

UNIDAD 7

Lenguaje SQL (Structured Query Language)

Introducción

2

Características del lenguaje

- SQL constituye un factor fundamental en el éxito de las bases de datos relacionales
- Es un lenguaje declarativo (no procedural), es decir, se indica “que” se busca pero no el “como”
- Resultado del esfuerzo conjunto de dos grupos:
 - ANSI – American National Standards Institute
 - ISO – International Standards Organisation

SQL

3

Historia

- 1975: Prototipo creado por IBM, conocido como SEQUEL.
- 1977: Cambia el nombre a SQL (Structured Query Language).
- 1979: Primer SGBDR comercial basado en el (ORACLE).
- 1986: Aprobada la norma SQL/ANSI.
- 1987: Primer estándar internacional de ISO-9075.
- 1989: Nueva versión ISO SQL-89 que añade integridad referencial básica.
- 1992: Versión SQL-92 que amplía notablemente la anterior:
 - ▣ Nuevos operadores relacionales: OUTER JOIN y JOIN
 - ▣ SQL dinámico
 - ▣ El parámetro SQLSTATE para gestión de errores
 - ▣ Cursores de desplazamiento (scroll cursor)
 - ▣ Modo de acceso (lectura o lectura/escritura) y nivel de aislamiento de las transacciones.
 - ▣ Definir dominios (CREATE DOMAIN).

Historia

- 1999: Versión SQL-99 que incorpora aspectos de orientación a objetos y amplía notablemente el lenguaje. Ya está incorporado a los principales SGBD-OR
 - ▣ Nuevos tipos de datos: LOB, BOOLEAN, ROW, ARRAY, DISTINCT.
 - ▣ Posibilidad de definir nuevos tipos de datos por parte del usuario.
 - ▣ Disparadores (triggers), vistas actualizables
 - ▣ Cursores (punteros) sensitivos. Consultas recursivas.
 - ▣ Definición de roles de usuario
 - ▣ Incorporación de las características de orientación a objetos: tipos de datos abstractos, generalización, herencia y polimorfismo.

SQL

5

Historia

□ 2003: Versión SQL:2003 - ISO/IEC 9075-n:2003

- Framework: Introducción con el marco de trabajo conceptual general.
- Foundation: Define las estructuras de datos y operaciones para trabajar con BD relacionales y especifica la semántica y sintaxis del lenguaje.
- Call-Level Interface(SQL/CLI): SQL embebido, uso de órdenes SQL desde otros lenguajes de programación generales.
- Persistent Stored Modules(SQL/PSM): Procedimientos almacenados.
- Host Language Bindings(SQL/Bindings): entre otros, incluye “On-Line Analytical Processing” (SQL/OLAP).
- Management of External Data(SQL/MED): gestión de datos externos a una BD relacional mediante el uso de “wrappers” y “datalinks”.
- Object Language Bindings (SQL/OLB): SQL embebido en Java (SQLJ).
- Information and Definition Schemas (SQL/Schemata): Establece las estructuras y contenido del DEFINITION_SCHEMA, es decir, los metadatos internos.
- SQL Routines and Types Using the Java TM Programming Language (SQL/JRT): utilizar métodos y clases Java como si fuesen rutinas-SQL y tipos estructurados.
- XML-Related Specifications (SQL/XML): Maneras de utilizar SQL en conjunción con XML.

SQL

6

Evolución

- ❑ SQL 1: Introducido por primera vez en un producto comercial por Oracle en 1979.
- ❑ SQL 2
- ❑ SQL 3
(extiende SQL 2 con conceptos de OO)

SQL

7

Tipos de Operaciones que permite especificar

1. *Definición de Datos (DDL)*
2. *Manipulación de Datos (DML): Consultas y Actualizaciones*
3. Definición de vistas
4. Definición de restricciones de integridad
5. Especificación de aspectos de seguridad y autorización
6. Especificación de control de transacciones
7. Reglas para inclusión en lenguajes (C, PASCAL, etc.)

SQL: DDL y DML

8

DDL

- Incluye sentencias que permiten definir los objetos de la Base de Datos
- Las sentencias DDL generan cambios en el diccionario de datos, el cual contiene los metadatos

DML

- Es un lenguaje que permite a los usuarios acceder o manipular los datos de la base de datos

SQL

9

DDL

- ☐ Create
- ☐ Alter
- ☐ Drop

DML

- ☐ Select
- ☐ Insert
- ☐ Update
- ☐ Delete

1 - DDL

10

CREATE
DOMAIN

```
CREATE DOMAIN nombre_dominio AS  
tipo_dato [DEFAULT valor];
```


1 – DDL (Integridad Referencial)

12

Integridad referencial

- `<definición de restricción referencial> ::=`
`FOREIGN KEY (<lista columnas referenciantes>) <esp.de ref>`
 - `<esp. de ref> ::= REFERENCES <nombre tabla referenciada>`
`[(<colum.ref.>)] [<acc. ref>]`
 - `<acc.ref> ::=`
`{ <regla de modificación> [<regla de borrado>]`
`| <regla de borrado> [<regla de modificación>] }`
 - `<regla de modificación> ::=`
`ON UPDATE { CASCADE | SET NULL | SET DEFAULT | RESTRICT | NO`
`ACTION }`
 - `<regla de borrado> ::=`
`ON DELETE { CASCADE | SET NULL | SET DEFAULT | RESTRICT | NO`
`ACTION }`

Tipos de Datos

13

Tipos de
Datos
Predefinidos

- ☐ Numéricos
- ☐ Caracteres
- ☐ Fecha
- ☐ Objetos Binarios

Tipos de Datos

14

Tipos de Datos Numéricos

- `NUMERIC(p,s)`
- `DECIMAL(p,s)`
- `INTEGER` (alias: `INT`)
- `SMALLINT` small integers
- `FLOAT(p,s)`
- `REAL` (for short floats)
- `DOUBLE` (for long floats)

En Oracle:

- `NUMBER(t,d)`: Para almacenar valores enteros o decimales, positivos o negativos. Su rango va de 1.0×10^{-130} hasta 9.999...(38 nueves). El parámetro "t" indica el número total de dígitos (contando los decimales) que contendrá el número como máximo (es la precisión). Su rango va de 1 a 38. El parámetro "d" indica el máximo de dígitos decimales (escala). La escala puede ir de -84 a 127. Para definir número enteros, se puede omitir el parámetro "d" o colocar un 0. Un atributo definido "number(5,2)" puede contener cualquier número entre -999.99 y 999.99.

Tipos de Datos

15

Tipos de
Datos
Caracter

- ❑ CHARACTER(n) (fixed length)
- ❑ CHARACTER (variable length)
- ❑ CHARACTER VARYING(n) (alias: VARCHAR(n))
- ❑ CLOB (Character Large Object, e.g., for large text),
- ❑ NCLOB (National CLOB)

En SGBDs:

- ❑ Oracle: VARCHAR2(size), CHAR(size), CLOB, RAW, LONG RAW
- ❑ PostgreSQL: CHARACTER(size), CHAR(size), VARYING(size), VARCHAR(size), TEXT
- ❑ MySQL: CHAR(M), VARCHAR(M), TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT

Tipos de Datos

16

Tipos de Datos Fecha

- **DATE** ej. DATE '1993-01-02'
- **TIME** ej. TIME '13:14:15'
- **TIMESTAMP** ej. TIMESTAMP '1993-01-02 13:14:15.000001'
- **INTERVAL FirstUnitofTime [to LastUnitofTime]** ej. INTERVAL '01-01' YEAR TO MONTH

En SGBDs:

- Oracle: DATE, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE
- PostgreSQL: DATE, TIME, TIME WITH TIMEZONE, TIMESTAMP, INTERVAL
- MySQL: DATE, DATETIME, TIMESTAMP[(M)], TIME, YEAR[(2 | 4)]

Tipos de Datos

17

Tipos de Datos Binarios

- ❑ BIT[(n)] ej. '01000100'
- ❑ BLOB[(n)] ej. X'49FE' (Binary Large Objects, por ejemplo para multimedia)

Otros:

- ❑ BOOLEAN (true, false or unknown)

En SGBDs:

- ❑ Oracle: BLOB, BFILE, RAW, LONG RAW, ROWID
- ❑ PostgreSQL: TEXT, BYTEA, or in large object
- ❑ MySQL: TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB

1 - DDL

18

DROP TABLE

- ❑ DROP TABLE nombre_tabla [RESTRICT/CASCADE];
 - ▣ RESTRICT: Elimina la tabla solo si no esta referenciada.
 - ▣ CASCADE: Elimina la tabla y la restriccion.

(ejemploDDL.sql)

1 - DDL

19

ALTER TABLE

- ALTER TABLE nombre_tabla
ADD nombre_columna
tipo_dato/dominio
[DEFAULT valor];
- ALTER TABLE nombre_tabla
DROP nombre_columna;

2 - DML

20

Lenguaje de
Manipulación
del SQL

- DML implementa (al menos) las Operaciones del Algebra Relacional
- Funcionalidad

DML “=” Algebra Relacional



Multiconjunto o
Bolsa



Conjunto



Provoca diferencias importantes entre ambos

2 - DML

21

SELECT

(Forma
general)

```
SELECT lista_atributos
FROM nombre_tabla/s
[WHERE condición]
[GROUP BY lista_atributos
  [HAVING condición]]
[ORDER BY lista_atributos];
```

Donde la/s tablas especificadas en la clausula FROM pueden ser:

- Tablas permanentes (es decir, tablas base creadas por la sentencia *create table*)
- Tablas temporales (es decir, filas devueltas por un *suquery*)
- Tablas virtuales (es decir, creadas por un *create view*)

2 - DML: Tablas Ejemplo

22

Tabla DEPT

Tabla DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 filas

2 - DML: Tablas Ejemplo

23

Tabla
DEPT

Tabla EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	1400	30
7521	WARD	SALESMAN	7698	22/02/81	1250		30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1250		30
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	ANALYST	7839	09/06/81	2450	0	10
7788	SCOTT	PRESIDENT	7566	19/04/87	3000		20
7839	KING	SALESMAN		17/11/81	5000		10
7844	TURNER	CLERK	7698	08/09/81	1500		30
7876	ADAMS	CLERK	7788	23/05/87	1100		20
7900	JAMES	ANALYST	7698	03/12/81	950		30
7902	FORD	CLERK	7566	03/12/81	3000		20
7934	MILLER		7782	23/01/82	1300		10

14 filas

2 - DML: Proyección (Atributos)

24

Algebra

$\pi_{\text{epno, ename}} \text{ EMP}$

SQL

```
SELECT epno, ename  
FROM EMP;
```

(proyección1.sql)

2 - DML: Proyección (Atributos)

25

Algebra

$\pi_{\text{job}} \text{ EMP}$

SQL

```
SELECT job  
FROM EMP;
```

```
SELECT distinct job  
FROM EMP;
```

(proyección2 y 3.sql)

2 - DML: Proyección (Todos los atributos)

26

Algebra

EMP

SQL

```
SELECT *  
FROM EMP;
```

Proyecta todos los
atributos de la tabla

([proyección4.sql](#))

2 - DML: Proyección (Atributo renombrado)

27

Algebra

$\rho_{\text{nombre}} (\pi_{\text{ename}} \text{EMP})$

SQL

```
SELECT ename AS  
       nombre  
FROM EMP;
```

(proyección5.sql)

2- DML: Proyección

28

Proyección

A diferencia del Algebra, SQL permite proyectar:

- Atributos
- **Constantes**
- **Expresiones**
- **Funciones**

2 - DML: Proyección

29

Expresiones y
Constantes

```
SELECT ename as nombre,  
       sal/2 as salario_quincena  
FROM EMP;
```

(proyección5.sql)

```
SELECT ename as nombre,  
       ' cobra quincenalmente ',  
       sal/2 as salario_quincena,  
       ' pesos'  
FROM EMP;
```

(proyección5.sql)

2 - DML: Proyección

30

Funciones

- ❑ `SELECT substr(ename,1,3)`
`FROM emp;`
- ❑ `SELECT concat(ename, job)`
`FROM emp;`
- ❑ `SELECT count(*)`
`FROM emp;`

Funciones sobre
cadenas de
caracteres en
Oracle

(proyección5.sql)

2 - DML: Proyección con Funciones

31

Funciones de Agregacion

- **COUNT**
 - ▣ `SELECT Count(nombre_atributo/distinct nombre_atributo/*)`
`FROM nombre_tabla;`
- **SUM**
 - ▣ `SELECT Sum(nombre_atributo)`
`FROM nombre_tabla;`
- **AVG**
 - ▣ `SELECT Avg(nombre_atributo)`
`FROM nombre_tabla;`
- **MAX (MIN)**
 - ▣ `SELECT Max(nombre_atributo)`
`FROM nombre_tabla;`

NOTA: Todas estas funciones se llaman funciones de RESUMEN o de AGREGACION

2 - DML: Restricciones de las Funciones de Agregación

32

Funciones de Agregación

- No se pueden componer funciones agregadas, por ej. `max(avg())`
- Para toda consulta que utiliza alguna función de agregación, cada elemento de la lista del SELECT debe ser:
 - ▣ Una función agregada, o
 - ▣ Un atributo presente en la lista de atributos del GROUP BY ([Se vera mas adelante](#)).
- Los valores NULL son ignorados por cualquier función de agregación con excepción de `count(*)`.
- Los valores NULL son tratados como un valor convencional en los agrupamientos por atributos ([Se vera mas adelante](#)).

2 - DML: Alias

33

Alias

```
SELECT lista_atributos  
FROM nombre_tabla nombre_alias  
....;
```

(alias.sql)

2 - DML: Selección

34

Algebra

$\pi_{\text{empno,ename}}(\sigma_{\text{deptno}=20} \text{ EMP})$

SQL

```
SELECT empno, ename  
FROM EMP  
WHERE DEPTNO=20;
```

([seleccion1.sql](#))

2 - DML: Selección

35

Condiciones
compuestas

```
SELECT empno, ename  
FROM EMP  
WHERE DEPTNO=20 and SAL<1000;
```

La condición puede utilizar cualquiera de los operadores relacionales y operadores lógicos para generar expresiones más complejas (para cualquier tipo de dato)

2- DML: Selección

36

Otros Operadores de comparación

Ademas de los operadores utilizados en el Algebra (<, >, =, and, etc.) SQL provee los siguientes:

1. Operadores de valores nulos
 - ▣ <columna> IS [NOT] NULL
2. Operador LIKE
 - ▣ <columna> LIKE <patron>
3. Operadores de condiciones de dominios
 - ▣ <columna> [NOT] BETWEEN <valor inferior> AND <valor superior>
4. Operadores de comparación de conjuntos
 - ▣ <columna> [NOT] IN (<lista de valores>)

2 - DML: Selección (Nulos)

37

Condiciones con nulos

1. Operadores de valores nulos

El valor NULL no es una constante, por lo tanto, no se puede utilizar explícitamente como un operando.

```
SELECT empno, ename, comm
FROM EMP
WHERE comm is null;
```

El valor nulo no es igual ni distinto a ningun valor

```
SELECT empno, ename, comm
FROM EMP
WHERE comm <> 300;
```

([nulos.sql](#))

2 - DML: Selección (Nulos)

38

Condiciones
con nulos

Los valores nulos pueden representar semanticamente:

- **Valor desconocido:** No disponible por el momento. Por ej. Si se desconoce la fecha de nacimiento.
- **Valor inaplicable:** Ningún valor tiene sentido. Ej. Un atributo “ nombre cónyuge ” para un soltero/a.
- **Valor retenido:** No se dispone de la suficiente jerarquía para conocer el valor. Por ej. Un número de teléfono, pin, etc.

2 - DML: Selección (Nulos)

39

Condiciones
con nulos

Operando con valores nulos:

1. Operadores aritmeticos (+,*,etc.): Cuando al menos uno de los operandos es null, el resultado de la operación es **null**.
2. Operadores relacionales (<,>,,etc.): Cuando al menos uno de los operandos es null, el resultado de la operación es **unknown (desconocido)**. El valor **DESCONOCIDO** es otro valor de verdad.

Por ejemplo, si x tiene valor NULL:

- $x + 3$ es NULL
- $NULL + 3$ no es válido (No se puede usar como una constante)
- $x = 3$ es UNKNOWN
- $NULL = 3$ no es válido (No se puede usar como una constante)

2 - Valor de Verdad DESCONOCIDO

Dado que las comparaciones expresadas en la clausula WHERE pueden ser combinadas usando los operadores logicos AND, OR y NOT...

Analicemos el resultado que se obtiene cuando alguno/s de los comparaciones devuelve el valor de verdad DESCONOCIDO (UNKNOWN)

2 – Operadores Logicos con el Valor de Verdad DESCONOCIDO

Tabla de Verdad para el Operador AND

A	B	A and B
V	V	V
V	F	F
V	D	D
F	∀	
F	F	F
F	D	F
D	∀	
D	F	
D	D	D

Analizando el resultado de A and B:

V -> Los dos son Verdaderos

F -> Alguno de los dos son Falsos

D -> Ninguno es FALSO, pero alguno es

DESCONOCIDO



2 – Operadores Logicos con el Valor de Verdad DESCONOCIDO

Tabla de Verdad para el Operador AND

A	B	A and B
V	V	V
V	F	F
V	D	D
F	∀	
F	F	F
F	D	F
D	∀	
D	F	
D	D	D

Analicemos nuevamente el resultado de A and B, dandole los siguientes valores numericos a los valores de verdad:

V = 1

F = 0

D = 0,5

Entonces...

$A \text{ and } B = \min\{A, B\}$



2 – Operadores Logicos con el Valor de Verdad DESCONOCIDO

Tabla de Verdad para el Operador OR y NOT=

A	B	A or B	Not A
V	V	V	F
V	F	V	
V	D	V	
F	∀		
F	F	F	V
F	D	D	
D	∀		
D	F		
D	D	D	D

$$A \text{ or } B = \max\{A, B\}$$

$$\text{Not } A = 1 - A$$



2 - DML: Selección

44

Condiciones
con operador
el "like "

2. Operador LIKE

```
SELECT *  
FROM scott.emp  
WHERE ename LIKE 'A%';
```

(selección2_1.sql)

El operador like compara parte de una cadena de caracteres, donde:

- % sustituye a una cantidad arbitraria de caracteres
- _ sustituye a un solo carácter

2 - DML: Selección

45

Condiciones
con operador
de rango

3. Operador de rango

```
SELECT *
```

```
FROM scott.emp
```

```
WHERE salary BETWEEN 1000 and 2000;  
(selección2_1.sql)
```

2 - DML: Selección

46

Condiciones
comparacion
con conjuntos

4. Operadores de comparación con conjuntos

```
Select *  
FROM scott.emp  
WHERE deptno IN (20,30);
```

2 - DML: Producto Cartesiano

47

Algebra

$EMP \times DEPT$

SQL

```
SELECT *  
FROM EMP , DEPT;
```

([producto.sql](#))

2 – DML: Join

48

join

[join] :: =

nom_tabla1 NATURAL JOIN nom_tabla2 /

nom_tabla1 JOIN nom_tabla2 ON condición /

nom_tabla1 INNER JOIN nom_tabla2 ON condición /

nom_tabla1 LEFT [OUTER] JOIN nom_tabla2 ON
condición /

nom_tabla1 RIGHT [OUTER] JOIN nom_tabla2 ON
condición

2 - DML: Natural Join

49

Algebra

$\pi_{ename, job, dname} (EMP * DEPT)$

SQL

```
SELECT ename, job, dname  
FROM EMP NATURAL JOIN  
DEPT;
```

([join2.sql](#))

(Los atributos sobre los cuales se hace el JOIN deben tener el mismo nombre en ambas tablas)

2 - DML: Natural Join

50

Natural Join
construido con
producto
cartesiano
Algebra

```
SELECT epno,ename, job, mgr, hiredate,  
       sal, comm, deptno, dname, loc  
FROM   EMP, DEPT  
WHERE  EMP.deptno= DEPT.deptno;
```

(join1.sql)

2 - DML: Join

51

Algebra

$\pi_{ename, job, dname}$
 $(EMP \bowtie_{EMP.deptno = DEPT.deptno} DEPT)$

SQL

- `SELECT ename, job, dname
FROM EMP JOIN DEPT ON
EMP.deptno = DEPT.deptno;
(join3.sql)`
- `SELECT ename, job, dname
FROM EMP INNER JOIN
DEPT ON EMP.deptno =
EPT.deptno;
(join3_1.sql)`

2 - DML: Join

52

Join con
restricción
extra

```
SELECT ename, job, dname  
FROM EMP JOIN DEPT ON EMP.deptno  
      = DEPT.deptno  
WHERE EMP.deptno = 10;
```

(join3_1.sql)

2 - DML: Outer Join

53

Outer Join

- ❑ *LEFT OUTER JOIN*
- ❑ *RIGHT OUTER JOIN*
- ❑ *FULL OUTER JOIN*

2 - DML: Outer Join

54

Left Outer Join

- `SELECT deptno,dname, ename, job
FROM DEPT NATURAL JOIN EMP;`

El departamento 40 no sale informado porque no existe ningún empleado asignado a él

([join5.sql](#))

- `SELECT DEPT.deptno,dname, ename, job
FROM DEPT LEFT OUTER JOIN EMP ON
DEPT.DEPTNO=EMP.DEPTNO;`

El departamento 40 sale informado a pesar de que no existen empleados asignados a él

([join6.sql](#))

2 - DML: Outer Join

55

Right Outer Join

- *SELECT deptno,dname, ename, job
FROM DEPT NATURAL JOIN EMP;*
El departamento 40 no sale informado porque no existe ningún empleado asignado a él
([join5.sql](#))
- *SELECT ename, job, dept.deptno,dname
FROM EMP RIGHT OUTER JOIN DEPT ON
DEPT.DEPTNO=EMP.DEPTNO;*
El departamento 40 sale informado a pesar de que no existen empleados asignados a él
([join7.sql](#))

2 - DML: Outer Join

56

Full Outer Join

```
SELECT lista_atributos  
FROM nombre_tabla FULL OUTER JOIN  
nombre_tabla  
ON condición;
```


2 - DML: Union

57

Algebra

- $\pi_{\text{ename}} (\sigma_{\text{deptno}=20 \text{ or } \text{deptno}=30} \text{EMP})$
- $\pi_{\text{ename}} (\sigma_{\text{deptno}=20} \text{EMP}) \cup \pi_{\text{ename}} (\sigma_{\text{deptno}=30} \text{EMP})$

SQL

- ```
SELECT ename FROM EMP
WHERE deptno=20
or deptno=30 ;
```
- ```
SELECT ename FROM EMP
WHERE deptno=20
UNION
SELECT ename
FROM EMP
WHERE deptno=30;
```

([union1.sql](#))

2 - DML: Union

58

Union

- En este caso, si existen tuplas repetidas se eliminan

Union All

- En este caso, si existen tuplas repetidas no se eliminan

2 - DML: Intersección

59

Intersección

Departamentos que tienen administrativos y analistas entre sus empleados (CLERK y ANALYST)

$$\pi \text{ deptno } (\sigma \text{ job} = \text{'CLERK'} \text{ EMP})$$
$$\cap$$
$$\pi \text{ deptno } (\sigma \text{ job} = \text{'ANALYST'} \text{ EMP})$$

2 - DML: Intersección

60

Intersección

```
Select deptno  
from emp  
where job='CLERK'  
  
INTERSECT  
  
Select deptno  
from emp  
where job='ANALYST';
```

(interseccion1.sql)
Ver
(depts_jobs.sql)

2 - DML: Minus

61

Algebra

$\pi_{\text{deptno}} \text{Dept} - \pi_{\text{deptno}} \text{EMP}$

SQL

□ `select deptno from dept
minus
select deptno from emp;`

Departamentos que no tienen
empleados

([minus1.sql](#))

2 - DML: División

62

Algebra

$\pi_{x,y}A \text{ DIVIDE BY } \pi_yB$

SQL

- La operación DIVIDE no tiene un operador equivalente en SQL, pero puede expresarse como un doble NOT EXISTS
- Se analizará mas adelante, cuando se explique el operador NOT EXISTS

```
Select A.x
```

```
from A
```

```
where not exists
```

```
(select *
```

```
from B
```

```
where not exists
```

```
(select *
```

```
from A AX
```

```
where AX.x=A.x
```

```
and AX.y=B.y));
```

63

Subconsultas

2 – DML: Subconsultas

64

Operadores
de
comparación
para consultas
más complejas

1. Operador IN/ NOT IN

- <expresión> IN (<subconsulta>)
 - ▣ En este caso <expresión> es comparada con el conjunto que devuelve la subconsulta:
 - El resultado es Verdadero si el elemento de la expresion esta dentro del conjunto devuelto por la subconsulta. Y Falso si el elemento no se encuentra en el.
 - Los atributos colocados en <expresion> deben coincidir con los atributos de la tabla que genera la subconsulta.
- <expresión> NOT IN (<subconsulta>)

2 – DML: Subconsultas

65

Operadores
de
comparación
para consultas
más complejas

2. Operador =, <, > con subconsultas

□ Ej: <expresión> = (<subconsulta>)

■ En este caso <expresión> es comparada con el VALOR que devuelve la subconsulta

2 – DML: Subconsultas

66

Operadores
de
comparación
para consultas
más complejas

2. Operador =, <, > con subconsultas: Uso de ANY y ALL

- **<expresión>** <operador de comparación> [**ANY/ALL**] (<subconsulta>)
 - <expresión> puede ser una columna o un valor computado
 - En este caso <expresión> es comparada con cada valor seleccionado por la subconsulta
 - **Any:** Evalúa verdadero si existe al menos una tupla seleccionada en el subquery que satisfaga la condición. Si el subquery da vacío, la condición es falsa.
 - **All:** Evalúa verdadero si todas las tuplas seleccionadas por el subquery satisfacen la comparación. Si el subquery es vacío, la condición es verdadera.

2 - DML: Subconsultas

67

Operadores
de
comparación
para consultas
más complejas

Mostrar todos los empleados que trabajan en el depto 10 y que ganan al menos tanto como algún empleado del departamento 30

```
SELECT *  
FROM scott.emp  
WHERE deptno=10  
and sal >= any  
  (SELECT sal  
   FROM scott.emp  
   WHERE deptno=30);
```

2 - DML: Subconsultas

68

Operadores
de
comparación
para consultas
más complejas



Mostrar todos los empleados que no trabajan en el depto 30 y que ganan más que todos los empleados que trabajan en el departamento 30

```
SELECT *  
FROM scott.emp  
WHERE deptno <> 30  
and sal >= all  
  (SELECT sal  
   FROM scott.emp  
   WHERE deptno=30);
```

2 – DML: Subconsultas

69

Equivalencias
de
operadores
para consultas
más complejas
(con
subqueries)

- IN  = ANY
- NOT IN  <> ALL

2 – DML: Subconsultas

70

Subconsultas en la clausula FROM

3. Subconsultas en la clausula FROM

Supongamos que queremos obtener el nombre del depto en el que trabaja SCOTT:

```
select DNAME
from dept, (select *
            from emp
            where ename='SCOTT') AUX
where dept.DEPTNO=AUX.DEPTNO;
```

```
select DNAME
from dept natural join (select *
                        from emp
                        where ename='SCOTT');
```

- Se puede observar en la clausula FROM una subconsulta en lugar de una tabla base.
- Dado que no tiene nombre si se necesita referenciar alguna atributo, debera colocarse un alias (en este caso, AUX).

2 – DML: Subconsultas

71

Operadores
de
comparación
para consultas
más complejas

4. Operador EXISTS/ NOT EXISTS

- where [NOT] EXISTS (<subconsulta>)
 - ▣ En este caso, se evalúa el conjunto que devuelve la subconsulta:
 - El resultado es Verdadero si el conjunto generado es no vacío. Es Falso si el conjunto es vacío.
- where NOT EXISTS (<subconsulta>)

2 - DML

72

Subconsultas

- Algunas operaciones del Algebra tambien se pueden implementar con subconsultas y operadores de conjunto

2 - DML: Intersección

73

Intersección
con subquery

```
SELECT dept.deptno, dname  
from dept  
where deptno in  
    (Select deptno  
        from emp  
        where job='CLERK')      (interseccion2.sql)  
and deptno in  
    (Select deptno  
        from emp  
        where job='ANALYST');
```

2 - DML: Minus

74

Minus con
subquery

- `select deptno from dept
where deptno not in
(select deptno
from emp);`

(minus2.sql)

2 - DML: Exists vs In

75

Exists vs. In

- ❑ Las subconsultas vinculadas con IN pueden seleccionar más de un atributo, obviamente compatible con los atributos expresados en el where.
- ❑ Todas las consultas resueltas con IN pueden ser resueltas con EXISTS
- ❑ Idem para los negados (NOT IN – NOT EXISTS)

2 - DML: División

76

Algebra

$\pi_{x,y} A$

DIVIDE BY

$\pi_y B$

SQL

*Select A.x
from A*

where not exists

*(select **

from B

where not exists

*(select **

from A AX

where AX.x=A.x

and AX.y=B.y));

2 - DML: Grupos

77

Group y
Having

SELECT

....

[GROUP BY lista_atributos

[HAVING condición]];

- Agrupa las tuplas que tienen igual valor del/de los atributos especificados en la lista_atributos.
- Muestra una tupla representativa del grupo (resume el grupo). Este valor debe ser único al grupo y además si es un atributo, debe estar especificado en la cláusula group by.

(agrup cantidad emp x depto.sql)

(agrup con join.sql)

2 - DML: Grupos

78

Group y
Having

SELECT
[GROUP BY lista_atributos
[HAVING condición]];

- Agrupa las tuplas que tienen igual valor del/ de los atributos especificados en lista_atributos.
- Muestra una tupla representativa del grupo (resume el grupo). Este valor debe ser único al grupo y además si es un atributo, debe estar especificado en la cláusula group by.
- El HAVING permite seleccionar los grupos que cumplen la condición indicada

(agrup cantidad emp x depto.sql)

(agrup con join.sql)

(agrup cantidad emp x depto con condicion.sql)

2 – DML: Ordenamiento

79

Ordenamiento

SELECT

....

[ORDER BY lista_atributos ASC/DESC];
(orden.sql)

Ordena la salida (tabla resultante)
según el/los atributos especificados

2 – DML: Actualización

80

Actualización

- ❑ Inserción de tuplas
- ❑ Modificación de tuplas
- ❑ Eliminación de tuplas

2 – DML: Inserción

81

Insert con
values

```
INSERT INTO <tabla> [<column_i, ..., column_j>]  
VALUES (<valor_i, ..., valor_j>);
```

- Para cada columna debe ser especificado un valor.
- Si no se especifica una columna de la tabla, se almacenará en ella un valor nulo.
- Si no se especifican las columnas se asumirá el orden especificado en el create.

3 – DML: Inserción

82

Insert con
subquery

```
INSERT INTO <tabla> [<column_i, ..., column_j>]  
<query>;
```

Ejemplo:

```
insert into hr.jobs  
select *  
from otrohr.jobs;
```

3 – DML: Modificación

83

Update

UPDATE <tabla>

SET <column_i>=<expr_i>, ..., <expr_j> = <expr_j>

[WHERE <condición>];

Donde:

<expr> puede ser una constante, una expresión aritmética o un query

Ejemplo:

UPDATE EMP

SET sal = (select min(sal) from EMP where
job='MANAGER')

WHERE job='SALESMAN' and deptno=20;

3 – DML: Modificación

84

Update

```
UPDATE <tabla>  
SET (<columna i, ..., columna j>) = <query>  
[WHERE <condición>];
```

3 – DML: Eliminación

85

DELETE

DELETE FROM <tabla> [WHERE <condición>];

3 - Vistas

86

Definición de Vistas

- Una vista es definida como una consulta sobre una o más relaciones, guardadas con un nombre en el diccionario de datos
- ¿Por qué usar vistas?
 - ▣ Ocultar información a algunos usuarios
 - ▣ Hacer más simples algunas consultas
 - ▣ Modularidad en el acceso a la base de datos

3 - Vistas

87

Creación de Vistas

CREATE VIEW <nombre de la vista> [<columna(s)>] AS
<sentencia select> [WITH CHECK OPTION];

- Si [<columna(s)>] no está especificado, las columnas toman el mismo nombre de los correspondientes a la tabla generada en la sentencia select.
- Una vista es evaluada cada vez que es accedida.
- La definición de una vista puede usar otra vista.
- La cláusula WITH CHECK OPTION provoca que si se intenta insertar en la vista una tupla que no satisface la condición de la vista, esa inserción es rechazada. Los Updates son similarmente rechazados si el nuevo valor no satisface la condición.

3 - Vistas

88

Uso de Vistas

- Una vista puede ser usada en una sentencia `SELECT` en forma idéntica que una tabla.
- No toda vista puede ser actualizada automáticamente (insert, update, delete).
- La dificultad de la actualización de vistas radica en que éstas deben ser traducidas a actualizaciones sobre las tablas bases sobre las que está definida la vista (*create trigger ... instead of ...*)

3 - Vistas

89

Condiciones para que una vista pueda actualizarse automáticamente

- La cláusula from debe usar una sola tabla
- La clausula select contiene sólo nombres de atributos, es decir, no tiene:
 - ▣ expresiones
 - ▣ funciones de agregación
 - ▣ especificación de distinct
- No debe incluir la cláusula group by en su definición
- No se ha incluido en la clausula select algún atributo que en la tabla base tiene la restricción de not null

4 – Restricciones de Integridad

90

Tipos

1. **NOT NULL** - Indica que la columna no puede contener valores nulos
2. **UNIQUE** - Asegura que el valor de una columna/s debe ser unico en toda la tabla
3. **PRIMARY KEY** - Asegura la Integridad de Entidad
4. **FOREIGN KEY** - Asegura la Integridad Referencial
5. **CHECK** - Asegura que se satisfaga la condicion especificada
6. **DEFAULT** - Especifica un valor por defecto

4 – Restricciones de Integridad

91

Tipos

(En cuanto a la forma de definirse)

Una restricción puede definirse sintácticamente de dos formas:

- Como parte de la definición de una columna
 - El nombre de la restricción es asignada automáticamente por el motor de base de datos.
- Como parte de la definición de la tabla.
 - El nombre de la restricción puede ser especificado por el usuario.

4 – Restricciones de Integridad

92

Comprobación

- Todas las restricciones son comprobadas con los mismos mecanismos y procedimientos:
- Cada restricción tiene una condición que se evalúa a cierto o a falso.
- Cada restricción tiene un modo (constraint mode):

- ▣ DEFERRED => diferido

Las restricciones con el modo DEFERRED son comprobadas al final de la ejecución de un grupo lógico de sentencias SQL (transacción).

- ▣ IMMEDIATE => inmediato

Las restricciones con el modo IMMEDIATE son comprobadas después de la ejecución de cada sentencia SQL.

4 – Restricciones de Integridad

93

Restricciones
de Tabla
/Columna

- Nulidad
 - ▣ Las filas de la tabla no pueden tener valores nulos

4 – Restricciones de Integridad

94

Restricciones
de Tabla
/Columna

- Unicidad (unique)
 - ▣ Dos filas de la tabla no pueden tener el mismo valor no nulo en las columnas indicadas.
 - ▣ Puede aplicarse al conjunto completo de columnas

4 – Restricciones de Integridad

95

Restricciones
de Tabla /
Columna

- Clave primaria (primary key)
 - ▣ Es una restricción de unicidad pero en la cual las columnas no pueden tomar valor nulo (integridad de entidades)

4 – Restricciones de Integridad

96

Restricciones
de Tabla /
Columna

- Integridad referencial (foreign key-references) (**VER CREATE TABLE**)
 - ▣ Representa una clave ajena o foránea.
 - ▣ Incluye una o más columnas referenciantes (referencing columns) y las correspondientes columnas referenciadas (referenced columns) de una tabla base referenciada (referenced table), que puede ser la referenciante.
 - ▣ Las columnas referenciadas deben tener una restricción de unicidad en la tabla referenciada.

4 – Restricciones de Integridad

97

Restricciones
de Tabla /
Columna

- Control (check)
 - ▣ Incluye una condición de búsqueda que sólo puede referir a elementos de la tabla que la incluye.

4 – Restricciones de Integridad

98

Restricciones
de Tabla /
Columna

- Valor por defecto
 - ▣ Especifica un valor por defecto que tomara la columna si no se indica un valor explicito

4 – Restricciones de Integridad

99

Sintaxis

- $\langle \text{definición de restricción de tabla} \rangle ::=$
[CONSTRAINT $\langle \text{nombre de restricción} \rangle$] $\langle \text{restricción de tabla} \rangle$ [$\langle \text{modo de restricción} \rangle$]

5 – Seguridad

100

Seguridad

- Debemos proteger la base de datos de actualizaciones malintencionadas
 - ▣ Cada usuario debe ver solo los datos que ellos necesiten
 - ▣ Los usuarios tienen privilegios, y deben solo podran operar segun el modo y el conjunto de datos que se les haya concedido
 - Sentencias Grant and Revoke
 - Tablas Virtuales: View

5 – Seguridad

101

Usuarios

□ create <nombre usuario>
identified by <password>;

5 – Seguridad

102

Privilegios

- Obtención de privilegios:
 - ▣ El creador de un objeto es el dueño
 - ▣ El dueño tiene todos los privilegios y puede conceder privilegios a otros usuarios

5 – Seguridad

103

Conceder/
Revocar
permisos

- `grant <lista de privilegios>`
 `[on <objeto/s>]`
 `to <usuario/s o rol/es>`
 `[with grant option];`

Ejemplos:

- `grant select on emp to usu1;`
- `grant select(a,b) on emp to usu1;`
- `grant insert on dept to usu2 with grant option;`

Esta ultima, no solo concede el derecho de insetar tuplas al usu2 sino que tambien usu2 pueda conceder derechos a otros usuarios.

5 – Seguridad

104

Conceder/
Revocar
permisos

```
□ revoke <lista de privilegios>  
    [on <objeto/s>]  
    from <usuario/s o rol/es>  
    [restrict | cascade];
```

Ejemplo:

```
□ revoke select on emp from usu1 cascade;
```

En este caso, la sentencia revoca el privilegio de hacer select sobre la tabla emp a usu1, pero ademas (en forma de cascada o transitivamente) tambien lo revoca a los usuarios que se lo haya concedido usu1.