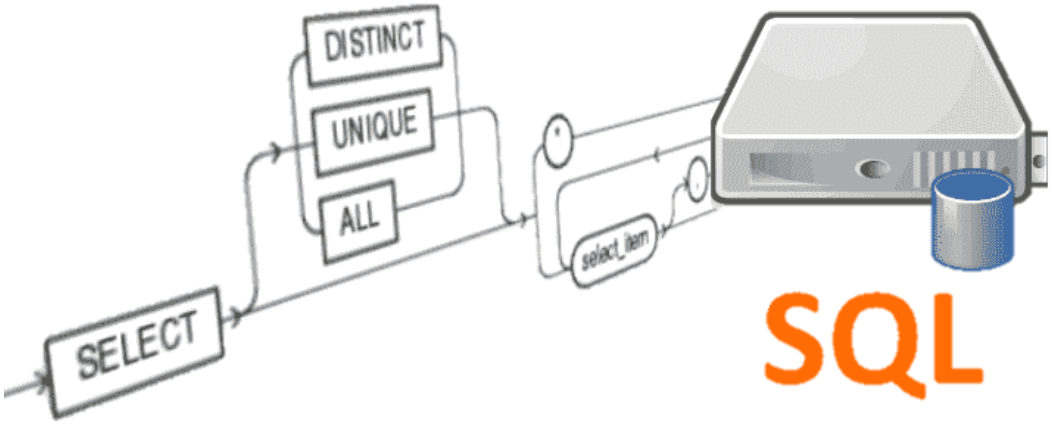
SQL

Structured Query Language

(Lenguaje de Consulta Estructurada)



Introducción

- Los lenguajes formales proporcionan una **notación concisa** para la representación de consultas.
- Sin embargo, los sistemas de base de datos necesitan un **lenguaje de consultas más cómodo** para el usuario.
- Aunque SQL se considere un lenguaje de consultas, contiene **muchas otras capacidades** que incluyen características para definir estructuras de datos, modificación de datos y la **especificación de restricciones de integridad.**
- Los sistemas de gestión de base de datos con soporte SQL más utilizados son: **DB2, Oracle, SQL Server, Sybase ASE, MySQL, PostgreSQL, Firebird**

Historia

- SQL se ha establecido como el lenguaje estándar de base de datos relacionales. Hay **numerosas versiones** de SQL.

- La versión original se desarrollo en el laboratorio de investigación de San Jose, California **(San Jose Research Center) de IBM**, este lenguaje originalmente denominado **Sequel**, se implementó como parte del proyecto System R, a principios de 1970 [McJones97].

- Desde entonces ha evolucionado a lo que ahora se conoce como **SQL** (**Structured Query Language**, lenguaje estructurado de consultas).

Evolución

En 1986, **ANSI** (American National Standards Institute, Instituto Nacional Americano de Normalización) e ISO (Organización Internacional de Normalización) publicaron una norma de SQL denominada **SQL-86.**

En 1987 IBM publicó su propia norma de SQL denominada **SAA-SQL** (System Application Architecture Database Interfaz, ***Interfaz de BDs para arquitecturas de aplicación de sistemas***).

Años más tarde SQL3 incorpora **consultas recursivas** y **Triggers** y luego el SQL2003 incorpora **XML**.

SQL proporciona más de dos tipos de lenguajes diferentes: en particular veremos comandos de, para ***especificar el esquema relacional y*** de para ***las consultas y actualizaciones*** de la base de datos.

SQL no es un lenguaje de programación

- Fue diseñado con el único propósito de **acceder a datos estructurales**
- Como en el caso de los más modernos lenguajes relacionales, SQL está basado en el ***cálculo relacional de tuplas***.
- Toda consulta formulada utilizando el **cálculo relacional** de tuplas (o su equivalente, el **álgebra relacional**) se puede formular también utilizando SQL.

Características

- El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones s/los mismos.

- Es un lenguaje **declarativo de alto nivel**, que gracias a su **fuerte base teórica** y su **orientación al manejo de conjuntos** de registros, y no a registros individuales, permite una alta productividad en codificación

- De esta forma **una sola sentencia puede equivaler a uno o más programas** que utilizasen un lenguaje de bajo nivel orientado a registro.

Otras características de SQL

SQL puede ser utilizado para otras características propias que no poseen los lenguajes formales de consultas:

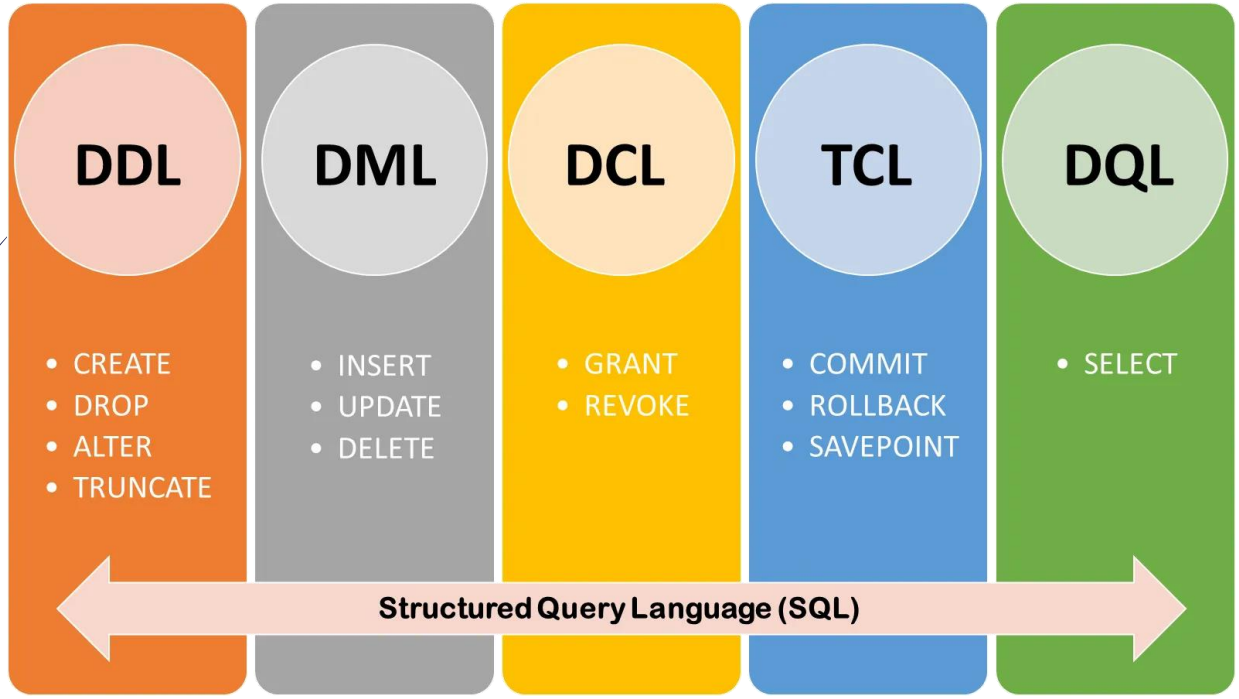
- **Definición de vistas**: El DDL de SQL incluye instrucciones para la definición de vistas.
- **Autorización**: El DDL de SQL incluye instrucciones para la especificación de los derechos de acceso a los objetos de la BD.
- **Integridad**: El DDL de SQL también incluye un conjunto de sentencias para la especificaciones de restricciones de integridad.
- **Control de transacciones**: SQL incluye ordenes para la especificación de los estados de una transacción, algunas implementaciones permiten además el bloqueo explicito de objetos de datos con el fin de manejar control de concurrencia.

SQL

SQL no es un lenguaje de programación

- Fue diseñado para **acceder a datos estructurales**
- SQL está basado en el **cálculo relacional de tuplas**. Toda consulta formulada utilizando el **álgebra relacional** se puede formular también utilizando SQL.
- El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.
- Es un lenguaje **declarativo de alto nivel**, que gracias a su **fuerte base teórica** y su **orientación al manejo de conjuntos** de registros, y no a registros individuales, permite una alta productividad en codificación

SQL se divide en Sub Lenguajes



SQL se divide en Sub Lenguajes

El SQL proporciona funcionalidad más allá de la simple consulta **(recuperación) de datos**.

(DDL) Data Definition Language - Lenguaje de Definición de Datos

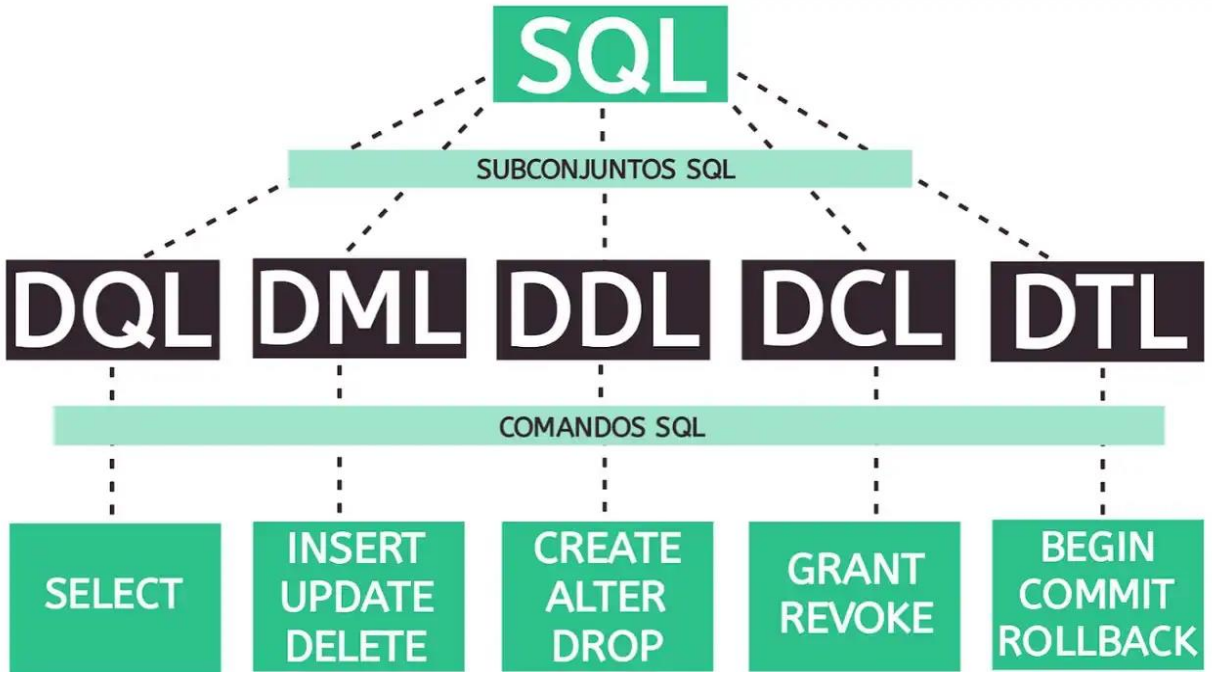
(LMD) Data Manipulation Language - Lenguaje de Manipulación de Datos

(DCL) Data Control Language - Lenguaje de Control de Datos

(TCL) Data Transaction Language - Lenguaje de Transacción de datos

(DQL) Data Query Language – Lenguaje de Consulta de Datos

SQL se divide en Sub Lenguajes



Lenguaje de Definición de datos (DDL)

El resultado de la compilación de estos comandos es un conjunto de tablas, relaciones y reglas cuyas definiciones quedan almacenadas en un archivo **(Componente estática)**

Este archivo comúnmente llamado diccionario de datos (o catalogo del sistema) es el que se consulta toda vez que se quiere leer, modificar o eliminar los datos de la base de datos (**Metadatos**)

CREATE: Crea Objetos en la Base de Datos

ALTER: Modifica Objetos en la Base de Datos

DROP: Elimina Objetos en la Base de Datos

Lenguaje de Manipulación de datos (DML)

Se entenderá por manipulación de datos la:

Recuperación de Información. **SELECT**

Inserción de nueva Información. **INSERT**

Eliminación de información existente. **DELETE**

Modificación de Información Almacenada. **UPDATE**

DDL - DML

DDL (Estructura)	DML (Datos)
CREATE	INSERT
ALTER	UPDATE
DROP	DELETE
	SELECT
	TRUNCATE

Otras características de SQL

Capacidades que van más allá del cálculo o del álgebra relacional.
Características EXTRAS proporcionadas por SQL:

Capacidades aritméticas

- En SQL es posible incluir operaciones aritméticas así como comparaciones, por ejemplo A < B + 3.

Asignación y comandos de impresión

- Es posible imprimir una relación construida por una consulta y asignar una relación calculada a un nombre de relación.

Funciones agregadas

- Operaciones tales como promedio (**average**), suma (**sum**), máximo (**max**), etc., se pueden aplicar a las columnas de una relación para obtener una cantidad única.

Sentencia SELECT

SELECT [ALL|DISTINCT]

{ * | *expr_1* [AS *c_alias_1*] [, ... [, *expr_k* [AS *c_alias_k*]]]}

FROM *table_name_1* [*t_alias_1*] [, ... [, *table_name_n* [*t_alias_n*]]]

[WHERE *condition*]

[GROUP BY *name_attr_i* [,... [, *name_attr_j*]]

[HAVING *condition*]]

[{UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]

[ORDER BY *name_attr_i* [ASC|DESC] [, ... [, *name_attr_j* [ASC|DESC]]]];

Cláusulas comando SELECT

“*” Indica que se incluyen ***TODOS*** los atributos, a menos que se mencione una lista

FROM Indica de cuál/es tabla/s se extraerá la información

WHERE Permite el ***filtrado de filas***, es decir, explicitar una condición de búsqueda de tuplas

ORDER BY Permite indicar el o los campos que liderarán y mostrarán el ordenamiento de las tuplas.

Algunos Ejemplos

SELECT * **FROM** cliente

Todos los atributos de la tabla

SELECT apellido, nombres, teléfono **FROM** cliente

Especificación de atributos → Proyección

SELECT monto*2 **AS** doble , fecha **FROM** venta

Nombre con el que aparece la columna

SELECT * **FROM** cliente **ORDER BY** provincia

Se ordena por un campo provincia, por defecto es ascendente

SELECT * **FROM** alumno **GROUP BY** carrera

Se agrupan por un campo común

Cláusulas comando SELECT

GROUP BY: Permite “agrupar” tuplas **según un campo común**. Permite funciones de agregación

HAVING: Permite un nuevo filtrado, **pero sobre las tuplas afectadas por el GROUP BY**, en función de una condición aplicable a cada grupo de filas

UNION [ALL] | INTERSECT | EXCEPT: permiten implementar **otras operaciones** del AR, y en especial, para la **implementación de subconsultas o consultas anidadas**

Condiciones WHERE

Comparación**: Operadores >, <, >=, <= y <>.**

SELECT campos **FROM** tabla **WHERE** campo > valor

Pertenencia a rango: **Operador BETWEEN**

SELECT campos **FROM** tabla **WHERE** campo **BETWEEN** valor1 **AND** valor2

Pertenencia a conjunto: **Operador IN** seguido de una lista de elementos entre paréntesis**.**

SELECT campos **FROM** tabla **WHERE** campo **IN** (valor, valor, valor, …)

Condiciones WHERE

Correspondencia con patrón: **Operador LIKE**

SELECT Lista_campos **FROM** table **WHERE** campo **LIKE** patrón

El patrón permite búsquedas con **comodines:**

“%” Representa 0 o más caracteres

“?” Representa un carácter

Test de valor nulo: **Operador IS NULL** permite determinar un campo es nulo o no

SELECT campos **FROM** tabla **WHERE** campo **IS NULL**

Ejemplos (LIKE)

Mostrar todos los empleados de nombre Ángel

SELECT Apenom **FROM** Empleado **WHERE** Apenom **LIKE** 'Ángel %'

Encuentra los empleados que tienen "Ángel" como PRIMER NOMBRE

SELECT Apenom **FROM** Empleado **WHERE** Apenom **LIKE** '% Ángel %'

Encuentra los empleados que tienen "Ángel" como PRIMER o SEGUNDO NOMBRE

Ejemplos (BETWEEN)

Listar a los empleados que ganan entre $30000 y $40000

SELECT empleado, sueldo **FROM** empleado
WHERE sueldo **>=** 30000 and sueldo **<=** 40000

Incluye los que ganan $30000 y $40000,

Consulta equivalente

SELECT empleado, sueldo **FROM** empleado
WHERE sueldo **BETWEEN** 30000 and 40000

EQUIVALENTE

Ejemplos (IN)

Listar hora que se dictan clases los lunes, miércoles y viernes

SELECT dia, hora **FROM** clase
WHERE dia = **“**Lunes” or dia=“Miercoles” or dia=“Viernes”

Las clases que se dictan los lunes o los miércoles o los viernes

EQUIVALENTE

SELECT dia, hora **FROM** clases
WHERE dia **IN** (Lunes, Miércoles, Viernes)

EQUIVALENTE

Condiciones WHERE

En una cláusula **WHERE** Se pueden combinar múltiples condiciones mediante los operadores lógicos **AND, OR y NOT,** determinando con el uso de **paréntesis** el orden de evaluación (y contrarrestando el efecto de la prioridad de operadores).

SELECT * **FROM** empleado **WHERE** salario > 35000

SELECT apellido, salario **FROM** empleado **WHERE** salario > 35000;

SELECT apellido, salario, sexo **FROM** empleado
WHERE sexo = 'F' **and** **(**apellido = 'PEREZ' **or** apellido = 'DIAZ'**)**;

Ejemplos (Comando SELECT)

Queremos conocer cuánto es el descuento para la jubilación del empleado (aporte jubilatorio), consistente en el 13% del salario, y listar sólo de los que ése monto no supere los $25000:

```
SELECT apellido, salario * 0.13 as aporte_jubilatorio
FROM empleado
WHERE salario * 0.13 <= 25000
```

Obtener datos de dos tablas (WHERE)

Para poder incluir datos contenidos en atributos dispuestos en más de una tabla, es necesario que ésas tablas se encuentren vinculadas o “relacionadas” a partir de atributos o campos comunes.

SELECT apellido, salario, cargo
FROM empleado, cargo
WHERE empleado.Id_cargo = cargo.Id_cargo

Si se omite la cláusula “**where**”, qué sucede?

Datos de mas de una tabla (WHERE)

Es posible listar campos de tablas relacionadas entre sí, mediante la cláusula "**where**" (sin utilizar JOIN).

En la cláusula **FROM** es posible introducir un "alias" al nombre de cada Tabla cuando existan atributos con el mismo nombre en las tablas. De este modo se diferencian los atributos con el mismo nombre.

SELECT L.titulo, L.editorial, A.apellido
FROM libro **AS** L, autor **AS** A
WHERE L.id_autor = A.id_autor

SELECT E.apellido, P.nombrep, T.horas
FROM empleado E, proyecto P, trabajan T
WHERE T.NroEmpT = E.NroEmp **AND** T.NroProy = P.NroProy

Obtener datos de dos tablas (JOIN)

```
SELECT L.titulo, L.editorial, A.apellido
FROM libro L
INNER JOIN autor A ON L.id_autor = A.id_autor
WHERE…
```

```
SELECT L.titulo, L.editorial, AU.apellido
FROM libro L
INNER JOIN autor A USING (id_autor)
WHERE…
```

```
SELECT E.apellido, P.nombrep, T.horas
FROM empleado E
INNER JOIN trabaja T ON T.NroEmpT = E.NroEmp
INNER JOIN proyecto P USING (NroProy)
WHERE ….
```

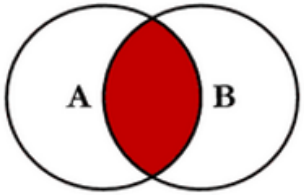
JOIN en el lenguaje relacional

La sentencia **JOIN** del SQL permite combinar registros de dos o más tablas en una base de datos relacional.

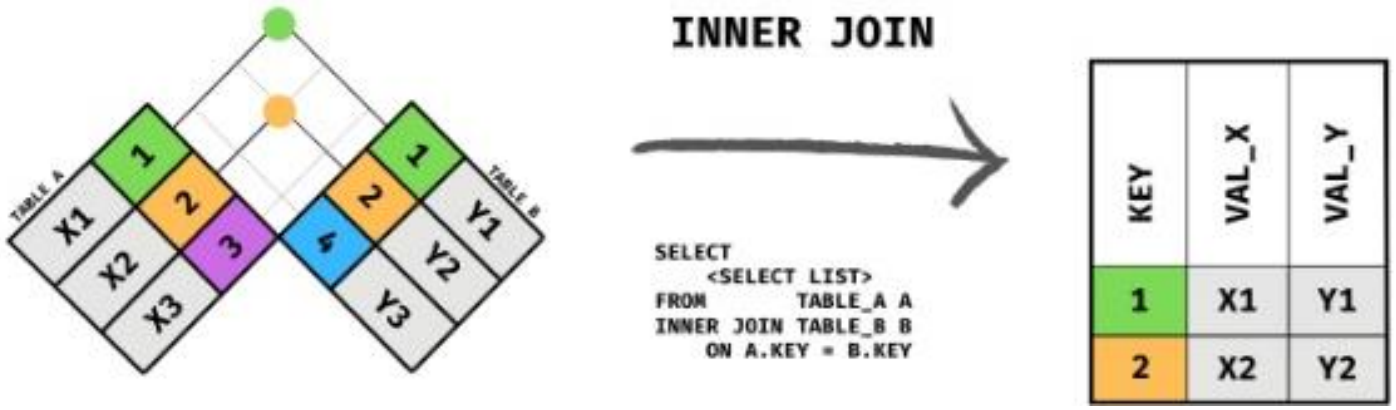
En el Lenguaje de Consultas Estructurado (SQL) y el estándar ANSI se especifican distintos tipos de JOIN:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- CROSS JOIN

INNER JOIN

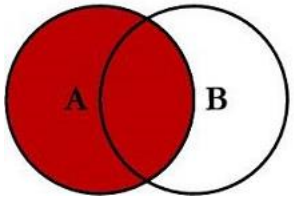


Este es el tipo más común. Devuelve las filas que tienen coincidencias en ambas tablas. Cada fila de la tabla A es combinada con las filas correspondientes de la tabla B, que satisfagan las condiciones que se especifiquen en el predicado del *JOIN*.

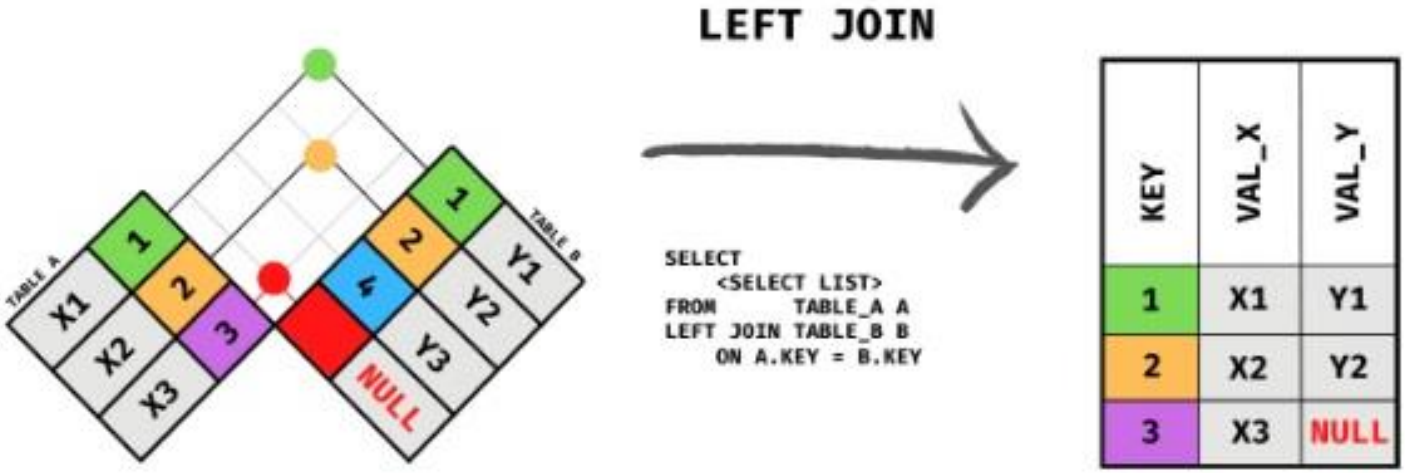


SELECT columnas **FROM** tablaA A **INNER JOIN** tablaB B
ON A.campo = B.campo

LEFT JOIN

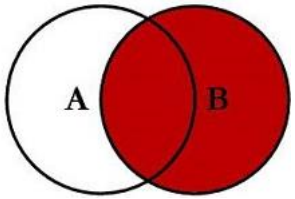


Devuelve **todas las filas de la tabla de la izquierda** (primera tabla mencionada) y las filas coincidentes de la tabla de la derecha. Si no hay coincidencia, las columnas de la tabla derecha mostrarán NULL

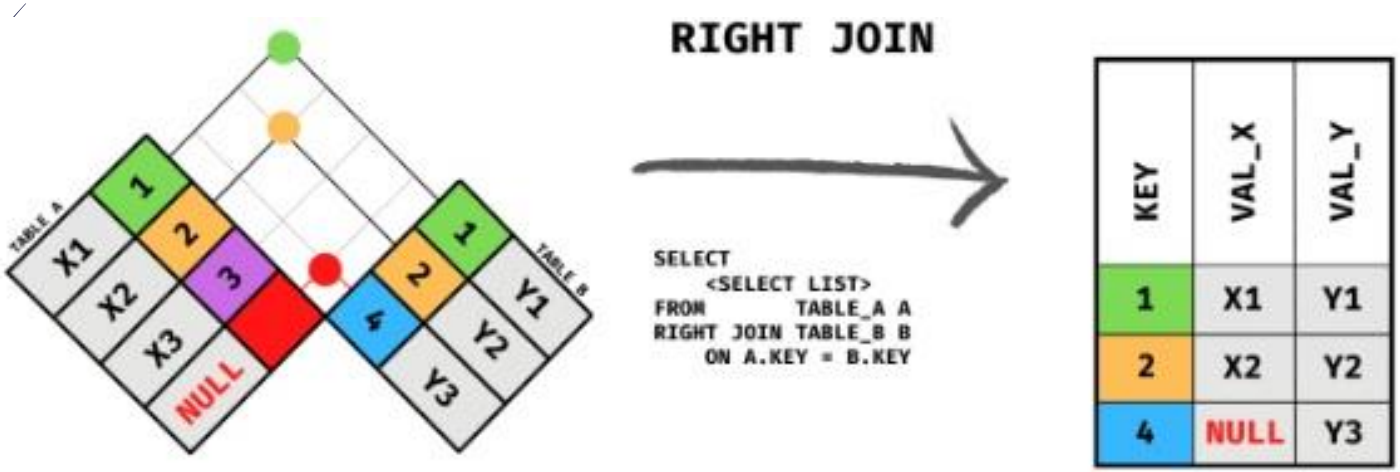


SELECT columnas **FROM** tablaA A **LEFT JOIN** tablaB B
ON A.campo = B.campo

RIGHT JOIN

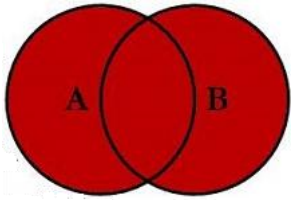


Devuelve **todas las filas de la tabla de la derecha** (segunda tabla mencionada) y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencia, las columnas de la tabla izquierda mostrarán NULL

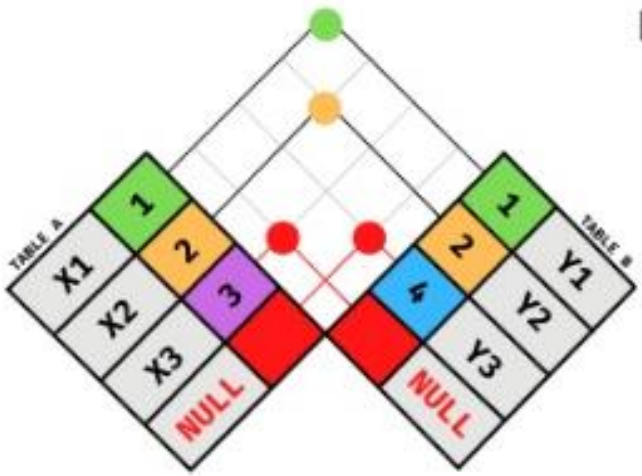


SELECT columnas **FROM** tablaA A **RIGHT JOIN** tablaB B
ON A.campo = B.campo

FULL OUTER JOIN



Devuelve todas las filas cuando hay una coincidencia en cualquiera de las tablas. Si no hay coincidencia, se rellena con NULL en la tabla que no tiene la coincidencia



FULL OUTER JOIN

```
SELECT
    <SELECT LIST>
FROM            TABLE_A A
FULL OUTER JOIN TABLE_B B
    ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2
3	X3	NULL
4	NULL	Y3

SELECT columnas **FROM** tablaA A **FULL OUTER JOIN** tablaB B **ON** A.campo = B.campo

CROSS JOIN

Devuelve el producto cartesiano de las dos tablas, es decir, combina cada fila de la primera tabla con cada fila de la segunda tabla.

SELECT columnas **FROM** tablaA **CROSS JOIN** tablaB

Resumen de JOIN

INNER JOIN: Coincidencias de ambas tablas.

LEFT JOIN: Todas las filas de la tabla izquierda, con coincidencias de la tabla derecha (o NULL si no hay).

RIGHT JOIN: Todas las filas de la tabla derecha, con coincidencias de la tabla izquierda (o NULL si no hay).

FULL JOIN: Todas las filas de ambas tablas, con coincidencias donde existan (o NULL si no hay).

CROSS JOIN: Producto cartesiano (combinación de todas las filas).

MySQL
y
Workbench

Introducción

- Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones.
- A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad
- MySQL es un motor de bases de datos de muy rápida en la lectura, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. **En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones**.

¿Qué es MySQL?

Es un dbms (sistema de gestión de base de datos) relacional desarrollado ***bajo licencia dual*** GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de **desarrollo web**

- MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius).
- **MySQL A.B.** fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Evolución

Poco a poco los elementos de los que carecía MySQL han ido siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Características disponibles en las últimas versiones:

- Amplio subconjunto del lenguaje SQL.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- Transacciones y claves foráneas.
- Conectividad segura. Replicación.
- Búsqueda e indexación de campos de texto.

Características

Multiplataforma: Para S.O. como Windows, Linux y Mac disponemos de nuestro servidor para instalarlo.

Multihilos: Está optimizado para equipos de múltiples procesadores

Fácil encontrar ayuda: Al ser una base de datos que se utiliza en multitud de aplicaciones web existen multitud de tutoriales, foros, …. en la red en los que podemos encontrar la información que necesitamos

Fácil de aprender: Simplemente con conocer el estándar de SQL podemos manejar la base de datos MySQL sin ningún problema

Probado: MySQL es un DBMS **ampliamente probado** por distintos usuarios y empresas con alto éxito.

Menos mantenimientos: Da la ventaja para que un programador pueda mantener la base de datos para sus aplicaciones. **Sin necesidad de ser un experto Administrador en Base de Datos (DBA).**

Código Abierto o no?

A diferencia de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, **MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código**

una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos.

Esto es lo que posibilita el **esquema de doble licenciamiento**: la base de datos se distribuye en varias versiones

Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y soporte oficial.

Herramientas

- Cada distribución de MySQL incluye herramientas para trabajar directamente con las bases de datos, especialmente las interfaces gráficas como Workbench

- ***MySQL Workbench*** es una **herramienta visual de diseño de bases de datos** que integra **desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento** para el administrador de sistemas de base de datos MySQL.

- MySQL Workbench es uno de los primeros productos de la familia MySQL que ofrece dos ediciones diferentes - una open source y una edición comercial

Muchas Gracias