

SELECT

- La instrucción SELECT nos permite consultar datos de una base de datos y su sintaxis es:

```
SELECT * | { [DISTINCT] column | expression [alias], ... }  
FROM    table;
```

- * se utiliza para consultar todas las columnas de una tabla
- alias (o as) puede utilizarse para poner un nombre temporal a la columna en los resultados de la consulta.

CONSULTANDO TODAS LAS COLUMNAS DE UNA TABLA

```
SELECT *  
FROM departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

CONSULTAR COLUMNAS ESPECÍFICAS

```
SELECT department_id, location_id  
FROM departments;
```



	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

ESTRUCTURA DE SENTENCIAS SQL

- No son sensibles a mayúsculas/minúsculas
- Se pueden introducir en una o varias líneas
- Las palabras clave no se pueden dividir en líneas ni se pueden abreviar
- Las cláusulas se suelen colocar en líneas aparte para facilitar su lectura
- Se recomienda 'dar formato' las sentencias utilizando sangrías
- Las palabras claves se suelen escribir en mayúsculas y el resto en minúscula.



UTILIZANDO UN ALIAS PARA COLUMNAS

```
SELECT last_name AS name, commission_pct comm  
FROM employees;
```

	 NAME	 COMM
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)

...

```
SELECT last_name "Name" , salary*12 "Annual Salary"  
FROM employees;
```

	 Name	 Annual Salary
1	King	288000
2	Kochhar	204000
3	De Haan	204000

...

CADENAS DE LITERALES

- Un literal es un carácter, un número o una fecha que se incluyó en un SELECT (no es un nombre de una columna ni alias).
- Se imprime en cada fila devuelta.
- Los literales de fecha y caracteres deben ir entre comillas simples.

```
SELECT idProducto, nombreProducto, '2020-08-14' fecha_actual  
FROM producto;
```

ELIMINAR REGISTROS DUPLICADOS DE UNA CONSULTA

- Por defecto las consultas mostrarán todos los registros que devuelva, incluyendo valores duplicados

```
SELECT department_id  
FROM employees;
```

1

	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60

```
SELECT DISTINCT department_id  
FROM employees;
```

2

	DEPARTMENT_ID
1	(null)
2	90
3	20
4	110

...

EXPRESIONES ARITMÉTICAS

- Es posible incorporar operaciones aritméticas en una consulta. Por ejemplo para convertir a otra moneda el precio de un artículo guardado en base de datos.

Operador	Descripción
+	Sumar
-	Restar
*	Multiplicar
/	Dividir

UTILIZANDO OPERADORES ARITMÉTICOS

```
SELECT last_name, salary, salary + 300  
FROM   employees;
```

	R 2	LAST_NAME	R 2	SALARY	R 2	SALARY+300
1		King		24000		24300
2		Kochhar		17000		17300
3		De Haan		17000		17300
4		Hunold		9000		9300
5		Ernst		6000		6300
6		Lorentz		4200		4500
7		Mourgos		5800		6100
8		Rajs		3500		3800
9		Davies		3100		3400
10		Matos		2600		2900

...

PRIORIDAD DE OPERADORES

- La multiplicación y la división tienen lugar antes que la suma y la resta
- Los operadores con la misma prioridad se calculan de izquierda a derecha
- Los paréntesis se utilizan para omitir la prioridad por defecto o para aclarar la sentencia

VALORES NULOS

- Si faltan valores en una fila para una columna, entonces el valor es nulo.
- Un valor nulo no está disponible (no es lo mismo que cero).
- Si utilizamos una columna con valor nulo en una operación aritmética, el resultado es nulo.


```
SELECT salary, salary + 300  
FROM employees;
```

OPERADORES DE COMPARACIÓN

Operador	Descripción
=	Igual a
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	Distinto a
BETWEEN ...AND...	Rango de valores (ambos inclusive)
IN(conjunto)	Match con alguno de los valores
LIKE	Match con un patrón de caracteres
IS NULL	Valor nulo

UTILIZANDO OPERADORES DE COMPARACIÓN

```
SELECT last_name, salary  
FROM employees  
WHERE salary <= 3000 ;
```

	 LAST_NAME	 SALARY
1	Matos	2600
2	Vargas	2500

UTILIZANDO 'BETWEEN' PARA DEFINIR RANGOS

- BETWEEN puede utilizarse para filtrar los resultados de una consulta en función a un rango

```
SELECT last_name, salary  
FROM employees  
WHERE salary BETWEEN 2500 AND 3500 ;
```





Lower limit Upper limit

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

UTILIZANDO 'IN' PARA DETERMINAR LA PERTENENCIA A UN CONJUNTO

- Utilizamos IN para validar si un valor pertenece a una lista (o subconsulta)

```
SELECT employee_id, last_name, salary, manager_id  
FROM employees  
WHERE manager_id IN (100, 101, 201);
```

	 EMPLOYEE_ID	 LAST_NAME	 SALARY	 MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

PATRÓN DE COMPARACIÓN PARCIAL USANDO 'LIKE'

- LIKE puede utilizarse para hacer búsquedas parciales en columnas del tipo cadenas de caracteres.
- Para las búsquedas pueden utilizarse los 'comodines' (wildcards):
 - % que simboliza uno o más caracteres
 - _ que simboliza un caracter

```
SELECT    first_name  
FROM      employees  
WHERE     first_name LIKE 'S%' ;
```


MÁS SOBRE EL OPERADOR 'LIKE' UTILIZANDO COMODINES

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE '_o%' ;
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

UTILIZANDO CONDICIONES CON 'NULL'

- Si necesitamos ubicar registros donde una columna sea nula, se utiliza el operador IS NULL

```
SELECT last_name, manager_id  
FROM employees  
WHERE manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

DEFINIENDO CONDICIONES CON OPERADORES LÓGICOS

Operador	Descripción
AND	Devuelve TRUE si ambos lados de la condición son verdaderos
OR	Devuelve TRUE si al menos uno de los lados de la condición es verdadero
NOT	Devuelve TRUE si la condición es falsa

OPERADOR 'AND'

- AND necesita que ambas condiciones sean verdaderas:





```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
AND job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	149	Zlotkey	SA_MAN	10500
2	201	Hartstein	MK_MAN	13000

OPERADOR 'OR'

- OR requiere que al menos una de las dos condiciones involucradas sea verdadera:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
OR job_id LIKE '%MAN%';
```

	 EMPLOYEE_ID	 LAST_NAME	 JOB_ID	 SALARY
1	100	King	AD_PRES	24000
2	101	Kochhar	AD_VP	17000
3	102	De Haan	AD_VP	17000
4	124	Mourgos	ST_MAN	5800
5	149	Zlotkey	SA_MAN	10500
6	174	Abel	SA_REP	11000
7	201	Hartstein	MK_MAN	13000
8	205	Higgins	AC_MGR	12000

OPERADOR 'NOT'

```
SELECT last_name, job_id  
FROM employees  
WHERE job_id  
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

Nota: El Operador NOT puede utilizarse acompañando al NULL, BETWEEN, IN o LIKE

REGLAS DE PRECEDENCIA

Operador	Descripción
1	Operadores aritméticos
2	Condiciones de comparación
3	IS [NOT] NULL, LIKE, [NOT] IN
4	[NOT] BETWEEN
5	NOT
6	AND
7	OR

Utilizando paréntesis, pueden forzarse el 'salto' de estas reglas

LIMIT

- Se utiliza para limitar el número de registros que retorna una consulta a base de datos.

SELECT {fieldname(s) | *}

FROM tableName(s)

[WHERE condition]

LIMIT N;

- Por ejemplo:

SELECT * FROM members LIMIT 2;

OFF SET




- Se utiliza conjuntamente con LIMIT y permite especificar desde qué registro se retornará datos de una consulta.
- Así OFFSET indicará desde qué registros se retornará datos y LIMIT cuántos traerá.
- Esta combinación LIMIT – OFFSET puede ser muy útil cuando tenemos sistemas que consultan mucha información y debe mostrarse al cliente en forma paginada.
- Por ejemplo:

```
SELECT * FROM members LIMIT 1, 2;
```

UTILIZANDO LA CLÁUSULA 'ORDER BY'

- ORDER BY permite ordenar los resultados de una consulta:
 - ASC: Orden ascendente (es el ordenamiento por defecto)
 - DESC: Orden descendente
- ORDER BY se escribe al final del SELECT:

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date ;
```

	 LAST_NAME	 JOB_ID	 DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

...

ORDENAMIENTO

- Ordenando en forma descendente:

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date DESC ;
```

1

- Utilizando el alias de una columna para ordenar:

```
SELECT employee_id, last_name, salary*12 annsal  
FROM employees  
ORDER BY annsal ;
```

2

ORDENAMIENTO

- Utilizando la posición numérica de una columna dentro del ordenamiento:

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY 3;
```

3

- Ordenando por varias columnas:

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id, salary DESC;
```

4