NOT NULL	VARCHAR(50)
COMPONENTES_LINEA	NOT NULL	VARCHAR(50)
CIUDAD		VARCHAR(50)
PAIS		VARCHAR(50)
NO_MAQUINA	NOT NULL	VARCHAR(50)
NO_PROYECTO	NOT NULL	VARCHAR(50)

Figura 4.17 Tabla planta_producción

Finalmente, la tabla "mercadeo" se presenta en la figura 4.18.

```
DESCRIBE Mercadeo;
```

Nombre	¿Nulo?	Tipo
NRO_PROYECTO_MERCADEO	NOT NULL	VARCHAR(50)
TIPO_MERCADEO	NOT NULL	CHAR(2)
COSTO_PROY_MERCADEO	NOT NULL	NUMBER(15,2)
NRO_PLAN_MERCADEO	NOT NULL	NUMBER(8,2)
COD_PRODUCTO		VARCHAR(50)

Figura 4.18 Atributos de la tabla mercadeo

4.5 Consultas simples

La cláusula SELECT permite extraer información de la base de datos; información que se puede filtrar, agrupar u ordenar sujeta al cumplimiento de las condiciones explicitadas en la cláusula. La cláusula SELECT permite al usuario la rápida productividad en ambientes de producción soportados por motores de bases de datos, y su potencia a nivel formal corresponde a las operaciones de selección (σ), proyección (Π) y combinación ($\bowtie$) del Algebra Relacional [36].

La información que puede extraer de la base de datos corresponde a i) el contenido de las tuplas de las relaciones y ii) la información almacenada en los atributos de

las tablas. El formato de la cláusula SELECT al utilizar una extensión de la notación Backus-Naur Form (BNF)[1] es el siguiente:

```
SELECT [DISTINCT|ALL]{*|[expresiónColumna[AS nuevoNombre]][…]}
FROM NombreTabla [alias[…]
[WHERE condición]
[GROUP BY listaColumnas][HAVING condición]
[ORDER BY listaColumnas]
```

SELECT es una palabra reservada del lenguaje de manipulación de datos del gestor de la base de datos SQL, la cual explicita los atributos o columnas que deben aparecer en cada una de las tuplas o filas de la base de datos.

De la relación o tabla de la base de datos identificada por NombreTabla se pueden seleccionar o todas las tupla con ALL o con asterisco (*) o se pueden extraer tuplas con características distintas con DISTINCT. La expresión Columna representa la columna o columnas de la tabla que se están consultando; la cual se puede explicitar mediante una expresión a través de campos calculados. Suponga la tabla de proyectos del SIM, entonces las cláusulas: i) SELECT * FROM proyectos selecciona todos los campos de la relación proyectos. ii) SELECT No_proyecto, costo FROM proyectos; selecciona únicamente los atributos del número del proyecto y el costo del proyecto del SIM. iii). Mientras que en la cláusula SELECT No_proyecto, costo*1,05 AS "PROYECTOS AUMENTADOS EN 5 %", la expresión Columna es costo*1,05, y la palabra AS indica el literal con el que se va a renombrar o nuevoNombre de columna.

FROM: explicita la relación o tablas de las cuales se va a extraer la información de la base de datos.

WHERE: en la selección opera a nivel condicional, seleccionando o filtrando únicamente aquellas tuplas de la tabla que cumplan con la condición.

GROUP BY: selecciona las tuplas o filas a las cuales correspondan el mismo valor de atributo o columna de la tabla.

[1] http://www-oei.eui.upm.es/Asignaturas/BD/BD/docbd/tema/LenguajeSQL-PAC.pdf Consultado el 4 de octubre de 2012.

HAVING: se utiliza para el filtrado de los grupos, bajo el cumplimiento de una condición.

ORDER BY: se emplea en el requerimiento de ordenar las salidas explicitadas en la cláusula SELECT por algún atributo de la tabla.

Es de la mayor importancia resaltar que de acuerdo con la notación explicitada en el formato de la instrucción SELECT, el orden de cada una de las cláusulas que integran el SELECT no se puede cambiar; siendo obligatorias en la estructuración de cualquier consulta SQL únicamente las cláusulas SELECT y FROM; las restantes son opcionales.

El resultado de la cláusula SELECT, teniendo en cuenta que la consulta se deriva de una tabla o de un conjunto de tablas, es otra tabla o relación, tal y como se muestra en el desarrollo de las próximas secciones con relación a la Base de Datos del Sistema Integrado de Manufactura (SIM).

4.5.1 Seleccionar todos los atributos de todas las tuplas

Los atributos de una tabla son las columnas de la tabla, y el cruce de la fila con la columna de la tabla contiene explícitamente la unidad de información de la tabla; cruce que al ser representado por algunos o todos los atributos de la tabla recibe el nombre de "tupla".

Ejemplo 4.1 Selección con atributos de tablas del SIM.

Producir un listado que muestre todos los atributos de la relación de empleados del SIM.

Análisis: Se seleccionan todos los atributos de la tabla de empleados.

Resultados:

```
SELECT /* Se muestran los atributos de la tabla de empleados */
id_empleado,nombre_empleado,direccion_fis,email_
empleado,telefono_empleado,ciudad_nac,fecha_nac,sexo_
empleado,cargo_empleado,fecha_ingreso,sueldo,No_proyecto
FROM empleados;
```

Los resultados de la consulta se presentan en la figura 4.19.

```
SELECT id_empleado,nombre_empleado,direccion_fis,email_empleado,telefono_empleado,ciudad_
nac,fecha_nac,sexo_empleado,cargo_empleado,fecha_ingreso,sueldo,No_proyecto
FROM empleados;
```

ID_EMPLEADO	NOMBRE_EMPLEADO	DIRECCION_FIS	EMAIL_EMPLEADO	TELEFONO_EMPLEADO	CIUDAD_NAC	FECHA_NAC	SEX	CARGO_EMPLEADO	FECHA_INGRESO	SUELDO	NO_PROYECTO
72415240	CONSUEGRA AMIN PEDRO DANIEL	CALLE 80 NO 34-12	pcons@hotmail.com	3001452140	MONTERIA	13-MA-80	M	TECNICO	01-JAN-15	1000000	P001BA0001
74415240	AYOS RENOWISKY ANDRÉS JULIAN	CALLE 47 NO 80-30	areno@hotmail.com	3001334544	SOLEDAD	20-JAN-95	M	TECNICO	01-JAN-18	1500000	P002NY0002
74415291	GUTIEREZ DÍAZ ANDRES	CALLE 101 NO 51B-15	gdiaz@gmail.com	3001425063	BARRANQUILLA	15-DEC-89	M	INGENIERO	01-JAN-18	3000000	P003MI0003
74415342	GOENAGA MAYA JULIO	SAN TELMO AVE. 100 NO 12-10	goema134@hotmail.com	3001452563	BARRANQUILLA	14-JAN-90	M	INGENIERO	01-JAN-15	4500000	P004SE0004
74415393	VILLAR ARTEAGA LUIS ALBERTO	TRANSV. 40 NO 234-15	villaar190@hotmail.com	3001156321	BARRANQUILLA	07-DEC-01	M	TECNICO	01-JAN-18	1000000	P005BA0005
74415444	VILLAMIZAR ATYOS ANDRÉS	CALLE 80 NO52-15 APTO.303	viaty@gamil.com	3147890123	BARRANQUILLA	04-APR-67	M	DOCTOR	01-JAN-18	5000000	P006SE0007
74415495	JULIO CONTRERAS JUAN CARLOS	CALLE 95 NO 34-12	jcjc1456@hotmail.com	3148963022	CARTAGENA	05-JAN-80	M	DOCTOR	01-JAN-18	5000000	P006NY0008
74415546	NANDIN LOPERA JULIANA	CARRERA 42A4 NO 23-18	nnlo456@hotmail.com	3146369999	CARTAGENA	10-DEC-88	F	INGENIERO	01-JAN-15	3000000	P005BA0009
74415597	PEREZ PEREZ ANDREA	CARRERA 68C NO 97-18	perchy@gamil.com	3001452129	MADRID	04-DEC-98	F	ASISTENTE	01-JAN-18	1200000	P003MI0003
74415648	AGUIRRE PEREZ RUTH ELENA	CARRERA 15H NO 97-10	agperz77@hotmail.com	3001452225	MIAMI	03-JAN-95	F	TECNICO	01-JAN-18	1000000	P004SE0004
74415699	SILVESTRE CUELLO ANA	CALLE 56C NO 43-12	silcuna4@hotmail.com	3001452489	SANTA MARTA	08-DEC-90	F	ASISTENTE	01-JAN-15	800000	P005BA0012
74415750	VISBAL PEREZ SOFIA	CALLE 80 NO 45C-34	vipe80@hotmail.com	3001452963	BARRANQUILLA	08-DEC-90	F	ASISTENTE	01-JAN-18	800000	P001BA0001
74415801	VEGA BAENA CARMEN	CARRERA 87 NO 32C-15	veba89@hotmail.com	3001452001	BOGOTA	12-APR-70	F	INGENIERO	01-JAN-18	3800000	P002NY0002
74415852	SANDOVAL HUCKE MARY	TRANSV 88 NO 56-18	santih@gamil.com	3001452458	NEW YORK	15-APR-77	F	INGENIERO	01-JAN-18	4500000	P003MI0003
74415903	LOPEZ DIAZ RUBEN DANIEL	CALLE 80B NO 45-44	lodiru677@hotmail.com	3014145214	BARRANQUILLA	27-DEC-88	M	TECNICO	01-JAN-15	1000000	P004SE0004
74415954	HERRERA LORA ANNY	CARRERA 18 NO 34-43	hlann@45@gamil.com	3001452141	SOLEDAD	17-JAN-80	F	INGENIERO	01-JAN-18	4300000	P005BA0005
74416005	GUERRA PORTO LUIS MIGUEL	CALLE 82 NO 21-25	guer@hotmail.com	3010145214	CALI	12-DEC-91	M	ADMINISTRADOR	01-JAN-18	900000	P006NY0008
74416056	FABREGAS COTO LUIS DANIEL	CARRERA 52H NO 34-15	facat45@hotmail.com	3141452148	MIAMI	15-JAN-90	M	ADMINISTRADOR	01-JAN-15	900000	P001BA0001
74416107	JUVENAL HINCAPIE ELEONORA	CALLE 14 NO 45-10	juvehin7@hotmail.com	3141452142	CARACAS	16-APR-92	F	ADMINISTRADOR	01-JAN-18	980000	P001BA0001
74416158	PUYANA KARKER CLAUDIA JULIANA	CARRERA 56B NO 78-15	tatalinda@hotmail.com	3141452140	MIAMI	17-DEC-95	F	ADMINISTRADOR	01-JAN-18	980000	PA11TA1111

Figura 4.19 Tuplas de la tabla de empleados enumerando cada uno de los atributos de la tabla

Ejemplo 4.2 Selección utilizando el selector asterisco (*)

Listar todas las líneas, filas o tuplas de la relación "productos" utilizando en la cláusula SELECT el selector asterisco (*).

Análisis: Se seleccionan todos los atributos de la tabla de productos.

Resultados:

```
SELECT *
FROM productos;
```

El carácter asterisco (*) en la cláusula SELECT da como salida de información todos los atributos de una tabla sin necesidad de nombrar cada una de las identificaciones de los campos de las tuplas dentro del SELECT, como se presenta en la figura 4.20.

```
SELECT *
FROM productos;
```

COD_PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
100-100-101-000	PORTATIL NGP	T	800000	20	P001BA0001
100-100-101-001	PROCESSOR XTS	P	400000	10	P002NY0002
100-100-101-002	INTERFACES MULTIPROPÓSITO	T	200000	3	P003MI0003
100-100-101-003	PHOTONIC CONTOLLED ONE SIM	T	20000	1	P004SE0004
100-100-101-004	DRIVE CONTROLLERS SIM	P	25000	2	P005BA0005
100-100-101-005	CONCECTOR RXL NGN	T	1800000	1	P006SE0007
100-100-101-006	TWO BRAIN MOTOR	T	3000000	1	P006NY0008
100-100-101-007	MOTOR DEDUCTIVE	T	1000000	20	P005BA0009
100-100-101-008	WEB 10.4 NGWEB	T	1800000	20	P005BA0010
100-100-101-009	MOTOR AIRPL. XT CONTROL TAB.	T	1000000	20	P005BA0011
100-100-101-010	MOTOR WEM DB 2.0	P	500000	4	P005BA0012

COD_PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
100-100-101-011	OPS - 2018	T	175000	8	P006BO0013
100-100-101-012	ROUT NO.1435	T	30000	9	P007TA0014
100-100-101-013	BODY TRACKING XC-XB-XD	T	180000	12	P008TA0015
100-100-101-014	OPTICAL COMPUTER	T	570000	5	P009TA0014
100-100-101-015	CHIP RSA	MT	40000	100	P010BA0015
100-100-101-016	LPS - NEW GENERATION	MT	80000	57	P020BA0016
100-100-101-017	CODY SYS. UVXA	MT	50000	30	P011BO0017
100-100-101-018	OPTICAL CONTROL FBA-12	T	100000	10	P012BO0222
100-100-101-019	CAD GRAPHIC SYSTEMS	P	5000	1	PA11TA1111

Figura 4.20 Utilización de * en la cláusula SELECT

4.5.2 Seleccionar algunos atributos de todas las tuplas

La selección de algunos atributos de una relación implica nombrar en el SELECT los atributos que se quieran extraer de la tabla.

Ejemplo 4.3 Selección de algunos atributos de una tabla

Producir un listado de proyectos del SIM en el que se muestren únicamente los atributos: el número del proyecto, el nombre del líder del proyecto y el costo asociado al proyecto. Los resultados de la consulta se presentan en la figura 4.21.

Análisis: De la tabla de proyectos se seleccionan únicamente los atributos requeridos de la consulta.

Resultados: Se presentan en la figura 4.21, al correr el código SQL.

```
/* Se selecciona solo los atributos de Número del proyecto, su
nombre, su líder y coste */

SELECT No_proyecto,nombre_proyecto,lider_proyecto,costo_
proyecto

FROM proyectos
```

```
SELECT No_proyecto,nombre_proyecto,lider_proyecto,costo_
proyecto

FROM proyectos
```

NO_PROYECTO	NOMBRE_PROYECTO	LIDER_PROYECTO	COSTO_ PROYECTO
P001BA0001	ENSAMBLE DE COMPUTADORES	HARKER JOHN	100000000
P002NY0002	NUEVOS PROCESADORES	HEARNE CHARLES	100000000
P003MI0003	INTERFACES GRÁFICAS TACTILES	DAY LIN	10000000
P004SE0004	SENSORES ÓPTICOS	KIM WELLINGTON	200000000
P005BA0005	PEN DRIVES	JIM HARNE	10000
P006SE0007	CONECTORES DE ROUTERS	LINDA DAY	450000
P006NY0008	BRAIN MACHINE	KIMBERLY NORTHON	50000
P005BA0009	MOTOR DEDUCTIVE DATABASE	SAIM KINCHNER	10000000
P005BA0010	WEB SEMANTIC SOFTWARE V.99	ANGELA LEIBNITZ	67890000
P005BA0011	MOTOR INFERENCIAL DB	CONRA LIBTZ	120000
P005BA0012	WEB DATABASE	MARIE CURLITZ	12457
P006BO0013	OPERATING SYSTEMS PROD.	JHON HAINSERBERG	100000
P007TA0014	SOFTWARE DE RUTEO	MIT CONNETLONELL	12452222
P008TA0015	SENSOR BODY SYSTEMS	RYAN BOSH	235645
P009TA0014	OPTICAL COMPUTER DEVICE	NEIL BANDERBOSH	12456897
P010BA0015	OPTICAL CHIP	ALDRIN HARKER	12456897
P020BA0016	LIGHT PEN SENSOR	PHILIPH FAIRHEAD	10000000
P011BO0017	EXPLORER BD SYSTEM	KIM STRAUUS	500000001
P012BO0222	INTERFACE FOR OPTICAL CON.	JUAN MIGUEL OTALORA	454545000
PA11TA1111	SOFTWARE GRAPHIC	JOHANA BEIBAM	5124600000

Figura 4.21 Selección de una serie de atributos de todas las tuplas de la tabla de proyectos

4.5.3 Uso de literales en la cláusula SELECT

Los campos de salida que aparecen como etiquetas en los atributos de las tablas se pueden renombrar en SQL con la cláusula AS, cuya sintaxis es la siguiente:

```
SELECT [expresiónColumna [AS nuevoNombre][…]
FROM NombreTabla
```

Donde AS identifica el literal que renombra los campos de salida de la tabla, y los literales son encerrados entre comillas: "ESTE ES UN LITERAL".

Ejemplo 4.4 Utilización de AS en la selección de tuplas

Listar del SIM, utilizando la relación de proveedores, los atributos de la tabla nit_proveedor, nombre_proveedor y país_proveedor, renombrándolos con las identidades en literales "CÓDIGO", "PROVEEDOR" y "PAIS".

Análisis: De la tabla de proveedores se seleccionan los atributos nit, nombre y país, y a cada uno de ellos se le asigna, respectivamente, los literales "CÓDIGO", "PROVEEDOR" y "PAIS".

Resultados: La figura 4.22 muestra la salida, cuya consulta es

```
SELECT Nit_proveedor AS "CÓDIGO",nombre_proveedor AS
"PROVEEDOR", pais_proveedor AS "PAIS"
FROM proveedores;
```

```
SELECT Nit_proveedor AS "CÓDIGO",nombre_proveedor AS
"PROVEEDOR", pais_proveedor AS "PAIS"
FROM proveedores;
```

CÓDIGO	PROVEEDOR	PAIS
72435228001	DELL CORPORATION	USA
72435228002	TOSHIBA COMPANY	JAPAN
72435228003	IBM	COLOMBIA

Figura 4.22 Uso de literales en las salidas de los atributos de las relaciones

4.5.4 Impresión de campos numéricos con formato predefinido

Ejemplo 4.5 Utilización de formatos numéricos en la selección de tuplas

Generar un listado mostrando el código de los proyectos del SIM, sus líderes y el costo de la inversión asociada, mostrando el formato de los campos numéricos.

Análisis: Se seleccionan las tuplas de la tabla de proyectos, con los atributos: el número del proyecto, el líder del proyecto y su costo; pero al costo se le asocia el formato TO_CHAR.

Resultados: La imagen de salida se encuentra en la figura 4.23, y el SELECT asociado a la consulta es:

```
SELECT NO_proyecto,lider_proyecto,
TO_CHAR (costo_proyecto, '$99G999G999G999D99', 'NLS_NUMERIC_
CHARACTERS = '',.'' ') AS "COSTO PROYECTO"
FROM proyectos;
```

Donde los campos de la máscara ('$99G999G999G999D99'): D representa la coma decimal y G el punto decimal, tal y como se ejemplifica en la figura 4.23.

```
SELECT NO_proyecto,lider_proyecto,
TO_CHAR (costo_proyecto, '$99G999G999G999D99', 'NLS_
NUMERIC_CHARACTERS = '',.'' ') AS "COSTO PROYECTO"
FROM proyectos;
```

NO_PROYECTO	LIDER_PROYECTO	COSTO PROYECTO
P001BA0001	HARKER JOHN	$100.000.000,00
P002NY0002	HEARNE CHARLES	$100.000.000,00
P003MI0003	DAY LIN	$10.000.000,00
P004SE0004	KIM WELLINGTON	$200.000.000,00
P005BA0005	JIM HARNE	$10.000,00
P006SE0007	LINDA DAY	$450.000,00
P006NY0008	KIMBERLY NORTHON	$50.000,00
P005BA0009	SAIM KINCHNER	$10.000.000,00
P005BA0010	ANGELA LEIBNITZ	$67.890.000,00
P005BA0011	CONRA LIBTZ	$120.000,00
P005BA0012	MARIE CURLITZ	$12.457,00
P006BO0013	JHON HAINSERBERG	$100.000,00
P007TA0014	MIT CONNETLONELL	$12.452.222,00
P008TA0015	RYAN BOSH	$235.645,00
P009TA0014	NEIL BANDERBOSH	$12.456.897,00
P010BA0015	ALDRIN HARKER	$12.456.897,00
P020BA0016	PHILIPH FAIRHEAD	$10.000.000,00
P011BO0017	KIM STRAUUS	$500.000.001,00
P012BO0222	JUAN MIGUEL OTALORA	$454.545.000,00
PA11TA1111	JOHANA BEIBAM	$5.124.600.000,00

Figura 4.23 Salida de atributos con formatos numéricos

4.5.5 Selección de tuplas distintas

El uso de DISTINCT en la cláusula SELECT permite seleccionar únicamente las tuplas distintas con relación a un determinado atributo de la tabla.

Ejemplo 4.6 Utilización de selección de tuplas distintas SELECT DISTINCT

Listar las ciudades de ejecución de los proyectos del SIM y compararlas con el número de ciudades distintas donde se están ejecutando los proyectos del Sistema Integrado de Manufactura.

Análisis: Para lograr la comparación de los resultados primero se seleccionan las ciudades donde se ejecutan proyectos, no importando si las ciudades se repiten o no; posteriormente se realiza una selección de las ciudades distintas empleando la cláusula DISTINCT.

Resultados: La selección en la que se muestran todas las ciudades donde se ejecutan los proyectos se presenta en la figura 4.24; por su parte, al seleccionar las ciudades distintas, las cuales están realizando proyectos, la sentencia SELECT con resultados en la figura 4.25 se muestra a continuación:

```
SELECT DISTINCT ciudad_ejecucion
FROM proyectos;
```

```
SELECT ciudad_ejecucion
FROM proyectos;
```

CIUDAD_EJECUCION
BARRANQUILLA
NEW YORK
MIAMI
SEATLE
BARRANQUILLA
SEATLE
NEW YORK
BARRANQUILLA
BARRANQUILLA
BARRANQUILLA
BARRANQUILLA
BOGOTA
TAMPA
TAMPA
TAMPA
BARRANQUILLA
BARRANQUILLA
BOGOTA
BOGOTA
TAMPA

Figura 4.24 Listado de todas las ciudades donde se encuentran proyectos en ejecución

```
SELECT DISTINCT ciudad_ejecucion
FROM proyectos;
```

CIUDAD_EJECUCION
MIAMI
SEATLE
BARRANQUILLA
BOGOTA
TAMPA
NEW YORK

Figura 4.25 Nombre de las ciudades distintas donde se están ejecutando proyectos

4.5.6 Generación de campos calculados

Las consultas SQL en sus atributos de salida en cada una de las tuplas pueden tener expresiones calculadas en las columnas o atributos de resultados, de acuerdo con el siguiente formato:

```
SELECT [DISTINCT][ALL] {*|[expresiónColumna [AS nuevoNombre ]] […]}
FROM NombreTabla
```

Luego, la expresión Columna puede ser una expresión matemática, la cual afecta a través de sus cálculos los resultados de la columna de la tabla.

Ejemplo 4.7 Selección de tuplas con campos calculados

Calcular el saldo actual de la tabla cuenta del SIM.

Análisis: La tabla "cuenta" en SIM tiene en sus atributos el Saldo_Anterior y el Movimiento, a partir de los cuales se calcula el Saldo_Actual.

Resultados: La salida de la consulta se presenta en la figura 4.26, generada de la operación del siguiente código:

```
SELECT cod_cuenta,nombre_cuenta,saldo_anterior+Movimiento AS
"SALDO"
FROM cuentas;
```

Notar que la expresión que aparece en la columna titulada SALDO es el resultado de una expresión matemática.

```
SELECT cod_cuenta,nombre_cuenta,saldo_anterior+Movimiento
AS "SALDO"
FROM cuentas;
```

COD_CUENTA	NOMBRE_CUENTA	SALDO
10-000-000-000-000	ACTIVO	1200000000
10-001-000-000-000	ACTIVO CORRIENTE	1010000
10-001-001-000-000	BANCOS	999500
10-001-001-001-000	BANCOS EXTRANJEROS	1010000
10-001-001-001-001	SEA FIRST BANK	9500
10-001-001-001-002	UNION BAN	11000
10-001-001-001-003	BANCO BILBAO	10050
10-001-001-001-004	INTERNATIONAL BANK	500
10-001-001-001-005	BANCO DE PORTUGAL	11000
10-001-001-011-000	BANCOS LOCALES	10000300
10-001-001-011-001	BANCO DE BOGOTÁ	1010000
10-001-001-011-002	BANCO DE COLOMBIA	2000200
10-001-001-011-003	BANCO SANTANDER	700000500
01-001-001-001-999	PROV. DELL	200500
01-001-001-001-998	PROV. TOSHIBA	50500
01-001-001-001-997	PROV. IBM	7000500

Figura 4.26 Campo calculado de SALDO a partir del Saldo_Anterior y el Movimiento

4.5.7 Búsqueda de tuplas basada en comparación

Los resultados de las consultas a la base de datos realizadas hasta este momento han implicado que en los resultados se muestran todas las tuplas de la tabla. Para filtrar por alguna condición las tuplas de salida de la tabla se utiliza la palabra clave WHERE, de acuerdo con el siguiente formato:

```
SELECT [DISTINCT][ALL] {* |[expresionColumna [ AS nuevoNombre ]] […]}
FROM   NombreTabla
[WHERE condición]
```

Donde la condición cumple la función de filtrar las tuplas de salida de la(s) tabla(s) de la base de datos, lo cual está sujeto a su cumplimiento.

Los predicados básicos de estructuración de condiciones para las búsquedas por comparación de acuerdo con el estándar ISO se presentan en la tabla 4.1.

Tabla 4.1 Condiciones o predicados básicos de búsqueda de acuerdo con ISO

Predicado	Significado
Comparación	Filtra comparando los resultados de dos expresiones (a,b), a versus b
Rango	Compara si el valor de una expresión (a) pertenece a un rango R, a $\in$[R]
Pertenencia	Verfica si el valor de una expresión (a) pertenece a un conjunto (C), a $\in$[C]
Correspondencia	Verifica si una cadena de caracteres (c) pertenece a un patrón (P), c $\in$[P]
Nulo	Compuebra si hay tuplas para las cuales sus atributos sean nulos

Los operadores aritmético/lógicos bases para la comparación simple se presentan en la tabla 4.2.

Tabla 4.2 Operadores básicos para estructuras consultas basadas en comparación

Operador	Significado
=	Igual
<>	Diferente, según el estándar ISO
¡=	Diferente, aceptado para algunos dialectos del SQL
<	Menor que
<=	Menor o igual que
>	Mayor
>=	Mayor o igual que
AND	Operador lógico que corresponde a A y B (A $\wedge$ B)
OR	Operador lógico que corresponde a A o B (A $\vee$ B)
NOT	Operador lógico que corresponde a la negación ($\neg$ B)

La precedencia de operadores se presenta en la tabla 4.3.

Tabla 4.3 Precedencia de Operadores

Precedencia	Significado
1	( ) Los paréntesis tienen la mayor prioridad y eliminan ambigüedades
2	Las expresiones son evaluadas de izquierda a derecha
3	Primero se evalúan las subexpresiones que están contenidas en corchetes
4	NOT se evalúa antes que los operadores AND y OR
5	AND se evalúa antes que el operador OR

Ejemplo 4.8 Selección de tuplas con base en comparación simple

Generar un listado con números de proyectos, nombres y costos de proyectos realizados en el SIM cuyos costos sean superiores o iguales a 100 millones de pesos.

Análisis: De la tabla de proyectos se listan los campos requeridos, teniendo en cuenta filtrar en el WHERE aquellas tuplas cuyos costos de proyectos sean superiores al valor de 100 millones de pesos.

Resultados: El resultado de la consulta se presenta en la figura 4.27, según el código.

```
SELECT No_proyecto,nombre_proyecto,lider_proyecto,
TO_CHAR (costo_proyecto, '$99G999G999G999D99', 'NLS_NUMERIC_
CHARACTERS = '',.'' ') AS "COSTO PROYECTO"
FROM proyectos
WHERE costo_proyecto >= 100000000
```

```
SELECT No_proyecto,nombre_proyecto,lider_proyecto,
TO_CHAR (costo_proyecto, '$99G999G999G999D99', 'NLS_
NUMERIC_CHARACTERS = '',.'' ') AS "COSTO PROYECTO"
FROM proyectos
WHERE costo_proyecto >= 100000000
```

NO_ PROYECTO	NOMBRE_PROYECTO	LIDER_PROYECTO	COSTO PROYECTO
P001BA0001	ENSAMBLE DE COMPUTADORES	HARKER JOHN	$100.000.000,00
P002NY0002	NUEVOS PROCESADORES	HEARNE CHARLES	$100.000.000,00
P004SE0004	SENSORES ÓPTICOS	KIM WELLINGTON	$200.000.000,00
P011BO0017	EXPLORER BD SYSTEM	KIM STRAUUS	$500.000.001,00
P012BO0222	INTERFACE FOR OPTICAL CON.	JUAN MIGUEL OTALORA	$454.545.000,00
PA11TA1111	SOFTWARE GRAPHIC	JOHANA BEIBAM	$5.124.600.000,00

Figura 4.27 Proyectos del SIM superiores o iguales a 100 millones de pesos

4.5.8 Búsqueda de tuplas basada en comparación compuesta

Ejemplo 4.9 Selección con base en la utilización de comparaciones compuestas

Listar los números de proyectos, nombre y líderes realizados en la ciudad de Barranquilla con prioridad uno.

Análisis: Teniendo en cuenta la relación proyectos del SIM, se seleccionan aquellas tuplas con base en la cláusula WHERE que cumplan las condiciones i) con prioridad 1 en sus proyectos y ii) con ciudad de ejecución igual a Barranquilla.

Resultados: Se presentan en la figura 4.28, de acuerdo con la siguiente consulta:

```
SELECT No_proyecto,nombre_proyecto,lider_proyecto,ciudad_
ejecucion,prioridad_proyecto
FROM proyectos
/* Ejemplificación de condición compuesta numérica y alfabética */
WHERE prioridad_proyecto = 1 AND ciudad_
ejecucion='BARRANQUILLA'
```

Se debe tener en cuenta que en la consulta la ciudad de ejecución es un literal ('BARRANQUILLA'), donde el literal se debe escribir exactamente como está en la base de datos; por cuanto si se escribe el literal en la consulta como 'Barranquilla', el resultado del motor de la base de datos es vacío.

```
SELECT No_proyecto,nombre_proyecto,lider_proyecto,ciudad_
ejecucion,prioridad_proyecto

FROM proyectos

WHERE prioridad_proyecto = 1 AND ciudad_
ejecucion='BARRANQUILLA'
```

NO_PROYECTO	NOMBRE_PROYECTO	LIDER_PROYECTO	CIUDAD_ EJECUCION	PRI
P005BA0005	PEN DRIVES	JIM HARNE	BARRANQUILLA	1
P005BA0009	MOTOR DEDUCTIVE DATABASE	SAIM KINCHNER	BARRANQUILLA	1
P005BA0010	WEB SEMANTIC SOFTWARE V.99	ANGELA LEIBNITZ	BARRANQUILLA	1
P005BA0011	MOTOR INFERENCIAL DB	CONRA LIBTZ	BARRANQUILLA	1
P005BA0012	WEB DATABASE	MARIE CURLITZ	BARRANQUILLA	1

Figura 4.28 Proyecto del SIM ejecutados en Barranquilla con prioridad uno

4.5.9 Selección de tuplas basada en rango

El desarrollo de condiciones de búsqueda con base en rangos en la cláusula SELECT se realiza con las palabras BETWEEN / NOT BETWEEN; palabras que ubican las tuplas de la consulta en los rangos relacionados en la consulta.

Ejemplo 4.10 Selección de productos del SIM por rango

Listar los productos fabricados del SIM cuya cantidad en el rango de producción esté ubicada en el intervalo cerrado [10;25].

Análisis: De la tabla de productos se seleccionan aquellas tuplas situadas en el rango cuya cantidad esté ubicada entre 10 y 25.

Resultados: Los resultados de la consulta se presentan en la figura 4.29.

```
SELECT *
FROM productos
/* La palabra BETWEEN es una palabra reservada que ubica el
rango de los productos */
WHERE cantidad_productos BETWEEN 10 AND 25;
```

```
SELECT *
FROM productos
WHERE cantidad_productos BETWEEN 10 AND 25;
```

COD_ PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
100-100-101-000	PORTATIL NGP	T	800000	20	P001BA0001
100-100-101-001	PROCESSOR XTS	P	400000	10	P002NY0002
100-100-101-007	MOTOR DEDUCTIVE	T	1000000	20	P005BA0009
100-100-101-008	WEB 10.4 NGWEB	T	1800000	20	P005BA0010
100-100-101-009	MOTOR AIRPL. XT CONTROL TAB.	T	1000000	20	P005BA0011
100-100-101-013	BODY TRACKING XC-XB-XD	T	180000	12	P008TA0015
100-100-101-018	OPTICAL CONTROL FBA-12	T	100000	10	P012BO0222

Figura 4.29 Búsqueda por rangos de los productos del SIM con número de producción que están en el intervalo [10;25]

Ejemplo 4.11 Listado de empleados del SIM por rangos salariales

Producir un listado de empleados no ubicados en los rangos salariales ente 2 y 5 millones de pesos.

Análisis: Con base en la relación o tabla de empleados del SIM se selecciona los atributos de identidad del empleado, teniendo en cuenta que en la cláusula WHERE se filtran aquellas tuplas no situadas (NOT BETWEEN) en el rango entre 2 y 5 millones de pesos.

Resultados: La consulta se estructura de la siguiente forma, con las identidades de los empleados en sus códigos y nombres:

```
SELECT Id_empleado,nombre_empleado, sueldo
FROM empleados
/* NOT BETWEEN son palabras reservadas */
WHERE sueldo NOT BETWEEN 2000000 AND 5000000
```

Los resultados se muestran en la figura 4.30.

```
SELECT Id_empleado,nombre_empleado, sueldo
FROM empleados
WHERE sueldo NOT BETWEEN 2000000 AND 5000000
```

ID_EMPLEADO	NOMBRE_EMPLEADO	SUELDO
72415240	CONSUEGRA AMIN PEDRO DANIEL	1000000
74415240	AYOS RENOWISKY ANDRÉS JULIAN	1500000
74415393	VILLAR ARTEAGA LUIS ALBERTO	1000000
74415597	PEREZ PEREZ ANDREA	1200000
74415648	AGUIRRE PEREZ RUTH ELENA	1000000
74415699	SILVESTRE CUELLO ANA	800000
74415750	VISBAL PEREZ SOFIA	800000
74415903	LOPEZ DIAZ RUBEN DANIEL	1000000
74416005	GUERRA PORTO LUIS MIGUEL	900000
74416056	FABREGAS COTO LUIS DANIEL	900000
74416107	JUVENAL HINCAPIE ELEONORA	980000
74416158	PUYANA KARKER CLAUDIA JULIANA	980000

Figura 4.30 Salarios de empleados del SIM no ubicados en el rango salarial entre 2 y 5 millones

4.5.10 Selección de tuplas por pertenencia a un conjunto

La pertenencia o no a un conjunto en la cláusula SELECT se hace con las palabras IN/NOT IN, respectivamente; en las cuales IN comprueba la pertenencia de un atributo

de la tabla a una lista dentro de base de datos; en el caso de NOT IN, cumple la función de la no pertenencia del atributo a la lista.

Ejemplo 4.12 Selección de tuplas del SIM por pertenencia a un conjunto

Generar un listado de la maquinaria del SIM que esté en mantenimiento preventivo.

Análisis: Con base en la tabla de maquinaria del SIM se seleccionan aquellas tuplas que en su tipo de mantenimiento tienen la letra P, que significa que están dentro del conjunto de mantenimientos preventivos.

Resultados: Se muestran en la figura 4.31, según la consulta.

```
/* Maquinaria en proceso de mantenimiento preventivo ( P ) */
SELECT *
FROM mantenimiento
WHERE tipo_mantenimiento IN ( 'P' )
```

```
SELECT *
FROM mantenimiento
WHERE tipo_mantenimiento IN ( 'P' )
```

COD_ MANTENIMIENTO	FECHA_ INICIAL	FECHA_ FINAL	TIP	COMENTARIOS	NO_MAQUINA
MT-RK1214001	12-DEC-20	13-DEC-20	P	Falta de Repuesto	MA-RK1214001
MT-RF1222002	01-DEC-18	02-DEC-18	P	Falta de Repuesto	MA-RF1222002
MT-RZ1214004	14-DEC-20	14-DEC-20	P	Sin verficación	MA-RZ1214004
MT-TU9456006	08-DEC-20	09-DEC-20	P	Base de inicio de linea	MA-TU9456006
MT-RF1222008	11-APR-19	11-APR-19	P	Aseguramiento de calidad	MA-RF1222008

Figura 4.31 Maquinaria que está en mantenimiento preventivo o pertenece al conjunto P

Ejemplo 4.13 Selección de tuplas con pertenencia a múltiples conjuntos

Listar los proyectos nacionales realizados por el Sistema Integrado de Manufactura con niveles de prioridad uno y dos.

Análisis: De la tabla de proyectos del SIM se seleccionan aquellos cuyas ciudades de ejecución estén situadas en Bogotá y Barranquilla; y adicionalmente, los códigos de prioridad de dichos proyectos estén ubicados en el conjunto (1,2); esta consulta corresponde a la ejemplificación de una combinación compuesta seleccionando las tuplas por varios conjuntos.

Resultados: Los resultados se presentan en la figura 4.32, de acuerdo con la siguiente consulta:

```
/*Condición de búsqueda basada en la pertenencia a un conjunto */
SELECT No_proyecto,Nombre_proyecto , ciudad_ejecucion,lider_
proyecto,prioridad_proyecto
FROM proyectos
/* Tuplas pertenecientes a proyecto nacionales */
WHERE ciudad_ejecucion IN ('BARRANQUILLA','BOGOTA')
/* Filtra las tuplas con proyectos cuyas prioridades son 1 y 2 */
AND prioridad_proyecto IN (1,2)
```

```
SELECT No_proyecto,Nombre_proyecto, ciudad_ejecucion,lider_
proyecto,prioridad_proyecto

FROM proyectos

WHERE ciudad_ejecucion IN ('BARRANQUILLA','BOGOTA') AND
prioridad_proyecto IN (1,2)
```

NO_ PROYECTO	NOMBRE_PROYECTO	CIUDAD_EJECUCION	LIDER_PROYECTO	PRI
P001BA0001	ENSAMBLE DE COMPUTADORES	BARRANQUILLA	HARKER JOHN	2
P005BA0005	PEN DRIVES	BARRANQUILLA	JIM HARNE	1
P005BA0009	MOTOR DEDUCTIVE DATABASE	BARRANQUILLA	SAIM KINCHNER	1
P005BA0010	WEB SEMANTIC SOFTWARE V.99	BARRANQUILLA	ANGELA LEIBNITZ	1
P005BA0011	MOTOR INFERENCIAL DB	BARRANQUILLA	CONRA LIBTZ	1
P005BA0012	WEB DATABASE	BARRANQUILLA	MARIE CURLITZ	1
P006BO0013	OPERATING SYSTEMS PROD.	BOGOTA	JHON HAINSERBERG	1
P010BA0015	OPTICAL CHIP	BARRANQUILLA	ALDRIN HARKER	2
P020BA0016	LIGHT PEN SENSOR	BARRANQUILLA	PHILIPH FAIRHEAD	2
P011BO0017	EXPLORER BD SYSTEM	BOGOTA	KIM STRAUUS	1
P012BO0222	INTERFACE FOR OPTICAL CON.	BOGOTA	JUAN MIGUEL OTALORA	1

Figura 4.32 Proyectos nacionales con prioridades asociadas en [1,2]

Ejemplo 4.14 Listado de tuplas por la no pertenencia (NOT IN) a un conjunto

Listar las identidades de los empleados no extranjeros que no sean técnicos ni ingenieros que trabajen en el SIM.

Análisis: Utilizando la tabla de empleados, proyectos los atributos de sus identidades (edificación del empleado, su nombre, su ciudad de nacimiento y el cargo) se filtran aquellas tuplas para las cuales los trabajadores no pertenezcan a los grupos de

técnicos ni ingenieros ni pertenezcan a las ciudades de nacimiento diferentes de las situadas en el territorio nacional; se especifican en el conjunto las ciudades extranjeras New York y Miami (USA).

Resultados: La consulta da como resultado la salida de la figura 4.33 al correr la consulta

```
/* Ejemplificación de la no pertenencia NOT IN a un conjunto */
SELECT Id_empleado, nombre_empleado, ciudad_nac,cargo_empleado
FROM empleados
/* Empleados no en los cargos de de técnicos e ingenieros */
WHERE cargo_empleado NOT IN ('TECNICO','INGENIERO')
/* Empleados no extranjeros */
AND ciudad_nac NOT IN ('NEW YORK','MIAMI')
```

```
SELECT Id_empleado,nombre_empleado,ciudad_nac,cargo_
empleado
FROM empleados
WHERE cargo_empleado NOT IN ('TECNICO','INGENIERO')
AND ciudad_nac NOT IN ('NEW YORK','MIAMI')
```

ID_EMPLEADO	NOMBRE_EMPLEADO	CIUDAD_NAC	CARGO_EMPLEADO
74415444	VILLAMIZAR ATYOS ANDRÉS	BARRANQUILLA	DOCTOR
74415495	JULIO CONTRERAS JUAN CARLOS	CARTAGENA	DOCTOR
74415597	PEREZ PEREZ ANDREA	MADRID	ASISTENTE
74415699	SILVESTRE CUELLO ANA	SANTA MARTA	ASISTENTE
74415750	VISBAL PEREZ SOFIA	BARRANQUILLA	ASISTENTE
74416005	GUERRA PORTO LUIS MIGUEL	CALI	ADMINISTRADOR
74416107	JUVENAL HINCAPIE ELEONORA	CARACAS	ADMINISTRADOR

Figura 4.33 Empleados del SIM no extranjeros y no técnicos ni ingenieros

4.5.11 Selección de tuplas utilizando patrones

La búsqueda de información en una base de datos por correspondencia de patrones se basa en la operación de la cláusula SELECT en las palabras LIKE y NOT LIKE.

Ejemplo 4.15 Selección de tuplas utilizando las cláusulas LIKE y NOT LIKE

Generar un listado que muestre todas las cuentas de los bancos de cuentas del plan contable del SIM; las cuentas de bancos locales son aquellas en las cuales el código de la cuenta tiene la sarta o cadena de caracteres 10-001-001-011.

Análisis: De la tabla cuentas del SIM se seleccionan aquellas tuplas de la tabla "cuentas" para las cuales el código de la cuenta cumple con el valor de los caracteres relacionados.

Resultados: La consulta SQL se muestra en la Figura 4.34 corriendo el siguiente código:

```
SELECT *
FROM cuentas
/* El patrón de caracteres que se va a utilizar se coloca
dentro de los porcentajes %___% */
WHERE cod_cuenta LIKE '%10-001-001-011%' ;
```

```
SELECT *
FROM cuentas
WHERE cod_cuenta LIKE '%10-001-001-011%' ;
```

COD_CUENTA	NOMBRE_CUENTA	SALDO_ ANTERIOR	MOVIMIENTO	SALDO_ ACTUAL	TIP
10-001-001-011-000	BANCOS LOCALES	10000000	300	10000300	P
10-001-001-011-001	BANCO DE BOGOTÁ	1000000	10000	1010000	P
10-001-001-011-002	BANCO DE COLOMBIA	2000000	200	2000200	P
10-001-001-011-003	BANCO SANTANDER	700000000	500	700000500	P

Figura 4.34 Selección de tuplas con base en una secuencia de caracteres (% … %) del patrón

Ejemplo 4.16 Producir un listado de empleados que tengan en sus nombres la letra D, seguida por cualquier conjunto de tres caracteres, así: D _ _ _ _ _

La siguiente consulta SQL muestra los resultados en la figura 4.35.

```
SELECT Id_empleado,nombre_empleado,email_empleado
FROM empleados
/* El patrón en la consulta es una cadena de 6 caracteres y el
primero es D */
WHERE nombre_empleado LIKE '%D_____'
```

```
SELECT Id_empleado,nombre_empleado,email_empleado
FROM empleados
WHERE nombre_empleado LIKE '%D_____'
```

ID_EMPLEADO	NOMBRE_EMPLEADO	EMAIL_EMPLEADO
72415240	CONSUEGRA AMIN PEDRO DANIEL	pcons@hotmail.com
74415903	LOPEZ DIAZ RUBEN DANIEL	lodiru677@hotmail.com
74416056	FABREGAS COTO LUIS DANIEL	facat45@hotmail.com

Figura 4.35 Búsqueda de patrones por cadena de caracteres de los cuales el primero es D, y se nota que en todas las tuplas está la palabra DANIEL

Ejemplo 4.17 Listar los productos asociados a los proyectos cuyo número de proyecto termine en 1

Los resultados de la consulta mostrados en la Figura 4.36 son ejecutados por la consulta

```
SELECT *
FROM productos
/* Se seleccionan cualquier secuencia de caracteres con
longitud de al menos 1, el último de los cuales es 1 */
WHERE No_proyecto LIKE '%1'
```

```
SELECT *
FROM productos
WHERE No_proyecto LIKE '%1'
```

COD_ PRODUCTO	NOMBRE_ PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
100-100-101-000	PORTATIL NGP	T	800000	20	P001BA0001
100-100-101-009	MOTOR AIRPL. XT CONTROL TAB.	T	1000000	20	P005BA0011
100-100-101-019	CAD GRAPHIC SYSTEMS	P	5000	1	PA11TA1111

Figura 4.36 Selección de cualquier secuencia de caracteres de No_proyecto, siendo el último carácter igual a 1

4.5.12 Selección de tuplas utilizando con atributos nulos

Las tuplas de una base de datos pueden ser consultadas en función de los atributos vacíos de las tablas de la base de datos. Los campos vacíos de dichos atributos se consultan con la palabra NULL; por su parte, los campos cuyos atributos no son vacíos o tienen información se pueden consultar con las palabras NOT NULL.

Ejemplo 4.18 Seleccionar las máquinas del SIM que estando en mantenimiento no tienen comentarios asociados a sus números de órdenes de mantenimiento

Los resultados de la consulta que se presentan en la figura 4.37 son generados a través de la siguiente consulta SQL:

```
SELECT *
FROM mantenimiento
/* Notar en la figura que el contenido de información de
comentarios está vacío */
WHERE comentarios IS NULL
```

```
SELECT *
FROM mantenimiento
WHERE comentarios IS NULL
```

COD_MANTENIMIENTO	FECHA_ INICIAL	FECHA_ FINAL	TIP	COMENTARIOS	NO_MAQUINA
MA-RK1214002	12-JUN-12	13-JUN-13	P		MA-RK1214002
MB-RK1214003	12-JUN-12	13-JUN-14	P		MB-RK1214003
MC-RK1214004	13-JUL-12	13-JUL-14	P		MC-RK1214004

Figura 4.37 Utilización de NULL en la selección de tuplas

Ejemplo 4.19 Seleccionar del SIM las órdenes de mantenimiento con comentarios asociados.

Los resultados de la consulta derivados de la proposición de SQL que se presentan a continuación se muestran en la figura 4.38.

```
SELECT *
FROM mantenimiento
/* Extrae de la base de datos del SIM los campos con contenido
de información no nulos */
WHERE comentarios IS NOT NULL
```

```
SELECT *
FROM mantenimiento
WHERE comentarios IS NOT NULL
```

COD_MANTENIMIENTO	FECHA_ INICIAL	FECHA_ FINAL	TIP	COMENTARIOS	NO_MAQUINA
MT-RK1214001	12-DEC-20	13-DEC-20	P	Falta de Repuesto	MA-RK1214001
MT-RF1222002	01-DEC-18	02-DEC-18	P	Falta de Repuesto	MA-RF1222002
MT-TZ9456003	02-JAN-19	05-JAN-19	C	Verificación de exactitud	MA-TZ9456003
MT-RZ1214004	14-DEC-20	14-DEC-20	P	Sin verificación	MA-RZ1214004
MT-RL1222005	11-JAN-21	18-JAN-21	C	En espera de repuesto prov.	MA-RL1222005
MT-TU9456006	08-DEC-20	09-DEC-20	P	Base de inicio de línea	MA-TU9456006
MT-RZ1214007	07-APR-20	08-APR-20	C	Fallo de corriente	MA-RZ1214007
MT-RF1222008	11-APR-19	11-APR-19	P	Aseguramiento de calidad	MA-RF1222008
MT-TY9456009	20-DEC-20	20-DEC-20	C	En espera de repuesto prov.	MA-TY9456009

Figura 4.38 Utilización NOT NULL en la selección de tuplas de la base de datos del SIM

El cierre de las tuplas totales de la tabla de mantenimiento se presenta en la figura 4.39.

```
SELECT *
FROM mantenimiento
```

COD_MANTENIMIENTO	FECHA_ INICIAL	FECHA_ FINAL	TIP	COMENTARIOS	NO_MAQUINA
MT-RK1214001	12-DEC-20	13-DEC-20	P	Falta de Repuesto	MA-RK1214001
MT-RF1222002	01-DEC-18	02-DEC-18	P	Falta de Repuesto	MA-RF1222002
MT-TZ9456003	02-JAN-19	05-JAN-19	C	Verificación de exactitud	MA-TZ9456003
MT-RZ1214004	14-DEC-20	14-DEC-20	P	Sin verificación	MA-RZ1214004
MT-RL1222005	11-JAN-21	18-JAN-21	C	En espera de repuesto prov.	MA-RL1222005
MT-TU9456006	08-DEC-20	09-DEC-20	P	Base de inicio de línea	MA-TU9456006
MT-RZ1214007	07-APR-20	08-APR-20	C	Fallo de corriente	MA-RZ1214007
MT-RF1222008	11-APR-19	11-APR-19	P	Aseguramiento de calidad	MA-RF1222008
MT-TY9456009	20-DEC-20	20-DEC-20	C	En espera de repuesto prov.	MA-TY9456009
MA-RK1214002	12-JUN-12	13-JUN-13	P		MA-RK1214002
MB-RK1214003	12-JUN-12	13-JUN-14	P		MB-RK1214003
MC-RK1214004	13-JUL-12	13-JUL-14	P		MC-RK1214004

Figura 4.39 Tuplas totales de la tabla de mantenimiento del SIM

4.5.13 Generación de resultados ordenados

4.5.13.1 Generación de tuplas con un atributo ordenado

Las tuplas generadas en las consultas anteriores no están ordenadas por ninguno de los atributos que componen las tablas. La estructura de sintaxis para seleccionar tuplas en SQL con la cláusula ORDER BY se presenta a continuación:

```
SELECT [DISTINCT] [ALL] {* | [expresionColumna [AS nuevoNombre]] […]}
FROM NombreTabla
[WHERE condición]
[ORDER BY listaColumnas]
```

Luego, el ORDER BY ordena los resultados de las tuplas por la(s) columna(s) especificada(s) en la lista de columnas. Cuando se requiere que las tuplas se ordenen ascendentemente se utiliza la palabra ASC (ASCENDING key) o, en su defecto, si se requiere ordenar las columnas en orden descendente, se usa DESC.

Ejemplo 4.20 Producir un listado de transacciones del SIM ordenadas ascendentemente por el campo o atributo de movimiento de la transacción.

La figura 4.40 presenta los resultados de la siguiente consulta:

```
SELECT *

FROM transacción

/* El atributo por el cual se va a ordenar la tabla es el campo
de movimiento de la transacción.

 Notar que los resultados del campo "movimiento" están
ordenados de menor a mayor */

ORDER BY movimiento ASC.
```

```
SELECT *
FROM transaccion
ORDER BY movimiento ASC
```

NO_TRANSACCION_ CONTABLE	FECHA_ TRANSACCION	MOVIMIENTO	TIP	DETALLE_ TRANSACCION	COD_CUENTA
TC-22-12-19-0001	15-APR-19	1000	D	REGISTRO A ACTIVO	10-000-000-000-000
TC-18-12-19-0003	21-DEC-19	1000	D	COMPRA DE MÁQUINA	10-001-001-001-005
TC-19-12-19-0010	03-DEC-19	1000	D	PAGO A PROVEEDOR	10-001-001-001-004
TC-22-12-19-0020	26-DEC-19	5000	C	PAGO EN DOLARES A PROV.INT	01-001-001-001-998
TC-20-12-19-0005	20-DEC-19	5000	D	RETIRO ELECTRÓNICO	10-001-001-001-003
TC-22-12-19-0016	21-DEC-19	40000	D	FIN DEL CONTRATO	10-001-001-011-001
TC-22-12-19-0003	22-DEC-19	40000	D	REGISTRO A BANCOS	10-001-001-000-000
TC-14-12-18-0002	28-DEC-19	50000	D	RETORNO DE MERC. NO ENVIADA	10-001-001-011-000
TC-22-12-19-0002	15-DEC-19	50000	D	TRANSF. BANCARIA	10-001-000-000-000
TC-22-12-19-0017	22-DEC-19	60000	D	FIN DE PAGO DE LA IMPORTACIÓN	10-001-001-011-002
TC-21-15-18-0001	24-DEC-19	60000	D	TRANF. CTA. EXTRANJERA	10-001-001-001-000
TC-21-12-19-0004	15-DEC-19	70000	D	REGISTRO	10-001-001-001-001
TC-22-12-19-0018	13-DEC-19	80000	D	COMPRA DE MÁQUINA	10-001-001-011-003
TC-22-10-19-0002	16-DEC-19	100000	D	TRANF. BANCARIA	10-001-001-001-002
TC-22-12-19-0021	15-DEC-19	500000	C	DEVOLUCIÓN DE COMPRA	01-001-001-001-997
TC-22-12-19-0019	24-DEC-19	1005000	D	PAGO A PROVEEDOR	01-001-001-001-999

Figura 4.40 Utilización de la cláusula ORDER BY

4.5.13.2 Generación de tuplas con múltiples atributos ordenados

La cláusula ORDER BY se puede aplicar en el ordenamiento de los resultados de varias columnas, según la lista de columnas de acuerdo con la sintaxis:

```
SELECT [DISTINCT] [ALL] {* | [expresionColumna [AS nuevoNombre]] […]}
FROM NombreTabla
[WHERE condición]
[ORDER BY Columna1, Columna2,…,Columnai,…Columnan ASC DESC]
```

Con la sintaxis presentada, las tuplas se organizan en orden por la columna1, y dentro de la columna1 por la columna2, y así sucesivamente, tal y como se presenta a continuación.

Ejemplo 4.21 Generar un listado que muestre los movimientos de la tabla de transacciones del SIM ordenados ascendentemente, y dentro de cada intervalo de movimiento ordenar las tuplas por detalle de la transacción en orden ascendente

---

Los resultados de las consultas se muestran en la figura 4.41, según la siguiente consulta:

```
SELECT *
FROM transacción
/* El primer atributo por el cual se ordenan las tuplas es por
movimiento (Columna1); el segundo atributo de orden ascendente
es por detalle la transacción (Columna2).
Notar que en cada intervalo 1000 dentro de él se ordenan los
detalles <C...,P…,R…>. */
ORDER BY movimiento, detalle_transaccion ASC
```

```
SELECT *
FROM transaccion
ORDER BY movimiento,detalle_transaccion ASC
```

NO_TRANSACCION_ CONTABLE	FECHA_ TRANSACCION	MOVIMIENTO	TIP	DETALLE_ TRANSACCION	COD_CUENTA
TC-18-12-19-0003	21-DEC-19	1000	D	COMPRA DE MÁQUINA	10-001-001-001-005
TC-19-12-19-0010	03-DEC-19	1000	D	PAGO A PROVEEDOR	10-001-001-001-004
TC-22-12-19-0001	15-APR-19	1000	D	REGISTRO A ACTIVO	10-000-000-000-000
TC-22-12-19-0020	26-DEC-19	5000	C	PAGO EN DOLARES A PROV.INT	01-001-001-001-998
TC-20-12-19-0005	20-DEC-19	5000	D	RETIRO ELECTRÓNICO	10-001-001-001-003
TC-22-12-19-0016	21-DEC-19	40000	D	FIN DEL CONTRATO	10-001-001-011-001
TC-22-12-19-0003	22-DEC-19	40000	D	REGISTRO A BANCOS	10-001-001-000-000
TC-14-12-18-0002	28-DEC-19	50000	D	RETORNO DE MERC. NO ENVIADA	10-001-001-011-000
TC-22-12-19-0002	15-DEC-19	50000	D	TRANSF. BANCARIA	10-001-000-000-000
TC-22-12-19-0017	22-DEC-19	60000	D	FIN DE PAGO DE LA IMPORTACIÓN	10-001-001-011-002
TC-21-15-18-0001	24-DEC-19	60000	D	TRANF. CTA. EXTRANJERA	10-001-001-001-000
TC-21-12-19-0004	15-DEC-19	70000	D	REGISTRO	10-001-001-001-001
TC-22-12-19-0018	13-DEC-19	80000	D	COMPRA DE MÁQUINA	10-001-001-011-003
TC-22-10-19-0002	16-DEC-19	100000	D	TRANF. BANCARIA	10-001-001-001-002
TC-22-12-19-0021	15-DEC-19	500000	C	DEVOLUCIÓN DE COMPRA	01-001-001-001-997
TC-22-12-19-0019	24-DEC-19	1005000	D	PAGO A PROVEEDOR	01-001-001-001-999

Figura 4.41 Selección de tupla por ordenamiento de varias columnas o atributos de una tabla

La cláusula ORDER BY también se puede utilizar con expresiones y campos calculados, tal y como se presenta a continuación.

Ejemplo 4.22 Generar un listado de empleados del SIM ordenado ascendentemente por el cargo del funcionario, y dentro de los cargos ordenarlo ascendentemente por el salario del empleado, considerando un aumento del 5 por ciento

Los resultados se presentan en la Figura 4.42, de acuerdo con la siguiente consulta:

```
SELECT id_empleado,nombre_empleado,cargo_empleado,sueldo*1.05
FROM empleados
/* Los empleados se presentan ascendentemente por cargos; y
dentro de los cargos se ordenan los salarios de los empleados
con un aumento del 5% de menor a mayor salario. */
ORDER BY cargo_empleado ASC,sueldo*1.05 ASC
```

```
SELECT id_empleado,nombre_empleado,cargo_
empleado,sueldo*1.05
FROM empleados
ORDER BY cargo_empleado ASC,sueldo*1.05 ASC
```

ID_EMPLEADO	NOMBRE_EMPLEADO	CARGO_ EMPLEADO	SUELDO*1.05
74416056	FABREGAS COTO LUIS DANIEL	ADMINISTRADOR	945000
74416005	GUERRA PORTO LUIS MIGUEL	ADMINISTRADOR	945000
74416158	PUYANA KARKER CLAUDIA JULIANA	ADMINISTRADOR	1029000
74416107	JUVENAL HINCAPIE ELEONORA	ADMINISTRADOR	1029000
74415750	VISBAL PEREZ SOFIA	ASISTENTE	840000
74415699	SILVESTRE CUELLO ANA	ASISTENTE	840000
74415597	PEREZ PEREZ ANDREA	ASISTENTE	1260000
74415444	VILLAMIZAR ATYOS ANDRÉS	DOCTOR	5250000
74415495	JULIO CONTRERAS JUAN CARLOS	DOCTOR	5250000
74415291	GUTIEREZ DÍAZ ANDRES	INGENIERO	3150000
74415546	NANDIN LOPERA JULIANA	INGENIERO	3150000
74415801	VEGA BAENA CARMEN	INGENIERO	3990000
74415954	HERRERA LORA ANNY	INGENIERO	4515000
74415852	SANDOVAL HUCKE MARY	INGENIERO	4725000
74415342	GOENAGA MAYA JULIO	INGENIERO	4725000
74415903	LOPEZ DIAZ RUBEN DANIEL	TECNICO	1050000
74415393	VILLAR ARTEAGA LUIS ALBERTO	TECNICO	1050000
72415240	CONSUEGRA AMIN PEDRO DANIEL	TECNICO	1050000
74415648	AGUIRRE PEREZ RUTH ELENA	TECNICO	1050000
74415240	AYOS RENOWISKY ANDRÉS JULIAN	TECNICO	1575000

Figura 4.42 Utilización de la cláusula ORDER BY con campos calculados

4.5.14 Generación de resultados con funciones de agregación

Las funciones de agregación se utilizan para generar en los reportes derivados de la base de datos, síntesis o resumen de resultados. Las principales funciones de agregación se muestran en la tabla 4.3.

Tabla 4.3 Funciones de agregación según el estándar ISO

Nombre de la Función	Utilidad
AVG	Calcula el promedio de los valores contenidos en los atributos de una tabla.
COUNT	Calcula el número de valores contenidos en una columna específica.
SUM	Calcula la suma de los contenidos de una columna específica de una tabla.
MIN	Calcula el valor más pequeño de una columna de una tabla.
MAX	Calcula el mayor valor de una columna de una tabla.

4.5.14.1 Conteo de tuplas

La estructura sintáctica para el conteo de tuplas en SQL se representa como:

```
SELECT COUNT [DISTINCT] {* [expresionColumna [AS nuevoNombre ]] […]}
FROM NombreTabla
[WHERE condición]
```

Ejemplo 4.23 Contar el número de proyectos de la base de datos del SIM con alta prioridad (aquellos en los cuales la prioridad del proyecto es igual a uno).

Los resultados se presentan en la figura 4.43, al operar en el gestor de la base la siguiente consulta:

```
/* COUNT (*) Se cuentan los proyectos de prioridad de proyecto
asociada igual a 1 */
SELECT COUNT (*) AS "PROYECTOS DE ALTA PRIORIDAD"
FROM proyectos
WHERE prioridad_proyecto = 1
```

```
SELECT COUNT (*) AS "PROYECTOS DE ALTA PRIORIDAD"
FROM proyectos
WHERE prioridad_proyecto = 1
```

PROYECTOS DE ALTA PRIORIDAD
14

Figura 4.43 Utilización de la cláusula COUNT (*)

4.5.14.2 Conteo de tuplas distintas

Cuando se requiere contar el número de tuplas diferentes de una tabla de la base de datos, se utiliza la palabra DISTINCT en la cláusula SELECT.

Ejemplo 4.24 Contar el número de mantenimientos distintos que se utilizan en el Sistema Integrado de Manufactura

La solución se muestra en la figura 4.44 y es generada por esta consulta:

```
/* Se cuentan los mantenimientos distintos, P = preventivo y C
= Correctivo */
SELECT COUNT (DISTINCT tipo_mantenimiento) AS "TIPOS DE
MANTENIMIENTO"
FROM mantenimiento
```

```
SELECT COUNT (DISTINCT tipo_mantenimiento) AS "TIPOS DE
MANTENIMIENTO"
FROM mantenimiento
```

TIPOS DE MANTENIMIENTO
2

Figura 4.44 Utilización de COUNT (DISTINCT)

4.5.14.3 Utilización de las funciones de agregación Mínimo, Máximo y Promedio

Ejemplo 4.25 Calcular los valores mínimos, máximos y promedios de los montos de los costos de inversión de los proyectos del SIM

Los resultados de la resolución de la consulta se presentan en la figura 4.45 al ejecutar la siguiente consulta:

```
SELECT
/* La función MIN devuelve el valor más pequeño del atributo
costo_proyecto */
MIN(costo_proyecto) AS "MONTO MÍNIMO",
/* La función MAX devuelve el mayor valor del atributo costo_
proyecto */
MAX(costo_proyecto) AS "MONTO MÁXIMO",
/* La función AVG devuelve el valor promedio del atributo
costo_proyecto */
AVG(costo_proyecto) AS "MONTO PROMEDIO"
FROM proyectos
```

```
SELECT
MIN(costo_proyecto) AS "MONTO MÍNIMO",
MAX(costo_proyecto) AS "MONTO MÁXIMO",
AVG(costo_proyecto) AS "MONTO PROMEDIO"
FROM proyectos
```

MONTO MÍNIMO	MONTO MÁXIMO	MONTO PROMEDIO
10000	5124600000	330768956

Figura 4.45 Utilización de MIN, MAX y AVG

4.5.15 Generación de resultados con funciones de agrupación

4.5.15.1 Selección de tuplas utilizando agrupación (GROUP BY)

Los resultados generados de una base de datos se pueden clasificar en tres tipos: i) Listados, ii) Reportes y iii) Informes.

Los *listados* en la mayoría de los casos son un vaciado de una o varias tablas de la base de datos, cuyo fin es la validación o verificación de las informaciones contenidas en la base de datos. Los *reportes* son listados con un mayor nivel de procesamiento, dentro de los cuales se incluyen campos de resumen o de totales correspondientes a uno o varios atributos de contenido de una tabla de la base de datos; son necesarias en este caso las funciones de agregación anteriormente enunciadas. Los *informes* son más elaborados en sus resultados y pueden requerir agrupación de datos, y en dicha agrupación se utilizan funciones de agregación; para la generación de este tipo de resultados se utiliza la cláusula GROUP BY, cuya funcionalidad es agrupar los datos de las tablas indicadas en el SELECT por los atributos de agrupamiento. Los informes para dar una respuesta más significativa al usuario se pueden complementar con la utilización en el SELECT de la cláusula ORDER BY.

La sintaxis de la cláusula SELECT junto con las funciones de ordenamiento y agrupación es:

```
SELECT [DISTINCT] [ALL] {* | [expresionColumna [AS nuevoNombre]] […]}
FROM   NombreTabla
[WHERE condición]
[GROUP BY listaColumnas] [HAVING condición]
[ORDER BY listaColumnas]
```

Ejemplo 4.26 Producir un informe por números de proyectos del SIM ordenados ascendentemente por números productos en el que para cada proyecto se cuente el número de empleados asociados y la suma de sus salarios por proyecto

Los resultados del informe derivados de la consulta que se muestra a continuación se presentan en la figura 4.46.

```
SELECT No_proyecto,COUNT(Id_empleado), SUM (sueldo)
/* Dentro de cada proyecto, se cuentan las idenficaciones de
los empleados */
FROM empleados
/* El informe es agrupado por los números de proyectos */
GROUP BY No_proyecto
/* Los proyectos son ordenados por No_proyecto */
ORDER BY No_proyecto
```

```
SELECT No_proyecto,COUNT(Id_empleado), SUM (sueldo)
FROM empleados
       GROUP BY No_proyecto
ORDER BY No_proyecto
```

NO_PROYECTO	COUNT(ID_EMPLEADO)	SUM(SUELDO)
P001BA0001	4	3680000
P002NY0002	2	5300000
P003MI0003	3	8700000
P004SE0004	3	6500000
P005BA0005	2	5300000
P005BA0009	1	3000000
P005BA0012	1	800000
P006NY0008	2	5900000
P006SE0007	1	5000000
PA11TA1111	1	980000

Figura 4.46 Uso de la cláusula GROUP BY

4.5.15.2 Selección de tuplas utilizando totales de cuenta (HAVING COUNT)

Ejemplo 4.27 Generar un informe agrupado por números de proyectos del SIM ordenados ascendentemente por números de proyectos, en el que para cada proyecto se cuente el número de empleados asociados y la suma de sus salarios por proyecto, para aquellos proyectos que reporten más de dos empleados asociados al proyecto

Los resultados de la consulta se presentan en la figura 4.47 al operar el siguiente SELECT:

```
SELECT No_proyecto,COUNT(Id_empleado), SUM (sueldo)
FROM empleados
/* Se agrupan los proyectos por No_proyecto, contando los
empleados que pertenecen al Proyecto */
GROUP BY No_proyecto
/* Se filtra el conteo para aquellos proyectos cuya cuenta
tenga más de dos empleados */
HAVING COUNT (Id_empleado) > 2
/* Se realiza el ordenamiento por el número del proyecto */
ORDER BY No_proyecto
```

```
SELECT No_proyecto,COUNT(Id_empleado), SUM (sueldo)
FROM empleados
GROUP BY No_proyecto
HAVING COUNT (Id_empleado) > 2
ORDER BY No_proyecto
```

NO_PROYECTO	COUNT(ID_EMPLEADO)	SUM(SUELDO)
P001BA0001	4	3680000
P003MI0003	3	8700000
P004SE0004	3	6500000

Figura 4.47 Utilización de la cláusula HAVING COUNT

4.6 Consultas compuestas

4.6.1 Selección de tuplas utilizando subconsultas

Las subconsultas en SQL son estructuras con consultas anidadas, en las que en una instrucción SELECT se encapsula otra instrucción SELECT, y dentro de esta última a su vez se encapsula otra, y así sucesivamente; sujeto el proceso de encapsulamiento al cumplimiento de una determinada condición, como se presenta en el siguiente formato:

```
SELECT [DISTINCT][ALL] {* | [expresionColumna [AS nuevoNombre]] […]}
FROM NombreTabla
[WHERE condición
   [SELECT [DISTINCT][ALL] {* [expresionColumna [ AS
     nuevoNombre ]] […]}
     FROM NombreTabla
     [WHERE condición… | [HAVING condición]
     [GROUP BY listaColumnas] [HAVING condición]
     [ORDER BY listaColumnas]]
```

Las instrucciones de los SELECT anidados o internos reciben el nombre de "subconsultas", y para obtener los resultados las subconsultas utilizan la ejecución de los SELECT externos; teniendo en cuenta que para el cumplimiento de la condición de la subconsulta se pueden utilizar las cláusulas WHERE y HAVING.

4.6.1.1 Generación de subconsultas con el operador de igualdad

Ejemplo 4.28 Generar un listado de productos que pertenezcan a proyectos liderados por Linda Day.

Los resultados de la consulta se presentan en la figura 4.48, según el texto SQL mostrado a continuación:

```
/* Consulta externa */
SELECT cod_producto,nombre_producto,tipo_producto,costo_
producto
FROM productos
WHERE No_proyecto = ( /* Consulta anidada o subconsulta */
SELECT No_proyecto
             FROM proyectos
             WHERE lider_proyecto = 'LINDA DAY')
```

```
SELECT cod_producto,nombre_producto,tipo_producto,costo_
producto
FROM productos
WHERE No_proyecto = ( SELECT No_proyecto
              FROM proyectos
              WHERE lider_proyecto = 'LINDA DAY')
```

COD_PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_PRODUCTO
100-100-101-005	CONCECTOR RXL NGN	T	1800000

Figura 4.48 Anidamiento de consultas con el operador de igualdad

Ejemplo 4.29 Generar un listado de las identificaciones, los nombres y los sueldos de los empleados que pertenecen a los proyectos ubicados en la planta de producción de la ciudad de Madrid (España).

Los resultados de la consulta se muestran en la figura 4.49, de acuerdo con el siguiente conjunto de instrucciones de selección:

```
/* Consulta Externa */
SELECT Id_empleado,nombre_empleado,sueldo
FROM empleados
WHERE No_proyecto = ( /* Primera subconsulta anidada */
SELECT No_proyecto
       FROM proyectos
       WHERE No_proyecto = ( /* Segunda subconsulta anidada */
       SELECT No_proyecto
       FROM planta_produccion
       WHERE ciudad = 'MADRID'))
```

```
SELECT Id_empleado,nombre_empleado,sueldo
FROM empleados
WHERE No_proyecto = (SELECT No_proyecto
       FROM proyectos
       WHERE No_proyecto = (SELECT No_proyecto
       FROM planta_produccion
       WHERE ciudad = 'MADRID'))
```

ID_EMPLEADO	NOMBRE_EMPLEADO	SUELDO
74415495	JULIO CONTRERAS JUAN CARLOS	5000000
74416005	GUERRA PORTO LUIS MIGUEL	900000

Figura 4.49 Anidamiento de cruce de las tablas de empleados, proyectos y plantas de producción del SIM

4.6.2 Generación de subconsultas con funciones de agregación

Ejemplo 4.30 Generar un listado de transacciones del SIM mostrando el número de la transacción contable, su fecha, el movimiento asociado y la diferencia con el valor promedio total de las transacciones para aquellas transacciones que hayan registrado un movimiento superior al valor promedio de las transacciones.

Teniendo en cuenta que no es posible utilizar las funciones de agregación en conjunto con la cláusula WHERE, para la realización de esta consulta es necesario, en primer lugar, construir una subconsulta con al fin de hallar el promedio de las transacciones (AVG); y en segundo lugar, se debe construir la instrucción SELECT externa para sacar las tuplas de la tabla de transacciones que sean superiores al promedio de la totalidad de las transacciones contables del Sistema Integrado de Manufactura.

Los resultados de la consultas se muestran en la figura 4.50 al correr la proposición SQL que se presenta a continuación:

```
/* Listado de transacciones con movimientos superiores al
promedio del total de las transacciones */

/* Subconsulta externa */

SELECT No_transaccion_contable,fecha_transaccion,movimiento,

movimiento - (SELECT AVG(movimiento) FROM transacción) AS "DIF.
CON PROMEDIO"

FROM transacción

/* Subconsulta anidada con la función de agregación promedio (AVG) */

WHERE movimiento > (SELECT AVG(movimiento) FROM transacción)
```

Ejemplo 4.31 Construir un listado de órdenes de transporte, con sus rutas de distribución asociadas y costos de transporte, que sean mayores que el valor mínimo del mínimo costo de transporte

Los resultados de la consulta se muestran en la figura 4.51, de acuerdo con la ejecución dela siguiente consulta:

```
/* Listado de costos de transporte del SIM cuyos valores
son mayores que el mínimo costo de transporte mostrando su
diferencia */

/* Consulta externa */

SELECT No_orden_transporte,No_ruta_distribucion,coste_
transporte, coste_transporte - (SELECT MIN(coste_transporte)
FROM transporte) AS "DIF. MÍNIMO"

FROM transporte

/Subconsulta o consulta interna con la función de agregación MIN

WHERE coste_transporte > (SELECT MIN(coste_transporte) FROM
transporte)
```

```
/* Listado de transacciones con movimientos superiores al
promedio del total de las transacciones */
SELECT No_transacción_contable,fecha_
transaccion,movimiento,movimiento - (SELECT AVG(movimiento)
FROM transacción) AS "DIF. CON PROMEDIO"
FROM transacción
WHERE movimiento > (SELECT AVG(movimiento) FROM
transacción)
```

NO_TRANSACCION_ CONTABLE	FECHA_TRANSACCION	MOVIMIENTO	DIF. CON PROMEDIO
TC-22-12-19-0019	24-DEC-19	1005000	875750
TC-22-12-19-0021	15-DEC-19	500000	370750

Figura 4.50 Utilización de subconsulta con la función de agregación promedio (AVG)

```
/* Listado de costos de transporte del SIM cuyos valores
son mayores que el mínimo costo de transporte mostrando su
diferencia */
SELECT No_orden_transporte,No_ruta_distribucion,coste_
transporte, coste_transporte - (SELECT MIN(coste_
transporte) FROM transporte) AS "DIF. MÍNIMO"
FROM transporte
WHERE coste_transpote > (SELECT MIN(coste_transpote) FROM
transporte)
```

NO_ORDEN_ TRANSPORTE	NO_RUTA_DISTRIBUCION	COSTE_ TRANSPOTE	DIF. MÍNIM
TA-VE-05-01-19-004	RD-BAQ-CO-SPA-BA-004	70000	65000
TA-VE-05-01-22-006	RD-BAQ-CO-MI-USA-006	30000	25000
TA-VE-24-12-19-008	RD-BAQ-CO-LA-USA-008	40000	35000
TA-VE-16-01-20-012	RD-SAM-CO-NY-USA-012	50000	45000
TA-VE-15-04-20-014	RD-BOG-CO-CA-CO-014	19999	14999
TA-VE-22-04-20-018	RD-SAM-CO-NY-USA-015	800000	795000
TA-VE-24-12-25-020	RD-SAM-CO-NY-USA-015	55000	50000

Figura 4.51 Utilización de subconsulta con la función de agregación MINIMO (MIN)

4.6.2.1 Generación de subconsultas anidadas con la utilización de la cláusula IN

Ejemplo 4.32 Generar un listado de facturas en el que se muestren las órdenes de venta con las descripciones de los productos, las cantidades vendidas y NIT a los clientes situados en áreas residenciales de apartamentos

La realización de esta consulta requiere: en primer lugar, ubicar en la tabla de ventas; las áreas residenciales donde hay compradores que vivan en apartamentos; en segundo lugar, identificar los productos comprados por estos compradores consultando la tabla de productos; y finalmente, en tercer lugar, consultar en la tabla de facturación los números de las órdenes de ventas, sus descripciones y las cantidades vendidas; para lo cual se utiliza en cada una de las consultas la cláusula SELECT en conjunto con la cláusula IN; resultados que se presentan en la figura 4.52, de acuerdo con la operación de la consulta SQL, que se presenta a continuación:

```
/* Consulta externa, que selecciona los atributos de la factura
*/
SELECT No_orden_venta, descripcion, cantidad_vendida,Nit_
comprador
FROM facturación
/* Subconsulta anidada que ubica el producto vendido */
WHERE cod_producto IN (SELECT cod_producto
       FROM productos
/*Subconsulta que selecciona las tuplas de apartamentos */
       WHERE cod_producto IN( SELECT cod_producto
       FROM ventas
 /* Direcciones iguales a APTO */
WHERE direccion_comprador LIKE '%APTO%'))
```

```
SELECT No_orden_venta, descripcion,cantidad_vendida,Nit_
comprador
FROM facturacion
WHERE cod_producto IN (SELECT cod_producto
       FROM productos
       WHERE cod_producto IN( SELECT cod_producto
       FROM ventas
       WHERE direccion_comprador LIKE '%APTO%'))
```

NO_ORDEN_VENTA	DESCRIPCION	CANTIDAD_VENDIDA	NIT_COMPRADOR
VE-05-01-19	SENSORES ÓPTICOS	42	72480234

Figura 4.52 Subconsulta anidada utilizando la cláusula IN

4.6.3 Selección de alguna o todas las tuplas

4.6.3.1 Selección de algunas tuplas (ANY)

Ejemplo 4.33 Generar un listado de cuentas del SIM cuyo saldo actual sea mayor que el valor de la compra de las cuentas por pagar en dólares; o sea, aquellas cuentas que registran el valor de 'D' en el tipo de moneda.

Los resultados de la consulta se muestran en la figura 4.53, los cuales son derivados de la siguiente consulta:

```
/* Consulta externa */
SELECT cod_cuenta,nombre_cuenta,saldo_actual
FROM cuenta
/Subconsulta anidada con la utilización de la cláusula SOME */
WHERE saldo_actual > SOME (SELECT valor_compra
       FROM cuentas_por_pagar
/* Filtro correspondiente al tipo de moneda en Dólares (D) */
       WHERE tipo_moneda = 'D')
```

```
SELECT cod_cuenta,nombre_cuenta,saldo_actual
FROM cuenta
WHERE saldo_actual > SOME (SELECT valor_compra
       FROM cuentas_por_pagar
       WHERE tipo_moneda = 'D')
```

COD_CUENTA	NOMBRE_CUENTA	SALDO_ACTUAL
10010101	Banco Colombia	5000000

Figura 4.53 Utilización de la cláusula ANY

4.6.3.2 Selección de todas las tuplas (ALL)

Ejemplo 4.34 Seleccionar todas las transacciones contables del SIM cuyos movimientos sean menores que los movimientos de las cuentas del sistema contable que hayan registrado un valor de $5'000.000,oo.

```
/* Selecciona los atributos de la tabla transacción del SIM */
SELECT No_transacción_contable, fecha_transacción, movimiento
FROM transacción
/* Selecciona todas las transacciones cuyo movimiento es menor
que las cuentas del Sistema Contable que hayan registrado un
valor de movimiento igual a $5'000.000,oo. */
WHERE movimiento < ALL (SELECT movimiento
       FROM cuentas
/* Cumplimiento de la condición de igual de $5´000.000 */
       WHERE movimiento = 5000000)
```

La operación de la selección de tuplas con la cláusula ALL se muestra en la figura 4.54.

```
/* Selección de todas las transacciones contables cuyos
movimientos son menores que todas las cuentas con
movimientos iguales a $5'000.000
*/
SELECT No_transacción_contable,fecha_transacción,
movimiento
FROM transacción
WHERE movimiento < ALL (SELECT movimiento
       FROM cuentas
       WHERE movimiento = 5000000)
```

NO_TRANSACCION_CONTABLE	FECHA_TRANSACCION	MOVIMIENTO
TC-22-12-19-0019	24-DEC-19	1005000
TC-22-12-19-0021	15-DEC-19	500000
TC-22-10-19-0002	16-DEC-19	100000
TC-22-12-19-0018	13-DEC-19	80000
TC-21-12-19-0004	15-DEC-19	70000
TC-21-15-18-0001	24-DEC-19	60000
TC-22-12-19-0017	22-DEC-19	60000
TC-22-12-19-0002	15-DEC-19	50000
TC-14-12-18-0002	28-DEC-19	50000
TC-22-12-19-0016	21-DEC-19	40000
TC-22-12-19-0003	22-DEC-19	40000
TC-20-12-19-0005	20-DEC-19	5000
TC-22-12-19-0020	26-DEC-19	5000
TC-19-12-19-0010	03-DEC-19	1000
TC-22-12-19-0001	15-APR-19	1000
TC-18-12-19-0003	21-DEC-19	1000

Figura 4.54 Utilización de la cláusula ALL

4.6.4 Generación de tuplas a través de consultas multitablas

4.6.4.1 Combinación simple

Las consultas multitablas tienen como finalidad combinar la información contenida en los atributos de una o más tablas con el fin de generar tuplas cuyos contenidos son las parejas de las tablas combinadas, lo cual corresponde a la operación definida en inglés como JOIN. La operación del JOIN implica incluir en la cláusula FROM una o más tablas; para lo cual se incluye una variable *dummy* (alias) que se asocia con la tabla de la cual se van a consultar sus atributos.

Ejemplo 4.35 Producir un listado de compradores registrados en la tabla de ventas del SIM que hayan adquirido productos producidos por el SIM.

La resolución de la consulta combina las tablas de ventas y productos del SIM, por su campo de asociatividad en los compradores que tienen registrados productos vendidos de los proyectos del SIM. Los resultados de la consulta se muestran en la figura 4.55 al operar la consulta que se muestra a continuación:

```
/* Combinación simple de dos tablas ventas y productos */

SELECT c.Nit_comprador, c.nombre_comprador, p.nombre_producto,
p.Cod_producto

/* Las variables c y p son alias que identifican los campos
asociados c a ventas y p a productos */

FROM ventas c, productos p

WHERE c.cod_producto = p.cod_producto
```

```
SELECT c.Nit_comprador, c.nombre_comprador, p.nombre_
producto, p.Cod_producto
FROM ventas c, productos p
WHERE c.cod_producto = p.cod_producto
```

NIT_ COMPRADOR	NOMBRE_COMPRADOR	NOMBRE_PRODUCTO	COD_ PRODUCTO
72480241	UNIVERSIDAD TECNOLOGÍA DEL SUR	PORTATIL NGP	100-100-101-000
18452699	SYSTEMS COMPUTERS	PROCESSOR XTS	100-100-101-001
77287568	ALFA CORPORATION COMPANY	INTERFACES MULTIPROPÓSITO	100-100-101-002
72480234	UNIVERSIDAD TECNOLOGÍA DEL ORIENTE	PHOTONIC CONTOLLED ONE SIM	100-100-101-003
72480235	IMPERIAL UNIVERSITY	DRIVE CONTROLLERS SIM	100-100-101-004
72485555	UNIVERSIDAD DE LAS AMÉRICAS	CONCECTOR RXL NGN	100-100-101-005
72480243	COLEGIO GREGORIANO	TWO BRAIN MOTOR	100-100-101-006
72480242	UNIVERSIDAD DE LOS ALPES	MOTOR DEDUCTIVE	100-100-101-007
72480246	UNIVERSITY OF ALBERTA	WEB 10.4 NGWEB	100-100-101-008
13456987	SOUTH TAMPA UNIVERSITY	MOTOR AIRPL. XT CONTROL TAB.	100-100-101-009
15458963	CHARLOTTE UNIVERSITY	MOTOR WEM DB 2.0	100-100-101-010
72481235	PACIFIC COMPANY	OPS - 2018	100-100-101-011
72480333	CENTRAL STATE UNIVERSITY	ROUT NO.1435	100-100-101-012
72480125	ATEL COMPANY	BODY TRACKING XC-XB-XD	100-100-101-013
72480487	LANCE CORP. INDUSTRY	OPTICAL COMPUTER	100-100-101-014
13845269	ALFA-OMEGA CORPORTATION	CHIP RSA	100-100-101-015
13287568	ENTERPRISE LAB. HK	LPS - NEW GENERATION	100-100-101-016

Figura 4.55 Utilización de consultas multitablas en su combinación simple

La consulta anterior es equivalente a la consulta que se muestra a continuación, cuyos resultados se presentan en la figura 4.56, en la cual se ejemplifica la cláusula JOIN.

```
SELECT
/* Atributos asociados a la tabla de ventas con el alias c */
 c.Nit_comprador, c.nombre_comprador, c.cod_producto,
/* Atributos asociados a la tabla de productos con el alias p */
 p.cod_producto, nombre_producto
/* Utilización de la claúsula JOIN */
 FROM ventas c JOIN productos p ON c.cod_producto=p.cod_
producto
```

```
SELECT
c.Nit_comprador, c.nombre_comprador,c.cod_producto,
p.cod_producto,nombre_producto
FROM ventas c JOIN productos p ON c.cod_producto=p.cod_producto
```

NIT_ COMPRADOR	NOMBRE_COMPRADOR	COD_PRODUCTO	COD_PRODUCTO	NOMBRE_PRODUCTO
72480241	UNIVERSIDAD TECNOLOGÍA DEL SUR	100-100-101-000	100-100-101-000	PORTATIL NGP
18452699	SYSTEMS COMPUTERS	100-100-101-001	100-100-101-001	PROCESSOR XTS
77287568	ALFA CORPORATION COMPANY	100-100-101-002	100-100-101-002	INTERFACES MULTIPROPÓSITO
72480234	UNIVERSIDAD TECNOLOGÍA DEL ORIENTE	100-100-101-003	100-100-101-003	PHOTONIC CONTOLLED ONE SIM
72480235	IMPERIAL UNIVERSITY	100-100-101-004	100-100-101-004	DRIVE CONTROLLERS SIM
72485555	UNIVERSIDAD DE LAS AMÉRICAS	100-100-101-005	100-100-101-005	CONCECTOR RXL NGN
72480243	COLEGIO GREGORIANO	100-100-101-006	100-100-101-006	TWO BRAIN MOTOR
72480242	UNIVERSIDAD DE LOS ALPES	100-100-101-007	100-100-101-007	MOTOR DEDUCTIVE
72480246	UNIVERSITY OF ALBERTA	100-100-101-008	100-100-101-008	WEB 10.4 NGWEB
13456987	SOUTH TAMPA UNIVERSITY	100-100-101-009	100-100-101-009	MOTOR AIRPL. XT CONTROL TAB.
15458963	CHARLOTTE UNIVERSITY	100-100-101-010	100-100-101-010	MOTOR WEM DB 2.0
72481235	PACIFIC COMPANY	100-100-101-011	100-100-101-011	OPS - 2018
72480333	CENTRAL STATE UNIVERSITY	100-100-101-012	100-100-101-012	ROUT N° 1435
72480125	ATEL COMPANY	100-100-101-013	100-100-101-013	BODY TRACKING XC-XB-XD
72480487	LANCE CORP. INDUSTRY	100-100-101-014	100-100-101-014	OPTICAL COMPUTER
13845269	ALFA-OMEGA CORPORTATION	100-100-101-015	100-100-101-015	CHIP RSA
13287568	ENTERPRISE LAB. HK	100-100-101-016	100-100-101-016	LPS - NEW GENERATION

Figura 4.56 Resultados de la utilización de la cláusula JOIN

4.6.4.2 Generación de tuplas ordenando una combinación

Ejemplo 4.36 Producir un listado de empleados ordenado alfabéticamente por nombres en forma ascendente en el que se muestre el cargo, su sueldo y el número del proyecto en el cual está laborando

La resolución de la consulta requiere la combinación de la tabla de empleados y la tabla de proyectos, y para los atributos en los cuales el número del proyecto en la tabla de empleados coincida con el número del proyecto en la tabla de proyectos, se seleccionan los atributos de la tabla empleados, de los cuales se ordenan los nombres de los empleados a nivel ascendente.

Los resultados se muestran en la figura 4.57, derivados de la ejecución de esta consulta:

```
/* Selección de los atributos de identidad del empleado, sueldo
y cargo */
SELECT E. Id_empleado, nombre_empleado, cargo_empleado, sueldo,
 p.No_proyecto
/* Consulta multitabla en la cual se seleccionan atributos de
tablas de empleados y proyectos */
 FROM empleados e, proyectos p
 WHERE e.No_proyecto = p.No_proyecto
/* Ordenamiento ascedente (ASC) de la consulta por el nombre
del empleado */
 ORDER BY nombre_empleado ASC
```

```
SELECT E. Id_empleado,nombre_empleado, cargo_
empleado,sueldo,

p.No_proyecto

FROM empleados e, proyectos p

WHERE e.No_proyecto = p.No_proyecto

ORDER BY nombre_empleado ASC
```

ID_ EMPLEADO	NOMBRE_EMPLEADO	CARGO_ EMPLEADO	SUELDO	NO_ PROYECTO
74415648	AGUIRRE PEREZ RUTH ELENA	TECNICO	1000000	P004SE0004
74415240	AYOS RENOWISKY ANDRÉS JULIAN	TECNICO	1500000	P002NY0002
72415240	CONSUEGRA AMIN PEDRO DANIEL	TECNICO	1000000	P001BA0001
74416056	FABREGAS COTO LUIS DANIEL	ADMINISTRADOR	900000	P001BA0001
74415342	GOENAGA MAYA JULIO	INGENIERO	4500000	P004SE0004
74416005	GUERRA PORTO LUIS MIGUEL	ADMINISTRADOR	900000	P006NY0008
74415291	GUTIEREZ DÍAZ ANDRES	INGENIERO	3000000	P003MI0003
74415954	HERRERA LORA ANNY	INGENIERO	4300000	P005BA0005
74415495	JULIO CONTRERAS JUAN CARLOS	DOCTOR	5000000	P006NY0008
74416107	JUVENAL HINCAPIE ELEONORA	ADMINISTRADOR	980000	P001BA0001
74415903	LOPEZ DIAZ RUBEN DANIEL	TECNICO	1000000	P004SE0004
74415546	NANDIN LOPERA JULIANA	INGENIERO	3000000	P005BA0009
74415597	PEREZ PEREZ ANDREA	ASISTENTE	1200000	P003MI0003
74416158	PUYANA KARKER CLAUDIA JULIANA	ADMINISTRADOR	980000	PA11TA1111
74415852	SANDOVAL HUCKE MARY	INGENIERO	4500000	P003MI0003
74415699	SILVESTRE CUELLO ANA	ASISTENTE	800000	P005BA0012
74415801	VEGA BAENA CARMEN	INGENIERO	3800000	P002NY0002
74415444	VILLAMIZAR ATYOS ANDRÉS	DOCTOR	5000000	P006SE0007
74415393	VILLAR ARTEAGA LUIS ALBERTO	TECNICO	1000000	P005BA0005
74415750	VISBAL PEREZ SOFIA	ASISTENTE	800000	P001BA0001

Figura 4.57 Consulta multitabla utilizando el ordenamiento de una combinación

4.6.4.3 Generación de tuplas por combinación de *n* tablas

Ejemplo 4.37 Seleccionar del SIM los códigos de los máster de producción, identificando las máquinas que operan en dichas líneas de las plantas de producción que tengan en ejecución proyectos del Sistema Integrado de Manufactura

La solución de la consulta requiere la combinación de cuatro (n=4) tablas: producción, maquinaria, planta de producción y proyectos del SIM. Los resultados de la consulta se presentan en la figura 4.58, derivados del siguiente código SQL:

```
/* Se seleccionan las tuplas distintas */

       SELECT DISTINCT

/* Se imprimen los campos claves de las cuatro tablas */

 r.Cod_master_produccion,m.No_maquina, d.No_linea_produccion,p.
No_proyecto

/* Las cuatro tablas son producción, maquinaria, planta de
producción y proyectos */

FROM produccion r,maquinaria m, planta_produccion d, proyectos p

WHERE  r.No_maquina = m.No_maquina /* Combinación entre
producción y maquinaria */

AND    m.No_maquina = d.No_maquina /* Maquinaria y planta de
producción relación */

AND    d.No_proyecto = p.No_proyecto /* Combinación planta de
producción proyectos */
```

```
SELECT DISTINCT r.Cod_master_produccion,m.No_maquina, d.No_
linea_produccion,p.No_proyecto
FROM produccion r,maquinaria m, planta_produccion d,
proyectos p
WHERE r.No_maquina = m.No_maquina
AND    m.No_maquina = d.No_maquina
AND    d.No_proyecto = p.No_proyecto
```

COD_MASTER_ PRODUCCION	NO_MAQUINA	NO_LINEA_ PRODUCCION	NO_PROYECTO
MP002-L200-PRO-0001	MA-RF1222002	L100-COM-0001	P006SE0007
MP003-L300-IGA-0002	MA-TZ9456003	L100-COM-0002	P006NY0008

Figura 4.58 Combinación de n(n=4) tablas del SIM

4.6.4.4 Generación de tuplas por múltiples columnas de agrupamiento

Ejemplo 4.38 Generar un listado agrupado por proyectos en el que se especifiquen los cargos de desempeño de cada empleado y cuente el número de empleado(s) por cargo perteneciente a cada proyecto del SIM, ordenado ascendentemente tanto por los números de los proyectos y dentro de cada proyecto por los cargos de los empleados

El listado se debe agrupar por los números de los proyectos; y dentro de los proyectos agrupar por los cargos de los empleados, haciendo el conteo de los empleados por cargo que participan en los proyectos. Los resultados de la consulta se muestran en la figura 4.59 al correr el siguiente código SQL:

```
SELECT p.No_proyecto,e.cargo_empleado, COUNT(*) AS "EMPLEADOS/
PROYECTO"
FROM proyectos p, empleados e
/* Combinación de las tablas de proyectos y empleados por el
número del proyecto */
WHERE p.No_proyecto = e.No_proyecto
/* Se agrupan las tuplas por el No_proyecto y dentro de los
proyectos por los cargos de los empleados */
GROUP BY p.No_proyecto, cargo_empleado
```

```
ORDER BY p.No_proyecto, cargo_empleado ASC
```

```
SELECT p.No_proyecto,e.cargo_empleado, COUNT(*) AS
"EMPLEADOS/PROYECTO"
FROM proyectos p, empleados e
WHERE p.No_proyecto = e.No_proyecto
GROUP BY p.No_proyecto, cargo_empleado
ORDER BY p.No_proyecto, cargo_empleado
```

NO_PROYECTO	CARGO_EMPLEADO	EMPLEADOS/PROYECTO
P001BA0001	ADMINISTRADOR	2
P001BA0001	ASISTENTE	1
P001BA0001	TECNICO	1
P002NY0002	INGENIERO	1
P002NY0002	TECNICO	1
P003MI0003	ASISTENTE	1
P003MI0003	INGENIERO	2
P004SE0004	INGENIERO	1
P004SE0004	TECNICO	2
P005BA0005	INGENIERO	1
P005BA0005	TECNICO	1
P005BA0009	INGENIERO	1
P005BA0012	ASISTENTE	1
P006NY0008	ADMINISTRADOR	1
P006NY0008	DOCTOR	1
P006SE0007	DOCTOR	1
PA11TA1111	ADMINISTRADOR	1

Figura 4.59 Cláusula SELECT con múltiples columnas de agrupamiento

4.6.5 Generación de tuplas por combinaciones

4.6.5.1 Selección de tuplas por combinación externa izquierda

Ejemplo 4.39 Producir un listado ordenado por los nombres de los proveedores que tengan asociados los productos del SIM a los cuales proveen sus materias primas

La solución de la consulta es una combinación externa izquierda, la cual utiliza en el SELECT la cláusula LEFT JOIN, la selecciona las tuplas de los proveedores que suministran materias primas a los productos, indicando el código del producto al cual proveen la materia prima. En este caso, todos los proveedores registrados suministran materia prima a algún producto; razón por la cual todas las tuplas de la tabla proveedores tienen sus correspondientes tuplas en la tabla de productos, cuya información se presenta en la figura 4.60, derivada de la consulta que se presenta a continuación:

```
/* De la tabla de proveedores ( r ) selecciona todas (*) las
tuplas,

 De la tabla de productos (p) selecciona todas ( * ) las tuplas

 Las tuplas se homologan en su combinación con el atributo del
código del producto */

SELECT s.*,p.*

FROM provedores s LEFT JOIN productos p ON s.cod_producto =
p.cod_producto

ORDER BY s.nombre_proveedor
```

Ejemplo 4.40 Listar la maquinaria asociada a las líneas de las planta de producción del SIM, mostrando tanto las máquinas asignadas como las no asignadas, y mostrar las máquinas por el número de la maquinaria.

En este caso se seleccionan las tuplas de la tabla "maquinaria", las cuales tienen asignadas líneas de producción en la tabla de planta_producción; pero las máquinas que no están asignadas a ninguna línea de fabricación aparecen como vacías (nulas) con relación a los atributos de las líneas de producción. Los resultados de la consulta se presentan en la figura 4.61; resultados obtenidos de la ejecución de la consulta SQL:

```
/* La tabla ubicada en la parte izquierda son las máquinas */
```

```
/* La tabla de la parte derecha es la planta_producción */
SELECT m.*,t.*
/* La combinación se realiza por el número de las máquinas */
FROM maquinaria m LEFT JOIN planta_produccion t ON m.No_maquina
= t.No_maquina
ORDER BY m.No_maquina
```

```
SELECT s.*,p.*
FROM proveedores s LEFT JOIN productos p ON s.cod_producto = p.cod_
producto
ORDER BY s.nombre_proveedor
```

NIT_ PROVEEDOR	NOMBRE_ PROVEEDOR	DIRECCION_ PROVEEDOR	CIUDAD_ PROVEEDOR	PAIS_ PROVEEDOR	EMAIL_PROVEEDOR	COD_ PRODUCTO
72435228001	DELL CORPORATION	HIGH STREET 101 No. 14-28	SEATLE	USA	main_100dell@del.usa.com.co	100-100-101-002
72435228003	IBM	Carrera 53 No. 100 - 25. Bogotá Colombia	BOGOTÁ	COLOMBIA	main_010ibm@ibm.com.co	100-100-101-004
72435228002	TOSHIBA COMPANY	Tokyo-to Chuo-ku Yaesu 1-Chome 5-ban 3-go	TOKIO	JAPAN	main_001TOSHIBAI@TOSHIBA.com.co	100-100-101-003

COD_CUENTA	COD_ PRODUCTO	NOMBRE_ PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
10-001-001-001-004	100-100-101-002	INTERFACES MULTIPROPÓSITO	T	200000	3	P003MI0003
10-001-001-011-000	100-100-101-004	DRIVE CONTROLLERS SIM	P	25000	2	P005BA0005
10-001-001-001-005	100-100-101-003	PHOTONIC CONTOLLED ONE SIM	T	20000	1	P004SE0004

Figura 4.60 Combinación externa izquierda LEFT JOIN

```
SELECT m.*,t.*
FROM maquinaria m LEFT JOIN planta_produccion t ON m.No_
maquina = t.No_maquina
ORDER BY m.No_maquina
```

NO_MAQUINA	NOMBRE_MAQUINA	EST	NO_LINEA_ PRODUCCION
MA-RF1222002	ENSAMBLADOR AUTOMÁTICO GENÉRICO	P	L100-COM-0001
MA-RF1222008	SELECTOR AUTOMÁTICO DE COMPONENTES	P	
MA-RK1214001	SELECTOR ÓPTICO	P	
MA-RL1222005	ARMADOR DE DISPOSITO	P	
MA-RZ1214004	CONTROLADOR DE VELOCIDAD	M	L100-COM-0003
MA-RZ1214007	ENSAMBLADOR DE CORE	P	
MA-TU9456006	GENERARON DE ARQUITECTURA ÓPTICA	M	
MA-TY9456009	GRAFICADOR DE DISEÑO POR ESPECIFICACIÓN	M	
MA-TZ9456003	SENSOR DE CALIDAD DE ESPACIO	P	L100-COM-0002

COMPONENTES_ LINEA	CIUDAD	PAIS	NO_MAQUINA	NO_ PROYECTO
CAD	BARRANQUILLA	COLOMBIA	MA-RF1222002	P006SE0007
CAD	SAO PAULO	BRASIL	MA-RZ1214004	P005BA0009
CAD	MADRID	ESPAÑA	MA-TZ9456003	P006NY0008

Figura 4.61 Combinación externa izquierda LEFT JOIN entre la maquinaria asignada a las líneas de producción

4.6.5.2 Selección de tuplas por combinación externa derecha

Ejemplo 4.41 Generar un listado en el que se presenten las ventas de productos utilizando RIGHT JOIN con los productos producidos por el SIM en el que se muestren los productos que no han tenido compradores en la tabla de ventas

La figura 4.62 muestra los resultados de la consulta en la cual la tabla de productos se encuentra en la parte derecha (RIGHT JOIN), y la tabla de ventas se presenta en la parte izquierda, donde se registran los compradores.

```
SELECT v.*,p.*
FROM ventas v RIGHT JOIN productos p ON v.cod_producto = p.cod_
producto
```

En la consulta se observa que hay productos que no han registrado ventas.

4.6.5.3 Selección de tuplas por combinación externa completa

Ejemplo 4.42 Construir un listado que muestre en su selección tanto las cuentas por pagar registradas en la tabla de cuentas del SIM como las cuentas del sistema contable del SIM que no están relacionadas con ninguna tupla de cuentas por pagar

La figura 4.63 muestra los resultados de la consulta al operar el código SQL:

```
/* Selecciona todas las tuplas (*) de la tabla cuentas por
pagar ( c ) */
/* Selecciona todas las tuplas (*) de la tabla cuentas (t) */
SELECT c.* , t.*
/* La combinación externa completa FULL JOIN no solo incluye
en los resultados las cuentas_por_pagar sin cuentas contables
asociadas, sino también las cuentas sin cuentas_por_pagar */
FROM cuentas_por_pagar c FULL JOIN cuentas t ON c.cod_cuenta =
t.cod_cuenta
```

```
SELECT v.*,p.*
FROM ventas v RIGHT JOIN productos p ON v.cod_producto =
p.cod_producto
```

NIT_ COMPRADOR	NOMBRE_COMPRADOR	DIRECCION_COMPRADOR	TELEFONO_ COMPRADOR	TIP	NUMERO_ ORDEN	COD_PRODUCTO
72480241	UNIVERSIDAD TECNOLOGÍA DEL SUR	TRANS. 100 NO. 200-141	5670280	N	VE-09-01-19	100-100-101-000
18452699	SYSTEMS COMPUTERS	CALLE 84 NO. 51-B-12	3509277	N	VE-15-12-19	100-100-101-001
77287568	ALFA CORPORATION COMPANY	CARRERA 140 NO. 100-15	3509217	N	VE-17-04-20	100-100-101-002
72480234	UNIVERSIDAD TECNOLOGÍA DEL ORIENTE	HIGH STREET 215 APTO. 101	9871578	N	VE-05-01-19	100-100-101-003
72480235	IMPERIAL UNIVERSITY	CALLE 101 NO. 45-80	5641255	N	VE-07-12-19	100-100-101-004
72485555	UNIVERSIDAD DE LAS AMÉRICAS	CARRERA 40 NO. 78-15	3602141	N	VE-05-01-22	100-100-101-005
72480243	COLEGIO GREGORIANO	CALLE 87 NO. 114-34	5670280	N	VE-24-01-19	100-100-101-006
72480242	UNIVERSIDAD DE LOS ALPES	TRANS. 90 NO. 200-141	5670123	N	VE-24-12-19	100-100-101-007
72480246	UNIVERSITY OF ALBERTA	TRANS. 250 NO. 200-141	8678467	N	VE-24-05-19	100-100-101-008
13456987	SOUTH TAMPA UNIVERSITY	ROAD FIRST NO. 180 - 14	9802222	N	VE-24-12-20	100-100-101-009
15458963	CHARLOTTE UNIVERSITY	HIGH AVENUE NO. 180-24	5670233	N	VE-16-01-20	100-100-101-010
72481235	PACIFIC COMPANY	HIGH STREET 214 NO. 20-333	9781255	N	VE-16-01-20	100-100-101-011
72480333	CENTRAL STATE UNIVERSITY	SECOND ROAD NO. 180-24	9802141	N	VE-16-01-20	100-100-101-012
72480125	ATEL COMPANY	CCC LOCAL 101	5670243	N	VE-15-04-20	100-100-101-013
72480487	LANCE CORP. INDUSTRY	CCC BUENAVISTA L 115 P 2	3330280	N	VE-19-04-20	100-100-101-014
13845269	ALFA-OMEGA CORPORTATION	ALBERTA HIGH AVE. NO. 12-18	3421578	N	VE-14-04-20	100-100-101-015
13287568	ENTERPRISE LAB. HK	GLASGOW, STREET ONE 45-156	9781255	N	VE-07-04-20	100-100-101-016

Figura 4.62 Combinación externa derecha RIGHT JOIN

COD_CUENTA	COD_ PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
10-000-000-000-000	100-100-101-000	PORTATIL NGP	T	800000	20	P001BA0001
10-001-000-000-000	100-100-101-001	PROCESSOR XTS	P	400000	10	P002NY0002
10-001-001-000-000	100-100-101-002	INTERFACES MULTIPROPÓSITO	T	200000	3	P003MI0003
10-001-001-001-000	100-100-101-003	PHOTONIC CONTOLLED ONE SIM	T	20000	1	P004SE0004
10-001-001-001-001	100-100-101-004	DRIVE CONTROLLERS SIM	P	25000	2	P005BA0005
10-001-001-001-002	100-100-101-005	CONCECTOR RXL NGN	T	1800000	1	P006SE0007
10-001-001-001-003	100-100-101-006	TWO BRAIN MOTOR	T	3000000	1	P006NY0008
10-001-001-001-004	100-100-101-007	MOTOR DEDUCTIVE	T	1000000	20	P005BA0009
10-001-001-001-005	100-100-101-008	WEB 10.4 NGWEB	T	1800000	20	P005BA0010
10-001-001-011-000	100-100-101-009	MOTOR AIRPL. XT CONTROL TAB.	T	1000000	20	P005BA0011
10-001-001-011-001	100-100-101-010	MOTOR WEM DB 2.0	P	500000	4	P005BA0012
10-001-001-011-002	100-100-101-011	OPS - 2018	T	175000	8	P006BO0013
10-001-001-011-003	100-100-101-012	ROUT NO.1435	T	30000	9	P007TA0014
01-001-001-001-999	100-100-101-013	BODY TRACKING XC-XB-XD	T	180000	12	P008TA0015
01-001-001-001-998	100-100-101-014	OPTICAL COMPUTER	T	570000	5	P009TA0014
01-001-001-001-997	100-100-101-015	CHIP RSA	MT	40000	100	P010BA0015
10-001-001-001-005	100-100-101-016	LPS - NEW GENERATION	MT	80000	57	P020BA0016
	100-100-101-019	CAD GRAPHIC SYSTEMS	P	5000	1	PA11TA1111
	100-100-101-018	OPTICAL CONTROL FBA-12	T	100000	10	P012BO0222
	100-100-101-017	CODY SYS. UVXA	MT	50000	30	P011BO0017

```
SELECT c.* , t.*
FROM cuentas_por_pagar c FULL JOIN cuentas t ON c.cod_
cuenta = t.cod_cuenta
```

NIT_ PROVEEDOR	NO_ORDEN_ COMPRA	VALOR_ COMPRA	TIP	MONTO_30	MONTO_60	MONTO_90
72435228001	19-09-2015-0001	500000	D	500000	500000	500000
72435228002	13-10-2020-0002	1000000	D	500000	5026500	2640000
72435228003	15-12-2018-0003	750000	D	250000	250000	250000

Figura 4.63 Combinación externa completa FULL JOIN

COD_CUENTA	NOMBRE_ CUENTA	SALDO_ ANTERIOR	MOVIMIENTO	SALDO_ ACTUAL	TIP
10-001-000-000-000	ACTIVO CORRIENTE	1000000	10000	1010000	D
10-001-001-000-000	BANCOS	1000000	-500	999500	D
10-001-001-001-000	BANCOS EXTRANJEROS	1000000	10000	1010000	D
10-001-001-011-000	BANCOS LOCALES	10000000	300	10000300	P
10-001-001-011-003	BANCO SANTANDER	700000000	500	700000500	P
10-001-001-001-003	BANCO BILBAO	10000	50	10050	D
01-001-001-001-997	PROV. IBM	7000000	500	7000500	D
10-000-000-000-000	ACTIVO	1000000000	200000000	1200000000	D
10-001-001-001-005	BANCO DE PORTUGAL	1000	10000	11000	D
10-001-001-001-004	INTERNATIONAL BANK	1000	-500	500	D
10-001-001-011-002	BANCO DE COLOMBIA	2000000	200	2000200	P
01-001-001-001-998	PROV. TOSHIBA	50000	500	50500	D
10-001-001-001-001	SEA FIRST BANK	10000	-500	9500	D
01-001-001-001-999	PROV. DELL	200000	500	200500	D
10-001-001-001-002	UNION BAN	1000	10000	11000	D
10-001-001-011-001	BANCO DE BOGOTÁ	1000000	10000	1010000	P

4.6.6 Selección de tuplas utilizando las cláusulas EXISTS y NOT EXISTS

4.6.6.1 Generación de tuplas utilizando la cláusula EXISTS

Ejemplo 4.43 Seleccionar del SIM los número de las órdenes de transporte con sus números de órdenes de venta asociadas, sus fechas de cargas y costes de seguros para aquellas órdenes de transporte que en sus rutas de distribución en sus entregas son atendidas por transportes aéreos; o para aquellas compañías en las que la identidad de la empresa de transporte tiene el token "AIR"

Los resultados se muestran en la figura 4.64; resultados obtenidos del procesamiento de la siguiente consulta SQL:

```
SELECT No_Orden_transporte,No_orden_venta,fecha_carga,coste_
   seguro
FROM transporte t
/*
WHERE EXISTS (SELECT *
   FROM entrega e
   WHERE t.No_ruta_distribucion = e.No_ruta_distribucion AND
   empresa_transportadora like '%AIR%')
/* Las rutas de distribución consideras en las entregas son las
   aéreas */
```

```
SELECT No_Orden_transporte,No_orden_venta,fecha_
carga,coste_seguro
FROM transporte t
WHERE EXISTS (SELECT *
   FROM entrega e
   WHERE t.No_ruta_distribucion = e.No_ruta_distribucion AND
   empresa_transportadora like '%AIR%')
```

NO_ORDEN_ TRANSPORTE	NO_ORDEN_VENTA	FECHA_CARGA	COSTE_SEGURO
TA-VE-17-04-20-003	VE-17-04-20	17-APR-20	5000
TA-VE-05-01-22-006	VE-05-01-22	05-JAN-22	5000
TA-VE-24-01-19-007	VE-24-01-19	24-JAN-19	3452
TA-VE-24-12-19-008	VE-24-12-19	24-DEC-19	5000
TA-VE-16-01-20-011	VE-16-01-20	16-JAN-20	5000

Figura 4.64 Selección de tuplas utilizando la cláusula EXISTS

4.6.6.2 Generación de tuplas utilizando la cláusula NOT EXISTS

Ejemplo 4.44 Seleccionar del SIM las identificaciones de los productos distintos, con todas sus especificaciones, para los cuales no se haya presentado comprador en la tabla de ventas.

La resolución de la consulta se presenta en la figura 4.65; resolución obtenida de la ejecución de la siguiente código SQL:

```
SELECT
DISTINCT /* Se selecciona de la base de datos los productos
distintos */
cod_producto,Nombre_producto,Tipo_producto,Costo_
producto,Cantidad_productos,No_proyecto /* Atributos de los
productos*/
FROM productos p
```

```
WHERE NOT EXISTS /* Productos para los cuales no se han
registrado compradores */
       (SELECT *
       FROM ventas v /* En la tabla de ventas */
       WHERE v.cod_producto = p.cod_producto)
```

```
SELECT DISTINCT cod_producto,Nombre_producto,Tipo_producto,Costo_
producto,Cantidad_productos,No_proyecto
FROM productos p
WHERE NOT EXISTS
       (SELECT *
       FROM ventas v
       WHERE v.cod_producto = p.cod_producto)
```

COD_ PRODUCTO	NOMBRE_PRODUCTO	TIPO_P	COSTO_ PRODUCTO	CANTIDAD_ PRODUCTOS	NO_ PROYECTO
100-100-101-017	CODY SYS. UVXA	MT	50000	30	P011BO0017
100-100-101-018	OPTICAL CONTROL FBA-12	T	100000	10	P012BO0222
100-100-101-019	CAD GRAPHIC SYSTEMS	P	5000	1	PA11TA1111

Figura 4.65 Utilización de la cláusula NOT EXISTS

4.6.7 Generación de tuplas por combinación de tablas de resultados

Con base en los conceptos del Algebra Relacional, con la estructura de relacional de tablas de una base de datos en su organización de tuplas o filas y columnas o atributos se pueden realizar operaciones tal y como se realizan en la Teoría de Conjuntos. En tal sentido, en dialecto SQL existen las operaciones de unión, intersección y diferencia de tablas.

4.6.7.1 Generación de tuplas por unión de tablas UNION

La unión de tablas T1 ∪ T2, o relaciones en una base de datos, es una nueva relación o tabla, la cual contiene las tuplas de la tabla T1 unidas con las tuplas de la tabla T2, eliminando las tuplas duplicadas en el resultado de la unión, y para las cuales debe existir un atributo compatible de unión entre las tablas. La cláusula que opera la unión de conjunto en SQL es UNION.

Ejemplo 4.45 UNION

Producir un listado en el que se muestre el código del producto del SIM, unido con los proveedores y a su vez con los proyectos de fabricación de productos.

La consulta se presenta en la figura 4.66, con la ejecución del siguiente código:

```
(SELECT cod_producto /* Se selecciona el código del producto de
la tabla de proveedores */
FROM proveedores)
UNION
(SELECT cod_producto /* Código del producto seleccionado de la
tabla de productos */
FROM productos)
UNION
(SELECT cod_producto /* Código del producto seleccionado de la
tabla de proyectos */
FROM proyectos)
```

```
(SELECT cod_producto
FROM proveedores)
UNION
(SELECT cod_producto
FROM productos)
UNION
(SELECT cod_producto
FROM proyectos)
```

COD_PRODUCTO
100-100-101-000
100-100-101-001
100-100-101-002
100-100-101-003
100-100-101-004
100-100-101-005
100-100-101-006
100-100-101-007
100-100-101-008
100-100-101-009
100-100-101-010
100-100-101-011
100-100-101-012
100-100-101-013
100-100-101-014
100-100-101-015
100-100-101-016
100-100-101-017
100-100-101-018
100-100-101-019

Figura 4.66 Empleo de la cláusula UNION para la unión de tablas

4.6.7.2 Generación de tuplas por intersección de tablas INTERSECT

La intersección de tablas T1 ∩ T2, o relaciones en una base de datos, es una nueva relación o tabla, la cual contiene tanto las tuplas de la tabla T1 como las tuplas de la tabla T2 intersectadas por un atributo común de las dos tablas, eliminando las tuplas duplicadas en el resultado de la intersección. La cláusula que opera la intersección de conjuntos en SQL es INTERSECT.

Ejemplo 4.46 Seleccionar los códigos de las cuentas relacionadas con proveedores que hayan tenido a su vez un registro en la tabla de ventas

Los resultados de la consultas se presentan en la figura 4.67, y se intersectan tres tablas: cuentas, proveedores y ventas. La operación del SIM permite que el proveedor puede tomar el rol a su vez de comprador, de acuerdo con los resultados de la siguiente consulta:

```
(SELECT cod_cuenta FROM cuentas)
/* Se selecciona el código de la cuenta de la tabla de cuentas */
INTERSECT
(SELECT cod_cuenta FROM proveedores)
/* Selección del código de la cuenta de la tabla de proveedores */
INTERSECT
/* Selecciona el código de la cuenta de la tabla de ventas */
(SELECT cod_cuenta FROM ventas)
```

```
(SELECT cod_cuenta FROM cuentas)
INTERSECT
(SELECT cod_cuenta FROM proveedores)
INTERSECT
(SELECT cod_cuenta FROM ventas)
```

COD_CUENTA
10-001-001-001-004
10-001-001-001-005
10-001-001-011-000

Figura 4.67 Utilización de la cláusula INTERSECT

4.6.7.3 Generación de tuplas por diferencia de tablas MINUS

La diferencia de relaciones o tablas en SQL corresponde a las operaciones de diferencia de dos conjuntos, A y B, denotado por A - B. Dadas dos tablas, T1 y T2, la operación de diferencia de tablas representada por T1 - T2 tiene como resultado una nueva tabla compuesta por las tuplas que pertenecen a la tabla T1 pero no pertenecen a la tabla T2; en los dialectos SQL la operación de diferencia de relaciones se opera con las cláusulas EXCEPT, MINUS o NOT EXISTS.

Ejemplo 4.47 Seleccionar los números de proyectos que habiendo sido asignados a los empleados no estén registrados en la tabla de proyectos

La figura 4.68 presenta la resolución de la consulta, y la respuesta es que no hay tuplas seleccionadas, por lo cual es lógico pensar que si se tiene un empleado contratado al SIM esté asignado a un proyecto existente, según la ejecución de la siguiente consulta SQL:

```
SELECT No_proyecto
FROM empleados)
       MINUS /* Operador de diferencia en SQL */
       (SELECT No_proyecto
       FROM proyectos)
```

```
(SELECT No_proyecto
FROM empleados)
       MINUS
       (SELECT No_proyecto
       FROM proyectos)
```

Figura 4.68 Utilización de la cláusula MINUS

Ejemplo 4.48 MINUS con resultados de tuplas del SIM

Producir un listado de productos del Sistema Integrado de Manufactura en el que se muestren los códigos de los productos para los cuales no hay un proveedor asignado.

Análisis:

De la tabla de productos del SIM, especificando el código del producto, se aplica el operador diferencia entre la tabla de productos, restándole los productos que son suministrados por los proveedores.

Resultados:

Las tuplas resultado de la consulta se muestran en la figura 4.69 al procesar la siguiente consulta SQL:

```
(SELECT cod_producto /* Productos para los cuales no hay
asignados proveedores */
FROM productos)
       MINUS /* Utilización del operador de diferencia MINUS */
       (SELECT cod_producto
       FROM proveedores)
```

```
(SELECT cod_producto
FROM productos)
       MINUS
       (SELECT cod_producto
       FROM proveedores)
```

COD_PRODUCTO
100-100-101-000
100-100-101-001
100-100-101-005
100-100-101-006
100-100-101-007
100-100-101-008
100-100-101-009
100-100-101-010
100-100-101-011
100-100-101-012
100-100-101-013
100-100-101-014
100-100-101-015
100-100-101-016
100-100-101-017
100-100-101-018
100-100-101-019

Figura 4.69 Uso de la cláusula MINUS

La consulta anterior es equivalente a si se utiliza NOT EXISTS, tal y como se presenta en la figura 4.70.

```
SELECT DISTINCT cod_producto
FROM productos p
WHERE NOT EXISTS
       (SELECT *
       FROM proveedores r
       WHERE p.cod_producto = r.cod_producto )
       ORDER BY cod_producto ASC
```

COD_PRODUCTO
100-100-101-000
100-100-101-001
100-100-101-005
100-100-101-006
100-100-101-007
100-100-101-008
100-100-101-009
100-100-101-010
100-100-101-011
100-100-101-012
100-100-101-013
100-100-101-014
100-100-101-015
100-100-101-016
100-100-101-017
100-100-101-018
100-100-101-019

Figura 4.70 Equivalencia de cláusulas MINUS y NOT EXISTS

4.7 Edición de la base de datos

La dinámica de los sistemas de información al servicio de los requerimientos de información de una organización hace que el contenido de los datos de la base de datos no sean estáticos; por lo tanto, se requiere la edición de los datos y no solo su consulta (SELECT) a través del gestor de la base de datos.

Siendo el SQL un lenguaje de manipulación de datos, este dispone de un conjunto de cláusulas que permiten la modificación de los datos, entre los cuales se encuentran: i) INSERT, o inserción de datos a la base de datos. ii) UPDATE, o edición de los datos de la base de datos. iii) DELETE, o cancelación de datos contenidos en la base de datos.

4.7.1 Adición de tuplas a la base de datos INSERT

Una vez creada la base de datos, es necesario cargar o poblar los datos en esta; para lo cual se requiere la cláusula INSERT. El propósito de esta cláusula es adicionar líneas o filas a una tabla, a una vista o a una tabla base de una vista en la base de datos. Se aclara que en el caso de que la base de datos sea orientada por objetos, se adicionarán los datos a un objeto tabla de la base de datos.

Es lógico suponer el cumplimiento de unos prerrequisitos previos a la operación de la inserción. Luego, el INSERT debe cumplir como prerrequisitos: i) La tabla debe estar definida dentro de su esquema de bases de datos. ii) El usuario debe tener el privilegio de poder operar la cláusula INSERT en la tabla de la base de datos, o lo que es equivalente, estar autorizado para insertar datos en una tabla autorizada. iii) Con base en la creación y la autorización de la tabla, el usuario finalmente puede insertar los datos en la base de datos, de acuerdo con la siguiente sintaxis:

```
INSERT INTO NombreTabla[(ListadeColumnas)]
VALUES (ListadeValoresdeDatos)
```

- NombreTabla identifica el nombre de la tabla o de la vista en la cual se van a insertar los datos.
- ListadeColumnas es el conjunto de los nombres de una o más columnas que componen la tabla (o lo que es equivalente, los atributos de la entidad en el Modelo Entidad-Relación), campos que son opcionales dentro de la cláusula

INSERT. En el caso de que en la inserción se coloquen los nombres de las columnas, estos deben coincidir exactamente en nombre, número y orden con las columnas tal y como fueron creadas en la tabla.

– ListaValoresdeDatos es el conjunto de valores que se van a insertar en la base de datos, separados por comas, y deben cumplir con los siguientes requisitos en su estructura de entradas de datos: i) Los valores de los datos deben corresponderse en número y orden con las columnas tal y como fueron definidas en la tabla. ii) Los datos de entrada deben corresponderse en tipo de datos con los definidos en las columnas de las tablas, con el fin de que la entrada de datos sea compatible.

Ejemplo 4.49 Inserción de tuplas de una tabla de la base de datos

Teniendo como base el Sistema Integrado de Manufactura cuya descripción se presenta en la figura 4.71, insertar los datos del empleado con la identificación 77745678, teniendo en cuenta insertar todos los datos de la tabla.

Name	Null?	Type
ID_EMPLEADO	NOT NULL	VARCHAR(10)
NOMBRE_EMPLEADO	NOT NULL	VARCHAR(50)
DIRECCION_FIS		VARCHAR(50)
EMAIL_EMPLEADO	NOT NULL	VARCHAR(50)
TELEFONO_EMPLEADO	NOT NULL	NUMBER(12,2)
CIUDAD_NAC		VARCHAR(50)
FECHA_NAC		VARCHAR(50)
SEXO_EMPLEADO	NOT NULL	CHAR(1)
CARGO_EMPLEADO	NOT NULL	VARCHAR(50)
FECHA_INGRESO	NOT NULL	VARCHAR(50)
SUELDO	NOT NULL	NUMBER(15,2)
NO_PROYECTO	NOT NULL	VARCHAR(50)

Figura 4.71 Descripción de la definición de columnas de tabla EMPLEADOS de la base de datos del SIM

```
INSERT INTO Empleados
VALUES('77745678','STRIDENGER BROWN ANNIE', 'Rua Mayor 87-40 ,
Salamanca, España','heartannie@gmail.com',300268789,'Bonn','17-
JAN-99','F','INGENIERA','02-JAN-19',5000000,'PA11TA1111').
```

Se debe tener en cuenta que los datos están siendo insertados en el mismo orden y número en que fueron creadas las columnas de la tabla; luego en la tabla hay 12 columnas definidas y en los valores de los datos hay 12. Los datos se corresponden con su tipo. La identificación del empleado está definida como un VARCHAR de longitud 10, mientras que el dato de entrada se especifica como carácter ('77745678') y con una longitud menor que 10. A su vez, el sueldo, como campo numérico, se especifica con caracteres numéricos.

Ejemplo 4.50 Inserción de tuplas a la base de datos especificando solo algunas columnas de la tabla

Insertar una nueva tupla en la tabla de EMPLEADOS de la base de datos del SIM; insertar obligatoriamente los campos no nulos de la tabla.

```
INSERT INTO Empleados
(Id_empleado,nombre_empleado,email_empleado,teléfono_
emopleado,sexo_empleado,cargo_empleado,fecha_ingreso,sueldo,No_
proyecto)
VALUES
('77845678','SUHEN',su218@gmail.
com',301268789,'F','INGENIERO','02DEC19',10000000,
'PA11TA1111')
```

Es importante enfatizar que los nueve campos de datos corresponden a las nueve columnas de la tabla de empleados que son no nulas; por esta razón se nombran exactamente las nueve columnas de la tabla de empleados, de acuerdo como aparecen en la figura 4.71.

Ejemplo 4.51 Inserción de tuplas en una tabla considerando campos nulos

Teniendo como base el SIM insertar la tupla del empleado con identidad 77945678, y especificar NULL en aquellas columnas definidas como no nulas en la base de datos.

```
INSERT INTO Empleados
VALUES
('77945678','SALAMANCA ROUX JUAN DAVID', NULL, 'eljully@
gmail.com', 31268789, NULL, NULL,'F','INGENIERA','02-JAN-
19',5000000,'PA11TA1111')
```

Luego, los tres campos: dirección física, ciudad de nacimiento y fecha de nacimiento de la tabla de empleados, son definidos como nulos (NULL) en la tabla de empleados.

4.7.2 Modificación de tuplas de la base de datos UPDATE

La actualización del contenido de los datos de una base de datos se hace con la cláusula UPDATE, cuyo formato es el siguiente:

```
UPDATE NombreTabla
SET NombreColumna1 = ValorDato1 [,NombreColumna1 = ValorDato1.
. .]
[WHERE condicionBusqueda]
```

El propósito de la cláusula UPDATE es permitir la edición de los contenidos de los datos de una tabla en una base de datos. La cláusula modifica el contenido de la fila de la tabla, en la columna que se indique en la tabla.

- NombreTabla es la identidad de la tabla que se va a modificar en la base de datos.
- NombreColumna corresponde al nombre de la(s) columna(s) de la tabla donde se desea modificar el contenido de los datos; las columnas se especifican a través de asignación SET.
- WHERE es opcional, y en el caso de que no se especifique el UPDATE, actualizará todas las filas de la tabla en la(s) columna(s) especificadas en el SET. En el caso de que WHERE se especifique, entonces se debe cumplir la condición de búsqueda con el fin de actualizar la(s) filas(s) del cruce columna_fila de la tabla donde se cumpla la condición.

Los requisitos para la ejecución de la cláusula UPDATE son los siguientes:

- La tabla que se va a modificar debe estar en el esquema del usuario de la base de datos.
- En el caso de que la tabla no pertenezca al esquema de la base de datos definida por el usuario, este debe tener privilegios de modificación de la tabla.

Ejemplo 4.52 Actualización de columnas, de una tabla en todas sus filas

Con base en el SIM actualizar la tabla PRODUCTOS aumentando en un 1% el costo de fabricación del producto en todas las filas de la columna.

```
UPDATE Productos    /*Nombre de la tabla que se va modificar
SET costo_producto=costo_producto*1.01; Nombre de la columna
cuyo contenidose va a modificar con el porcentaje indicado */.
```

Se debe tener en cuenta que en este caso se omite la cláusula WHERE por cuanto la actualización de la tabla de productos en sus costos de fabricación se va a aumentar en un 1%.

Ejemplo 4.53 Edición de tuplas o filas específicas de una tabla

Utilizando el esquema de la base de datos del SIM modificar la ciudad del proveedor asociado con el NIT 72228002 cambiando su sede de Tokio a Osaka.

```
UPDATE proveedores
/*El nombre de la tabla que se va a modificar es proveedores */
SET ciudad_proveedor = 'OSAKA'
WHERE NIT=72228002;
/*
Se cambia el contenido del dato por Osaka
Se cambia solo la ciudad del proveedor asociada al nit
72228002*/.
*/
```

Ejemplo 4.54 Actualización de varias columnas de una tabla

Se requiere modificar la factura correspondiente a la orden de venta numerada como VE-09-01-19 en la tabla de FACTURACIÓN cambiando la cantidad de unidades vendidas por 100, el descuento por 200 y editando el nit del comprador 18452699.

```
UPDATE facturación  /*El nombre de la tabla que se va a
modificar es facturación */

SET cantidad_vendida=100,  /*Se modifican los campos cantidad,
descuento y nit del descuento=200, comprador */

       nit_comprador=18452699

WHERE No_Orden_Venta = 'VE-09-01-19´; /* Ubicando los campos
mencionados de la orden de venta */
```

4.7.3 Borrado de tuplas de la base de datos DELETE

El borrado de datos de la base de datos se efectúa utilizando la cláusula DELETE, cuya estructura de su sintaxis y significado es la siguiente:

```
DELETE FROM NombreTabla

[WHERE CondiciónBúsqueda]
```

NombreTabla indica el nombre de la tabla que se va a borrar en el caso de que no se especifique la opción WHERE, asociada a la condición de búsqueda. Se debe tener en cuenta que la operación de un DELETE NombreTabla sin la condición de búsqueda ejecutará el borrado de todas las filas o tuplas de una tabla, por lo cual se debe tener el mayor cuidado en su operación en términos de fiabilidad en la conservación de los datos en una base de datos.

Al especificar la cláusula WHERE asociada con la condición de búsqueda en combinación con DELETE se borran entonces aquellas tuplas de la base de datos que cumplan con la condición asociada a la columna que se indique.

Los requisitos para la operación de la instrucción DELETE en el SQL son:

– La tabla que se va a cancelar debe estar definida en el esquema del usuario que va a borrar la tabla.

– En el caso de que la tabla no esté definida en el esquema de base de datos de propiedad del usuario, entonces este debe tener los privilegios para poder cancelar datos de la base de datos, representados por tablas o átomos (relación cruce de fila_columna) de la base de datos.
– Es lógico pensar que si se opera el DELETE con una condición de búsqueda, la(s) tupla(s) que se va(n) a cancelar debe(n) cumplir la condición de búsqueda; de otro modo, hay un error en la semántica de la ejecución del DELETE, porque si bien la cláusula en su sintaxis puede estar bien, en un sentido evaluativo de la operación de la cláusula se ejecuta pero no cancela nada de la base de datos.

Ejemplo 4.55 Cancelación de una tabla de la base de datos

Cancelar la tabla de PRODUCCIÓN de la base de datos del SIM.

```
DELETE FROM producción;
DELETE producción;
```

Los dos comandos relacionados borran todas las tuplas o filas de la tabla de producción del Sistemas Integrado de Manufactura.

Ejemplo 4.56 Cancelación de tuplas de una tabla por cumplimiento de condición

Cancelar de la tabla de MAQUINARIA del SIM aquellas tuplas cuyo estado de mantenimiento de las máquinas en sus números esté en 'T', o mantenimiento terminado.

```
DELETE from maquinaria
WHERE EST = 'T';
```

La instrucción dada cancela de la base de datos del SIM, específicamente de la tabla MAQUINARIA, solo aquellas tuplas cuyo campo EST, o estado de mantenimiento de la base de datos, se encuentra con el indicador alfabético 'T', lo cual indica que el mantenimiento de la máquina está terminado.

4.8 Resumen

El manejo de los datos a través de un SGBD se desarrolla a través de la cláusula SELECT, en sus especificades de: i) *Consultas simples*, que permiten: la selección de tuplas distintas y la generación de campos calculados; las búsqueda de tuplas basadas en comparaciones simples o compuestas; seleccionar tuplas por rangos, de pertenencia a un conjunto, por patrones o por atributos nulos; la generación de tuplas ordenándolas por resultados y por múltiples atributos ordenados y, finalmente, la generación de tuplas utilizando funciones de agregación. ii) *Consultas compuestas,* que permiten: la generación de subconsultas; consultas multitabla; generación de tuplas por combinaciones; selección de tuplas utilizando EXISTS y NOT EXISTS y la generación de tuplas por combinación de tablas de resultados.

Finalmente, se presentan las opciones que cumplen con la edición de los datos en sus opciones de inserción (INSERT), modificación (UPDATE) y borrado (DELETE) de los datos de la base de datos.

Terminada la presentación de consultas a la base de datos, el siguiente capítulo desarrolla los conceptos relacionados con la proyección de la tecnología de bases de datos.

Ejercicios

Suponga que un Sistema Académico de Bases de Datos de Estudiantes (SABDE) está estructurado de acuerdo con el siguiente esquema de entidades y atributos, con sus respectivas **claves,** como se muestra en la tabla 4.4.

Tabla 4.4 Sistemas Académico de Bases de Datos de Estudiantes

ENTIDADES	ATRIBUTOS	SIGNIFICADO
ASIGNATURAS	**codasg**	Código de la asignatura
	nomasg	Nombre de la asignatura
	nhtasg	Número de horas teóricas de la asignatura
	nhpasg	Número de horas prácticas de la asignatura
	crasg	Créditos de la asignatura
MATRICULA	**codest**	Código del estudiante
	codasg	Código de la asigntura
	codprog	Código del Programa
	gruasg	Grupo de la asignatura
ESTUDIANTES	**codest**	Código del estudiante
	nomest	Nombre del estudiante
	sexest	Género del estudiante
	nivest	Nivel del estudiante
	codprog	Código del Programa
PROGRAMAS	**codprog**	Código del Programa
	nomprg	Nombre del Programa
	mneprog	Sigla del Programa
PENSUM	**codasg**	Código de la asignatura
	codprg	Código del Programa
	nivasg	Nivel de la asignatura

Construir las siguientes consultas de la base de datos SABDE.

1.

a. Seleccionar todos los atributos de la tabla de estudiantes de SABDE.

b. Seleccionar los códigos y nombres de la tabla de asignaturas.

c. Listar todos los nombres de los programas cuya identidad comiencen por "ING" (Ingenierías).

d. Generar un listado de estudiantes ubicados entre los niveles 5 y 10 del programa 13.

e. Listar los nombres de las asignaturas con 3 horas teóricas y 2 horas prácticas, con número de créditos igual 3.

2.

a. Generar un listado de estudiantes mujeres del programa 14 del último semestre (10).

b. Seleccionar todos los programas de la base de datos SABDE cuya sigla sea IC (Ingeniería Civil).

c. Listar los pénsum académicos de la base de datos SABDE cuyos códigos de las asignaturas empiecen por los dígitos 13, pertenecientes al programa 13 y con niveles de asignaturas del primero y el décimo semestre de pregrado.

d. Listar las identidades de los estudiantes hombres no ubicados entre los niveles [1;5] y que no pertenezcan al programa 16.

e. Producir un listado de asignaturas cuyos nombres empiezan por los caracteres "EST" y que pertenezcan al programa 12.

3.

a. Generar un listado de asignaturas ordenadas alfabéticamente por nombres, y dentro de estos, ordenados por números de créditos de menor a mayor.

b. Listar todos los estudiantes de SABDE en sus atributos de código y nombre ordenados de menor a mayor por el código del programa, y dentro de cada programa, ordenados alfabéticamente.

c. Contar el número de estudiantes del programa con código 16, mujeres y pertenecientes al nivel 5 de pregrado.

d. Calcular los valores mínimos, máximos y promedios de los créditos de las asignaturas.

e. Producir un informe de estudiantes agrupados por el código del programa, en el que para cada programa se cuente el número de alumnos asociados a cada unidad académica.

4.

a. Generar un listado de asignaturas matriculadas pertenecientes al código del programa 10.

b. Producir un listado de programas académicos en los que haya estudiantes matriculados.

c. Construir un listado de estudiantes matriculados con sus códigos e identificaciones, cuyos códigos de las asignaturas sean mayores que el valor mínimo del número de créditos de la tabla de asignaturas.

d. Generar un listado de programas con sus respectivos pénsum académicos (planes de estudio o semáforos) en el que se muestran las asignaturas por nivel asociados a cada semestre académico.

e. Generar un listado que contenga todas las asignaturas matriculadas menores o iguales a 3 créditos académicos.

5.

a. Producir un listado de programas académicos con estudiantes matriculados.

b. Producir un listado de estudiantes matriculados pertenecientes al código del programa identificado con la sigla de Ingeniería Industrial (II), ordenados alfabéticamente, en el que se muestra las asignaturas matriculadas por el estudiante con su respectivo grupo.

c. Producir un listado de asignaturas ordenadas alfabéticamente en el que se muestre el código del programa académico, con su respectiva sigla de identificación.

d. Producir un listado ordenado por nombres de estudiantes que tengan asociadas asignaturas matriculadas.

e. Generar un listado en el que se presenten las asignaturas de SABDE utilizando RIGHT JOIN con la tabla de asignaturas matriculadas.

6.

a. Seleccionar todos los atributos de la tabla de asignaturas con créditos de asignaturas iguales a 3 asociadas al nombre de programa de Ingeniería Civil.

b. Generar un listado de programas utilizando LEFT JOIN con la tabla de estudiantes de SABDE.

c. Listar todos los nombres de los programas cuya identidad comiencen por "ING" (Ingenierías), en su atributo menor haciendo LEFT JOIN con la tabla de pénsum de SABDE.

d. Seleccionar de SABDE las identificaciones de los estudiantes agrupados por género que no se hayan registrado como alumnos matriculados.

e. Seleccionar de la tabla de pénsum de SABDE los atributos de identidad de las asignaturas para las cuales existe en el nombre de la asignatura el token "EST" (Estructura de Datos, Estadística).

7.

a. Producir un listado en el que se muestre la tabla de matrículas unida con la tabla de estudiantes del Sistema Académico de Base de Datos de Estudiantes.

b. Se requiere la información de las tablas de asignatura y pénsum académicos con el fin de identificar los códigos de las asignaturas no asociados con ningún pénsum académico.

c. Producir un listado de programas académicos que no tengan asociados estudiantes matriculados.

d. Listar los estudiantes con asignaturas matriculadas utilizando la intersección de las tablas de estudiantes y matrículas.

e. Seleccionar de la tabla de matrícula las asignaturas que no estén matriculadas en la tabla de asignaturas.

8.

a. Seleccionar de la SABDE todos los estudiantes del pénsum de Ingeniería de Sistemas que estando matriculados en el grupo 01 tengan asignaturas con 3 horas teóricas, 2 horas prácticas y 3 créditos.

b. Seleccionar todas las asignaturas de SABDE clasificadas por niveles de los alumnos matriculados del último semestre del programa cuya identidad en su sigla sea II (Ingeniería Industrial).

c. Producir un listado de programas con estudiantes matriculados por niveles en el que se muestren las asignaturas matriculadas ordenadas ascendentemente por el código de la asignatura.

d. Producir un listado de asignaturas para las cuales no existan estudiantes matriculados; hacer la consulta utilizando las cláusulas SQL MINUS y NOT EXISTS.

e. Generar una consulta que muestre los programas académicos sin pénsums asociados; construir la consulta utilizando las cláusulas SQL MINUS y NOT EXISTS.

9.

a. Contar el número de tuplas contenidas en SABDE de las entidades ESTUDIANTES, ASIGNATURA y MATRÍCULA.

b. Calcular el número de estudiantes del programa de sistemas (13) que matricularon la asignatura de proyecto de grado (2991 y 2990).

c. Teniendo en cuenta que los atributos que componen la entidad PROGRAMAS se muestran en la tabla 4.5.

Tabla 4.5 Atributos de la tabla de PROGRAMAS

Name	Null?	Type
CODPRG	NOT NULL	NUMBER(2)
NOMPRG	NOT NULL	VARCHAR(50)
MNEPRG	NOT NULL	CHAR(2)

Insertar un nuevo programa con código de programa 30, de nombre GEOLOGÍA, el cual se identificá con la sigla GEO.

d. Al calcular el número de estudiantes matriculados en cada programa, ordenados ascendentemente por códigos de programas, las últimas tres tuplas de la consulta son...

e. Al realizar la consulta: para cada programa indicar el número de estudiantes matriculados en cada pénsum imprimiendo todas las características asociadas a las asignaturas matriculadas por los estudiantes. La matrícula de la alumna ZAPATA ALVEAR JUANITA MARÍA es...

10.

a. Al combinar las relaciones de estudiantes con programas, ordenadas ascendentemente por los nombres de los alumnos, las cuatro primeras tuplas resultado son...

b. Al producir un listado de asignaturas asociadas a cada uno de los planes de estudios (pénsum) ordenados descendentemente por los códigos de los programas, las cuatro primeras tuplas de la consulta son...

c. Al generar un listado en orden ascendente con los nombres de las asignaturas matriculadas pertenecientes a todos los planes de estudios del programa de Ingeniería Industrial, las cuatro primeras tuplas de la consulta son...

d. Al determinar el número de asignaturas ofertadas por cada programa ordenadas por el código de programa, los resultados del programa con el máximo y el mínimo número de asignaturas son...

e. Eliminar de la tabla de programas las unidades académicas (programas) creados por *default* para prueba de la base de datos SABDE, o aquellas unidades cuya sigla mneprg es igual a "DEF".

Capítulo 5

Diseño de bases de datos orientadas a objetos

5.1 Síntesis conceptual

El paradigma de modelado de base de datos orientado a objetos se ha convertido en una técnica emergente para el desarrollo del *software*, en virtud de que en gran parte se busca estructurar una línea de producción estandarizada que se integre con la programación orientada a objetos; por otra parte, el modelo de bases de datos orientado a objetos permite resolver problemas complejos que para los paradigmas relacionales resultan de mayor grado de dificultad, especialmente en la estructuración de datos heredados, agregados, derivados y embebidos con aplicaciones[39]. Es por ello que resulta clave presentar los aspectos generales más importantes para el modelado de bases de datos orientados a objetos a fin de comprender este nuevo enfoque de modelado que, por supuesto, encaja con metodologías de desarrollo de *software* recientes.

5.2 Objetivos

Al término de la lectura de este capítulo y con base en el aprendizaje de los capítulos anteriormente expuestos se pretende que el lector aprenda a:

- Comprender los conceptos y la evolución de las bases de datos orientados a objetos.
- Comprender el proceso de modelo de bases de datos orientado a objetos.
- Conocer los detalles del modelado de bases de datos orientado a objetos en lo referente a la aplicación de la conceptualización de paradigma orientado a objetos para el modelado de bases de datos (objetos de información, atributos, métodos, restricciones y características avanzadas del modelado).

5.3 Introducción BDOO

El diseño de las bases de datos tradicionales se ha desarrollado sobre la estructura de modelos relacionales, jerárquicos, red y en algunos casos soportados en lenguajes de especificación y etiquetados como RDF, XML y UML, los cuales hasta el momento han podido satisfacer con éxito los requerimientos funcionales[40].

No obstante la evolución de los lenguajes de programación, los sistemas operativos y la misma evolución de los sistemas gestores de base de datos, como también la complejidad de los problemas que se deben resolver desde el punto de vista de datos, estructuras de datos complejas, referenciales, descriptivas, móviles, distribuidas, multimediales y vectoriales ha provocado que el modelado y diseño de datos en sí mismo deba evolucionar para poder modelar los nuevos requerimientos de los sistemas.

Los requerimientos de estas aplicaciones son muy distintos de los convencionales; aquellos que por tradición se sitúan en el manejo de datos centralizados, federados, han migrado a modelos de base datos más abiertos, distribuidos y dinámicos. Por la naturaleza de los requerimientos de las aplicaciones emergentes, el modelo de datos resulta más complejo al aparecer nuevos tipos de relaciones más dinámicas, mayores restricciones, demanda mayor flexibilidad a la hora de factorizar o simplificar bases de datos; las transacciones igualmente son en tiempo real, complejas de extensa duración, manejan múltiples condiciones de excepción que deben ser modeladas, diseñadas e implementadas. Por lo tanto, todo esto obliga a crear nuevos tipos de datos para almacenar mapas, vectores, polígonos, imágenes, textos, símbolos, y por consiguiente requiere incorporar operaciones no estandarizadas para cada aplicación.

Para resolver en parte el problema descrito ha emergido la aplicación del modelado, diseño e implementación de bases de datos orientada a objetos, el cual realmente no es un concepto nuevo propiamente dicho; realmente se puede indicar que ha sido una evolución natural de las estructuras de datos que han estado presente en los lenguajes de programación orientados a objetos (C++, Smalltalk o Java), pero que en este caso, en lugar de ser dinámicos completamente, se orientan hacia la persistencia, es decir, a mantener datos a lo largo de tiempo, tiempo que es definido dentro las reglas de la transacción y los datos que la soportan. Una de las características importante es que el modelo y diseño de bases de datos orientado a objetos (MDBDOO) adopta completamente los conceptos asociados con la programación orientada a objetos (POO), y este aplica en todas las etapas del proceso de desarrollo de las bases

de datos. El mismo modelo de dominio que se identifica en el plano conceptual durante la especificación y análisis de necesidades se transforma directamente en un modelo de objetos de *software* interconectados. Un ejemplo de modelo de dominio en un contexto de una universidad se visualiza en la figura 5.1, el cual consiste en identificar relaciones los objetos de información primarios, en este caso objetos como estudiante, profesor, curso, programa, facultad y universidad.

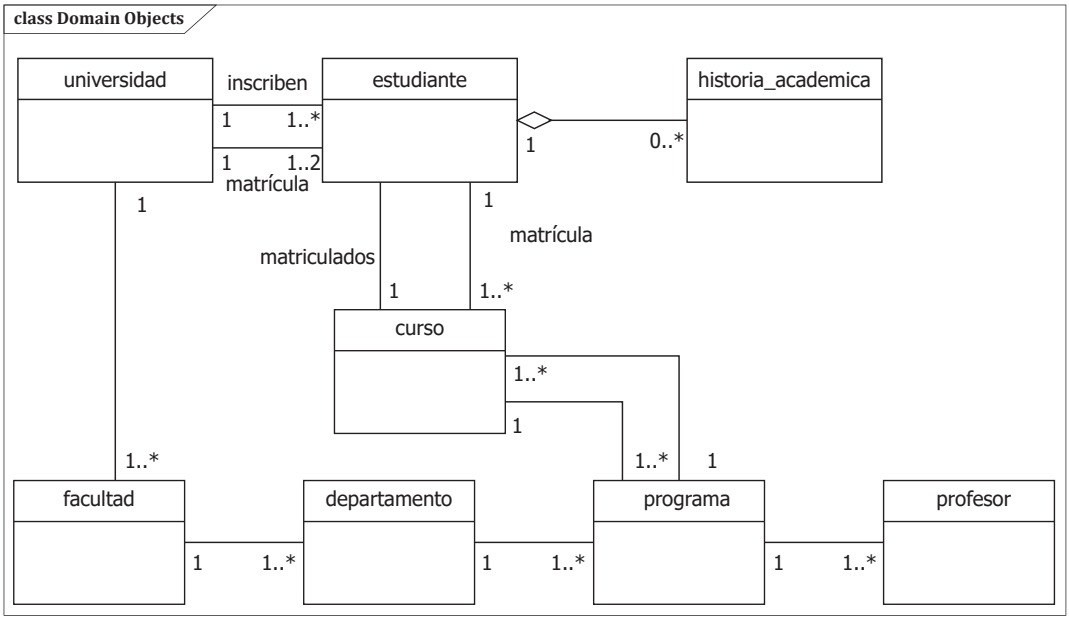


Figura 5.1 Modelo de dominio de un contexto universitario

5.4 Orientación a Objetos

El paradigma orientado a objetos aporta cambios importantes en la manera como se describen, modelan, diseñan en implementan las estructuras de datos. Los conceptos básicos del POO tienen origen en Simula 67, un lenguaje diseñado para hacer simulaciones, creado por Ole Johan Dahl y Kristen Nygaard; de allí en adelante han emergido múltiples lenguajes de programación que han integrado funciones, datos extendidos, métodos, funciones y procedimientos y que han enriquecido y redefinido el enfoque del modelo de datos. De esta forma, los conceptos asociados

al paradigma orientado a objetos son válidos para este contexto. Veamos a continuación los conceptos básicos [41].

Clase: es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones, restricciones y se describen de acuerdo con una semántica.

Objeto: es una unidad de información compuesta por propiedades (atributos) y comportamiento (funcionalidad, métodos) que buscan representar un objeto real del mundo o del contexto en análisis.

Atributo: es una propiedad de una clase identificada con un nombre, dentro de un rango de valores y con restricciones.

Operación: es la implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre un comportamiento; es una abstracción que describe unas acciones lógicas que son ejecutadas por el objeto.

Mensaje: son peticiones lanzadas desde una aplicación a otro objeto con el fin de que realice alguna función; por ejemplo, un objeto puede aceptar un mensaje y como respuesta realizará una acción y devolverá un valor.

La herencia: consiste en que una clase puede heredar sus variables y métodos a varias subclases (la clase que hereda es llamada "superclase" o "clase padre"). Esto significa que una subclase, aparte de los atributos y métodos propios, tiene incorporados los atributos y métodos heredados de la superclase.

Encapsulamiento: consiste en unir en la clase las características y comportamientos, esto es, las variables y métodos. Es tener todo esto en una sola entidad. La importancia del encapsulamiento permite manejar la complejidad de las clases como cajas negras donde solo se conoce el comportamiento pero no los detalles internos; esto es clave, ya que es conveniente conocer qué hace la Clase, pero no será necesario saber cómo lo hace.

Abstracción: Consiste en captar las características de un objeto y su comportamiento. Veamos, por ejemplo, las características de una clase llamada estudiante en un contexto de una universidad; tendríamos que las características o atributos son:

nombre, apellidos, dirección, género, fecha de nacimiento, dirección contacto (sin tener en cuenta la descomposición relacional). Y en cuanto a su comportamiento, acciones como: crear, modificar, consultar, datos del estudiante, etc.

Tipos de clases: En el contexto del modelado de dominio se maneja el mismo esquema de los programas orientados a objetos, es decir, se manejan tres tipos de clases: clases de control, clases entidad y clases interface.

Las clases de control gestionan el flujo de operación o transacción en la modelación de una aplicación (por ejemplo, donde se encapsulan las operaciones o métodos); las *clases entidad* son las que se utilizan para crear objetos que manejan datos (por ejemplo, clases para personas, objetos tangibles o eventos), y las *clases interface* son las que manejan la entrada y la salida de información (por ejemplo, las interfaces gráficas y los formularios de entrada y salida de datos utilizados por una aplicación).

Tipos de métodos: Existen varios métodos que son comunes a la mayoría de las clases: *Método constructor*: tiene el mismo nombre que la clase. Se ejecuta cuando se crea un objeto de una clase. Por lo tanto, un constructor contiene instrucciones para inicializar las variables de un objeto. *Método destructor*: es un método que se utiliza para destruir un objeto. *Método accesor*: es un método que devuelve el valor de un atributo privado de otro objeto. De esta forma es como los objetos externos pueden acceder a los datos encapsulados. *Método mutador*: es un método que almacena un nuevo valor en un atributo. Así los objetos externos pueden modificar los datos encapsulados.

5.5 El proceso de modelo de bases de datos orientado a Objetos

Una de las características claves del enfoque de desarrollo orientado a objetos es que los mismos conceptos básicos de la programación, el diseño y el modelado orientado a objetos se pueden aplicar en todas las etapas del proceso de desarrollo del *software* (base de datos y aplicaciones).

El mismo modelo de dominio que se identifica en el plano conceptual durante la especificación y análisis de requerimientos se transforma directamente en un modelo de objetos de información relacionados. Muchos de los conceptos de orientación a objetos básicos (que modelan el mundo con clases de objetos, la integración, el

comportamiento de los datos, la herencia, encapsulación y polimorfismo) se pueden aplicar sin problemas en diferentes niveles de abstracción del modelo de las bases de datos orientada a objetos. A través de los sistemas de gestión de bases de datos orientadas a objetos (OODBMS) se proporcionan los mecanismos para la definición de datos, manipulación de datos y control de datos [42].

No es práctico para aplicaciones orientadas a objetos mantener todos los objetos relevantes en memoria en tiempo de ejecución todo el tiempo; por lo tanto, a pesar del fracaso de la OODBMS de hacerse popular, está claro que los entornos de desarrollo orientado a objetos necesitan mecanismos para almacenar los estados del objeto entre las sesiones de ejecución de la aplicación.

Almacenar el estado de un objeto entre las sesiones de ejecución de la aplicación se llama *persistencia del objeto*. Los lenguajes orientados a objetos proporcionan un mecanismo para almacenar el estado persistente de un objeto. *La serialización* es el proceso de escribir un objeto en un medio de almacenamiento o un canal de comunicación como un flujo de datos.

El proceso de modelado orientado a objeto se puede resumir así:

1. Identificación de las funcionalidades de los datos (reglas semánticas).
2. Identificación de los objetos de información.
3. Elaboración del Modelo de Dominio.
4. Mapeo de datos: Descripción de los objetos de información (atributos, métodos y restricciones).
5. Identificación y establecimiento de la multiplicidad (fuente y destino).
6. Identificación de las clases asociativas.
7. Identificación de las clases abstractas, concretas.
8. Documentación del modelo de datos.

5.5.1 Identificación de las funcionalidades de los datos (reglas semánticas)

Es un proceso de abstracción que busca identificar las reglas semánticas y sintácticas del sistema, como por ejemplo, qué tipo relaciones se generan entre los diferentes

objetos; en el contexto del ejemplo de la Universidad; veamos, por ejemplo, el caso en que se tiene una facultad, la cual tiene asociados varios profesores; como objetos de información se describen, pero adicionalmente se deben describir y establecer los aspectos relacionados con la jerarquía y organización; en este caso suponga que varios tipos de profesores en la facultad tienen la categoría de *titulares, asociados y asistentes* y que se hace necesario modelar estas características y métodos particulares y vinculantes para cada grupo; en este caso es clave representar las características asociadas a la herencia o composición.

5.5.2 Las reglas semánticas

Agregación: Un conector de agregación es un tipo de asociación que muestra que un elemento contiene o se compone de otros elementos. Se utiliza en los modelos de clase para mostrar cómo los elementos más complejos (agregados) se construyen a partir de una colección de elementos más simples (partes componentes, por ejemplo, un coche de ruedas, llantas, motor, etc.). En la figura 5.2, en el ejemplo en el contexto de la Universidad, se puede establecer que los departamentos están conformados por laboratorios y programas; el modelo se representa ahora así:

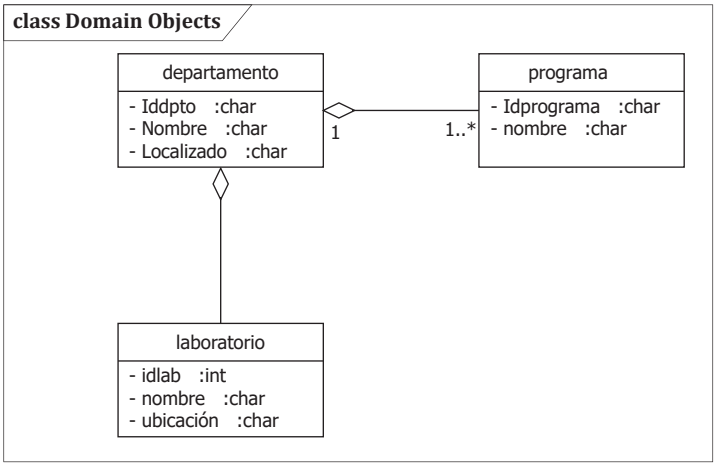
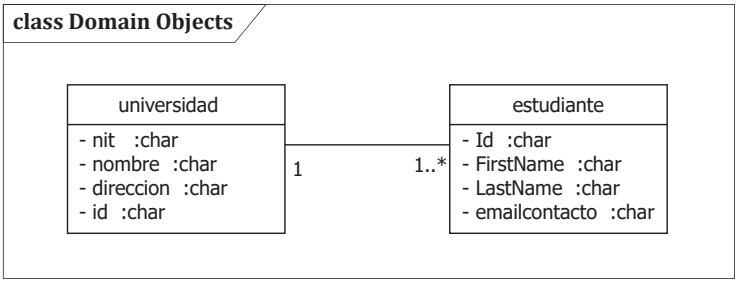


Figura 5.2 Conector de agregación del ejemplo del contexto de la Universidad

Asociación: Una asociación implica la relación entre dos elementos del modelo normalmente. Este conector puede incluir funciones nombradas en cada extremo, la

multiplicidad, dirección y restricciones. La asociación es el tipo de relación general entre elementos. Para conectar más de dos elementos en una asociación puede utilizar el elemento de asociación N-aria.



En la figura 5.3, en el ejemplo en el contexto de la Universidad, se puede establecer la asociación entre universidad y estudiante de carácter binario; mientras que en la figura 5.4 se puede visualizar una relación n-aria.

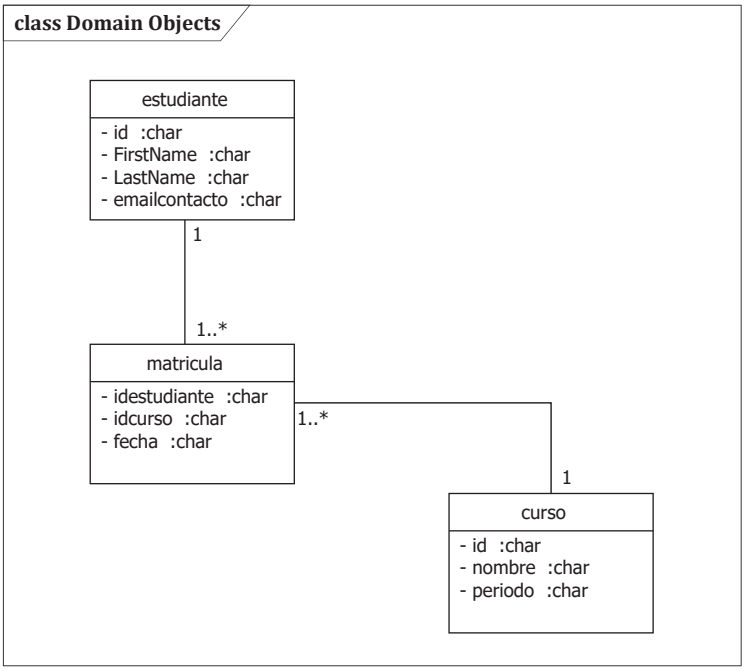


Figura 5.3 Asociación entre universidad y estudiante de carácter binario

Una clase de asociación es una construcción que permite una asociación de atributos y operaciones. Esto resulta en una relación híbrida con las características de una asociación y una clase.

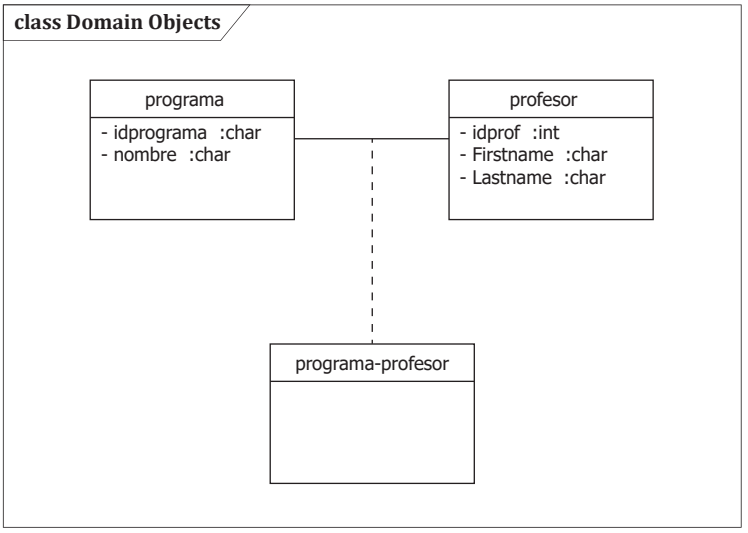


Figura 5.4 Visualización de una relación n-aria

Generalización: Es utilizada para indicar la herencia. Es decir, la relación entre dos entidades. Las implicación de la generalización consiste en que la entidad fuente (padre) hereda las características a la entidad destino (hija).

En la figura 5.5, en el ejemplo en el contexto de la Universidad, se puede establecer una relación de generalización entre la entidad "estudiante" (padre) y la entidad "aspirante" (hijo); está última hereda las características y operaciones de la entidad padre.

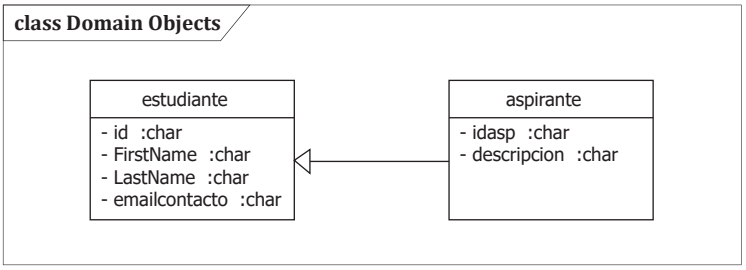


Figura 5.5 Relación de generalización

Dependencia: Relaciones de dependencia se utilizan para modelar una amplia gama de relaciones de dependencias entre las entidades. Aunque su aplicación no es común en las base de datos, pueden emplearse para detonar una relación de dependencia funcional entre una entidad y un tipo de datos definido por el usuario.

En la figura 5.6, en el ejemplo en el contexto de la Universidad, se puede establecer una relación de dependencia entre la entidad "facultad" y el clasificador "tipos de facultades", para de esta manera denotar un grupo enumerado de facultades (salud, ingeniería, educación, negocios.)

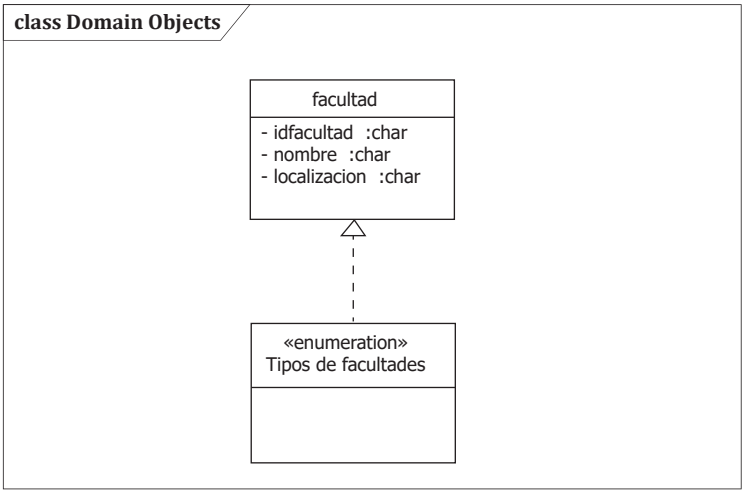


Figura 5.6 Relación de dependencia

5.5.3 Identificación de los objetos de información

Clase abstracta: Este tipo permite crear "estructuras de datos genéricas", que son extraídas del contexto del sistema que se va a modelar, pueden contener atributos y comportamientos generales sin especificar cómo lo hacen. Eso significa que en su descripción sus métodos no tienen "declaración"; estas clases, como son generalidades, no pueden ser instanciadas por ningún objeto, entonces su único fin es ser heredado/extendido por otras clases.

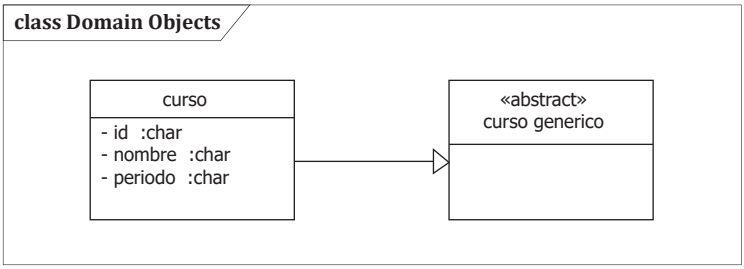


Figura 5.7 Clase abstracta

En la figura 5.7, en el ejemplo del contexto de la Universidad, se puede establecer una clase abstracta, en la cual se establece un patrón o estructura de datos genérica que contiene los datos estandarizados de los cursos de la universidad; la entidad "curso" hereda las características de la clase abstracta en mención.

Clase concreta: Es una entidad de información simple que en contexto de base de datos orientada a objetos se modela para describir las características y el comportamiento de la misma; por ejemplo, la entidad "profesor", "curso", "estudiantes", entre otras, en el modelo de la Universidad.

Clase hoja: Es una entidad de información simple que en contexto de base de datos orientada a objetos se modela para describir las características y el comportamiento de la misma de una clase que depende de una clase padre, que ya se mostró cuando se habló de la generalización.

5.5.4 Elaboración del modelo de Dominio

El modelo de dominio es un modelo de clases de alto nivel que muestran una vista estática del sistema de datos, o parte del modelo, que describe atributos y el comportamiento a nivel general sin lugar a detallar los métodos para lograr operaciones. El modelo de dominio es más útil para ilustrar las relaciones entre las clases y las interfaces. Las generalizaciones, agregaciones y las asociaciones son todas valiosas en lo que refleja la herencia, composición o uso, y las conexiones, respectivamente. En la figura 5.8 se ilustra las relaciones de agregación entre clases.

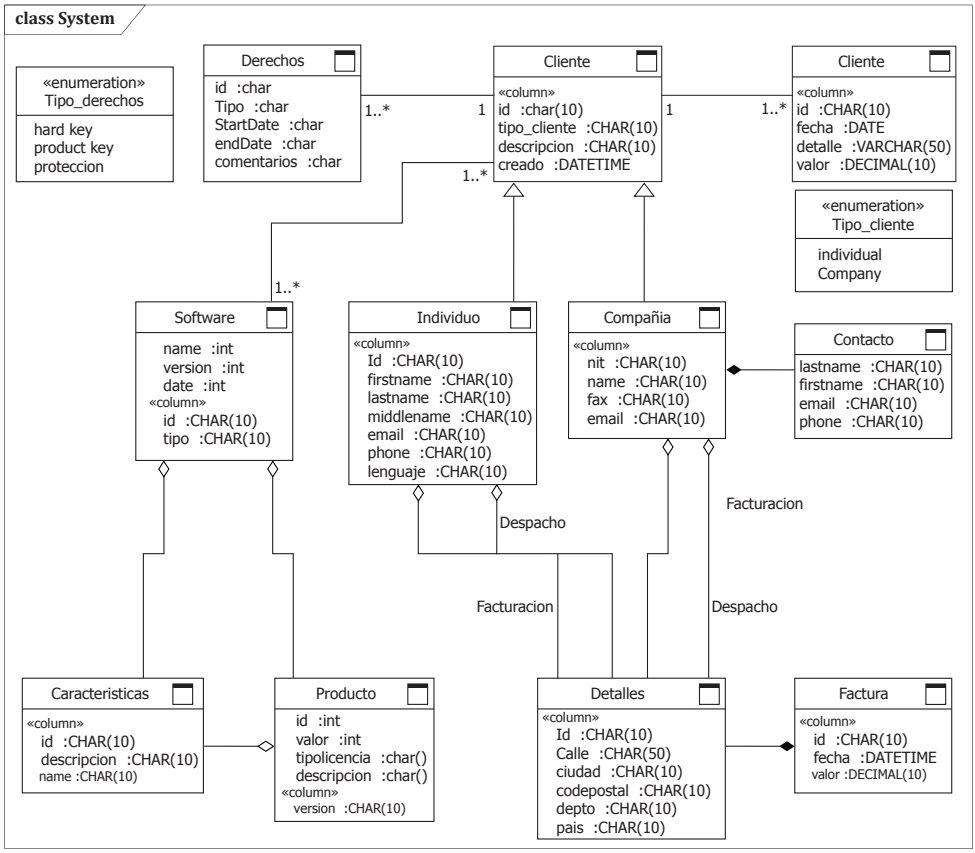


Figura 5.8 Relaciones de agregación entre clases

5.5.5 Mapeo de datos

El proceso de modelado orientado a objetos consiste en desarrollar el mapeo de datos, que en este caso busca describir los detalles derivados desde cada clase, como también incorporar aquellos elementos relacionados con la integración de las clases de control, entidad e interface.

- *Nombre de la clase*: debe tener un nombre simple, un sustantivo y en singular.

Las clases en su descripción tienen el siguiente detalle.

- *Atributo*: es una característica de una clase que se deriva del contexto del sistema que se modela o del mundo real.

El atributo contiene los siguientes elementos:

- *Nombre del atributo*: nombre del atributo, por ejemplo: edad.
- *Tipo de dato*: dependiendo del sistema de base de datos orientado a objetos puede tomar boolean, byte, char, doublé, float, int, long, short, none.
- *Valor inicial*: valor que toma inicialmente el atributo dentro del objeto; por ejemplo, edad=0.
- *Alias*: un nombre sustituto que facilita las operaciones dentro del proceso de gestión de datos.
- *Notas*: información que no tiene efectos semánticos pero facilita la documentación del modelo.
- *La visibilidad*: es una característica específica que permite establecer el ámbito de operación del atributo.
- *Public*: cualquier clase externa con visibilidad hacia la clase dada puede utilizar la característica o el atributo. Se denota con el símbolo +.
- *Protected*: cualquier descendiente del clasificador puede utilizar la característica. Se denota con el símbolo #.
- *Private*: Solo el propio clasificador puede utilizar la característica. Se denota con el símbolo.

En 5.9 se presenta un esquema del modelo de base de datos Universidad con más detalles y mayor descripción que en el modelo de dominio.

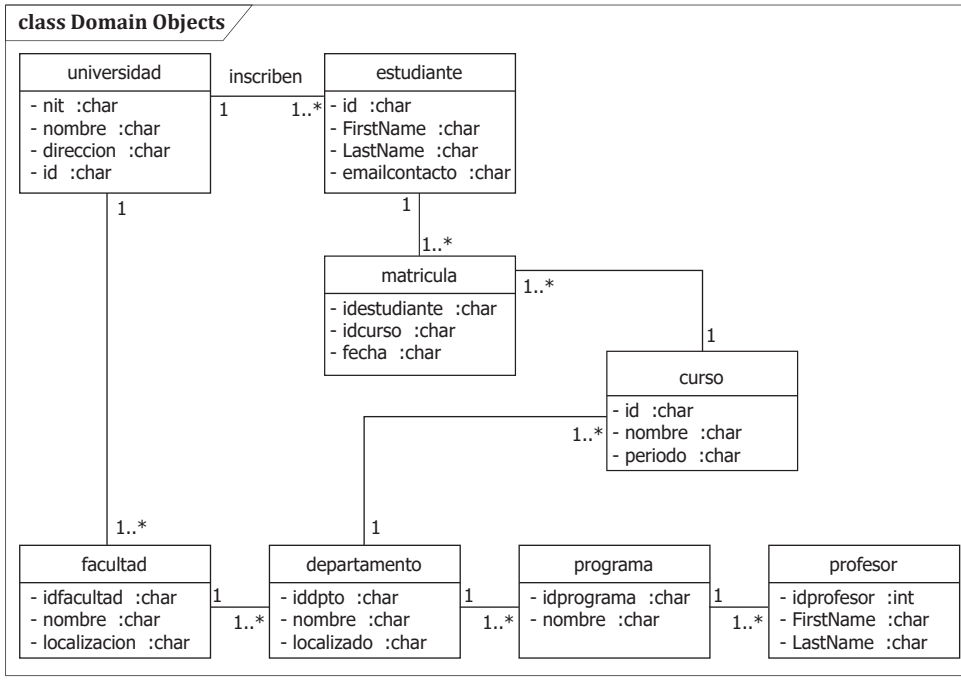


Figura 5.9 Esquema detallado del modelo de la base de datos universitaria

5.5.6 Identificación y establecimiento de la multiplicidad (fuente y destino)

Cuando se utiliza una clase es razonable asumir que puede existir cualquier número de instancias de ella; por ejemplo, en el caso de la clase "estudiante" es seguro que existan muchas de ellas, cada uno de los objetos estudiante representa la instanciación de la clase. Lo más frecuente es que dentro del modelado de un sistema una clase tenga al menos una instancia, varias bien definidas o muchas. El número de instancias que puede tener una clase en tiempo de ejecución es su *multiplicidad*. Esta consiste en el rango de cardinalidades permitidas que puede asumir una entidad. En la figura 5.10 se puede ver un ejemplo de la multiplicidad entre la clase "programa" y la clase "profesor".

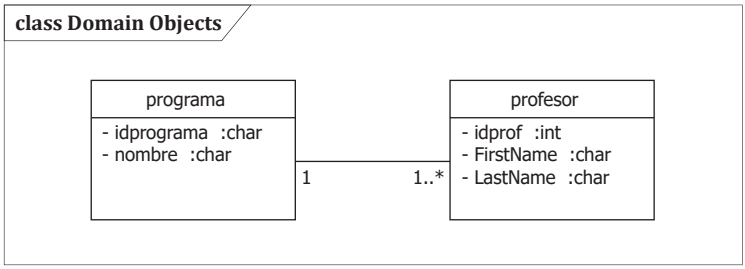


Figura 5.10 Multiplicidad entre las clases "programa" y "profesor"

Notaciones utilizadas en este caso son:

- **1:** Indica el número de instancias de la clase "programa" que se relacionan con la clase "profesor"; en este caso 1, la cual se extrae del contexto del problema que esté resolviendo o del contexto del sistema de datos que se esté diseñando; un programa de la universidad tiene asociados varios profesores (muchos).
- **1 ..*:** Indica el número de instancias de la clase "profesor" que se relacionan con la clase "programa"; en este caso, al menos 1 o más profesores están asociados con un programa.
- 1-.5: Indica el número de instancias de la clase "estudiantes" que se relacionan con la clase "programa"; en este caso, al menos 1 estudiante se relaciona con 5 matrículas (figura 5.11) de manera histórica (en un contexto supuesto).

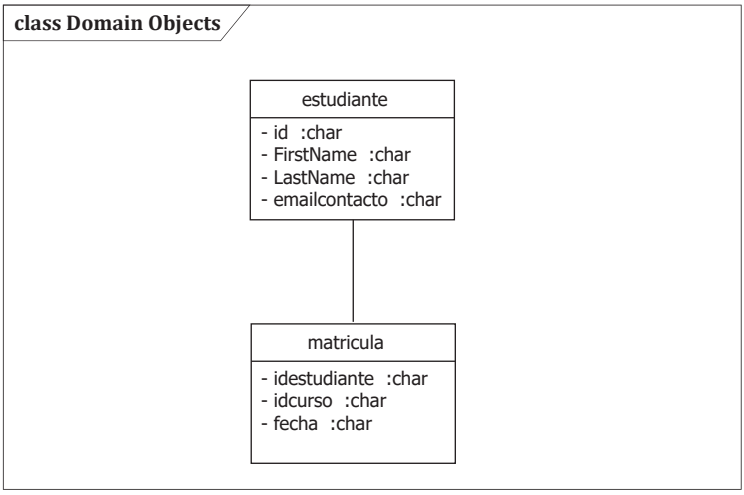


Figura 5.11 Instanciación de la relación histórica de un estudiante con sus matrículas

5.5.7 Recomendaciones para modelar una base de datos orientado a objetos

- Identificar aquellas clases del modelo cuyo estado debe trascender el tiempo de vida de las aplicaciones.
- Crear un modelo de clases (entidades) que contenga estas clases y marcarlas como persistentes; se puede definir un conjunto de valores etiquetados o enumerados para tal propósito y cubrir los detalles de la base de datos que se va a modelar.
- Expandir los detalles estructurales de las clases; esto significa detallar los atributos, las asociaciones, la multiplicidad en general.
- Buscar patrones comunes que permitan detallar el modelo, tales como asociaciones cíclicas, asociados uno a uno, asociaciones n-arias y asociativas.
- Crear las abstracciones necesarias, como generalizaciones, dependencias, herencia y agregación.
- Tener en cuenta el comportamiento de las clases expandiendo las operaciones que sean importantes para el acceso a los datos, la integridad y la seguridad de los datos.

- Integrar las reglas del contexto (sistema, negocio, ámbito) que incorporan restricciones (llaves principales, llaves foráneas, valores "not null", valores condicionales).
- Integrar herramientas que permitan transformar el modelo de datos en instancias de base de datos.

5.6 Paradigmas emergentes de modelo de datos NoSQL o base de datos NoSQL

Las base de datos NoSQL, también llamadas en el contexto "base de datos no solamente SQL", es un enfoque reciente o un nuevo paradigma de gestión y diseño de base de datos que puede utilizarse para definir, gestionar y controlar grandes volúmenes de datos con modelos centralizados, distribuidos o virtualizados. Se pude decir que la base de datos NoSQL es una clase de base de datos muy diferente del modelo tradicional de base de datos relacional, las cuales se conocen principalmente porque usan SQL como el lenguaje nativo de consultas de datos.

En base de datos NoSQL, los datos se almacenan es estructuras dinámicas, normalmente no soportan operaciones JOIN, ni permiten plenamente operaciones tipo ACID (atomicidad, consistencia, aislamiento y durabilidad), pero también hay que afirmar que la base de datos NoSQL también pueden soportar lenguajes de consulta de tipo SQL. Los sistemas de base de datos NoSQL evolucionaron particularmente en el entorno de los sistemas Web y de los más recientes frameworks de desarrollo, como Ruby on Rails, Django, Turbogears de Python, entre otros. Este nuevo paradigma ha estado presente en las aplicaciones sociales más reconocidas del mundo, como Twitter y Facebook.

5.7 Tipos de base de datos NoSQL

Hay cuatro tipos generales de las bases de datos NoSQL, cada uno con sus propios atributos específicos, así:

Key-Value: Este tipo de base de datos NoSQL es menos compleja. Estas base de datos está diseñada para el almacenamiento de datos de una manera esquema claves-valor, todos los datos están conformado por una clave y un valor, de ahí el nombre. Ejem-

plos de este tipo de base de datos incluyen: Cassandra, DyanmoDB, Azure Storage Table (ATS), Riak, BerkeleyDB.

Column store: También denominado como almacenamiento en columnas, en lugar de almacenar los datos en filas, esta base de datos está diseñada para almacenar las tablas de datos como secciones de columnas de datos, en lugar de filas de datos. Pueden verse como un modelo inverso de una base de datos estándar. Ejemplos de este tipo de base de datos incluyen: HBase, BigTable y Hypertable y otras.

Document database: se expande en la idea básica de almacenamiento: key-value, donde "documentos" son más complejos, porque contienen los datos y a cada documento se le asigna una clave única, que se utiliza para recuperar el documento. Estos están diseñados para almacenar, recuperar y gestionar la información orientada a documentos, también visto como datos semiestructurados. Los ejemplos incluyen sistemas de base de datos como MongoDB y CouchDB.

Graph database: Soportada en la teoría de grafos, esta base de datos está diseñada para representar datos cuyas relaciones están en forma de gráfico y tiene elementos que están interconectados entre ellos. Los ejemplos incluyen sistemas de bases de datos como Neo4J y políglota.

5.8 Porqué utilizar base de datos NoSQL

Las razones para que los diseñadores utilicen sistemas de base de datos NoSQL sobre base de datos relacionales tienen mucho que ver con los siguientes factores:

El crecimiento de los grandes volúmenes como Big Data: Big Data es uno de los principales paradigmas que están impulsando el crecimiento y la popularidad de los sistemas de base de datos a NoSQL. La amplia gama de tecnologías emergentes que permiten la recolección de datos que van desde transacciones "on line" hasta sistemas transacciones "on line", como tiendas virtuales, tiendas "on lines", utilizando dispositivos para captura de información tipo GPS, con teléfonos inteligentes y tabletas, hasta sofisticados sensores, están contribuyendo a capturar grandes columnas de datos en tiempo real. Entre las características de los proyectos de Big Data están:

– *Velocidad*: Alta velocidad de datos. Una gran cantidad de datos que provienen de diferentes fuentes, a una alta velocidad, y desde diferentes lugares.

- *Variedad de datos*: almacenamiento de datos estructurados, semiestructurados y no estructurados.
- *Volumen de datos*: datos que involucra a muchos terabytes o pentabytes de tamaño.
- *Complejidad de datos*: los datos almacenados y gestionados en diferentes lugares o centros de datos.

Modelos, Diseños de datos flexibles, escalables y funcionales: Las organizaciones cada vez más están interesadas en migrar a sistemas de base de datos NoSQL; por lo general son modelos de datos más flexibles que los modelos relacionales. El modelo de datos relacional se basa en relaciones definidas entre las tablas, que a su vez se definen por una estructura de atributos y restricciones determinada; todas ellas se estructuran de forma explícita en un esquema de base de datos de forma muy estricta y uniforme, de tal forma que cualquier cambio en la estructura es altamente costosa desde el punto de vista de la administración, con toda; la carga de riesgos que puede generarse con el manejo la integridad referencial. Por ejemplo, modificar la estructura de datos de un cliente puede afectar otras entidades que estén relacionadas con este, verbigracia, pedidos, despacho, pago, facturación, todo ello en un contexto de un sistema de ventas.

Regularmente los modelos de datos NoSQL pueden soportar muchos de estos casos de escalabilidad y flexibilidad que los mismos sistemas de base de datos soportados en un RDBMS. Una base de datos NoSQL es capaz de integrar todo tipo de datos estructurados, semiestructurados y no estructurados de una manera más fácil que una base de datos relacional. Esta característica asociada a una base de datos relacional puede ser un problema en cuanto a la flexibilidad, debido a que un esquema predefinido determina la rigidez de cómo se organizan los datos dentro de las bases de datos. Muchas de las aplicaciones en los ámbitos de las organizaciones buscan contar con sistemas flexibles que les permita escalar y modificar sus estructuras de datos, de tal forma que puedan responder rápidamente a los cambios que demanda su entorno, como por ejemplo, adoptar rápidamente ofertas de productos, modificación de catálogos de productos y otro tipos de reglas que se hace necesario implementar en tiempo real.

Analíticas e inteligencia de negocios: Un elemento clave para la implementación de base de datos NoSQL es dotar de la capacidad para extraer los datos o conoci-

miento a partir de patrones o estadística descriptiva, que se recopilan con el fin de obtener la mejor información de negocio en una ventaja competitiva.

La extracción de conocimiento desde las diferentes fuentes de información, base de datos estructuradas y no estructuradas, encaja dentro de las aplicaciones de inteligencia de negocio, la cual recopila altos volúmenes de datos. Los sistemas de bases de datos NoSQL no solo proporcionan un almacenamiento y gestión de datos a los negocios, sino que también ofrecen análisis de datos integrados que proporcionan la comprensión instantánea de conjuntos de datos complejos y facilitan la toma de decisiones en las organizaciones; por ejemplo, patrones de clientes, comportamiento de compras de los mismos, información resumida, agrupada por atributos y en general consultas basadas en agregación para la toma de decisiones.

5.9 Recomendaciones prácticas para seleccionar sistemas de base de datos NoSQL

A continuación se presentan algunas consideraciones claves para elegir una plataforma de Base de datos NoSQL:

Rendimiento: Una de las exigencias en un entorno de aplicaciones "on line" es garantizar tiempos de respuesta y disponibilidad aceptables, regularmente en nanosegundos, los sistema de Big Data deben moverse a velocidades extremadamente altas, sin importar en qué momento se cambia la escala o qué cargas de trabajo debe realizar su base de datos. El rendimiento de su entorno, es decir, sus aplicaciones, debe ser alto y estar articulado con la lista de requisitos para la implementación de una plataforma de NoSQL.

La disponibilidad: La disponibilidad de los datos para llevar a cabo las operaciones es una consideración importante para el rendimiento; la disponibilidad demanda sistemas habilitados 24 horas durante 7días; los datos siempre deben estar disponibles; no debe haber ningún tipo de fallos fallo en un entorno NoSQL, por lo cual se puede asegurar que las aplicaciones están siempre disponibles.

Escalabilidad: Con los grandes volúmenes de datos que hoy en día se deben manejar en las organizaciones, el sistema debe ser capaz de escalar muy rápidamente y elásticamente, cuando y dondequiera. Esto se aplica a todas las situaciones, ya sea escalando a través de múltiples centros de datos, e incluso a la nube, si es necesario;

para ello se suelen utilizar tecnologías bajo el modelo de entrega de computación elástica que permita instanciar modelos de bases de datos**.**

Costos: es una de las variables más importantes para hacer el cambio a una plataforma NoSQL que cumplen incluso una de las consideraciones que aquí se presentan con la tecnología de base de datos relacional, es decir, que pueden ser comparativamente costosas. Sin embargo, la implementación de sistemas de base de datos NoSQL en una línea de corto plazo puede generar beneficios importantes que reducen los costes operativos.

La gestión de los datos: La complejidad operacional de una plataforma NoSQL debe mantenerse a un mínimo. De tal forma que se pueda asegurar la administración y el desarrollo a través de las herramientas; y para ello se requiere mantener y maximizar los beneficios para pasar a un entorno NoSQL.

5.10 Algunos ejemplos de modelos de datos tipo NoSQL

5.10.1 Base de datos NoSQL –Apache Cassandra–

Apache Cassandra es uno de los más importantes motores de base de datos No SQL de carácter distribuido y basado en un modelo de almacenamiento de «Key-Value», de código abierto, que está escrito en Java. Este motor de base de datos permite grandes volúmenes de datos en forma distribuida. Existen varias aplicaciones importantes que utilizan Cassandra; entre ellos Twitter para su plataforma. Además es una de las base de datos NoSQL más relevantes a nivel mundial, la cual es utilizada por compañías como Netflix, eBay, Twitter, Urban Airship, Constant Contact, Reddit, Cisco, OpenX, Digg, CloudKick, Ooyala. Una de sus características consiste en que facilita la escalabilidad y la disponibilidad. La arquitectura de Cassandra está basada en una serie de nodos iguales que se comunican con un protocolo P2P, debido a lo cual la redundancia es máxima. Fue desarrollada por Apache Software Foundation. Cassandra puede manejar varios terabytes de datos si lo necesita y puede manejar fácilmente millones de ficheros, incluso en un clúster pequeño (Big Data).

5.10.2 Modelo de datos en Cassandra

Cassandra contiene un modelo de datos híbrido, entre un modelo Clave-Valor y una base de datos Tabular (orientado a columnas); la familia de columnas se asemeja a

una tabla en un RDBMS. Estos modelos de datos están compuestos por filas y columnas. Cada fila tiene múltiples columnas y cada una de estas tiene a su vez un nombre, un valor y un "timestamp".

Se diferencia de un base de datos RDBMS, en la que las tablas tienen diferentes filas en una misma familia de columnas, porque en esta no se tiene que compartir el mismo conjunto de columnas, y además una columna puede ser añadida a una o a múltiples filas en cualquier momento.

Casa Key en Cassandra corresponde a un valor que es a su vez un objeto. Cada clave tiene valores como columnas, y las columnas son agrupadas en conjuntos llamados "familias de columnas". Así, cada key identifica una fila con un número variable de elementos. Estas familias de columnas pueden ser consideradas como tablas. Una "tabla" en Cassandra es un mapa multidimensional distribuido, indexado por una clave. Además, las aplicaciones pueden especificar el orden de las columnas dentro de una Supercolumna, o una Familia de Columnas Simples.

5.11 Conceptos de base de datos en Cassandra

- **Column**. Es la unidad más básica en el modelo de datos de Cassandra. Una *column* es un triplete compuesto por una "key" (un nombre), un "value" (un valor) y un "timestamp". Los valores son todos suministrados por el usuario. El tipo de dato del "key" y el "value" son matrices de bytes de Java; el tipo de dato del "timestamp" es un "long primitive", mientras que las "column" son inmutables, para evitar problemas de multithreading. Las "column" se organizan dentro de las "columns families". Las "column" se ordenan por un tipo, que pueden ser uno de los siguientes: AsciiType, BytesType, LongType, TimeUUIDType y UTF8Type.

Row Key	Column Key i	Column Key j	Column Key k
	Value i	Value j	Value k

- **SuperColumn**. Es una columna cuyos "values" (valores) son una o más columnas, las cuales suelen llamarse "subcolumnas". Por su parte, las subcolumnas están ordenadas, y el número de columnas que se puede definir es ilimitado. Las supercolumns, a diferencia de las columnas, no tienen un "timestamp" definido.

Row Key i	Super Column Key i		Super Column Key j	
	Subcolumn 1	Subcolumn 2	Subcolumn 3	Subcolumn 4
	Column Value 1	Column Value 2	Column Value 3	Column Value 4

– **Column Family**. Es similar a una tabla en un RDMS. Se trata de un recipiente para una colección ordenada de "columns". Cada "column family" se almacena en un archivo separado.

Row Key i	Column key 1 Column key 2	Column key 3 Column key 4	Column key 5 Column key 6
	Column Value 1	Column Value 2	Column Value 3

– **Keyspace**. Es el contenedor para las "column family". Es similar a una base de datos en un RDMS. Un *keyspace* es una colección ordenada de "columns family".

– **Clúster**. Conjunto de recursos (nodos) que dan soporte a Cassandra y son vistas por los clientes como una única máquina.

El siguiente modelo presenta una analogía entre el modelo relacional y un modelo de datos tipo cassandra.

Modelo Relacional	Modelo Cassandra
Base de datos	Keyspace
Tabla	Column Family (CF)
Primary Key	Row Key
Column name	Column name/key
Column value	Column Value

5.12 Cassandra: una opción de BD NoSQL

En primer lugar se requiere contar con el *software*, el cual puede descargarse desde http://cassandra.apache.org/download/ (la última versión, a septiembre de 2015, es 2.2.1).

En la siguiente dirección web puede encontrar una guía actualizada de Casandra: http://cassandra.apache.org/doc/cql3/CQL.html

5.13 Resumen

El desarrollo de bases de datos orientadas a objetos se ha convertido en una herramienta importante para modelar las complejidades actuales de los requerimientos de gestión de la información; como también facilita la articulación con los entornos de programación orientada a objetos, aportando grandes ventajas, como son los aspectos de reutilización, generación de código, productividad desde el punto de vista el desarrollo de sistemas. No obstante, la evolución del modelo y diseño de base de datos está también marcada por el desarrollo de las infraestructuras *hardware*, como son los aspectos de virtualización, tecnologías móviles y distribuidas, que, por supuesto, demandan la gestión de datos, y a las cuales se busca responder con tecnologías de gestión de base de datos.

Los modelos de base de datos NoSQL emergen como un nuevo paradigma de modelado de información y soportan los diferentes entornos de desarrollo emergentes, tales como Ruby on Rails, Dganjo de Python.

Ejercicios

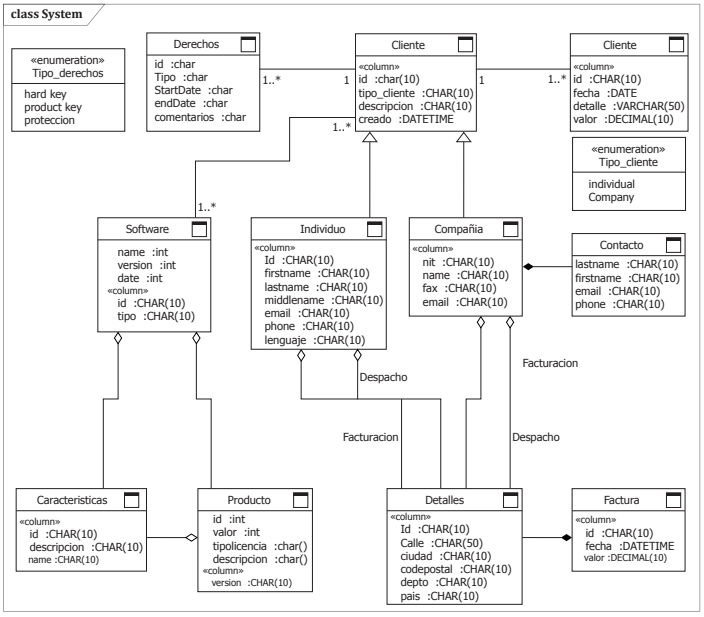
A continuación se describe la siguiente especificación de un problema de un sistema de información, la cual es importante leerla detenidamente para resolver los ejercicios de este capítulo.

"EMPRESA DE SERVICIOS GENERALES ABC"

Una empresa de servicios de servicios generales de la ciudad necesita diseñar un sistema su información para gestionar las operaciones y servicios asociadas con sus clientes. La empresa cuenta con N clientes, que son: personas y empresas en general. Los servicio que presta son: aseo y limpieza, seguridad y servicios domésticos. Para atender a cada cliente la compañía asigna a cada servicio: personal, insumos (productos de limpieza) y equipos. Los servicios son contratados por los clientes, por lo tanto es importante almacenar los detalles del contrato: fecha, detalle de los servicios contratados, valor. Cada cliente puede contratar uno o más servicios, y cada servicio está asociado a un cliente. Se desea igualmente monitorear la asignación de empleados a cada cliente y controlar el horario de entrada y salida de los mismos. La empresa genera una factura mensual por los servicios a sus clientes. La herramienta es manejada del lado de la empresa por un supervisor de servicio que desea generar informes de los servicios, monitorear los empleados, el inventario de los insumos y los equipos asignados; adicionalmente se necesita gestionar las novedades del personal que presta los servicios. Cada cliente debe contar con una funcionalidad que le permita reportar novedades del servicio y hacerles seguimiento a las novedades reportadas; la empresa de servicios debe responder las novedades en tiempo real a sus clientes. El sistema de información debe ser interoperable, usable, fácil de mantener, escalable en cuanto al manejo de datos y debe correr sobre un ambiente web, bajo computación en la nube.

Desarrollar:

1. El modelo de clases es un diagrama que le permite representar los datos de un problema, ¿cuáles son los elementos básicos de un diagrama de clases y qué significado semántico tienen?
2. De acuerdo con la especificación del problema, EMPRESA DE SERVICIOS GENERALES ABC, elaboré una tabla comparativa en la que establezca las diferencias de modelar la base de datos para el paradigma relacional y el paradigma orientado a objetos.
3. Elabore el modelo de datos conceptual de la Empresa de Servicios Generales; solamente modele las clases, las asociaciones y las restricciones relacionadas con la cardinalidad.
4. Elabore el modelo de datos detallado de la Empresa de Servicios Generales; solamente modele las clases, los atributos, las asociaciones y las restricciones relacionadas con la cardinalidad.
5. De acuerdo con el siguiente modelo de clases, extraiga la semántica o las reglas del modelo.



6. Siguiendo el paradigma de modelado de datos NoSQL, elabore el modelo de datos conceptual de la Empresa de Servicios Generales ABC bajo el concepto Key, Value.
7. Utilizando la descripción del ejercicio citado, Empresa de Servicios Generales ABC, elabore el diseño conceptual detallado para el sistema de base datos utilizando la técnica de Modelo de Entidad-Relación, describa las entidades, relaciones y sus atributos principales. Y utilice la simbología de Peter Chen.
8. Elabore una tabla comparativa donde encuentre las diferencias conceptuales entre el modelado orientado a objetos, el modelado NoSQL y el modelado relacional.
9. De acuerdo con la descripción del ejercicio citado, Empresa de Servicios Generales ABC, ¿qué recomendaciones prácticas propone para seleccionar un Sistema de base de datos NoSQL para este problema?
10. Qué analogías se pueden presentar entre el modelo relacional elaborado en Mysql y un modelo de datos NoSQL elaborado en un sistema NoSQL tipo CassandraTM.

Bibliografía

[1] M. Castells, *La sociedad red: una visión global*. Alianza Editorial, 2006.

[2] M. Sahami, M. Guzdial, A. McGettrick, and S. Roach, "Setting the stage for computing curricula 2013: computer science--report from the ACM/IEEE-CS joint task force", en *Proceedings of the 42nd ACM technical symposium on Computer science education,* 2011, pp. 161-162.

[3] "Top 10 Enterprise Database Systems to Consider". [En línea]. Disponible en http://www.serverwatch.com/article.php/3883441. [Accedido: 6 de mayo de 2013].

[4] "Top 10 Largest Databases in the World | Reviews, Comparisons and Buyer's Guides". [En línea]. Disponible en http://www.comparebusinessproducts.com/fyi/10-largest-databases-in-the-world. [Accedido: 6 de mayo de 2013].

[5] R. Elmasri y S. B. Navathe, *Sistemas de Bases de Datos: conceptos fundamentales*. 2000.

[6] T. M. Connolly y C. E. Begg, *Sistemas de Bases de Datos: Un Enfoque Practico Para Diseno, Implementacion y Gestion/Database Systems*. Pearson Education, 2006.

[7] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database system concepts*, vol. 3. McGraw-Hill, 1997.

[8] J. Martin y A. Di Marco, *Organización de las Bases de Datos*. Prentice-Hall, 1982.

[9] D. D. Chamberlin, M. M. Astrahan, K. P. Eswaran, P. P. Griffiths, R. A. Lorie, J. W. Mehl, P. Reisner, and B. W. Wade, "SEQUEL 2: A unified approach to data definition, manipulation, and control", *IBM Journal of Research and Development*, vol. 20, nº 6, pp. 560-575, 1976.

[10] J. Green y R. Johnson, "ProSQL: A prototyping tool for SQL temporal language extensions", *New Horizons in Information Management*, pp. 164-164, 2003.

[11] A. Oppel, *Fundamentos de bases de datos*. México: McGraw-Hill Professional Publishing, 2010.

[12] G. O'Regan, "Larry Ellison", en *Giants of Computing*. Springer, 2013, pp. 103-105.

[13] "Historia Sybase - Documentos - Juantaroo". [En línea]. Disponible en http://www.buenastareas.com/ensayos/Historia-Sybase/32037.html. [Accedido: 7 de marzo de 2013].

[14] "Historia De Microsoft SQL Server – Base de Datos Relacional – RDBMS | YAMIL Lªmß3ят - ESPOL - Yamil Lambert". [En línea]. Disponible en: http://blog.espol.edu.ec/ylambert/2011/05/31/historia-de-microsoft-sql-server-%E2%80%93-base-de-datos-relacional-rdbms/. [Accedido: 7 de marzo de 2013].

[15] J. Minker, *Foundations of deductive databases and logic programming*. Morgan Kaufmann, 2014.

[16] A. Dennis, B. H. Wixom, and D. Tegarden, *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons, 2015.

[17] J. Melton and A. R. Simon, *SQL: 1999: understanding relational language components*. Morgan Kaufmann Pub., 2002.

[18] "ScienceDirect.com - Computers and Standards - A technical overview of the information resource dictionary system". [En línea]. Disponible en http://www.sciencedirect.com/science/article/pii/016780518590035X. [Accedido: 7 de marzo de 2013].

[19] E. C. Foster and S. V. Godbole, "Introduction to Database Systems", en *Database Systems*. Springer, 2014, pp. 3-11.

[20] "Top 15 Most Popular Social Networking Sites". [En línea]. Disponible en http://www.ebizmba.com/articles/social-networking-websites. [Accedido: 6 de mayo de 2013].

[21] J. Ullman, "rinciples of DATABASE", *Knowledge-Base Systems*, vol. 1, pp. 1988-89, 1982.

[22] M. Hammer y D. Mc Leod, "Database description with SDM: a semantic database model", *ACM Transactions on Database Systems (TODS),* vol. 6, n.º 3, pp. 351-386, 1981.

[23] P. P. S. Chen, "The entity-relationship model-toward a unified view of data", *ACM Transactions on Database Systems (TODS),* vol. 1, nº 1, pp. 9-36, 1976.

[24] C. Batini, S. Ceri, and S. B. Navathe, *Conceptual database design*. Benjamin/Cummings, 1992.

[25] R. Elmasri and S. B. Navathe, *Fundamentals of database systems*. Pearson, 2014.

[26] D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, "Translation of extended entity-relationship database model into object-oriented database model", en *Interoperable Database Systems (DS-5): Proceedings of the IFIP WG2. 6 Database Semantics Conference on Interoperable Database Systems (DS-5),* Lorne, Victoria, Australia, 16-20, November, 1992, 2014, p. 87.

[27] D. Whitfield, S. Joshi, F. Hulick, and P. Mullins, "The application of a layered database model", *Journal of Computing Sciences in Colleges*, vol. 22, nº 4, pp. 228-234, 2007.

[28] M. de Tebar Peralta and R. Musach, "Estudi i implementació d'una solució Business Intelligence per a una empresa", 2015.

[29] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language–User's Guide*. Addison-Wesley Reading, MA, 1999.

[30] G. Wagner, *Foundations of Knowledge Systems: With Applications to Databases and Agents,* vol. 13. Springer Science & Business Media, 2012.

[31] A. O. Allen, *Probability, statistics, and queueing theory: with computer science applications*. Academic Press, 1990.

[32] R. N. Cardinal and M. R. Aitken, *ANOVA for the behavioural sciences researcher*. Lawrence Erlbaum Associates Publishers, 2006.

[33] N. Duffy, "Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design", *New Library World,* vol. 100, nº 3, pp. 147-147, 1999.

[34] R. E. Maria, L. A. R. Junior, L. E. G. de Vasconcelos, A. F. M. Pinto, P. T. Tsoucamoto, H. N. A. Silva, A. Lastori, A. M. da Cunha, and L. A. V. Dias, "Applying Scrum in an Interdisciplinary Project Using Big Data, Internet of Things, and Credit Cards", en *Information Technology-New Generations (ITNG), 2015 12th International Conference on*, 2015, pp. 67-72.

[35] K. E. Wiegers, *Software requirements*. Microsoft press, 2009.

[36] C. Date, *Database Design and Relational Theory: Normal Forms and All That Jazz*. O'Reilly Media, Inc., 2012.

[37] C. Coronel, S. Morris, and P. Rob, *Database systems: design, implementation, and management*. Cengage Learning, 2009.

[38] A. Dogac y P. P. Chen, "Entity-Relationship model in the ANSI/SPARC framework", en *Proceedings of the Second International Conference on the Entity-Relationship Approach to Information Modeling and Analysis*, 1981, pp. 357-374.

[39] A. Alberca M., *Bases de datos Orientadas a Objetos*. Universidad de Castilla-La Mancha, 2005.

[40] C. Richardson, 2006. POJOs in Action. Greenwich, CT: Manning.Tate, B., and J. Gehtland. 2005. Spring: A Developer's Notebook. Sebastopol, CA: O'Reilly.

[41] C. Bauer and G. King, *Java Persistence with Hibernate*. Greenwich, CT: Manning, 2006.

[42] S. Ambler, *Mapping Objects to Relational Databases: O/R Mapping in Detail*. 2006. Disponible en http://www.agiledata.org/essays/mappingObjects.html

[43]] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language*. Libro Pearson 2006.

[44] Recursos en linea www.sparxsystems.com guia práctica UML parte 2.

[45] The Object Management Group (OMG®) is an international, open membership, not-for-profit technology standards

[46] C Imhoff, N. Galemmo, and J. G. Geiger, *Masterinf Data Warehouse Design* . Wiley, 2003.

Esta obra, editada en Barranquilla por
Editorial Universidad del Norte en junio de 2017,
se compuso en Garamond Premier Pro, Electra LT Std y Courier New.

Este libro recoge la experiencia de los autores en el desarrollo de sus actividades de docencia, investigación y consultoría, en temas asociados con las arquitecturas de información en las organizaciones. Sin lugar a dudas, será de gran ayuda y una referencia conceptual, metodológica y práctica para iniciados y profesionales en el modelado, diseño e implementación de base de datos, y para estudiantes de Ingeniería de Sistemas y afines.

UNIVERSIDAD DEL NORTE