Unidad 3. Diseño

Clave:
Telemática
21141209 / 22141209
Desarrollo de Software
15141211 / 16141211

Universidad Abierta y a Distancia de México

Ciencias Exactas, Ingeniería y Tecnología

Índice

Presentación de la unidad

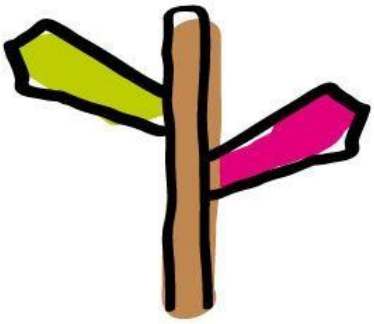
¡Felicidades!, estás a punto de terminar el curso, no ha sido fácil, pero las personas de éxito son aquellas que, con base en la perseverancia y constancia, llegan a cumplir sus metas y objetivos, así como tú has logrado llegar a esta unidad. Recuerda que todo ser humano tiene la capacidad de alcanzar todo lo que se proponga, contamos contigo.

En esta unidad conocerás temas que te ayudarán a obtener una mayor comprensión en materia de las bases de datos: el concepto de *Structured Query Language* (SQL), los diferentes tipos de datos que utilizan las base de datos, lo que es un diccionario de datos y cómo se genera; así mismo, se revisarán las instrucciones SQL: *Data Definition Language* (DDL) y *Data Manipulation Language* (DML), para finalizar con el tema de álgebra relacional. Se espera que con estos tópicos logres tener un panorama de los temas que conforman la tercera, y última, unidad de la asignatura de Base de datos.

¡Aviso importante!

Para el uso y actualización de actividades, es necesario considerar versiones de Sistemas operativos y gestores, en concordancia con las actualizaciones de software y hardware, estables que permitan llevar a cabo las actividades aquí propuestas.

Propósitos de la unidad



En esta unidad documentarás las técnicas para poder construir el prototipo de la base de datos, teniendo como base el concepto general de Structured Query Language (SQL), los tipos de datos, el diccionario de datos, las instrucciones SQL: Data Definition Language (DDL) y Data la Manipulation Language (DML), y el álgebra relacional.

Competencia específica

Elaborar modelados de Bases de datos documentales para representar la información que satisfaga las necesidades de gestión de información, mediante las técnicas de modelaje utilizadas para generar un prototipo.

3.1. Diseño de prototipo de Bases de datos

Antes de iniciar con el desarrollo de los subtemas de esta unidad, es esencial explicar a qué nos referimos con diseño de prototipo de base de Datos. Como se explicó anteriormente, se tiene contemplado que mediante los subtemas que se verán en esta unidad, y lo realizado en las unidades anteriores, realices el prototipo del diseño de la base de datos, entendiendo como prototipo a la primera versión de un nuevo tipo de producto, en el que se han incorporado solo algunas características del sistema final, o no se han realizado completamente, lo que permitirá al diseñador evaluar los requerimientos del usuario, verificar su estructura y flujo de información. Los prototipos son una visión previa o preliminar del sistema que se implementará a futuro.

Los prototipos de un sistema de información se utilizan para la recopilación rápida de los requerimientos de los diferentes usuarios para satisfacer sus necesidades de información, este prototipo es importante ya que está dentro del ciclo de vida del desarrollo de sistemas de bases de datos, que es una fase determinante para el perfeccionamiento e implementación del sistema.

El propósito de los prototipos es permitir a los diseñadores realizar adecuaciones, con la revisión de los usuarios, de tal forma que cumplan con las reglas del sector productivo al que va dirigido, y con los requerimientos del usuario.

Hay tres tipos de información que se busca al realizar un prototipo: las i**nnovaciones**, las **sugerencias del usuario** y el **plan de revisión**. Las **innovaciones** son parte de la información que se busca al realizar los prototipos, son capacidades nuevas, comúnmente conocidas como el plus, o capacidades nuevas del sistema no contempladas antes de la interacción del prototipo. Las **sugerencias del usuario** son la observación y las entrevistas que se le realizan al usuario, la reacción que tiene cuando se le presenta el prototipo y cuando interactúa con él. El usuario retroalimenta y sugiere (presentación de información, resultados correctos, faltantes de información, etc.) nuevas ideas para el mejoramiento del sistema. Es importante mencionar que esto permitirá al analista afinar y realizar cambios al prototipo. Finalmente, el **plan de revisión** permite al diseñador identificar las prioridades de construcción y redirigir los planes sin realizar gastos.

Existen cuatro tipos de prototipos:
Prototipo parchado. Es un sistema de información operable con características de información necesarias para la visualización del flujo de información, pero es ineficiente.
Prototipo no operacional. Es un modelado a escala no funcional de un sistema de información, se utiliza cuando la codificación requerida por las aplicaciones es muy extensa para la realización de un prototipo, pero aun así se pude obtener una idea clara del sistema por medio de la elaboración de prototipos solamente de entradas y salidas de información.
Prototipo de una serie. Es utilizado cuando se tienen planeadas muchas instalaciones del mismo sistema de información.
Prototipo de características seleccionadas. Es la construcción del prototipo operacional que incluye solo algunas características que tendrá el sistema final; este prototipo permite que el sistema se vaya

construyendo de forma modular, de tal manera que al ser evaluados de forma independiente y sean satisfactorios, se incorporen en el sistema final. Es importante la utilización de prototipos porque permiten comunicar, discutir y definir ideas entre los diferentes diseñadores y los usuarios responsables de cada información, apoyan el trabajo evaluando productos, clasificando los requerimientos de los usuarios y definiendo alternativas.

Es importante destacar que el prototipo que se pretende realizar en esta materia no incluirá la parte de la programación de interface, ya que la materia solo se enfoca en la parte documental y ciertas instrucciones SQL (que veremos a continuación) para la creación de la base de datos; se recomienda utilizar un software libre para la generación de la base.

Actividad 1: Tablas lógicas

1. **Consulta** el documento de actividades de la unidad.

***Revisa** los criterios de evaluación.

3.1.1. Concepto de *Structured Query Language* (SQL)

En este apartado veremos qué es el SQL, el cual se puede traducir como Lenguaje Estructurado de Consultas (*Structured Query Language)*. SQL es un lenguaje de consultas estructurado y de programación de bases de datos relacionales utilizado para accesar, consultar, actualizar y gestionar información, que permite realizar diversos tipos de operaciones; es utilizado casi siempre con el álgebra relacional, que permite explotar de manera más eficiente la información y tiene la capacidad de aplicar cálculos matemáticos. SQL es considerado un lenguaje de alta generación (4GL), y estándar, pero en algunos Sistemas Gestores de Bases de Datos tendrán pequeñas variaciones en su estructura.

Hay que destacar que los sistemas de bases de datos comerciales necesitan un lenguaje de consulta cómodo para el usuario, y el lenguaje de mayor influencia es SQL, que usa una combinación de álgebra relacional y construcciones de cálculo relacional.

Aunque el lenguaje SQL se considere un lenguaje de consultas de Bases de datos, contiene muchas otras capacidades: incluye características para definir la estructura de los datos, para la modificación de ellos en la base de datos y para la especificación de restricciones de seguridad. Actualmente numerosos productos son compatibles con el lenguaje SQL.

Dado que SQL es el lenguaje abierto y más comercial en el mercado, se debe remarcar la importancia de cada uno de sus componentes de manera general, los cuales se abordan a continuación.

Componentes de SQL
El lenguaje SQL está conformado básicamente por los siguientes componentes:

- Lenguaje de definición de datos (LDD). Proporciona órdenes para la definición de vistas y de esquemas de relación, borrado de relaciones, creación de índices, modificación de índices y de esquemas de relación. También incluye órdenes para la especificación de las restricciones de integridad que deben satisfacer los datos almacenados en la Base de datos, y las actualizaciones que violen las restricciones de integridad las rechaza. Además, incluye órdenes para especificar derechos de acceso para las relaciones y vistas.
- Lenguaje de manipulación de datos (LMD). Incluye un lenguaje de consultas, basado tanto en álgebra relacional como en cálculo relacional de tuplas (tablas), y órdenes para insertar, borrar y modificar tuplas de las bases de datos.
- Control de transacciones. Órdenes para la especificación del comienzo y final de transacciones.

SQL tiene la facultad de poder trabajar con diferentes lenguajes de programación de propósito general, dado que puede ser incorporado o dinámico. SQL incorporado trabaja con código fuente estructurado por el programador, mientras que el dinámico no requiere de estructuras de código fuente, dado que tiene la facultad de trabajar con asistentes que no lo requieren. Algunos ejemplos de lenguajes de programación de propósito general son C, C++, Java. PL/I, Cobol, entre otros.

Al momento de programar SQL se requiere tener claridad sobre el significado de las estructuras que utiliza para las expresiones básicas, las cuales son: **SELECT, FROM** y **WHERE.** Estas estructuras se comparan con la aplicación que tienen en el álgebra relacional, tema que se aborda más adelante.

- **SELECT.** Se usa para listar los atributos deseados del resultado de una consulta. Corresponde a la operación proyección de álgebra relacional.
- **FROM.** Lista las relaciones que deben ser analizadas en la evaluación de la expresión. Corresponde a la operación producto cartesiano de álgebra relacional.
- **WHERE.** Es un predicado que engloba los atributos de las relaciones que aparecen en la expresión **FROM.**
  Corresponde al predicado selección de álgebra relacional.

Es importante mencionar que el término **SELECT** tiene un significado diferente en SQL y en álgebra relacional, que se verá a profundidad más adelante.

Para qué sirve SQL
SQL permite la realización de consultas y actualizaciones sobre datos almacenados en tablas relacionales, como ya se ha mencionado antes. Este es el principal uso que harán los programadores y usuarios del lenguaje, pero también hay otras tareas que se pueden realizar mediante las sentencias SQL, aunque pertenecen más a la responsabilidad de lo que suelen llamarse administrador de bases de datos; las tareas son las siguientes:

1. Definición y destrucción de objetos.

Antes de poder usar una tabla para almacenar o consultar datos en ella, hay que describirla al SGBD, dándole su nombre y características, esto se hace definiéndola. Hay otros muchos objetos que maneja un SGBD con sentencias SQL, tales como espacios para ficheros físicos, índices, claves principales, campos, entre otros. Normalmente, la definición de tablas y vistas son responsabilidad del administrador de bases de datos, pero puede haber casos en que se permita a algunos usuarios definir tablas para datos de uso privado. En este caso, el usuario emplea las correspondientes sentencias de SQL.

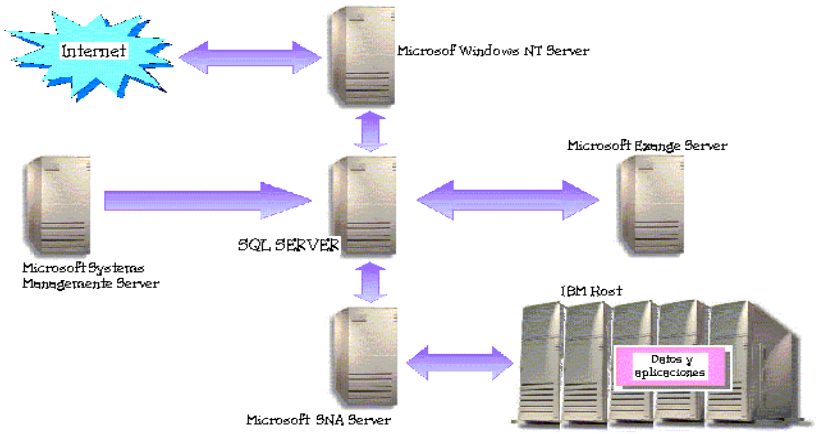
2. Gestión de las autorizaciones de acceso.

El usuario final no podrá consultar o actualizar datos de una tabla si previamente no ha sido autorizado para ello. Tampoco podrá hacerlo un programa si la persona que ha solicitado su ejecución no ha sido previamente autorizada. Estas autorizaciones se conceden o deniegan mediante sentencias SQL. Normalmente, las autorizaciones las concede el administrador de bases de datos, pero puede haber casos en que lo haga un usuario con respecto a sus datos privados, si se le permite tenerlos.

Existen otras tareas propias del administrador de bases de datos, en las que el SQL puede ayudar, sin embargo se necesitan otras herramientas adicionales. Así, por ejemplo, la prevención y corrección de problemas de rendimiento y la toma de copias de seguridad de los datos para el caso de fallos por catástrofes o averías.

Para concluir con este tema, tenemos que SQL permite:

- Definir y destruir objetos de las bases de datos.
- Conceder y denegar autorizaciones para usar estos objetos.
- Consultar y actualizar datos.



3.1.2. Tipos de datos

Un tipo de dato es un conjunto de valores y operaciones primitivas aplicables a las tablas que conforman las bases de datos. En los lenguajes de programación se proporcionan tipos de datos ya definidos agrupados en clases distintas. Los tipos de datos también son vistos como una limitación impuesta en la interpretación de datos en un sistema de tipificación (lenguaje de programación que clasifica valores y expresiones en tipos), describiendo la representación, interpretación y estructura de los valores u objetos en la memoria del ordenador. La declaración de un tipo de dato requiere la especificación de un nombre y la definición (descripción) del mismo.

Ejemplo: podemos decir en la entidad VENDEDOR que su campo identificador tiene el ID_VENDEDOR, y este campo tiene un tipo de dato que será definido por las reglas del usuario, pudiendo ser de dos tipos: numérico y entero o alfanumérico, este tipo de dato deberá indicar de forma específica la cantidad de caracteres requerida.
Es importante definir los tipos de datos para diseñar las tablas o entidades, dado que de aquí se definirá el dominio para el uso de las relaciones en la base de datos.

Los tipos de datos con restricciones y valores por defecto se pueden combinar en la definición de dominios. Una definición de dominio es un tipo de dato especializado que puede estar definido dentro de un esquema y utilizado en la definición de columnas. Por ejemplo, se desea definir un dominio de identificadores para usar en la definición de columnas como ID_VENDEDOR e ID_EDIFICIO. Estas definiciones son complejas, ya que se involucra un tipo de dato, un valor por defecto y una restricción no nula. Puesto que se usará esta una y otra vez en el esquema de la base de datos, se requiere simplificar el trabajo. Por tanto se crea un dominio de la siguiente forma:

```
CREATE DOMAIN IDENTIFICACOR NUMERIC (4) DEFAULT 0
CHECK (VALUE IS NOT NULL)
```

Se dice que un dominio llamado IDENTIFICADOR tiene las siguientes propiedades:

- Su tipo de dato es numérico de cuatro dígitos.
- Su valor por defecto es cero y nunca puede ser nulo.
- Una columna que define este dominio como su tipo de dato tendrá otras propiedades.
- SQL requiere lo que se denomina una restricción CHECK, para esto se pueden definir columnas en el esquema que tenga identificador como su tipo de dato.

En el modelo relacional, un dominio es un conjunto del cual toma sus valores una columna de una relación. En este sentido, los tipos de datos predefinidos son dominios. Sin embargo, SQL define los dominios de forma ligeramente diferente, proporciona tipos de datos predefinidos y permite dominios definidos por el usuario.

Los tipos de datos en SQL se clasifican en:

- Numéricos exactos

- Aproximados
- Cadenas de caracteres
- Cadenas de bits
- Fechas y horas
- Intervalos

Numéricos exactos
- Integer (enteros)
- SmALL integer (enteros cortos)
- Numeric (p,e) (numéricos)
- Decimal (p,e) (decimales)

Tipos de datos numéricos exactos.

En los últimos dos tipos de datos (numérico y decimal) se indica una precisión (p) y una escala (e). La precisión indica el total de números o dígitos en el número y la escala indica cuántos de estos están a la derecha del punto decimal.

Aproximados
- Real (real)
- Double precision (doble precisión)
- Float (flotante)

Tipos de datos aproximados.

Estos tipos de datos se usan normalmente para cálculos científicos y de ingeniería.

Cadenas de caracteres
- *Character* (n) (carácter)
- *Character varying* (n) (carácter variable)

Tipos de datos cadenas de caracteres.

Los campos de *Character* siempre almacenan “n” caracteres, aun cuando tengan que rellenar con blancos a la derecha para completar la longitud “n”. Los campos *Character varying* solo almacenan el número real de caracteres que se introdujeron (hasta un máximo de n). La diferencia entre ellos es que el primero está definido, y en el caso del segundo la palabra *varying* lo define, es variante constantemente.

Cadena de Bits
- Bit (n)
- Bit varying (n)

Tipos de datos cadenas de bits.

Estos campos se usan para banderas u otras máscaras de bits para el control.

Fechas y horas

- Date (fecha)
- Time (hora)
- Timestamp (Sello de tiempo)
- Time con tiempo zona
- Timestamp con tiempo zona

Tipos de datos fechas y horas

Date (fecha) se da en el orden: año, mes y día, con cuatro dígitos para el año.
Time se da en horas (0 a 23), minutos, segundos y décimas de segundo.
Timestamp es la fecha más la hora (*date plus time*).

Intervalos

- Year – month (año-mes)
- Date_time (dia-hora)

Tipos de datos de intervalo

El intervalo es la diferencia entre dos fechas (año-mes) o entre dos horas (día-hora). Por ejemplo, entre diciembre de 1994 y enero de 1996, el intervalo es un año y un mes.

3.1.3. Generación de Diccionario de datos

El propósito de un diccionario de datos es que contenga las características lógicas de los datos que se van a utilizar en un sistema (bases de datos), incluyendo nombre, descripción, alias, contenido y organización.

Los diccionarios de datos son desarrollados durante el análisis de flujo de datos, y ayudan a los analistas que participan en la determinación de los requerimientos del sistema, evitando así malas interpretaciones o ambigüedades, su contenido también se emplea durante el diseño del proyecto.

En un diccionario de datos se encuentra la lista de los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos. Desde el punto de vista estadístico, este diccionario debe tener la variable, el tipo de variable, su definición y su delimitación espacial.

Ejemplo:

Nombre Campo	Descripción	Tipo	Longitud	CAMPO LLAVE PK	CAMPO LLAVE FK	Tabla en PK	Tablas en FK	OTRAS	CAMPO OBLIGATORIO
CveAutor	Campo identificador del autor	CHAR	4	SI	SI	AUTOR	LIBRO		SI
NomAutor	campo que contiene el nombre del Autor	VARCHAR	60	NO	NO			AUTOR	SI
CveEditorial	Campo identificador de la editorial	INT		SI	SI	EDITORIAL	LIBRO		SI
DescEdi	Campo que contiene la descripción de editorial	VARCHAR	40	SI	SI	EDITORIAL	LIBRO		SI

Ejemplo de diccionario de datos.

Las ventajas de diseñar un diccionario de datos son:

- Permite la localización de información de manera más rápida.
- Aclara dudas con respecto a las características de los campos y, en su caso, realizar altas, bajas y cambios.
- Permite visualizar las tablas relacionadas entre sí y los campos que las relacionan.

Actividad 2: Diccionario de datos

Consulta las actividades de la unidad.

*__Revisa__ los criterios de evaluación de la actividad.

Actividad 3: Entrega de tablas lógicas y diccionario de datos

Revisa el documento de actividedes.

***Consulta** los criterios de evaluación de la actividad.

3.1.4. Instrucciones SQL: *Data Definition Language* (DDL) y *Data Manipulation Language* (DML)

Las peticiones sobre los datos se expresan mediante sentencias, que deben escribirse de acuerdo con las siglas sintácticas y semánticas de SQL. Para esto es necesario conocer las instrucciones primordiales de SQL: DDL (*Data Definition Language)* y DML (*Data Manipulation Language*).

DDL
La instrucción DDL (*Data Definition Language*) está conformada por los siguientes comandos.

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos, vistas e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas, agregando campos o cambiando la definición de los campos.

Instrucciones DDL

Así como se muestra en la tabla la descripción de cada comando, a continuación se describe la sintaxis de cada uno de ellos.

Sintaxis de CREATE
CREATE TABLE “nombre_tabla”
(“columna 1” “tipo_de_datos_para_columna_1”, “columna 2” “tipo_de_datos_para_columna_2”,….)

Sintaxis de DROP
DROP DATABASE , dataBase_name | dataBase_snapshot_name - * ,…n + *;+

Sintaxis de ALTER
ALTER TABLE “nombre_tabla” *modificar especificación+

DML
DML (*Data Manipulation Language*) está conformado por comandos que permiten al usuario seleccionar, añadir, modificar y borrar información contenida en el repositorio (Base de datos), es decir,

cuenta con los comandos: SELECT, INSERT, UPDATE Y DELETE.

Comando	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
INSERT	Utilizado para cargar lotes de datos en la Base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una Base de datos

Instrucciones DML.

Así como se muestra en la tabla la descripción de cada comando, a continuación se describe la sintaxis de cada uno de ellos.

Sintaxis de SELECT
SELECT "nombre de columna" FROM "nombre_tabla"

Sintaxis de INSERT
INSERT INTO "nombre_tabla" ("columna 1", "columna 2", …) valor S ("valor 1", "valor 2", …)

Sintaxis de UPDATE
UPDATE "nombre_tabla" SET "columna 1" = *nuevo valor+ WHERE ,condición-

Sintaxis de DELETE
DELETE FROM "nombre_tabla" WHERE {condición}

Existen otros comandos básicos, que se vuelven condiciones de modificación utilizadas para definir los datos que se desea seleccionar o manipular.

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

Comandos básicos.

Así como se muestra la descripción de cada comando (de condición), en la tabla, a continuación se describe la sintaxis de cada uno de ellos.

Sintaxis de FROM
FROM "nombre_tabla" WHERE {condición}

Sintaxis de WHERE
SELECT "nombre_columna" FROM "nombre_tabla" WHERE "condición"

Sintaxis de GROUP BY
SELECT "nombre_columna 1", SUM ("nombre_columna 2") FROM "nombre_tabla" GROUP BY "nombre_columna 1"

Sintaxis de HAVING
SELECT "nombre_columna 1", SUM ("nombre_columna 2") FROM "nombre_tabla" GROUP BY "nombre_columna 1" HAVING (condición de función aritmética)

Sintaxis de ORDER BY
SELECT "nombre_columna" FROM "nombre_tabla" [WHERE "condición"] ORDER BY "nombre_columna" [ASC, DESC]

Aparte de los comandos antes mencionados, SQL maneja operadores lógicos y de comparación, los cuales también tienen sus propias sintaxis. En el caso de los operadores lógicos, en la siguiente figura se muestra su definición en español junto con su uso.

Operador	Uso
AND es "y"	Evalúa dos condiciones y devuelve un valor de verdad solo si ambas son ciertas.
OR es "o"	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT es negación	Devuelve el valor contrario de la expresión.

Operadores lógicos.

A continuación se muestran las sintaxis de cada uno de los operadores lógicos.

Sintaxis de AND
SELECT "nombre_columna" FROM "nombre_tabla" WHERE "condición simple" { [AND] "condición simple"}+

Sintaxis de OR
SELECT "nombre_columna" FROM "nombre_tabla" WHERE "condición simple" { [OR] "condición simple"}+

Sintaxis de NOT
[NOT] boolean_expresión

En el caso de los operadores de comparación expuestos en la figura, se muestra la simbología así como su uso.

Operador (simbología)	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo.
IN	Utilizado para especificar registros de una base de datos.

Operadores de comparación.

A continuación se muestran las sintaxis de cada uno de los operadores de comparación.

Sintaxis de BETWEEN
campo [Not] Between valor1 AND valor2

Sintaxis de LIKE
expresión Like modelo

Sintaxis de IN
expresión [Not] In (valor1, valor2, . . .)

En esta parte es importante mencionar las funciones de los operadores agregados que se usan dentro de un comando SELECT en grupos de registros para devolver un único valor ya registrado.

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de la selección.
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado.
MAX	Utilizada para devolver el valor más alto de un campo especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado.

Funciones

Cada una de estas funciones se utiliza cuando el manejador de Base de datos es SQL y tienen las siguientes sintaxis:

Sintaxis de AVG
AVG ([ALL | DISTINCT] expresión)

Sintaxis de COUNT
SELECT COUNT ("nombre_columna") FROM "nombre_tabla"

Sintaxis de SUM
SUM ([ALL | DISTINCT] expresión)

Sintaxis de MAX
MAX ([ALL | DISTINCT] expresión)

Sintaxis de MIN
MIN ([ALL | DISTINCT] expresión)

3.1.5. Álgebra relacional

Para hablar de este tema, es importante conocer y tener bien definidos los términos básicos de las estructuras SQL mencionados y presentados en las figuras del subtema anterior, así como su funcionamiento. Ahora bien, este tema servirá para generar las operaciones más básicas en las bases de datos relacionales, y está constituido de manera práctica para su mejor comprensión.

El álgebra relacional es definida como un conjunto de operadores de alto nivel que trabajan sobre relaciones. Esto significa que estas operaciones usan una o dos relaciones existentes para crear una nueva relación. Esta nueva relación puede entonces usarse como entrada para una nueva operación. Este poderoso concepto – la creación de una nueva relación a partir de las relaciones existentes— hace posible una variedad infinita de manipulaciones sobre los datos. Eso también hace considerablemente más fácil la solución de las consultas, debido a que se puede experimentar con soluciones parciales hasta encontrar una proposición con la que se trabajará.

CLIENTE					
ID_CLIENTE	NOMB_CLIENTE	DIRECCION	PAÍS	SALDO INICIAL	SALDO ACTUAL
100 101 105 110	José López Maritza Julián Gómez	Box 241, Sonora B 918, Aguascalientes Santiago	Japón Austria USA Chile	45.551 75.314 49.333 27.400	52.113 77.200 57.811 35.414
VENDEDOR					
ID_VENDEDO	NOMB_VENDEDO	ID_JEFE	OFICINA	%_COMISIÓN	
10 14 23 37 39 27 44 35 12	Blanca Soto Mónica Márquez Francisco Moreno Elena Hernández Gabriela Acuña Teresa Cardoso Alberto Sánchez Berenice Medina Claudia Ortega	27 44 35 12 44 27 27 27	Aguascalientes Sonora Nayarit B.C. Sonora Aguascalientes Sonora Nayarit B.C	10 11 9 13 10 15 12 11 10	
PRODUCTO					
ID_PRODUCTO	DESC_PRODUCTO	ID_FABRICANTE	COSTO	PRECIO	

1035	abrigo	210	1,25	2,00
2241	lámpara de mesa	317	2,25	3,25
2249	lámpara de mesa	317	3,55	4,80
2518	escultura	253	0,60	1,20
VENTA				
FECHA	ID_CLIENTE	ID_VENDEDOR	ID_PROD	CANTIDAD
28/02	100	10	2241	200
12/02	101	23	2518	300
15/02	101	23	1035	150
19/02	100	39	2518	200
02/02	101	23	1035	200
05/02	105	10	2241	100
22/02	110	37	2518	150
14/02	105	10	2249	50
01/02	101	23	2249	75
04/02	101	23	2241	250
FABRICANTE				
ID_FABRICANTE	NOMB_FABRICANTE	DIRECCION	PAÍS	
210	Nike	AuklAND	Estados Unidos	
253	Hugo boss	Lagos	Nigeria	
317	Llama Llamps	Lima	Perú	

Ejemplo de datos en una base de datos.

El álgebra relacional consta de nueve operaciones: unión, intersección, diferencia, producto, selección, proyección, reunión, división y asignación. Las cuatro primeras operaciones se toman de la teoría de conjuntos de Matemáticas, y son considerablemente parecidas a las operaciones encontradas allí. Esto es razonable, debido a que las relaciones son en sí mismas conjuntos, de esta manera se le pueden aplicar las operaciones de conjunto. Las cuatro siguientes son operaciones nuevas, que se aplican específicamente al modelo de datos relacional. Mientras que la última operación (asignación) es la operación estándar de los lenguajes de computación, de dar un valor a un nombre. En este caso, la asignación se usa para dar un nombre a una nueva relación que se crea de relaciones existentes. Se usará el signo ":=" (es el nombre asignado a) para indicar la asignación de nombres a relaciones. A continuación se abordan cada una de las nueve operaciones del álgebra relacional.

1. Unión

La operación de unión (∪) permite combinar los datos de dos relaciones. Por ejemplo, suponga que la relación VENDEDOR no existe en la base de datos. En cambio, se tienen las dos relaciones de la figura que le sucede. Estas dos
relaciones representan todos los vendedores que están subordinados a otros vendedores (VENDEDOR_SUBORDINADO) y todos los vendedores que son jefes de otros vendedores (VENDEDOR_JEFE). Obviamente, existe redundancia de datos, esto sucede a través de la ejecución de una serie de órdenes de algebra relacional antes de utilizar el operador unión. Si se desea obtener

una relación que contenga todos los vendedores, se debe realizar la unión de VENDEDOR_SUBORDINADO y VENDEDOR_JEFE.

VENDEDOR:= VENDEDOR_SUBORDINADO ∪ VENDEDOR_JEFE

VENDEDOR es el nombre asignado a la relación resultante y consiste en las filas que están en VENDEDOR_SUBORDINADO o VENDEDOR_JEFE, o ambas.

Es importante observar que una fila que existe en ambas relaciones aparece solo una vez en la relación unión. Esto sigue la definición de una relación como un conjunto, debido a que un elemento dado se encuentra solo una vez en conjunto.

VENDEDOR_SUBORDINADO				
ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%_COMISIÓN
10	Blanca Soto	27	Aguascalientes	10
14	Mónica Márquez	44	Sonora	11
23	Francisco	35	Nayarit	9
37	Moreno Elena	12	B.C.	13
39	Hernández	44	Sonora	10
44	Gabriela Acuña	27	}Sonora	12
35	Alberto Sánchez	27	Nayarit	11
12	Berenice Medina	27	B.C.	10
	Claudia Ortega			
VENDEDOR_JEFE				
ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%_COMISION
1035	Teresa Cardoso		Aguascalientes	15
2241	Alberto Sánchez	27	Sonora	12
2249	Berenice Medina	27	Nayarit	11
2518	Claudia Ortega	27	B.C.	10

Representación relacional alternativa de vendedor.

VENDEDOR				
ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%_COMISION
10	Blanca Soto	27	Aguascalientes	10
14	Mónica Márquez	44	Sonora	11
23	Francisco Moreno	35	Nayarit	9
37	Elena Hernández	12	B.C.	13
39	Gabriela Acuña	44	Sonora	10
27	Teresa Cardoso		Aguascalientes	15
44	Alberto Sánchez	27	Sonora	12
35	Berenice Medina	27	Nayarit	11
12	Claudia Ortega	27	B.C.	10

Unión de vendedor subordinado y vendedor jefe.

Se debe señalar un requisito especial de la operación de unión que la hace ligeramente diferente de la unión de conjuntos de Matemáticas. En Matemáticas, dos conjuntos cualesquiera pueden ser combinados a través de la operación unión. Pero en el álgebra relacional, antes de aplicar la unión a las dos relaciones, estas deben tener exactamente las mismas columnas, tanto en número de columnas como en el dominio de las mismas. Si este es el caso, se dice que las dos relaciones son unión compatible. Obviamente, VENDEDOR_SUBORDINADO y VENDEDOR_JEFE son unión compatible.

Se requiere la compatibilidad de la unión para que el resultado sea una relación. Si se toma la unión de CLIENTE y PRODUCTO se obtendrá un conjunto y una relación. Las filas en el conjunto resultante no tendrían columnas comunes, de este modo ellas no se agruparían en una tabla relacional. La compatibilidad de la unión es claramente esencial para la operación de unión. Por la misma razón, es esencial para las operaciones de intersección y diferencia.

2. Intersección

La operación de intersección (∩) permite identificar los campos que son comunes en dos relaciones. Por ejemplo, si se requiere identificar a los vendedores que son subordinados de algún jefe, y que son jefes, se toma la intersección de VENDEDOR_SUBORDINADO y VENDEDOR_JEFE, llamando al resultado VENDEDOR_SUBORDINADO_JEFE:

VENDEDOR_SUBORDINADO_JEFE := VENDEDOR_SUBORDINADO ∩VENDEDOR_JEFE

Esto produce la siguiente relación:

VENDEDOR_SUBORDINADO_JEFE				
ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%_COMISIÓN
44	Alberto Sánchez	27	Sonora	12
35	Berenice Medina	27	Nayarit	11
12	Claudia Ortega	27	B.C.	10

El resultado de una operación de intersección es la relación que consiste en todas las filas que están en ambas relaciones. Esto significa que si C es la intersección de A y B:

C:= A ∩ B

Entonces C consta de aquellas filas que están en A y B. Al igual que antes, A y B deben ser unión compatible.

3. Diferencia

La operación de diferencia (-) permite identificar filas que están en una relación y no en otra. Suponga que es de interés identificar los jefes que no son subordinados de otros jefes, entonces se toma la diferencia entre VENDEDOR_JEFE y VENDEDOR_SUBORDINADO, en ese orden.

VENDEDOR_JEFE_JEFE := VENDEDOR_JEFE - VENDEDOR_SUBORDINADO

Esto produce la siguiente relación:

VENDEDOR_JEFE_JEFE

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN
27	Teresa Cardoso	15	Aguascalientes	11

La diferencia entre dos relaciones se define como la relación consistente en todas las filas que están en la primera relación, y no están en la segunda relación. Así, si:

C:= A – B

Entonces una fila está en C si y solo si está en A y no está en B.

Observe que A – B no es lo mismo que B – A. Si se invierte el orden de las relaciones usadas en el ejemplo anterior se obtendría:

VENDEDOR_SUBORDINADO - VENDEDOR_JEFE

La relación resultante consistiría de todos aquellos vendedores que no son jefes de nadie, lo cual es justamente lo opuesto de los vendedores que no son dirigidos por nadie. Así es que el orden de las relaciones en una operación de diferencia es muy importante. Una vez más es importante recordar que ambas relaciones deben ser unión compatible.
La operación de diferencia puede también ser llamada operación de sustracción. Esta operación particular es muy valiosa en la solución de algunos problemas difíciles, que de otro modo no se solucionarían.

4. Producto

La operación producto (*) es valiosa como un bloque para la construcción de una reunión (JOIN), que es probablemente la operación más importante en el álgebra relacional. La operación producto es idéntica a la operación en Matemáticas que crea el producto cartesiano de dos conjuntos. Ahora se explicará qué significa esto, ilustrándolo con la operación de producto en un ejemplo abstracto muy simple.

Considere las relaciones de la figura que sigue, A y B son relaciones de dos columnas que tienen los atributos X, Y, W y Z, respectivamente. El producto de A y B es C, que se muestra en el inciso (b) de la figura.

(a) La relación A y B	
A	
X	Y
10 11	22 25

B	
W	Z
33 37 42	54 98 100

(b) El producto de A y B

C (:= A * B)

X	Y	W	Z
10	22	33	54
10	22	37	98
10	22	42	100
11	25	33	54
11	25	37	98
11	25	42	100

Ejemplo de la operación producto.

C:= A * B

Observe que el producto se crea:

1. Concatenando los atributos de las dos relaciones.
2. Uniendo cada fila en A, con cada una de las filas en B.

Esto significa que los atributos de C son todos los atributos de A y B juntos. Debido a que A y B tienen dos atributos cada uno, C tiene cuatro atributos. Las filas de C se crean ensartando filas de A y B juntas. Cada fila de A se corresponde tres veces con B, y entonces aparece en tres diferentes filas en A, por el número de filas en B.

Otro ejemplo se da si tomamos el producto de PRODUCTO y VENTA.

P_V:= PRODUCTO * VENTA

P_V tendría 10 columnas y 40 filas, por lo que habría un problema en este caso, debido a que una columna en PRODUCTO y una columna en VENTA (ID_PRODUCTO) tienen el mismo nombre. Esto se resuelve modificando el nombre de la columna en cada caso, añadiendo el nombre de la relación original. Así, en P_V se tienen columnas llamadas PRODUCTO.ID_PRODUCTO y VENTA.ID_PRODUCTO.

Esto parece ser una aplicación no natural para la operación producto, lo que significa que no está claro qué tipo de consulta se responde tomando el producto de dos relaciones. No obstante el producto se usa como la operación de construcción de un bloque para la reunión (JOIN), también usado en el lenguaje de consulta SQL.

5. Selección

La operación de selección (=) se usa para crear una relación a partir de otra relación, seleccionando solo aquellas filas que satisfacen una condición específica. Por ejemplo:

Instrucción: dar toda la información de los vendedores en la oficina de Sonora.

Esto se soluciona seleccionando las filas de la relación VENDEDOR de la figura, sujetas a la condición de que una fila se selecciona solo si el atributo OFICINA es igual a “Sonora”. Esto se hace en el álgebra relacional como sigue:

```
VEND_SONORA:= SELECT  (VENDEDOR: OFICINA  = “Sonora”)
```

El nombre “VEND_SONORA” se ha definido para identificar la relación. La palabra clave “SELECT” se usa para indicar que está realizando una operación de selección. A continuación de “SELECT” se pone entre paréntesis el nombre de la relación de la cual son seleccionadas las filas, seguido por dos puntos (:); seguido por una condición de selección. Esas filas que satisfacen la condición de selección serán seleccionadas y puesta en relación resultante. El resultado de esta operación de selección es la siguiente:

VEND_SONORA				
ID_VENDEDOR	NOMB_VENDEDO	ID_JEFE	OFICINA	% COMISIÓN
14	Mónica Márquez	44	Sonora	11
39	Gabriela Acuña	44	Sonora	10
44	Alberto Sánchez	27	Sonora	12

Las condiciones de selección son esencialmente las mismas condiciones usadas en las instrucciones IF (si) en los lenguajes tradicionales de programación. Sin embargo, los nombres de las columnas usados en la condición de la selección dada deben encontrarse en la relación nombrada en la operación de selección. Algunos ejemplos de condiciones de selección que podrían usarse con la relación VENDEDOR son:

```
ID_VENDEDOR=23
NOMB_VENDEDOR=Berenice Medina
ID_JEFE>=20
not OFICINA = “B.C.”
% COMISION < 11
```

Pueden usarse operadores de comparación tales como < y >. Hay cinco operadores de comparación mencionados en el subtema anterior. Para cada uno de estos hay un operador correspondiente que usa el operador booleano “NOT”. También se pueden usar los conectores booleanos “AND” y “OR”. El operador NOT se puede usar para negar una condición entera. Ejemplos

Instrucción: ¿Qué vendedor tiene ID 23?

Solución: SELECT (VENDEDOR: ID_VENDEDOR=23)

Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN
23	FRANCISCO MORENO	35	NAYARIT	9

Instrucción: ¿Quiénes son los vendedores con un jefe con un ID mayor o igual que 20?

Solución: SELECT (VENDEDOR: ID_JEFE > = 20)

Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN
10	Blanca Soto	27	Aguascalientes	10
14	Mónica Márquez	44	Sonora	11
23	Francisco Moreno	35	Nayarit	9
39	Gabriela Acuña	44	Sonora	10

Instrucción: dar toda la información sobre los vendedores excepto aquellos de la oficina de Buenos Aires.

Solución: SELECT: (VENDEDOR:OFICINA=”B.C.”

Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN

10	Blanca Soto	27	Aguascalientes	10
14	Mónica Márquez	44	Sonora	11
23	Francisco Moreno	35	Nayarit	9
39	Gabriela Acuña	44	Sonora	10
27	Teresa Cardoso	0	Aguascalientes	15
44	Alberto Sánchez	27	Sonora	12
35	Berenice Medina	27	Nayarit	11

Instrucción: ¿Qué vendedores tienen una comisión menor a 11%?

Solución: SELECT (VENDEDOR: %COMISION<11)

Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN
10	Blanca Soto	27	Aguascalientes	10
23	Francisco Moreno	35	Nayarit	9
39	Gabriela Acuña	44	Sonora	10

Instrucción: ¿Quiénes son los vendedores de la oficina de Sonora que tienen una comisión >11%?

Solución: SELECT (VENDEDOR: OFICINA =”Sonora” AND %_ COMISION>11)

Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN
14	Mónica Márquez	44	Sonora	11
44	Alberto Sanchez	27	Sonora	12

Instrucción: ¿Quiénes son los vendedores cuyo jefe tiene identificador 27 o una comisión mayor que 10?

Solución: SELECT (VENDEDOR: ID_JEFE =27 AND %_ COMISION>10)

Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%COMISIÓN

10	Blanca Soto Mónica	27	Aguascalientes	10
14	Márquez Elena	44	Sonora	11
37	Hernández Teresa	12	B.C.	13
27	Cardoso Alberto		Aguascalientes	15
44	Sánchez Berenice	27	Sonora	12
35	Medina Buster	27	Nayarit	11
12	Sánchez	27	B.C.	10

6. Proyección

Para ilustrar la operación de selección preguntaban “Quiénes…”, lo cual siguiere que los usuarios querían solo los nombres de los vendedores que satisfacían la condición de la consulta. Sin embargo, la respuesta a cada consulta incluía filas enteras de datos tomadas de la relación VENDEDOR, debido a que la operación de selección siempre selecciona filas enteras. Está claro que se necesita alguna manera de eliminar las columnas no deseadas. Si la operación de selección puede pensarse como la eliminación de las filas no deseadas, la operación de proyectar se piensa como la eliminación de las columnas no deseadas. La relación que resulta de una operación de proyectar se llama una proyección de la relación original.

A diferencia de otras operaciones del álgebra relacional, la operación de proyectar no requiere de una palabra clave especial o símbolo. Más bien, para crear una proyección —una relación consistente solo de ciertas columnas identificadas de otra relación— se enlista simplemente la relación original seguida de las columnas que se quieren conservar encerradas entre corchetes.
Por ejemplo, si se desea identificar a los vendedores de la oficina de Sonora, se proyectaría la columna nombre de la relación VENDEDOR-SONORA que visualizamos en los ejemplos anteriores.

```
VENDEDOR_SONORA [ NOMB_VENDEDOR]
```

Esto resulta de la siguiente relación:

NOMB_VENDEDOR
Mónica Márquez
Gabriela Acuña
Alberto Sánchez

Se podrían haber escogido más de una columna, si se quiere el ID, el nombre y el jefe de esos vendedores se introduce:

```
VENDEDOR_SONORA (ID_VENDEDOR, NOMB_VENDEDOR, ID_JEFE)
```

Resultante:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE
14	Mónica Márquez	44
39	Gabriela Acuña	44
44	Alberto Sánchez	27

Es de interés conocer todos los diferentes % de comisión pagados a los vendedores. Se puede obtener esto proyectando simplemente la columna %_COMISION de la relación VENDEDOR: VENDEDOR [%COMISION]

Resultante:

%COMISIÓN
10
11
9
13
15
12

En cada tasa de comisión aparece solo una vez, aún cuando distintos vendedores tienen la misma tasa de comisión. Ya que una relación es un conjunto, una tasa dada aparece solo una vez. Esto es una característica importante de la operación de proyectar. Automáticamente elimina filas duplicadas de la operación resultante. Esto acurre cuando la relación resultante consiste de más de una columna. Si cualesquiera dos filas enteras de una relación son idénticas columna por columna, la fila aparece solo una vez en la relación.

Esta operación presenta una oportunidad conveniente para mostrar el anidamiento de operaciones en el álgebra relacional. Por anidamiento se entiende la ejecución de más de una operación sin asignar explícitamente un nombre a las relaciones intermedias resultantes. Por ejemplo, añadamos una operación de proyectar a una de las consultas usadas para ilustrar las operaciones de selección:

Instrucción: ¿quiénes son los vendedores que tienen una comisión < que 11%?

Solución: SELECT (VENDEDOR: %COMISION<11) [NOMB_VENDEDOR]

Resultado:

NOM_VENDEDOR
Blanca Soto
Francisco Moreno
Gabriela Acuña
Claudia Ortega

Este ejemplo muestra la operación de selección que se realiza primero seguida por la proyección sobre la columna NOMB_VENDEDOR resultante. Esto es permitido para anidar operaciones del álgebra relacional, como se necesite, usando paréntesis donde se necesite mostrar el orden de las operaciones.

7. Reunión

Conocida como JOIN, sirve para conectar datos a través de relaciones, es la función más importante en cualquier lenguaje de base de datos. Se conocen varias versiones que son las siguientes: la reunión natural (*natural join)* la reunión theta (*theta join*) y la reunión externa (*outer join*) de las cuales la más importante es la reunión natural. Esta relación almacena al cliente, al vendedor y al producto involucrado en una transacción de venta particular, incluyendo los IDs de estos tres elementos de datos. La información permite hacer conexiones lógicas entre las relaciones CLIENTE, VENDEDOR y PRODUCTO. Pensemos en un ejemplo: se deja conocer los nombres de los clientes que han hecho compras al vendedor 10, lo primero es seleccionar aquellas ventas aplicadas solo al vendedor 10, y se ponen entonces en una relación que se llama VENTA_10. Resultante:

VENTA_10

FECHA	ID_CLIENTE	ID_VENDEDOR	ID-PRODUCTO	CANTIDAD
28/02	100	10	2241	200
05/02	105	10	2241	100
14/02	105	10	2249	50

Se obtiene la información deseada reuniendo las relaciones VENTAS_10 y CLIENTE. Esta operación procede como sigue:

1. Se crea el producto de VENTA_10 y CLIENTES. Se obtiene una relación con 11 columnas (5 de VENTA_10 y 6 de ID_CLIENTE) y 12 filas (3 en VENTA_10 * 4 en CLIENTE).

2. Todas las filas de esta relación producto se eliminan, excepto aquellas en las cuales el ID_CLIENTE de VENTA_10 es igual al ID_CLIENTE de CLIENTE. se debe observar que hay dos columnas ID_CLIENTE en la relación y que en cada fila los valores en estas dos columnas son idénticos.

3. Dado que las dos columnas ID_CLIENTE contienen idéntica información, una de ellas puede

eliminarse. Esto resulta en la reunión natural de VENTA_10 y CLIENTE mostrada en la figura.

FECHA	ID_CLIENTE	ID_VENDEDOR	ID_PRODUCT O	CANTIDAD	
28/02 05/02 14/02	100 105 105	10 10 10	2241 2241 2241	200 100 50	
ID_CLIENTE	NOMB_CLIENTE	DIRECCIÓN	PAÍS	SALDO INICIAL	SALDO ACTUAL
100 105 105	José López Julián Julián	Box 241, Sonora B 918, Aguascalientes B 918 Aguascalientes	Mérida Culiacán Culiacán	45.551 49.333 49.333	52.113 57.811 57.811

Resultado de la reunión natural.

Se ha obtenido mucha información de solo los nombres de los clientes. Si se quiere solo el nombre, se tendrá que proyectar la columna NOMB_CLIENTE de la relación.

FECHA	ID_CLIENTE	ID_PRODUCTO	CANTIDAD		
28/02 05/02 14/02	10 10 10	2241 2241 2241	200 100 50		
ID_CLIENTE	NOMB_CLIENTE	DIRECCIÓN	PAÍS	SALDO INICIAL	SALDO ACTUAL
100 105 105	José López Julián Julián	Box 241,Sonora B 918, Aguascalientes B 918, Aguascalientes	Mérida Culiacán Culiacán	45.551 49.333 49.333	52.113 57.811 57.811

Reunió natural de VENTA_10 y CLIENTE.

La definición general de la reunión natural es la siguiente: se asume que se quiere tomar la reunión natural de dos relaciones, A y B, las cuales tienen las columnas C1,…,Cn en común. Entonces JOIN (A, B) se obtiene a través de los siguientes pasos:

a) Tomar el producto de A y B. La relación resultante tendrá las dos columnas para cada C1,…,Cn.
b) Eliminar todas las filas del producto, acepto aquellas en las cuales los valores de las columnas C1,…Cn en A son iguales, respectivamente, a los valores de esas columnas en B.
c) Proyectar una copia de las columnas C1,…Cn.

Ejemplos de las funciones y uso de la reunión (JOIN) con varias consultas simples. **Instrucción:** adjunte la información de las ventas a la información sobre los vendedores. **Solución:** JOIN (VENDEDOR, VENTA).
Resultado:

ID_VENDEDOR	NOMB_VENDEDOR	ID_JEFE	OFICINA	%_COMISIÓN
10	Blanca Soto	27	Aguacalientes	10
10	Blanca Soto	27	Aguascalientes	10
10	Blanca Soto	27	Aguascalientes	10
23	Francisco Moreno	35	Nayarit	9
23	Francisco Moreno	35	Nayarit	9
23	Francisco Moreno	35	Nayarit	9
23	Francisco Moreno	35	Nayarit	9
23	Francisco Moreno	35	Nayarit	9
37	Elena Hernández	12	B.C.	13
39	Gabriela Acuña	44	Sonora	10

FECHA	ID_CLIENTE	ID_PRODUCTO	CANTIDAD
28/02	100	2241	200
05/02	105	2241	100
14/02	105	2249	50
12/02	101	2518	300
15/02	101	1035	150
02/02	101	1035	200
01/02	101	2249	75
04/02	101	2241	250
22/02	110	2518	150
19/02	100	2518	200

La reunión natural de VENDEDOR y VENTA.

Se visualiza la misma consulta que en el ejemplo anterior, excepto que en este caso se ha usado la relación VENDEDOR en lugar de la relación CLIENTE.
Instrucción: ¿Cómo se llama el cliente involucrado en cada venta?

Solución: A := CLIENTE [ID_CLIENTE,NOMBRE_CLIENTE] B := JOIN (VENTA, A)
Resultado: **B**

FECHA	ID_VENDEDOR	ID_PROD	CANTIDAD	ID_CLIENTE	NOMB_CLIENTE
28/02	10	2241	200	100	José López
12/02	23	2241	100	105	Maritza
15/02	23	2249	50	105	Maritza
19/02	39	2518	300	101	José López
02/02	23	1035	150	101	Maritza
05/02	10	2249	200	101	Julián
22/02	37	2241	75	101	Gómez
14/02	10	2518	250	101	Julián
01/02	23	2518	150	110	Maritza
04/02	23	2518	200	100	Maritza

En este ejemplo no se proyectó la información no deseada de CLIENTE antes de hacer la reunión (JOIN). La respuesta a la consulta se encuentra en la relación que se ha nombrado B. En la columna ID_CLIENTE de la relación VENDEDOR se considera una columna redundante y se ha eliminado.

Instrucción: dar los nombres de los clientes que han comprado el producto 2218.

Solución: A:= SELECT (VENTA: ID_PROD= 2218) B:= JOIN (A, CLIENTE) [NOMB_CLIENTE]

Resultado:

B
NOMB_CLIENTE
Maritza
José López
Gómez

Esta solución del primer ejemplo crea una relación VENTA reducida, y en la relación A solo aquellas ventas que involucran el producto 2518. Entonces se hace la reunión de esta relación con la relación CLIENTE y se proyecta en NOMB_CLIENTE, dejando el resultado deseado en la relación B.
La operación de selección podría haber sido especificada en segundo lugar, en vez de la primera. Y en ese caso la solución habría sido:
A:= JOIN (VENTA, CLIENTE)

B:= SELECT (A: ID_PROD=2518) [NOMB_CLIENTE]

La respuesta es lógicamente equivalente a la primera solución.

Instrucción: ¿Quiénes han comprado lámpara de mesa?

Solución: A:= SELECT (PRODUCTO:DESCRIPCION=”lámpara de mesa”) B:=JOIN (A, VENTA) C:=JOIN (B, CLIENTE) [NOMB_CLIENTE]

Resultado:

C
NOMB_CLIENTE
José López
Julián
Maritza

El ejemplo debe seguir un camino lógico “lámpara de mesa” en la relación PRODUCTO hasta NOMB_CLIENTE en la relación CLIENTE. Para esto se identifican todos los productos “lámpara de mesa” y se sigue entonces el camino desde PRODUCTO a VENTA hasta CLIENTE.

Instrucción: ¿Qué vendedores han vendido productos manufacturados en Perú?

Solución: A:= SELECT (FABRICANTE: PAIS= Perú:) B:= JOIN (A, PRODUCTO) C:= JOIN(B, VENTA)

D:= JOIN (C, VENDEDOR) [NOMB_VENDEDOR]

Resultado:

D
NOMB_VENDEDOR
Blanca Soto
Francisco Moreno
DIVISION

8. División

Para trabajar con la división ejemplifiquemos la siguiente consulta.

Instrucción: liste los vendedores que han vendido todos los productos.

La base de datos contiene cuatro productos diferentes con ID: 1035, 2241, 2249 y 2218. Un vendedor satisface la consulta si él o ella han vendido cada uno de estos productos al menos una vez. Es decir,

para cada uno de estos productos debe haber al menos una fila en VENTA y que contenga el ID-VENDEDOR de este vendedor. Una consulta como esta puede solucionarse usando la operación división (divide) del álgebra relacional. ¿Cómo resolver esta consulta? Se seguirá un procedimiento cercano al que se usaría intuitivamente, y se mostrará cómo la operación de división corresponde con este procedimiento. El primer paso será entonces obtener una relación consistente del atributo clave para todos los productos en la base de datos. Se hace esto proyectando PRODUCTO en ID_PROD.

```
PI:= PRODUCTO[ID_PROD]
```

PI es una relación que contiene todos los valores de ID_PROD y se obtendrá una relación que contenga todas las instancias donde su vendedor y un producto están juntos en una única venta. Esto se hace proyectando la relación VENTA en ID_PROD e ID_VENDEDOR:

```
PI_VI := VENTA[ID_PROD, ID_VENDEDOR]
```

La instancia PI_VI consiste en un ID_PROD y un ID_VENDEDOR y significa que el producto representado por ID_PROD fue vendido por el vendedor representado por ID_VENDEDOR. Entonces PI_VI: consiste en todas las combinaciones producto/vendedor, donde el vendedor ha vendido el producto.

Resultante:

PI
ID_PROD
1035
2241
2249
2518
ID_VENDEDOR ID_PROD
10 2241
23 2518
23 1035
39 2518
37 2518
10 2249
23 2249
23 2241

Ahora solo se necesita determinar si cada vendedor presentado en PI_VI se asocia con cada ID del producto en PI, esto se hace automáticamente con la operación de división.
A:= PI_VI/PI

Resultado:

A
ID: VENDEDOR
23

En la operación de división se asume que A, B y C son relaciones, y se desea dividir B por C, dando A como resultado.

1. Las columnas de C deben ser un subconjunto de las columnas de B. Las columnas de A son todas y solo aquellas de B que no son columnas de C, esto corresponde al ejemplo anterior. Las columnas de PI/VI son ID_PROD e ID_VENDEDOR, mientras que la columna de PI es ID_PROD y la columna de A es ID_VENDEDOR.
2. Una fila se encuentra en A solo si está asociada con B con la fila de C. Esto también se corresponde al ejemplo citado, debido a que una fila de A (ID_VENDEDOR=23) está asociada en PI_VI con todas las filas de PI y es además el único ID_VENDEDOR asociado.
3. Esta operación es la opuesta a la operación producto. Es fácil verificar que si una relación es el producto de dos relaciones B y C, entonces se puede obtener B, dividiendo el producto de C: (B*C)/C=B entonces vemos la analogía con la aritmética común del porqué se ha llamado operación de división. Codd la incluyó en el álgebra relacional para proporcionar la capacidad necesaria para el cuantificador universal del cálculo relacional. De manera práctica, la operación de división se proporcionó para que se pudieran relacionar consultas que involucraran “cada” o “todo” como parte de la condición.

9. Asignación

Asignación es la operación estándar de los lenguajes de computación, que consiste en dar un valor a un nombre. En este caso la asignación se usa para dar un nombre a una nueva relación que se crea de relaciones existentes. Se usa el signo “:=” (es el nombre asignado a) para indicar la asignación de nombres a relaciones.
Para sintetizar los temas abordados a lo largo de la presente unidad, se sugiere revisar el material de apoyo titulado
SQL, el cual es una presentación de Power Point.

Actividad 4. Ejercicio: Funciones básicas de SQL y álgebra relacional

Revisa el documento de actividades.

***Consulta** los criterios de evaluación de la actividad.

Evidencia de aprendizaje. Desarrollo e integración de prototipo

Revisa el documento de actividades para las indicaciones de la actividad.

***Consulta** los criterios de evaluación de la evidenvia de aprendizaje.

Cierre de la unidad

Se ha concluido la tercera, y última, unidad de la asignatura de Bases de datos. A lo largo del curso se abordaron los temas básicos de informática, amarrando las ideas, poco a poco, hasta llegar a los conceptos generales de las bases de datos, que fueron el punto clave para conocer los tipos de bases de datos y gestores, dando una explicación de sus funciones y usos. Finalmente, en la unidad tres, se revisaron los modelos de bases de datos y su aplicación de manera práctica, utilizando SQL y sus instrucciones DDL y DML, pasando a la aplicación de ejercicios con álgebra relacional.

Es aconsejable que se revise nuevamente la unidad en caso de que los temas que se mencionan no sean familiares o no se recuerden, de no ser este el caso, se puede decir que se ha concluido satisfactoriamente con la unidad. ¡Muy bien! Enhorabuena.

Fuentes de consulta

- Celma, M.; Casamayor, J.C.; Mota, L. (2003) *Base de datos relacionales.* Madrid: Pearson-Prentice HALL.
- Pérez L, César. (2008) *Oracle 10g: administración y análisis de Base de datos*. Segunda edición. México: Alfaomega.
- Quiroz, Javier. (2003) “El modelo relacional de Base de datos” en *Boletín de Política Informática* Núm. 6 (Versión electrónica). Recuperado el 26 de enero de 2011, de http://www.inegi.org.mx/inegi/contenidos/espanol/prensa/contenidos/articulos/tecnologia/relacional.pdf
- Real Academia Española (2001) *Diccionario de la lengua española*. Vigésima segunda edición (Versión digital). Recuperado el 19 de enero de 2011, de http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=inform%E1tica
- Silberschatz, Abraham. (2006). *Fundamentos de Base de Datos*. España: McGraw-Hill.
- MasterMagazine (2005) *Definición de ASCII*. Recuperado el 21 de enero de 2011, de http://www.mastermagazine.info/termino/3926.php