

El Lenguaje SQL

Departamento de Informática y Estadística
Escuela Técnica Superior de Ingeniería Informática
www.etsii.urjc.es

1. INTRODUCCIÓN

1.1.- ¿Qué es SQL?

1.2.- Estado actual y futuro del SQL

2. SINTÁXIS BÁSICA DEL SQL-92

2.1.- Lenguaje de Definición de Datos

2.1.1.- Definición del esquema

2.1.2.- Evolución del esquema

2.2.- Lenguaje de Manipulación de Datos






2.2.1.- Actualizaciones

2.2.2.- Consultas

2.3.- Lenguaje de Control de Datos

2.3.1.- Recuperación y concurrencia

2.3.2.- Seguridad y confidencialidad

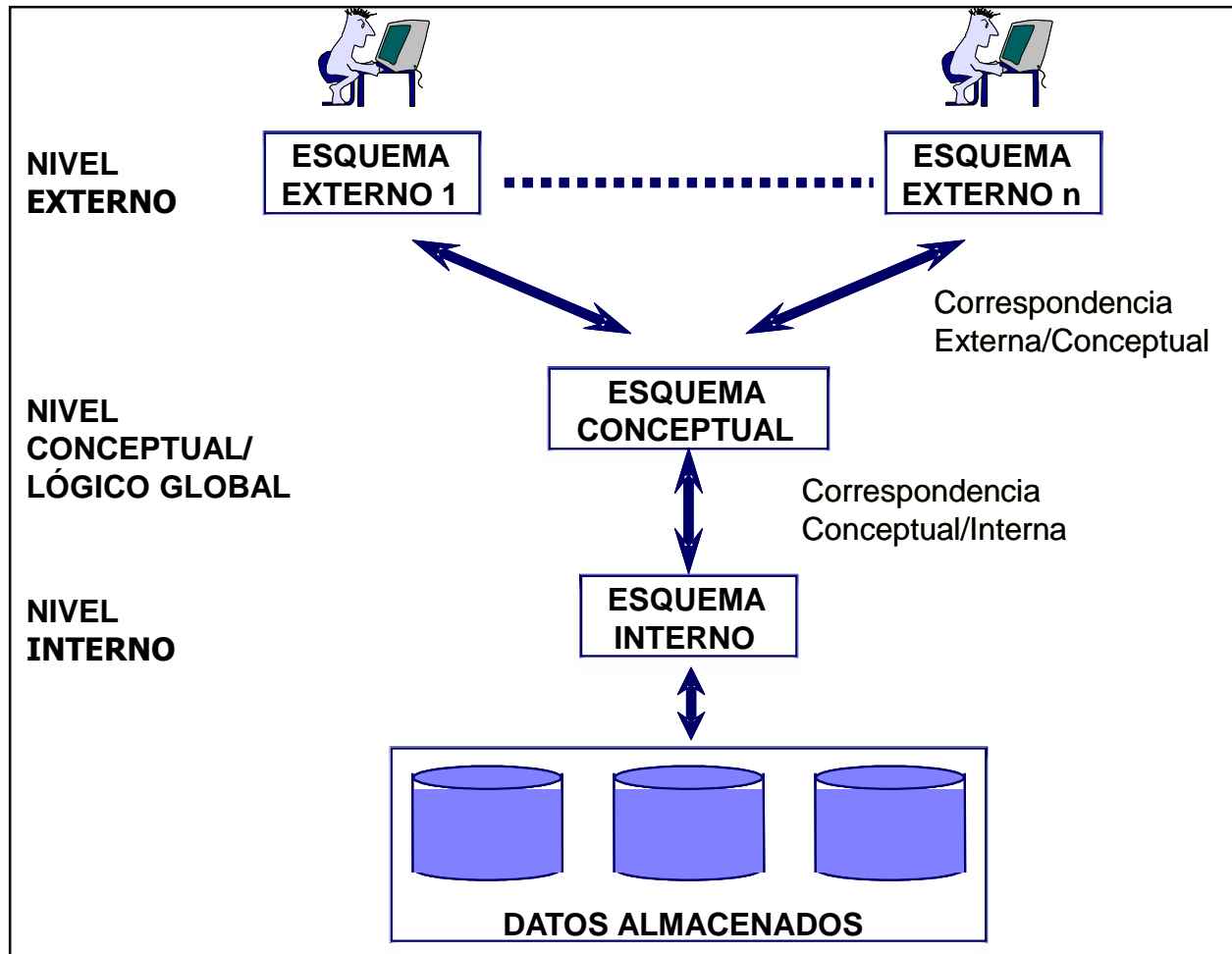
-  Tecnología y Diseño de Bases de Datos M.Piattini, E. Marcos, C.Calero y B. Vela Ed.: RA-MA, 2006, Parte II, capítulo 8
-  Fundamentos y Modelos de Bases de Datos. A. de Miguel y M. Piattini Ed.: RA-MA, 1997 Capítulo 7 (Pág. 215-265) y Apéndice A (Pág. 453-468)
-  Introducción al SQL para Usuarios y Programadores Rivero, E. et al. Ed.: Thomson, Segunda Edición, 2003
-  Diseño de Bases de Datos. Problemas Resueltos. A. de Miguel et al. RA-MA 2001
-  Documentación de Oracle
(https://docs.oracle.com/cd/E11882_01/index.htm)

■ SQL (Structured Query Language)

- Evolución de **SEQUEL** de IBM para prototipo System R
- Primer producto comercial: Oracle
 - INGRES, IBM/DS, IBM DB2...
- Lenguaje de:
 - **Definición**
 - **Manipulación**
 - Consulta **interactiva**
 - **Programación**
 - **Control**
- Características:
 - OPERA CON **CONJUNTOS DE REGISTROS**
 - LENGUAJE **RELACIONAL y DECLARATIVO**

Versión	Características
SQL-86-87	Primera formalizada por ANSI. Intersección de implementaciones existentes
SQL-89	Integridad Referencial...
SQL-92	SQL2 Mejor tratamiento de restricciones, sintaxis de dominios, tablas temporales, nuevos tipos de datos, Lenguaje de manipulación de esquema, Combinación externa, SQL dinámico...
SQL:1999	SQL3 Disparadores, Orientación al Objeto (TAD, Encapsulamiento, Jerarquías, Herencia, IDO...), OLAP
SQL:2003	XML , nuevos tipos, Sequences, Valores autogenerados...
SQL:2006	Interacción XML-SQL, Xquery,...
SQL:2008	INSTEAD OF en triggers...
SQL:2011	Bases de datos temporales...

La Arquitectura ANSI/X3/SPARC



2. Sintaxis Básica del SQL-92: Lenguaje de Definición de Datos - LDD

		Nivel Lógico Global	Nivel Externo	Nivel Físico
LENGUAJE DE DEFINICIÓN DE DATOS - LDD	Definición del Esquema (CREATE)	DOMAIN, TABLE, ASSERTION...	VIEW	INDEX
	Evolución del Esquema (ALTER, DROP)	DOMAIN, TABLE, ASSERTION...	VIEW	INDEX
LENGUAJE DE MANIPULACIÓN DE DATOS – LMD	Actualizaciones: Altas, Bajas, Modificaciones	INSERT ... DELETE... UPDATE...		
	Consultas	SELECT...		
LENGUAJE DE CONTROL DE DATOS – LCD	Recuperación y control de concurrencia	COMMIT, ROLLBACK...		
	Seguridad y Protección	GRANT, REVOKE...		

2. Sintaxis Básica del SQL-92

2.1. Lenguaje de Definición de Datos

Tipos de datos de ANSI SQL	Tipos de Datos de Oracle
CHARACTER(n), CHAR(n)	CHAR (n)
CHARACTER VARYING(n), CHAR VARYING(n)	VARCHAR2 (n)
NATIONAL CHARACTER(n), NATIONAL CHAR(n), NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n) NATIONAL CHAR VARYING(n) NCHAR VARYING(n)	NVARCHAR2(n)
NUMERIC(p,s), DECIMAL(p,s)	NUMBER (p,s)
INTEGER, INT, SMALLINT	NUMBER(p,0)
FLOAT DOUBLE PRECISION REAL	FLOAT(126) FLOAT(126) FLOAT(63)
DATE, TIME, TIMESTAMP, INTERVAL	DATE , TIMESTAMP, INTERVAL

Nomenclatura utilizada para la sintaxis de algunas sentencias del SQL-92:

Extensión de la Forma Normal de Backus (BNF), siendo:

- < > símbolos no terminales del lenguaje
- ::= operador de definición
- [] elementos opcionales
- { } elementos en una fórmula
- | alternativa
- ... repetición

LDD: Definición del esquema

Nivel lógico global

DOMINIOS: Sintaxis

<definición de dominio> ::=

```
CREATE DOMAIN <nombre de dominio> [AS] <tipo de datos>  
[ DEFAULT <opción por defecto> ]  
[ <restricción de dominio> ]
```

literal,
función de valor tiempo o
fecha, o bien USER, SYSTEM
USER o NULL

<restricción de dominio> ::=

```
[ <definición de nombre de restricción> ]  
<definición de restricción de verificación>
```

<definición de nombre de restricción> ::=

```
CONSTRAINT <nombre de restricción>
```

<definición de restricción de verificación> ::=

```
CHECK ( <condición> )
```

- **LDD: Definición del esquema**
- ***Nivel Lógico Global: Dominios***
 - CREATE DOMAIN **nomb_valido** AS CHARACTER;
 - CREATE DOMAIN **nota_valida** NUMERIC
CHECK (VALUE BETWEEN 0 AND 10);
 - CREATE DOMAIN **turno_valido** CHARACTER
DEFAULT 'M'
CONSTRAINT Mañana_o_Tarde
CHECK (VALUE IN ('M','T'));

ORACLE no admite Dominios

■ *Nivel Lógico Global: Tablas*

<definición de tabla> ::=

CREATE [TEMPORARY] TABLE <nombre de tabla>

<lista de elementos de tabla>

<lista de elementos de tabla> ::=

(<elemento de tabla> [{ , <elemento de tabla> } ...])

<elemento de tabla> ::= <definición de columna>

| <definición de restricción de tabla>

<definición de columna> ::= <nombre de columna> { <tipo de datos> | <nombre de dominio> } [<cláusula de defecto>] [<definición de restricción de columna> ...]

<definición de restricción de columna> ::=

[<definición de nombre de restricción>]

<restricción de columna> [<atributos de restricción>]

<<restricción de columna> ::=

NOT NULL | <especificación de unicidad> |
 <especificación de referencia> | <definición de restricción de verificación>

<definición de restricción de tabla> ::=

[<nombre de definición de restricción>]

<restricción de tabla> [<atributos de restricción>]

<restricción de tabla> ::=

<definición de restricción de unicidad>

| <definición de restricción referencial>

| <definición de restricción de verificación>

<definición de restricción de unicidad> ::=

<especificación de unicidad> (<lista de columnas únicas>)

<especificación de unicidad> ::=

UNIQUE | PRIMARY KEY

<lista de columnas únicas> ::= <lista de nombre de columnas>

<definición de restricción referencial> ::=

FOREIGN KEY (<columnas que ref.>)

<especificación de la referencia>

<especificación de la referencia> ::=

REFERENCES <columnas y tabla referenciadas>

[<acción referencial disparada>]

<columnas que ref.> ::= <lista de columnas de referencia>

<columnas y tabla referenciadas> ::=

<nombre de tabla> [(<lista de columnas de referencia>)]

<lista de columnas de referencia> ::= <lista de nombres de columnas>

<acción referencial disparada> ::=

<regla de modificación> [<regla de borrado>]

| <regla de borrado> [<regla de modificación>]

<regla de modificación> ::= ON UPDATE <acción referencial>

<regla de borrado> ::= ON DELETE <acción referencial>

<acción referencial> ::= CASCADE | SET NULL | SET DEFAULT

2. Sintaxis Básica del SQL-92

2.1. Lenguaje de Definición de Datos

- Más resumido:
 - CREATE TABLE nombre (
 columna tipodedatos | **dominio** [DEFAULT valorpordefecto] [restricciones de columna],
 ...
 [restricciones de tabla],
 ...);
- Restricciones de columna:
 - [CONSTRAINT nombrer restricción]
 {[NOT] NULL |
 UNIQUE/PRIMARY KEY |
 REFERENCES tabla [(columna)] [{ON DELETE | **ON UPDATE**} CASCADE | SET NULL | **SET DEFAULT**] |
 CHECK (condición) }
- Restricciones de tabla
 - [CONSTRAINT nombrer restricción]
 {UNIQUE |
 PRIMARY KEY (columna/s) |
 FOREIGN KEY (columnas)
 REFERENCES tabla [(columnas)] [{ON DELETE | **ON UPDATE**} CASCADE | SET NULL | **SET DEFAULT**] |
 CHECK (condición)}

- **LDD: Definición del esquema**
- ***Nivel lógico global***
 - Opciones de integridad referencial:
 - **ON UPDATE | ON DELETE**
 - CASCADE
 - SET NULL
 - SET DEFAULT
 - ***NO ACTION***
 - Por defecto, borrado y actualización restringidos (NO ACTION)

¡ORACLE **no** soporta
ON UPDATE ni
ON DELETE SET DEFAULT!

EJEMPLO:

ALUMNO (num_mat, nombre, ciudad, cod_grupo)

GRUPO (cod_grupo, curso, turno)

Modificación: Cascada
Borrado: Puesta a nulos

LDD: Definición del esquema

Nivel lógico global

Restricción de Tabla

Restricción de Columna

```
CREATE TABLE grupo  
(cod_grupo CHARACTER,  
curso CHARACTER NOT NULL,  
turno TURNO_VALIDO,  
PRIMARY KEY (cod_grupo),  
CHECK (curso > '0' AND curso < '4'));
```

ó

```
CREATE TABLE grupo  
(cod_grupo CHARACTER PRIMARY KEY,  
curso CHARACTER NOT NULL,  
turno TURNO_VALIDO,  
CHECK (curso > '0' AND curso < '4'));
```

```
CREATE TABLE alumno (  
    num_mat CHARACTER,  
    nombre NOMB_VALIDO UNIQUE,  
    ciudad CHARACTER NOT NULL,  
    cod_grupo CHARACTER,  
    PRIMARY KEY (num_mat),  
    FOREIGN KEY (cod_grupo) REFERENCES grupo  
    ON UPDATE CASCADE  
    ON DELETE SET NULL);
```


- En Oracle
 - CREATE TABLE grupo
(cod_grupo CHAR(3) PRIMARY KEY,
curso CHAR(1) NOT NULL,
turno CHAR(1) DEFAULT 'M'
CONSTRAINT Manana_o_Tarde CHECK (turno IN ('M','T')),
CHECK (curso > '0' AND curso < '4'));
 - CREATE TABLE alumno
(num_mat CHAR(3),
nombre VARCHAR2(20) UNIQUE,
ciudad CHAR(25) NOT NULL,
cod_grupo CHAR(3),
PRIMARY KEY (num_mat),
FOREIGN KEY (cod_grupo) REFERENCES grupo ON DELETE SET NULL);

LDD: Definición del esquema

Nivel lógico global

Esquema:

Alumno

num_mat	nombre	ciudad	<i>cod_grupo</i>

Grupo

cod_grupo	curso	turno

LDD: Definición del esquema

Nivel lógico global

Una forma de ver la estructura de una tabla en ORACLE:

⇒ **DESC[RIBE] nombre_tabla;**

```
SQL> desc alumno;
```

Nombre	¿Nulo?	Tipo
NUM_MAT	NOT NULL	CHAR(3)
NOMBRE		VARCHAR2(20)
CIUDAD	NOT NULL	CHAR(25)
COD_GRUPO		CHAR(3)

```
SQL> desc grupo;
```

Nombre	¿Nulo?	Tipo
COD_GRUPO	NOT NULL	CHAR(3)
CURSO	NOT NULL	CHAR(1)
TURN0		CHAR(1)

LDD: Definición del esquema *Nivel lógico global*

ASERCIONES:

¡ORACLE **no** soporta
la creación de aserciones!

*Todos los alumnos de Madrid tienen que estar matriculados en
el turno de tarde.*

```
CREATE ASSERTION madrid_tarde  
CHECK (NOT EXISTS
```

*(Consulta que devuelva los alumnos de Madrid
que están matriculados en el turno de tarde));*

LDD: Definición del esquema *Nivel lógico global*

ASERCIONES:

¡ORACLE **no** soporta
la creación de aserciones!

*Todos los alumnos de Madrid tienen que estar matriculados en
el turno de tarde.*

```
CREATE ASSERTION madrid_tarde  
CHECK (NOT EXISTS
```

*Alumnos de
Madrid que están
matriculados en
el turno de tarde*

```
(SELECT *  
FROM ALUMNO, GRUPO  
WHERE alumno.cod_grupo=grupo.cod_grupo  
AND alumno.ciudad='MADRID'  
AND grupo.turno='T'));
```

LDD: Definición del esquema

Nivel externo

Vistas:

```
CREATE [OR REPLACE] VIEW nombre_de_vista  
[(lista de columnas)]  
AS <cláusula SELECT>
```

```
CREATE VIEW alumnos_madrid  
AS (SELECT *  
FROM alumno  
WHERE ciudad='Madrid');
```

Todos los atributos de los
alumnos de Madrid

```
CREATE VIEW alumnos_madrid  
(nombre_alumno)  
AS (SELECT nombre  
FROM alumno  
WHERE ciudad='Madrid');
```

Nombre de los alumnos de
Madrid

LDD: Definición del esquema

Nivel físico

Índices:

```
CREATE INDEX ind_alumno  
ON alumno (ciudad, cod_grupo);
```

o

```
CREATE UNIQUE INDEX ind_alumno  
ON alumno (ciudad, cod_grupo);
```

■ LDD: Evolución del esquema

■ *Nivel Lógico Global*

■ ALTER TABLE

- Sirve para modificar una tabla:
 - » añadir (ADD), modificar (MODIFY), borrar (DROP), renombrar (RENAME) columnas, restricciones ...
 - » Habilitar (ENABLE) o deshabilitar (DISABLE) restricciones
 - » ...

■ DROP TABLE, DROP DOMAIN, DROP ASSERTION

■ *Nivel Externo*

■ DROP VIEW, ALTER VIEW

■ *Nivel Físico*

■ DROP INDEX, ALTER INDEX

- Ejemplos
- LDD: Evolución del esquema
 - Nivel Lógico Global
 - Añadir el precio (atributo) que cada alumno paga al matricularse:
 - ALTER TABLE alumno ADD (precio INTEGER);
 - Para borrar la columna sería:
 - ALTER TABLE alumno DROP COLUMN precio;
 - Otros ejemplos:
 - DROP DOMAIN turno_valido;
 - DROP TABLE grupo [CASCADE CONSTRAINTS];
 - DROP ASSERTION ciudad_turno;
 - Nivel Externo
 - DROP VIEW alumnos_madrid;
 - Nivel Físico
 - DROP INDEX ind_alumno;

LMD: Actualizaciones

Altas

**INSERT INTO <nombre_tabla> [(lista_columnas)]
VALUES (lista_columnas_inserción);**

INSERT INTO grupo VALUES ('I11', '1', **DEFAULT**);

⇒ Todas las columnas de la tabla y en el mismo orden.

INSERT INTO grupo (curso, cod_grupo) VALUES ('2','I12');

⇒ Clave primaria tiene que estar incluida.

GRUPO

cod_grupo	curso	turno
I11	1	M
I12	2	M

LMD: Actualizaciones

Altas

```
INSERT INTO GRUPO_M  
SELECT * FROM Grupo  
WHERE turno='M';
```

GRUPO_M

cod_grupo	curso	turno
I11	1	M
I12	2	M

LMD: Actualizaciones

Altas

```
INSERT INTO GRUPO_Curso1 (Turno,Codigo)  
SELECT turno, cod_grupo FROM Grupo  
WHERE curso='1';
```

GRUPO_Curso1

Turno	Codigo
M	I11

LMD: Actualizaciones

Altas

Alumno:

num_mat	nombre	ciudad	cod_grupo	precio
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I21	25000

Grupo:

cod_grupo	curso	turno
I11	1	M
I12	2	M
I13	3	M
I21	1	T
I22	2	T
I31	3	T

LMD: Actualizaciones

Bajas

**DELETE [FROM] tabla
[WHERE condición];**

**DELETE FROM grupo
WHERE curso='1';**

¿Qué ocurre en
la tabla ALUMNO?

cod_grupo	curso	turno
I12	2	M
I13	3	M
I22	2	T
I23	3	T

DELETE FROM grupo;

cod_grupo	curso	turno

LMD: Actualizaciones

Modificaciones

UPDATE tabla

SET columna1=expresión1 [, columna2=expresión2...]
[WHERE condición];

UPDATE alumno

SET cod_grupo='I22', precio=precio +50

WHERE nombre>'Ramiro';

Alumno:

num_mat	nombre	ciudad	cod_grupo	precio
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I21	25000
			I22	25050

■ LMD: Consultas

```
SELECT [ALL|DISTINCT] {lista-atributos | *}  
FROM nombre_tabla [, nombre_tabla, ...]  
[WHERE condición]  
[cláusula GROUP BY]  
[cláusula HAVING]  
[cláusula ORDER BY];
```


LMD: Consultas

De qué tabla/s se
selecciona la
información

```
SELECT *  
FROM alumno;
```

Todos los atributos de la
tabla en el orden de su
creación

⇒ *Todos* los atributos de la tabla ALUMNO (todas las tuplas).

num_mat	nombre	ciudad	cod_grupo	precio
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I22	25000

LMD: Consultas

Proyección

```
SELECT nombre, ciudad  
FROM alumno;
```

⇒ Selección de los atributos nombre y ciudad de la tabla ALUMNO.

nombre	ciudad
Juan	Madrid
Ana	Leganés
María	Leganés
Pedro	Getafe
Salomé	Madrid

2. Sintaxis Básica del SQL-92

2.2. Lenguaje de Manipulación de Datos

LMD: Consultas

ALL: valor que se
asume por omisión

DISTINCT:
Eliminación de
duplicados

```
SELECT ciudad  
FROM alumno;
```

ciudad
<i>Madrid</i>
Leganés
Leganés
Getafe
<i>Madrid</i>

¿Proyección?

```
SELECT DISTINCT ciudad  
FROM alumno;
```

ciudad
Getafe
Leganés
Madrid

¡Proyección!

π _{ciudad} Alumno

LMD: Consultas

Posición: nº de orden de la columna dentro de la lista de columnas seleccionadas (*izda a dcha*)

Por omisión: **ASC**
Valores nulos los más altos

ORDER BY {columna | posición} [DESC|ASC]
[, {columna | posición} [DESC|ASC]]...

LMD: Consultas

```
SELECT ciudad  
FROM alumno  
ORDER BY ciudad ASC;
```

ciudad
Getafe
Leganés
Leganés
Madrid
Madrid

```
SELECT ciudad  
FROM alumno  
ORDER BY nombre;
```

ciudad
Leganés
Madrid
Leganés
Getafe
Madrid

```
SELECT DISTINCT ciudad  
FROM alumno  
ORDER BY nombre;
```

LMD: Consultas

```
SELECT 1 nombre, 2 precio*0.10  
FROM alumno  
ORDER BY 2, nombre;
```

Ordenación por una
columna calculada y por
nombre

nombre	precio*0.10
PEDRO	2000
JUAN	2500
SALOME	2500
MARIA	3000
ANA	8000

LMD: Consultas

Selección

```
SELECT *  
FROM ALUMNO  
WHERE cod_grupo='I21';
```

$\sigma_{\text{cod_grupo}='I21'}$ ALUMNO

Condición: combinación de *una o más expresiones* (usando operadores lógicos) que da como resultado: CIERTO, FALSO o DESCONOCIDO

num_mat	nombre	ciudad	cod_grupo	precio
3	Ana	Leganés	I21	80000
2	Pedro	Getafe	I21	20000

LMD: Consultas

Operadores

OPERADORES DE COMPARACIÓN:

- igual ("="), distinto ("<>"), menor que ("<"), mayor que (">"), menor o igual a ("<=") y mayor o igual a (">=")
(⇒ No hay espacio entre los símbolos)

OPERADORES ARITMÉTICOS:

- Suma ("+"), resta ("-"), multiplicación ("*") y división ("/")

•OPERADORES LÓGICOS:

- AND, OR y NOT

*SELECT * FROM Alumno WHERE precio >20000 AND (ciudad = 'MADRID' OR ciudad='GETAFE');*
*SELECT * FROM Alumno WHERE precio >20000 AND ciudad = 'MADRID' OR ciudad='GETAFE';*

Primero
condiciones con
AND y luego OR

1º

2º

•VALORES NULOS (desconocido):

- IS [NOT] NULL

*SELECT * FROM Alumno WHERE ciudad IS NULL;*
*SELECT * FROM Alumno WHERE cod_grupo IS NOT NULL;*

LMD: Consultas

RANGO DE VALORES: BETWEEN ... AND

```
SELECT * FROM Alumno WHERE precio BETWEEN 20000 AND 25000;  
= SELECT * FROM Alumno WHERE precio >=20000 AND precio <= 25000;  
SELECT * FROM Alumno WHERE precio NOT BETWEEN 20000 AND 25000;  
= SELECT * FROM Alumno WHERE NOT (precio BETWEEN 20000 AND 25000);
```

OPERADOR LIKE:

Se emplea para comparar el contenido de una columna con una serie de caracteres.

Caracteres comodín:

- Subrayado (_): sustituye a un carácter en la misma posición
- Tanto por ciento (%): sustituye a n caracteres, donde n puede ser 0.

```
SELECT * FROM Alumno WHERE nombre LIKE '_E%O';
```

OPERADOR IN:

Permite comprobar si un valor pertenece a un conjunto de valores determinados.

- **expresión IN (lista de valores)**

```
SELECT * FROM Alumno WHERE ciudad IN ('BARCELONA','MADRID');  
= SELECT * FROM ALUMNO WHERE ciudad ='BARCELONA' OR ciudad='MADRID';  
SELECT * FROM Alumno WHERE ciudad NOT IN ('BARCELONA','MADRID');
```

EJEMPLOS:

```
SELECT *  
FROM Alumno  
WHERE cod_grupo LIKE 'I2%'; (o, en este caso, 'I2_')
```

num_mat	nombre	ciudad	cod_grupo	precio
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I22	25000

EJEMPLOS:

```
INSERT INTO Alumno  
VALUES (9, 'Eva', 'Fuenlabrada', NULL, 60000);
```

```
SELECT *  
FROM Alumno  
WHERE cod_grupo IS NULL; (=NULL no siempre funciona)
```

```
SELECT *  
FROM Alumno  
WHERE precio BETWEEN 15000 AND 30000;
```

2. Sintaxis Básica del SQL-92

2.2. Lenguaje de Manipulación de Datos

LMD: Consultas

Funciones de Agrupación

Cuenta líneas incluyendo NULOS	{ SELECT COUNT (*) FROM alumno WHERE precio=25000;	 COUNT(*) 2
Sin incluirlos	{ SELECT COUNT ([DISTINCT] cod_grupo) FROM alumno;	 COUNT(cod_grupo) 5 (3)
Máximo	{ SELECT MAX (precio) FROM alumno;	 MAX(precio) 80000
Mínimo	{ SELECT MIN (precio) FROM alumno;	 MIN(precio) 20000
Promedio	{ SELECT AVG (precio) * 0.1 FROM alumno;	 AVG(precio)*0.1 4000
Suma	{ SELECT SUM (precio) FROM alumno;	 SUM(precio) 240000

LMD: Actualizaciones

Altas

Alumno:

num_mat	nombre	ciudad	cod_grupo	precio
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I21	25000
9	Eva	Fuenlabrada	NULL	60000

Grupo:	cod_grupo	curso	turno
	I11	1	M
	I12	2	M
	I13	3	M
	I21	1	T
	I22	2	T
	I31	3	T

2. Sintaxis Básica del SQL-92

2.2. Lenguaje de Manipulación de Datos

LMD: Consultas

Cláusulas de Agrupación

SELECT ciudad, AVG(precio) FROM alumno GROUP BY ciudad;	ciudad	AVG(precio)
	Fuenlabrada	60000
	Getafe	20000
	Madrid	25000
	Leganés	55000

SELECT ciudad, AVG(precio) FROM alumno WHERE num_mat<5 GROUP BY ciudad;	ciudad	AVG(precio)
	Getafe	20000
	Madrid	25000
	Leganés	80000

SELECT ciudad, AVG(precio) FROM alumno WHERE num_mat<5 GROUP BY ciudad HAVING AVG(precio)>20000;	ciudad	AVG(precio)
	Madrid	25000
	Leganés	80000

LMD: Consultas

ORDEN DE EJECUCIÓN:

- 1º) FROM
- 2º) WHERE
- 3º) GROUP BY
- 4º) HAVING
- 5º) SELECT
- 6º) ORDER BY

WHERE: la condición se aplica a **todas las filas** de la tabla

HAVING: es una condición del **grupo**

2. Sintaxis Básica del SQL-92

2.2. Lenguaje de Manipulación de Datos

LMD: Consultas

Alumno

NUM_MAT	NOMBRE	CIUDAD	COD_GRUPO	PRECIO
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I21	25000
9	Eva	Fuenlabrada		60000

Empleado

DNI	NEMP	CIUDAD
12345	Jesús García	Barcelona
12346	Pepe Pérez	Zaragoza
12347	Rosa Gómez	Toledo
12348	Juan	Madrid

Diferencia

```
SELECT Nombre, Ciudad FROM Alumno  
EXCEPT  
SELECT NEMP, Ciudad FROM Empleado;
```

NOMBRE	CIUDAD
Ana	Leganés
Eva	Fuenlabrada
María	Leganés
Pedro	Getafe
Salomé	Madrid

¡En ORACLE es
Minus!

2. Sintaxis Básica del SQL-92

2.2. Lenguaje de Manipulación de Datos

Unión

```
SELECT Nombre, Ciudad FROM Alumno  
UNION [ALL]  
SELECT NEMP, CIUDAD FROM Empleado;
```

NOMBRE	CIUDAD
Ana	Leganés
Eva	Fuenlabrada
JesúsGarcía	Barcelona
Juan	Madrid
María	Leganés
Pedro	Getafe
PepePérez	Zaragoza
RosaGómez	Toledo
Salomé	Madrid

Intersección

```
SELECT Nombre, Ciudad FROM Alumno  
INTERSECT  
SELECT NEMP, CIUDAD FROM Empleado;
```

NOMBRE	CIUDAD
Juan	Madrid

LMD: Consultas Producto Cartesiano

SQL-89

```
SELECT Alumno.*, Grupo.*
FROM Alumno, Grupo;
```

SQL-92

```
SELECT *
FROM Alumno CROSS JOIN Grupo;
```

```
SELECT * FROM Alumno, Grupo;
```

NUM_MAT	NOMBRE	CIUDAD	COD_GRUPO	PRECIO	COD_GRUPO_1	CURSO	TURN0
1	Juan	Madrid	I11	25000	I11	1	M
1	Juan	Madrid	I11	25000	I12	2	M
1	Juan	Madrid	I11	25000	I13	3	M
1	Juan	Madrid	I11	25000	I21	1	T
1	Juan	Madrid	I11	25000	I22	2	T
1	Juan	Madrid	I11	25000	I31	3	T
3	Ana	Leganés	I21	80000	I11	1	M
...
9	Eva	Fuenlabrada		60000	I22	2	T
9	Eva	Fuenlabrada		60000	I31	3	T

LMD: Consultas

Combinación

```
SELECT alumno.nombre, alumno.ciudad, grupo.curso  
FROM alumno JOIN grupo  
ON alumno.cod_grupo=grupo.cod_grupo;
```

```
SELECT alumno.nombre, alumno.ciudad, grupo.curso  
FROM alumno, grupo  
WHERE (alumno.cod_grupo=grupo.cod_grupo);
```

NOMBRE	CIUDAD	CURSO
Juan	Madrid	1
Salomé	Madrid	1
Pedro	Getafe	1
Ana	Leganés	1
María	Leganés	2

LMD: Consultas

Combinación Natural

SQL-89

```
SELECT Alumno.*, Grupo.*
FROM Alumno, Grupo
WHERE Alumno.Cod_Grupo=Grupo.Cod_Grupo;
```

En ORACLE no se pueden
especificar columnas
en un NATURAL JOIN

SQL-92

```
SELECT *
FROM Alumno JOIN Grupo
ON Alumno.Cod_Grupo=Grupo.Cod_Grupo;
```

Si las dos columnas que representan código de grupo se llaman igual:

```
SELECT * from Alumno NATURAL JOIN Grupo;
```

COD_GRUPO	NUM_MAT	NOMBRE	CIUDAD	PRECIO	CURSO	TURNOS
I11	1	Juan	Madrid	25000	1	M
I21	5	Salomé	Madrid	25000	1	T
I21	2	Pedro	Getafe	20000	1	T
I21	3	Ana	Leganés	80000	1	T
I22	8	María	Leganés	30000	2	T

LMD: Consultas

Combinación

```
SELECT alumno.nombre, alumno.ciudad, grupo.curso  
FROM alumno NATURAL JOIN grupo  
WHERE turno='M';
```

```
SELECT nombre, ciudad, curso  
FROM alumno A, grupo G  
WHERE (A.cod_grupo=G.cod_grupo)  
AND (G.turno='M')
```

ALIAS



nombre	ciudad	curso
Juan	Madrid	1

2. Sintaxis Básica del SQL-92

2.2. Lenguaje de Manipulación de Datos

Alumno

NUM_MAT	NOMBRE	CIUDAD	COD_GRUPO	PRECIO
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
8	María	Leganés	I22	30000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I21	25000
9	Eva	Fuenlabrada		60000

Grupo

COD_GRUPO_1	CURSO	TURNO
I11	1	M
I12	2	M
I13	3	M
I21	1	T
I22	2	T
I31	3	T

LMD: Consultas

Combinación

```
SELECT G.curso, G.cod_grupo, A.nombre  
FROM alumno A, grupo G  
WHERE (A.cod_grupo=G.cod_grupo);
```

CURSO	COR_GRUPO	NOMBRE
1	I11	Juan
1	I21	Salomé
1	I21	Pedro
1	I21	Ana
2	I22	María

“Queremos obtener los grupos y los alumnos matriculados en los mismos.”

¿ I12, I13, I31? ¿Eva?
⇒ Se pierden.

LMD: Consultas

Combinación Externa

```
SELECT G.curso, G.cod_grupo, A.nombre  
FROM alumno A RIGHT OUTER JOIN grupo G  
ON A.cod_grupo(+ )= G.cod_grupo;
```

```
SELECT G.curso, G.cod_grupo, A.nombre  
FROM alumno A, grupo G  
WHERE (A.cod_grupo(+ )= G.cod_grupo);
```

CURSO	COD_GRUPO	NOMBRE
1	I11	Juan
1	I21	Ana
2	I22	María
1	I21	Pedro
1	I21	Salomé
3	I31	
2	I12	
3	I13	

*Combinación
externa
derecha*

LMD: Consultas

Combinación Externa Izquierda

```
SELECT A.nombre, G.cod_grupo  
FROM Alumno A LEFT OUTER JOIN Grupo G  
ON (A.cod_grupo= G.cod_grupo);
```

NOMBRE	COD_GRUPO
Juan	I11
Ana	I21
María	I22
Pedro	I21
Salomé	I21
Eva	

LMD: Consultas

Combinación Externa Completa

```
SELECT *  
FROM Alumno FULL OUTER JOIN Grupo  
ON Alumno.Cod_Grupo=Grupo.Cod_Grupo;
```

NUM_MAT	NOMBRE	CIUDAD	COD_GRUPO	PRECIO	COD_GRUPO_1	CURSO	TURNOS
1	Juan	Madrid	I11	25000	I11	1	M
5	Salomé	Madrid	I21	25000	I21	1	T
2	Pedro	Getafe	I21	20000	I21	1	T
3	Ana	Leganés	I21	80000	I21	1	T
8	María	Leganés	I22	30000	I22	2	T
9	Eva	Fuenlabrada		60000			
					I31	3	T
					I12	2	M
					I13	3	M

LMD: Consultas

Consultas Anidadas

```
SELECT *  
FROM alumno  
WHERE cod_grupo IN  
      (SELECT cod_grupo  
       FROM grupo  
       WHERE curso='1');
```

NUM_MAT	NOMBRE	CIUDAD	COD_GRUPO	PRECIO
1	Juan	Madrid	I11	25000
3	Ana	Leganés	I21	80000
2	Pedro	Getafe	I21	20000
5	Salomé	Madrid	I21	25000

LMD: Consultas

PREDICADO ALL:

ALL significa que la comparación de la expresión con el resultado de la subconsulta será CIERTA si lo es para todos los valores devueltos por la subconsulta; es decir, la condición se satisface cuando la comparación es **CIERTA para TODOS los valores devueltos por la subconsulta**.

⇒ *Obtener los alumnos que paguen menos matrícula que todos los demás alumnos.*

*SELECT * FROM Alumno a WHERE precio < ALL*

(SELECT precio FROM ALUMNO aa WHERE a.num-mat <> aa.num-mat);

PREDICADOS ANY o SOME (sinónimos):

ANY significa que la comparación de la expresión con el resultado de la subconsulta será CIERTA si lo es para uno de los valores devueltos por la subconsulta; es decir, la condición se satisface cuando la comparación es **CIERTA para AL MENOS UNO de los valores devueltos por la subconsulta**.

⇒ *Obtener los alumnos que paguen más que alguno de los alumnos de MADRID.*

*SELECT * FROM Alumno WHERE precio > ANY*

(SELECT precio FROM ALUMNO WHERE ciudad='MADRID');

PREDICADO EXISTS:

La condición EXISTS será CIERTA si la subconsulta devuelve una fila que satisfaga las condiciones impuestas en la cláusula WHERE (se puede expresar con el operador IN).

⇒ *Obtener los alumnos que estén matriculados en algún grupo.*

*SELECT * FROM Alumno a WHERE EXISTS*

*(SELECT * FROM GRUPO g WHERE a.cod_grupo=g.cod_grupo);*

LMD: Consultas

Ejemplo de Vistas

```
SELECT *  
FROM alumnos_madrid;
```

```
CREATE VIEW alumnos_madrid  
AS (SELECT *  
    FROM alumno  
    WHERE ciudad='Madrid');
```

Alumnos_madrid

num_mat	nombre	ciudad	cod_grupo	precio
1	Juan	Madrid	I11	25000
5	Salomé	Madrid	I22	25000

Lenguaje de Control de Datos

Recuperación y control de concurrencia	COMMIT ROLLBACK
Seguridad y protección	GRANT REVOKE