SQL JOINS

Son sentencias que combinan registros entre diferentes tablas y esto se realiza con datos que tengan en común las tablas.

M. EN C. NIELS HENRIK NAVARRETE MANZANILLA

SQL JOINS

Crear dos tablas:

Clientes

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

Ordenes

SQL JOINS

Ejemplo de Join

```
SELECT * FROM
DBO.PRUEBA
 JOIN
DBO.TBL_EMPLEADO
 ON DBO.PRUEBA.ID = DBO.TBL_EMPLEADO.ID_EMPLEADO;
```

```
+----+----------+-----+---------+
| ID | NAME     | AGE | AMOUNT  |
+----+----------+-----+---------+
|  3 | kaushik  |  23 |    3000 |
|  3 | kaushik  |  23 |    1500 |
|  2 | Khilan   |  25 |    1560 |
```

```
SELECT ID, NAME, AGE, AMOUNT
   FROM CUSTOMERS, ORDERS
   WHERE  CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

SQL JOINS

- El uso del JOIN, debe combinarse con la clausula WHERE.
- Varios de los operadores que hemos visto son usados también en las tablas que se unen como: =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT;
- El operador mas habitual es: “ = “

Existen diferentes tipos de Joins tales como:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- SELF JOIN
- CARTESIAN JOIN

INNER JOIN
Unión Interno

- Es el mas utilizado de los JOIN, a veces se refiere como EQUIJOIN.
- Crea una nueva tabla de resultados mediante la combinación de valores de las columnas de dos tablas (Tabla1 y Tabla 2), con base a un predicado de unión. La consulta compara cada fila de la tabla 1 con cada fila de la tabla 2 para encontrar todos los pares de filas que satisfacen la de predicado de unión. Cuando se cumple el predicado regresa los resultados por cada fila que lo cumpla.

Sintaxis:

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.common_filed = table2.common_field;
```

LEFT JOIN
Combinación Izquierda

- Devuelve todas filas de la tabla izquierda, incluso si no hay coincidencia en la tabla derecha. Esto significa que si la clausula **ON** no coincide con algún registro de la tabla derecha, la unión aún va a regresar los resultados pero con valores **null** en cada columna de la tabla derecha que no hubo coincidencia.
- Esto significa que un Left Join devuelve todos los valores de la tabla izquierda, además de los valores que coinciden con la tabla derecha y valores **null** en caso de que no coincidieron con el predicado.

Sintaxis:

```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_filed = table2.common_field;
```

RIGHT JOIN
Combinación Derecha

- Devuelve todas filas de la tabla derecha, incluso si no hay coincidencia en la tabla izquierda. Esto significa que si la clausula **ON** no coincide con algún registro de la tabla izquierda, la unión aún va a regresar los resultados pero con valores **null** en cada columna de la tabla izquierda que no hubo coincidencia.
- Esto significa que un Right Join devuelve todos los valores de la tabla derecha, además de los valores que coinciden con la tabla izquierda y valores **null** en caso de que no coincidieron con el predicado.

Sintaxis:

```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_filed = table2.common_field;
```

FULL JOIN
Combinación Izquierda - Derecha

- Combina los resultados de ambas tablas (Izquierda y Derecha), regresando como resultado un conjunto de registros y aquellos que no coinciden los representa con valores **null.**

Sintaxis:

```
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_filed = table2.common_field;
```

*Si la base de datos no contiene la clausula FULL JOIN se puede utilizar UNION ALL

FULL JOIN
Combinación Izquierda - Derecha

Si la base de datos no contiene la clausula FULL JOIN se puede utilizar UNION ALL

```
SELECT  ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
LEFT JOIN ORDERS
```

```
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
UNION ALL
     SELECT  ID, NAME, AMOUNT, DATE
     FROM CUSTOMERS
     RIGHT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
```

SELF JOIN

- Se utiliza cuando se combina una tabla así misma, si la tabla fueran dos al mismo tiempo temporalmente.

Sintaxis:

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common filed = b.common field;
```

```
SELECT  a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

CARTESIAN JOIN / CROSS JOIN

- Es una combinación cartesiana, por lo que devuelve el producto cartesiano de la combinación de dos a mas tablas. Por lo tanto equivale a un Inner Join, donde el predicado del join siempre se evalúa como TRUE.

Sintaxis:

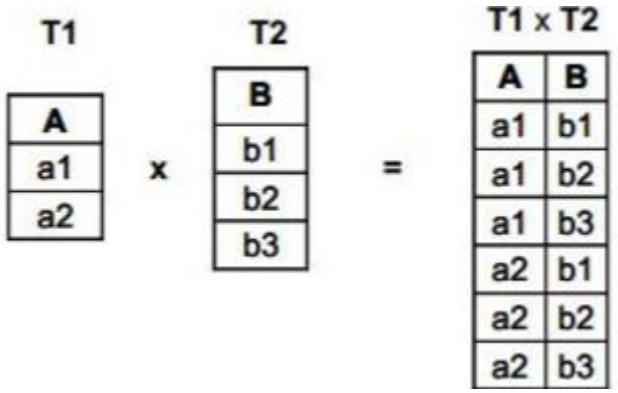
```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common filed = b.common field;
```

```
SELECT  a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

CARTESIAN JOIN / CROSS JOIN

- Es una combinación cartesiana, por lo que devuelve el producto cartesiano de la combinación de dos a mas tablas. Por lo tanto equivale a un Inner Join, donde el predicado del join siempre se evalúa como TRUE.

- Es una composición de tablas, con lo que se obtiene una tabla con las columnas de la primera tabla unidas a las columnas de la segunda tabla y las filas de la tabla resultante son todas las posibles concatenaciones de filas de la primera tabla con filas de la segunda tabla.



CARTESIAN JOIN / CROSS JOIN

- Ejercicio:

Tabla 1	
campo 1	campo 2
a1	b1
a2	b2
a3	b3

x

Tabla 2	
campo 11	campo 22
a4	b4
a5	b5

=

Tabla1 X Tabla 2			
campo 1	campo 2	campo 11	campo 22
a1	b1	a4	b4
a1	b1	a5	b5
a2	b2	a4	b4
a2	b2	a5	b5
a3	b3	a4	b4
a3	b3	a5	b5

UNION, INTERSECT Y EXCEPT

UNION

- Es usado para combinar los resultados de dos o mas SELECT, sin regresar valores (registros) duplicados.
- Para usar UNION, cada sentencia SELECT debe tener la misma cantidad de columnas, el mismo tipo de datos, el mismo orden, pero no debe tener la misma longitud

Sintaxis:

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

UNION ALL

- Es usado para combinar los resultados de dos o mas SELECT, duplicando los registros
- Las mismas reglas aplican para usar UNION ALL

Sintaxis:

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION ALL

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

INTERSECT

- Se usa para combinar dos consultas SELECT, pero solo va a regresar registros del primer SELECT que son idénticos a los registros del segundo SELECT.
- Es decir INTERSECT regresa solamente registros en común entre los dos SELECT.
- Igual se aplican las mismas reglas que UNION

Sintaxis:

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

INTERSECT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

```
SQL> SELECT  ID, NAME, AMOUNT, DATE
     FROM CUSTOMERS
     LEFT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER ID
INTERSECT
     SELECT  ID, NAME, AMOUNT, DATE
     FROM CUSTOMERS
     RIGHT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

EXCEPT

- Se usa para combinar dos consultas SELECT, solo regresa registros del primer SELECT que no se encuentren por el segundo SELECT
- Es decir EXCEPT regresa sol en el segundo SELECT
- Igual se aplican las mismas

Sintaxis:

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

EXCEPT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

ALIAS SQL

ALIAS

- Se puede renombrar una tabla o columna temporalmente asignándole otro nombre, este procedimiento se le conoce como **alias**

Sintaxis:

The basic syntax of **table** alias is as follows:

```
SELECT column1, column2....
FROM table_name AS alias_name
WHERE [condition];
```

The basic syntax of **column** alias is as follows:

```
SELECT column_name AS alias_name
FROM table_name
WHERE [condition];
```

```
SELECT C.ID, C.NAME, C.AGE, O.AMOUNT
   FROM CUSTOMERS AS C, ORDERS AS O
   WHERE  C.ID = O.CUSTOMER_ID;
```

MODIFICACIÓN DE TABLAS

ALTER TABLE

Permite modificar la estructura de una tabla existente

- Agregando
- Eliminando
- Modificando

Agregar una nueva columna

```
ALTER TABLE table_name ADD column_name datatype;
```

Quitando columna

```
ALTER TABLE table_name DROP COLUMN column_name;
```

ALTER TABLE

Cambiando el tipo de dato de la columna

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

Agregando restricción NOT NULL

```
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

Agregando restricción UNIQUE

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2..|.);
```

Agregando restricción CHECK

```
ALTER TABLE table name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);
```

TRUNCATE TABLE

- Es usado para borrar completamente los datos de una tabla existente.
- También se puede utilizar DROP TABLE para borrar completamente la tabla pero este comando elimina completamente la estructura de la tabla en la base de datos

Sintaxis:

```
TRUNCATE TABLE  table_name;
```

Funciones de uso general

Funciones

id	name	work_date	daily_typing_pages
1	John	2007-01-24	250
2	Ram	2007-05-27	220
3	Jack	2007-05-06	170
3	Jack	2007-04-06	100
4	Jill	2007-04-06	220
5	Zara	2007-06-06	300
5	Zara	2007-02-06	350

Funciones

- **COUNT:** Es una función muy utilizada para contar el numero de registros de una tabla o de una columna en especifico

Sintaxis:

```
SELECT COUNT(*) FROM employee_tbl ;
```

- **MAX:** Es una función muy utilizada para encontrar el valor máximo entre un conjunto de registros

Sintaxis:

```
SELECT MAX(daily typing pages)
> FROM employee_tbl;
```

Funciones

- **MIN:** Es una función muy utilizada para encontrar el valor mínimo entre un conjunto de registros

Sintaxis:

```
SELECT MIN(daily_typing_pages)
> FROM employee_tbl;
```

- Uno puede encontrar todos los registros con valor mínimo de cada registro con la clausula **GROUP BY**

Sintaxis:

```
SELECT id, name, work_date, MIN(daily_typing_pages)
> FROM employee_tbl GROUP BY name;
```

Funciones

- **AVG:** Es una función muy utilizada para encontrar el promedio entre un conjunto de registros

Sintaxis:

```
SELECT AVG(daily_typing_pages)
> FROM employee_tbl;
```

- Uno puede encontrar el promedio de todos los registros de la clausula **GROUP BY**

Sintaxis:

```
SELECT name, AVG(daily_typing_pages)
> FROM employee_tbl GROUP BY name;
```

Funciones

- **SUM:** Es una función muy utilizada para sumar los registros.

Sintaxis:

```
SELECT SUM(daily_typing_pages)
-> FROM employee_tbl;
```

- Uno puede encontrar el promedio de todos los registros de la clausula **GROUP BY**

Sintaxis:

```
SELECT name, SUM(daily_typing_pages)
-> FROM employee_tbl GROUP BY name;
```

Funciones

- **CONCAT:** Es una función utilizada para concatenar (unir) dos cadenas para crear una sola cadena

Sintaxis:

```
SELECT CONCAT('FIRST ', 'SECOND');
```

```
+----------------+
| FIRST SECOND   |
+----------------+
```