EAE Business School

SQL

Máster en Business Intelligence e Innovación Tecnológica

Índice de contenidos

SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales 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 bases de datos, así como hacer cambios en ellas.

Wikipedia

- Structured Query Language
- Es el lenguaje más universalmente usado para bases de datos relacionales
- Lenguaje declarativo de alto nivel
- Desarrollado por IBM (1974-1977) y aceptado por Codd con algunos matices.
- Se convirtió en un standard definido por :
  - ANSI (American National Standards Institute) e
  - ISO (International Standards Organization)
- El standard actual es el SQL:1999 (aunque existen revisiones del 2003, 2006 y 2008).
  - Se integra en la mayoría de SGBD del mercado: Oracle, IBM DB2, SQL Server, PostreSQL,…
  - A pesar de tratarse de un estándar, la mayoría de fabricantes no lo adoptan totalmente y presentan sus propias versiones…es común encontrar incompatibilidades entre sentencias.

- Las Sentencias del SQL se dividen en:
  - Sentencias DDL (Data Definition Language): Permiten crear/modificar/borrar estructuras de datos.
  - Sentencias DML (Data Manipulation Languaje): para manipular datos
- También provee sentencias para:
  - Definir permisos (control de acceso de usuarios)
  - Manejo de transacciones
  - Otros.
- Las sentencias tienen un formato de lenguaje natural (“pseudo-inglés”).

- Términos:
  - tabla → relación
  - fila → tupla
  - columna → atributo

Sentencias de consultas.

Sentencias de extracción de datos de una BDD, conforman uno de los tipos de comandos SQL más utilizados y una de las bases de las aplicaciones actuales.

- Forman parte del DML.
- Basadas en las definiciones de Codd.
- El resultado es una tabla lógica.

Ejemplo:

SELECT *jugador*
FROM *planillas*
WHERE *numero = 10*

SELECT *[ALL / DISTINC] [*] / [ListaColumnas_Expresiones] AS [Expresion]*
FROM *Nombre_Tabla_Vista*
WHERE *Condiciones*
ORDER BY *ListaColumnas [ASC / DESC]*

SELECT

Permite seleccionar las columnas que se van a mostrar y en el orden en que lo van a hacer. Simplemente es la instrucción que la base de datos interpreta como que vamos a solicitar información.

ALL / DISTINCT

Permite seleccionar las columnas que se van a mostrar y en el orden en que lo van a hacer. Simplemente es la instrucción que la base de datos interpreta como que vamos a solicitar información.

NOMBRES DE CAMPOS

Se especifica una lista de nombres de campos de la tabla queremos devolver.
Separamos cada nombre de los demás mediante comas.
Se puede anteponer el nombre de la tabla al nombre de las columnas, utilizando el formato *Tabla.Columna*. Además de nombres de columnas, en esta lista se pueden poner constantes, expresiones aritméticas, y funciones, para obtener campos calculados de manera dinámica.
Si queremos que nos devuelva todos los campos de la tabla utilizamos el comodín "*".
La tabla lógica resultante por defecto mostrará el nombre del campo en la tabla origen, se puede modificar utilizando la cláusula "AS".

FROM

Esta cláusula permite indicar las tablas o vistas de las cuales vamos a obtener la información. Podemos separar las tablas con ',' o bien aplicar sentencias de uniones especiales.

WHERE

Especifica la condición de filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en la mayoría de las consultas.

CONDICIONES

Son expresiones lógicas a comprobar para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica y en ella utilizar diversos operadores como:

- **>** (Mayor)
- **>=** (Mayor o igual)
- **<** (Menor)
- **<=** (Menor o igual)
- **=** (Igual)
- **<>** o **!=** (Distinto)
- IS [NOT] NULL (para comprobar si el valor de una columna es o no es nula, es decir, si contiene o no contiene algún valor)
- **LIKE**: para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: “%” y “_”.
- **BETWEEN**: para un intervalo de valores
- **IN( )**: para especificar una relación de valores concretos.

ORDER BY

Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.
El modo de ordenado puede ser ascendente (ASC), el utilizado por defecto, o descendiente (DESC).

Sentencias de consultas (ejemplos).

- Mostrar apellido, ciudad y región (LastName, city, region) de los empleados de USA:

```
SELECT E.LastName AS Apellido, City AS Ciudad, Region
FROM Employees AS E
WHERE Country = 'USA'
```

- Mostrar las distintas regiones de las que tenemos algún cliente, accediendo sólo a la tabla de clientes:

```
SELECT DISTINCT Region FROM Customers WHERE Region IS NOT NULL
```

- Mostrar los clientes que pertenecen a las regiones CA, MT o WA, ordenados por región ascendentemente y por nombre descendentemente:

```
CODE SELECT *
FROM Customers
WHERE Region IN('CA', 'MT', 'WA')
ORDER BY Region, CompanyName DESC
```

Sentencias de consultas (ejemplos).

- Mostrar los clientes cuyo nombre empieza por la letra “W”:

  *SELECT * FROM Customers WHERE CompanyName LIKE ‘W%’*

- Mostrar los empleados cuyo código está entre el 2 y el 9:

  *SELECT * FROM Employees WHERE EmployeeID BETWEEN 2 AND 9*

UNIONES DE TABLAS…JOIN

Es frecuente necesitar datos que podemos encontrar representados en tablas distintas. Éstos se pueden explotar directamente en una única consulta gracias al concepto ***join***.

Éste tipo de sentencias se basan en álgebra relacional para definir distintas agrupaciones y métodos de enlace entre los conjuntos que contiene cada tabla.

Se muestran los siguientes tipos de combinaciones:

- Producto cartesiano: tipo de unión básico, muestra el producto cartesiano de un conjunto de registros que cumplen la condición. Realmente se trata de un inner join simple.
- Interna (inner join): obtiene un producto cruzado de las dos tablas, por lo que se trata de una combinación entre los registros comunes que contienen.
- Externa (outer join): Mediante esta operación no se requiere que cada registro en las tablas a tratar tenga un registro equivalente en la otra tabla. El registro es mantenido en la tabla combinada si no existe otro registro que le corresponda.

INNER JOIN

Con esta operación se calcula el producto cruzado de todos los registros; así cada registro en la tabla A es combinado con cada registro de la tabla B; pero sólo permanecen aquellos registros en la tabla combinada que satisfacen las condiciones que se especifiquen. Este es el tipo de *JOIN* más utilizado, por lo que es considerado el tipo de combinación predeterminado.

Ejemplos:

- Explícito
  SELECT Campos FROM empleado INNER JOIN departamento ON empleado.IDDepartamento = departamento.IDDepartamento

- Implícito
  SELECT Campos FROM empleado, departamento WHERE empleado.IDDepartamento = departamento.IDDepartamento

OUTER JOIN

Realiza la combinación externa, sin vincular el resultado a la existencia de un registro. Permite las siguientes variables:

- Full: el resultado potencial es todo el conjunto de las dos tablas.
- Left: la sentencia LEFT OUTER JOIN retorna la pareja de todos los valores de la tabla izquierda con los valores de la tabla de la derecha correspondientes, o retorna un valor nulo NULL en caso de no correspondencia.
- Right: el resultado de esta operación siempre contiene todos los registros de la tabla de la derecha (la segunda tabla que se menciona en la consulta), aun cuando no exista un registro correspondiente en la tabla de la izquierda para uno de la derecha.

Ejemplos:

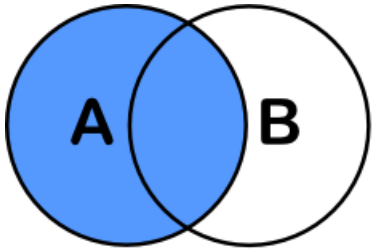
- SELECT * FROM empleado LEFT OUTER JOIN departamento ON empleado.IDDepartamento = departamento.IDDepartamento

- SELECT * FROM empleado RIGHT OUTER JOIN departamento ON empleado.IDDepartamento = departamento.IDDepartamento

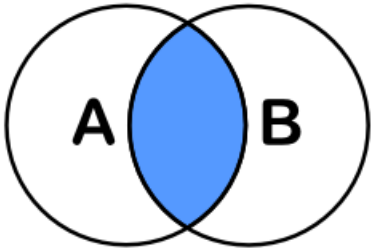
- SELECT * FROM empleado FULL OUTER JOIN departamento ON empleado.IDDepartamento = departamento.IDDepartamento

CHEATSHEET

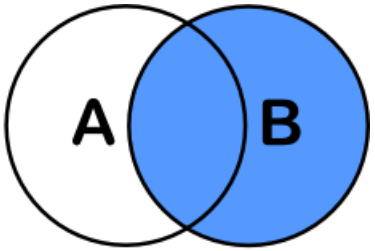
SQL JOINS



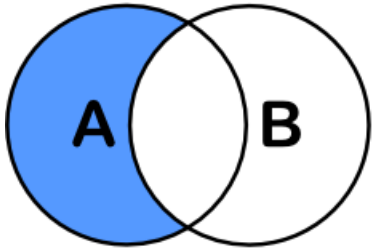

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```



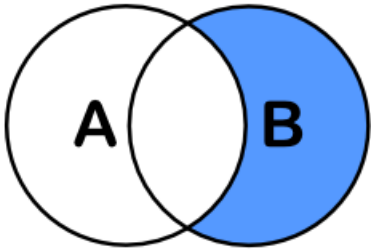

```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```



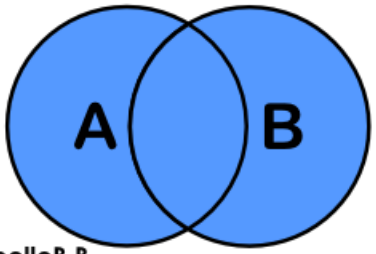

```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```



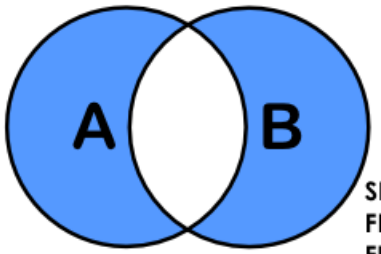

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

Consultas anidadas

Tambien llamadas subconsultas, se trata de utilizar una tabla lógica resultante de otra consulta como si de una tabla física o grupo de datos se tratara. Podemos incluir subconsultas en múltiples posiciones:

- Dentro del FROM, simulando una tabla.
- En el WHERE, devolviendo un conjunto de datos para utilizar en la condición.
- En la selección de campos, devolviendo una serie de valores como un campo.

Ejemplos:

- *SELECT sid, sname FROM (SELECT * FROM student) as example*

- *SELECT * FROM ciudades WHERE provincia in (SELECT codigoProvincia FROM provincias WHERE pais = 'ES')*

- *SELECT sid, sname, (SELECT COUNT (*) FROM apply A WHERE A.sid = S.sid and A.decision = 't') as acepted FROM student S ORDER BY acepted DESC*

Sentencias de Definición de Datos

DDL (Data Definition Language): Lenguaje que permite definir las estructuras de almacenamiento de los datos así como de los componentes lógicos y físicos de la BDD.

Permite:

- Crear elementos (*CREATE)*
- Borrar elementos *(DROP)*
- Modificar elementos (*ALTER).*

CREATE

Objetivo: bases de datos, esquemas, tablas, índices, vistas, procedures, funciones…

Ejemplo:

```
CREATE TABLE example_timestamp (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        data VARCHAR(100),
        cur_timestamp TIMESTAMP(8)
);
```

Restricciones (pueden variar en función del SQBD)

NOT NULL
UNIQUE
PRIMARY KEY
FOREIGN KEY
ENUM
SET

Otros atributos

DEFAULT

Tipos de datos (pueden variar en función del SQBD)

TINYINT[(longitud)] [UNSIGNED] [ZEROFILL] |
SMALLINT[(longitud)] [UNSIGNED] [ZEROFILL] |
MEDIUMINT[(longitud)] [UNSIGNED] [ZEROFILL] |
INT[(longitud)] [UNSIGNED] [ZEROFILL] |
INTEGER[(longitud)] [UNSIGNED] [ZEROFILL] |
BIGINT[(longitud)] [UNSIGNED] [ZEROFILL] |
REAL[(longitud,decimales)] [UNSIGNED] [ZEROFILL] |
DOUBLE[(longitud,decimales)] [UNSIGNED] [ZEROFILL] |
FLOAT[(longitud,decimales)] [UNSIGNED] [ZEROFILL] |
DECIMAL(longitud,decimales) [UNSIGNED] [ZEROFILL] |
NUMERIC(longitud,decimales) [UNSIGNED] [ZEROFILL] |
DATE |
TIME |
TIMESTAMP |
CHAR(longitud) [BINARY | ASCII | UNICODE] |
VARCHAR(longitud) [BINARY] |
TINYBLOB |
BLOB |
MEDIUMBLOB |
LONGBLOB |
TINYTEXT |
TEXT |
MEDIUMTEXT |
LONGTEXT

DROP

Objetivo: bases de datos, esquemas, tablas, índices, vistas, procedures, funciones…

Ejemplo:

DROP TABLE table_name ;

ALTER

Objetivo: tablas, columnas,…

Permite múltiples acciones: añadir columnas, modificar columnas, eliminar columnas, renombrar tablas, etc.

Ejemplos:

```
ALTER TABLE t1 RENAME t2;
ALTER TABLE t2 MODIFY a TINYINT NOT NULL;
ALTER TABLE t2 ADD d TIMESTAMP;
ALTER TABLE t2 ADD INDEX (d), ADD UNIQUE (a);
ALTER TABLE t2 DROP COLUMN c;
ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
      ADD PRIMARY KEY (c);
```

Sentencias de Manipulación de Datos

Data Manipulation Language ***(DML)*** es el lenguaje que permite realizar las tareas de mantenimiento y explotación de los datos almacenados en una base de datos.

- Se basa en las llamadas operaciones CRUD:
  - ***Create:*** insert
  - ***Read:*** select
  - ***Update:*** update
  - ***Delete***: delete

INSERT

Realiza la inserción de un registro en una tabla

Sintaxis: *INSERT INTO tbl_name VALUES();*
Ejemplo:

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway');

Puede incluir como valores el resultado de consultas, funciones, etc.

SELECT

Selecciona un conjunto de registros de una base de datos.

Sintaxis: *SELECT column_name,column_name FROM table_name;*
Ejemplo:

SELECT CustomerName,City FROM Customers;

Visto en detalle en los puntos anteriores.

UPDATE

Actualiza un registro ya existente en una tabla.

Sintaxis:

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

Ejemplo:

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste';
```

Puede incluir como valores el resultado de consultas, funciones, etc.

TABLE 7.1 SQL Data Definition Commands

COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Constraint used to validate data in an attribute
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows/columns from one or more tables
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table (and thus its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

Resumen DML

TABLE 7.2 SQL Data Manipulation Commands

COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute's values in one or more table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to their original values

TABLE 7.2 SQL Data Manipulation Commands (continued)

COMMAND OR OPTION	DESCRIPTION
COMPARISON OPERATORS	
=, <, >, <=, >=, <>	Used in conditional expressions
LOGICAL OPERATORS	
AND/OR/NOT	Used in conditional expressions
SPECIAL OPERATORS	Used in conditional expressions
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
AGGREGATE FUNCTIONS	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

SQLFiddle.com

A tool for easy online testing and sharing of database problems and their solutions.

Accede aquí…http://sqlfiddle.com/

SQL puede ser muy extenso y aquí vemos una pequeña parte. No dejes de revisar internet en busca de ejemplos y documentación ampliada!!