Unidad VI: Lenguaje SQL

6.1 Introducción

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés *structured query language*) es un lenguaje declarativo de acceso a bases de datosrelacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

6.2 Definición de datos

Algunos de los tipos de datos básicos de SQL son:

- **Date**: una fecha de calendario que contiene el año (de cuatro cifras), el mes y el día.
- **Time**: La hora del día en horas minutos segundos (el valor predeterminado es 0).
- **Timestamp**: la combinación de Date y Time.

6.3 Estructura básica de las consultas

Una base de datos relacional consiste en un conjunto de relaciones, a cada una de las cuales se le asigna un nombre
único. Cada relación tiene una estructura similar a la presentada en el Capítulo 3. SQL permite el uso de valores nulos
para indicar que el valor o bien es desconocido, o no existe. Se fijan criterios que permiten al usuario especificar a qué
atributos no se puede asignar valor nulo, como estudiaremos en el Apartado 4.11.

La estructura básica de una expresión SQL consiste en tres cláusulas: select, from y where.

• La cláusula select corresponde a la operación proyección del álgebra relacional. Se usa para listar los atributos deseados
del resultado de una consulta.

• La cláusula from corresponde a la operación producto cartesiano del álgebra relacional. Lista las relaciones que deben
ser analizadas en la evaluación de la expresión.

• La cláusula where corresponde al predicado selección del álgebra relacional. Es un predicado que engloba a los atributos
de las relaciones que aparecen en la cláusula from.

6.4 Operaciones sobre conjuntos

Las operaciones de SQL-92 union, intersect y except operan sobre relaciones y corresponden a las operaciones
del álgebra relacional ∪, ∩ y –. Al igual que la unión, intersección y diferencia de conjuntos en el álgebra
relacional, las relaciones que participan en las operaciones han de ser compatibles; esto es, deben tener el
mismo conjunto de atributos.

A continuación se demuestra cómo se pueden formular en SQL varias de las consultas de ejemplo consideradas
en el Capítulo 3 utilizando consultas que incluyen las operaciones union, intersect y except de dos conjuntos.
Los dos conjuntos utilizados serán: el conjuntos de todos los clientes que tienen una cuenta en el banco, que puede
obtenerse con:

select nombre-cliente
from impositor

y el conjunto de todos los clientes que tienen un préstamo en el banco, que puede obtenerse con:

select nombre-cliente
from prestatario

Apartir de ahora, las letras i y p se utilizarán para hacer referencia a las relaciones obtenidas como resultado de
las dos consultas anteriores.

6.5 Funciones de agregación

El SQL nos ofrece las siguientes funciones de agregación para efectuar varias operaciones sobre los datos de una base de datos:

Función	Descripción
COUNT	Nos da el número total de filas seleccionadas
SUM	Suma los valores de una columna
MIN	Nos da el valor mínimo de una columna
MAX	Nos da el valor máximo de una columna
AVG	Calcula el valor medio de una columna

En general, las funciones de agregación se aplican a una columna, excepto la función de agregación COUNT, que normalmente se aplica a todas las columnas de la tabla o tablas seleccionadas. Por lo tanto,COUNT (*) contará todas las filas de la tabla o las tablas que cumplan las condiciones. Si se
utilizase COUNT(distinct columna), sólo contaría los valores que no fuesen nulos

ni repetidos, y si se utilizase COUNT(columna), sólo contaría los valores que no fuesen nulos.
Ejemplo de utilización de la función COUNT (*)
Veamos un ejemplo de uso de la función COUNT, que aparece en la cláusula SELECT, para hacer la consulta "¿Cuántos departamentos están ubicados en la ciudad de Lleida?":

```
SELECT COUNT(*) AS numero_dep
FROM departamentos
WHERE ciudad_dep = 'Lleida';
```

6.6 Valores nulos

En este apartado se define la forma en que las diferentes operaciones del álgebra relacional tratan los valores
nulos y las complicaciones que surgen cuando los valores nulos participan en las operaciones aritméticas o en

las comparaciones. Como se verá, a menudo hay varias formas de tratar los valores nulos y, como resultado, las
siguientes definiciones pueden ser a veces arbitrarias.

Las operaciones y las comparaciones con valores nulos se deberían evitar siempre que sea posible.
Dado que el valor especial nulo indica «valor desconocido o no existente», cualquier operación aritmética (como +, –, * y /)
que incluya valores nulos debe devolver un valor nulo.

De manera similar, cualquier comparación (como <, <=, >, >= y ≠) que incluya un valor nulo se evalúa al valor
especial desconocido; no se puede decir si el resultado de la comparación es cierto o falso, así que se dice

que el resultado es el nuevo valor lógico desconocido. Las comparaciones que incluyan nulos pueden aparecer
dentro de expresiones booleanas que incluyan las operaciones y (conjunción), o (disyunción) y no (negación).
Se debe definir la forma en que estas operaciones tratan el valor lógico desconocido.

6.7 Consultas anidadas

Una consulta anidada, como su nombre indica, es aquella que está contenida dentro de otra. Los resultados de una la consulta anidada se utilizan como valores de comparación de la cláusula WHERE de la consulta que la anida. Se evaluará antes la sentencia SELECT anidada y una vez obtenido el valor o conjunto de valores se evaluará la otra. El formato es el siguiente:

```
SELECT col1, col1, ... , coln
FROM tabla1
WHERE col <operador lógico>
(SELECT col
FROM tabla2
WHERE condiciones);
```

6.8 Consultas complejas

El SQL soporta dos grupos de consultas multitabla:
- la unión de tablas.
- la composición de tablas.

La unión de tablas

Esta operación se utiliza cuando tenemos dos tablas con las mismas columnas y queremos obtener una nueva tabla con las filas de la primera y las filas de la

segunda. En este caso la tabla resultante tiene las mismas columnas que la primera tabla (que son las mismas que las de la segunda tabla).

Cuando hablamos de tablas pueden ser tablas reales almacenadas en la base de datos o tablas lógicas (resultados de una consulta), esto nos permite utilizar la operación con más frecuencia ya que pocas veces tenemos en una base de datos tablas idénticas en cuanto a columnas. Elresultado es siempre una tabla lógica.

Por ejemplo queremos en un sólo listado los productos cuyas existencias sean iguales a cero y también los productos que aparecen en pedidos del año 90. En este caso tenemos unos productos en la tabla de productos y los otros en la tabla de pedidos, las tablas no tienen las mismas columnas no se puede hacer una unión de ellas pero lo que interesa realmente es el identificador del producto (idfab, idproducto), luego por una parte sacamos los códigos de los productos con existencias cero (con una consulta), por otra parte los códigos de los productos que aparecen en pedidos del año 90 (con otra consulta), y luego unimos estas dos tablas lógicas.

6.9 Vistas

En el modelo de datos relacional la forma de guardar la información no es la mejor para ver los datos

Una vista es una consulta, que refleja el contenido de una o más tablas, desde la que se puede acceder a los datos como si fuera una tabla.

Dos son las principales razones por las que podemos crear vistas.

Seguridad, nos pueden interesar que los usuarios tengan acceso a una parte de la información que hay en una tabla, pero no a toda la tabla.

Comodidad, como hemos dicho el modelo relacional no es el más comodo para visualizar los datos, lo que nos puede llevar a tener que escribir complejas sentencias SQL, tener una vista nos simplifica esta tarea.

Las vistas no tienen una copia física de los datos, son consultas a los datos que hay en las tablas, por lo que si actualizamos los datos de una vista, estamos

actualizando realmente la tabla, y si actualizamos la tabla estos cambios serán visibles desde la vista.
Nota: No siempre podremos actualizar los datos de una vista, dependerá de la complejidad de la misma (dependerá de si el conjunto de resultados tiene acceso a la clave principal de la tabla o no), y del gestor de base de datos. No todos los gestores de bases de datos permiten actualizar vistas, ORACLE, por ejemplo, no lo permite, mientras que SQL Server sí.

6.10 Modificación de las bases de datos.

Las instrucciones para realizar estas operaciones son:

- CREATE TABLE: Nos permite crear una tabla de datos vacía.
- INSERT: Permite almacenar registros en una tabla creada.
- UPDATE: Permite modificar datos de registros almacenados en la tabla.
- DELETE: Borra un registro entero o grupo de registros de una tabla.
- CREATE INDEX: Crea un índice que nos puede auxiliar para las consultas.
- DROP TABLE: Permite borrar una tabla.
- DROP INDEX: Borra el índice indicado.