El Lenguaje SQL

TEMA V

Índice

¿QUÉ ES SQL?
(Structured Query Language)

- LENGUAJE DE:
  - **Definición**
  - **Manipulación**
    - Consulta **interactiva**
    - **Programación**
  - **Control**
- OPERA CON **CONJUNTOS DE REGISTROS**
- LENGUAJE **AUTOCONTENIDO Y NO PROCEDIMENTAL**

HISTORIA DE SQL

- IBM lo desarrolla en los años 70 (SEQUEL, SEQUEL-2, SQL)
- en los 80, la ANSI (American National Standard Institute) lo transforma en estándar para la definición y manipulación de datos en RDBMS. Aparecen numerosos SGBD basados en SQL o con la misma apariencia que este.
- en los 90 se amplían sus capacidades: SQL embebido (89), varias revisiones del estándar (SQL92, SQL9x), y versiones propias de ciertos SGBDR.
- hoy, su uso es generalizado en todos los SGBD relacionales

El Lenguaje SQL

1. Introducción

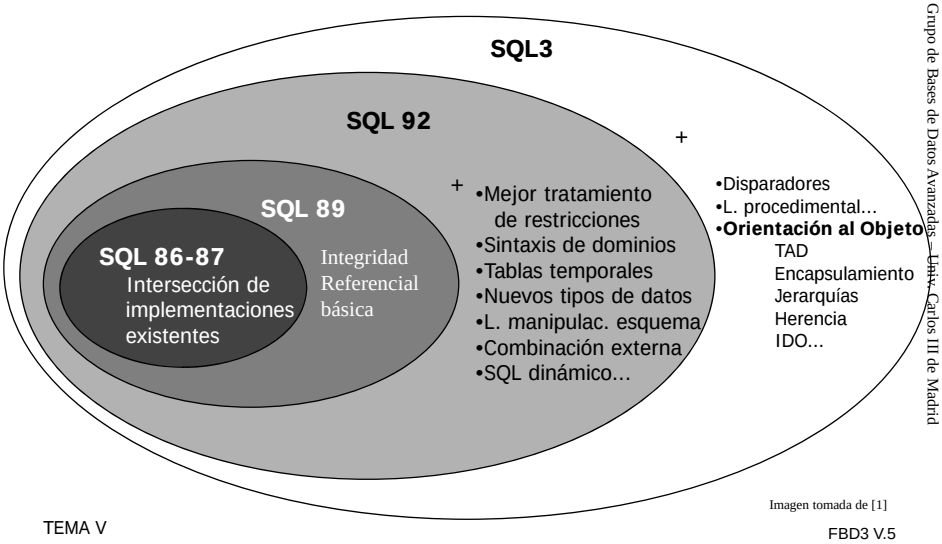

Imagen tomada de [1]

El Lenguaje SQL

2. SQL como Lenguaje de Definición de Datos

SENTENCIAS DE DEFINICIÓN DE DATOS

	Nivel Lógico Global	Nivel Externo	Nivel Físico
Definición del Esquema	CREATE DOMAIN CREATE TABLE CREATE ASSERTION	CREATE VIEW	CREATE INDEX
Evolución del Esquema	ALTER DOMAIN ALTER TABLE DROP DOMAIN DROP TABLE DROP ASSERTION	DROP VIEW	DROP INDEX

Definición de Dominios

<definición de dominio>::= **CREATE DOMAIN** nombre de dominio **[AS]**
<tipo de datos>
[**DEFAULT** <opción por defecto>]
[<restricción de dominio>]

<restricción de dominio>::= [<definición de nombre de restricción>]
<definición de restricción de verificación>
[<atributos de restricción>]

<definición de nombre de restricción>::= **CONSTRAINT** <nombre de restricción>

<definición de restricción de verificación>::= **CHECK** (<condición>)

El Lenguaje SQL

2. SQL como Lenguaje de Definición de Datos

Definición de Dominios

Ejemplo:

ALUMNO (num_mat, nombre, ciudad, cod_grupo)

DC/UC → **GRUPO (cod_grupo, curso, turno, num_alums)**

```
CREATE DOMAIN nomb_valido AS CHAR(20);

CREATE DOMAIN nota_valida INTEGER
        CHECK (VALUE BETWEEN 0 AND 10);

CREATE DOMAIN turno_valido CHAR(1)
        DEFAULT 'M'
        CHECK (VALUE IN ('M','T'));
```

Creación de tablas

<definición de tabla>::= **CREATE [TEMPORARY] TABLE** <nombre de tabla>
(<elemento de tabla> [{ , <elemento de tabla>}.....])

- Los elementos de tabla son:

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

<definición de restricción de tabla/columna>::= [<definición de nombre de restricción>]
<restricción de tabla/columna>
[<atributos de restricción>]

Creación de tablas

<definición de integridad referencial>::= **FOREIGN KEY** (<columnas que referencian>) **REFERENCES** <tabla y columnas referenciadas>
[**MATCH** <tipo de correspondencia>]
[<acción referencial disparada>]

- Donde:

<tabla y columnas referenciadas>::= <nombre de tabla> [(<lista de columnas referenciadas>)]
<acción referencial disparada>::=
<regla de modificación> [<regla de borrado>]
| <regla de borrado> [<regla de modificación>]

Creación de tablas

<regla de modificación> ::= **ON UPDATE** <acción referencial>
<regla de borrado> ::= **ON DELETE** <acción referencial>

<acción referencial> ::=
CASCADE
| **SET NULL**
| **SET DEFAULT**
| **NO ACTION**

El Lenguaje SQL

2. SQL como Lenguaje de Definición de Datos

Creación de tablas

Ejemplo:

```
CREATE TABLE grupo   (

        cod_grupo CHAR(3),
        curso CHAR(1) NOT NULL,
        turno TURNO_VALIDO,
        num_alums INTEGER(2),
        PRIMARY KEY (cod_grupo),
        CHECK (curso>'0' AND curso<'4')
);
```

Creación de tablas

Ejemplo:

```
CREATE TABLE alumno (

        num_mat INTEGER,
        nombre NOMB_VALIDO UNIQUE,
        ciudad CHAR(25) NOT NULL,
        cod_grupo CHAR(3),

        PRIMARY KEY (num_mat),
        FOREIGN KEY (cod_grupo) REFERENCES grupo
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```


Creación de aserciones

<definición de aserción>::= **CREATE ASSERTION**
<nombre de restricción>
<verificación de aserción>
[<atributos de restricción>]

<verificación de aserción> :: = **CHECK** (<condición>)

Creación de aserciones

Ejemplo:

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

```
CREATE ASSERTION madrid_tarde
CHECK (alumnos.ciudad<>'MADRID' OR grupo.turno='T');
```


El Lenguaje SQL

2. SQL como Lenguaje de Definición de Datos

Creación de vistas

<definición de vista>::= **CREATE VIEW** <nombre de vista>
[(<lista de columna>)] **AS**
<expresión consulta>
[WITH CHECK OPTION]

Ejemplo:

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

Creación de índices

<definición de índice>::= **CREATE [UNIQUE] INDEX**
<nombre del índice>
ON <nombre de tabla>
(<lista de columna>)

Ejemplo:

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


Actualización de esquemas

<borrar dominio>::= **DROP DOMAIN** <nombre de dominio>
<borrar tabla>::= **DROP TABLE** <nombre de tabla>
<borrar vista>::= **DROP VIEW** <nombre de vista>

- **Ejemplo:**

```
DROP DOMAIN turno_valido;

DROP TABLE alumno;

DROP ASSERTION ciudad_turno;
```

Actualización de esquemas

<modificación de tabla>::=**ALTER TABLE** <nombre de tabla>
ADD | MODIFY | DROP
<elementos>

donde elementos son **COLUMN**, **CONSTRAINT**

Ejemplo:

```
ALTER TABLE alumno
        DROP COLUMN nombre;

ALTER TABLE alumno
        ADD COLUMN precio INTEGER;
```


Disparadores

<definición de disparador>::=
[**CREATE**] **TRIGGER** [<nombre de disparador>]<tiempo acción disparador>
<evento disparador> **ON** <nombre tabla> [**REFERENCING** <lista de alias para valores antiguos o nuevos>]
<acción disparada>

Donde:

<tiempo de acción> ::=
AFTER
| **BEFORE**

<evento disparador< ::=
INSERT
| **DELETE**
| **UPDATE** [**OF** <lista de columnas disparador>]

Disparadores

```
<alias para valores antiguos o nuevos> ::=
    {OLD [ AS ] <nombre correlación valores antiguos>
     NEW [ AS ] <nombre correlación valores nuevos>}
  | {OLD_TABLE [ AS ]<alias de tabla de valores antiguos>
     NEW_TABLE [ AS ]<alias de tabla de valores nuevos>}

<acción disparada> ::=
    [ FOR EACH { ROW | STATEMENT}]
    [ WHEN ( <condición de búsqueda>) <sentencia SQL disparada>

<sentencia SQL disparada> ::=
            <sentencia de procedimiento SQL>
          | BEGIN ATOMIC
          | <sentencia de procedimiento SQL>;…
            END
```


El Lenguaje SQL

2. SQL como Lenguaje de Definición de Datos

Disparadores

- Ejemplo:

```
CREATE TRIGGER modificar_alum
   AFTER UPDATE OF cod_grupo ON Alumno
   REFERENCING OLD_TABLE AS v_alum
                NEW_TABLE AS n_alum
   FOR EACH ROW
   BEGIN ATOMIC
        (UPDATE Grupo SET num_alums = num_alums + 1
                WHERE Grupo.cod_grupo=n_alum.cod_grupo);
         (UPDATE Grupo SET num_alums = num_alums - 1
                WHERE Grupo.cod_grupo=v_alum.cod_grupo);
   END
```

Sentencias de manipulación

Operaciones de actualización:

INSERT
MODIFY
DELETE

Operación de consulta o recuperación

SELECT

Operación de actualización

Sintaxis para la inserción

<sentencia de inserción>::= **INSERT INTO** <nombre tabla>
[(columna [,columna]....)]
VALUES <(valor [,valor].....)>

Ejemplo:

INSERT INTO alumno VALUES ('016', 'Juan López', 'Barcelona', '3');

Operaciones de actualización

Sintaxis de la modificación

<sentencia de modificación**>**::= **UPDATE** <nombre tabla>
SET { <nombre de columna> = <valor>
[, <nombre de columna> = <valor>]...}
[**WHERE** <condición de búsqueda>]

Ejemplo:

```
UPDATE grupo SET turno='T' WHERE curso='3';
```


Operaciones de actualización

Sintaxis del borrado

<sentencia de borrado>::= **DELETE FROM** <nombre tabla>
[**WHERE** <condición de búsqueda>]

Ejemplo:

```
DELETE FROM alumno
WHERE num_mat=16;
```

Operación de consulta o recuperación

Sintaxis de la consulta

<sentencia de consulta>::=
SELECT (*|**ALL**|**DISTINCT**|<columna [,columna]...>)
FROM <nombre_tabla>[[,nombre_tabla]...]
WHERE <condición de búsqueda>
[**GROUP BY** <lista de columnas>
[**HAVING** <condición de búsqueda>] |
ORDER BY <lista de atributos> [**ASC**|**DESC**]]

ALL => selecciona todas las columnas
DISTINCT => suprime columnas duplicadas

Operación de consulta o recuperación

Sintaxis de la consulta

<condición de búsqueda>::= <atributo><operador>
<atributo|valor>[**AND**|**OR**|
<atributo><operador> <atributo|valor>]...
<operador>::=**<,>,<=,>=**,**between, like,in** ...

Ejemplo:
SELECT * FROM alumno WHERE nombre LIKE 'A%'

Operación de consulta o recuperación

Funciones de Agregado

- Funciones que se aplican sobre las tuplas que componen cada grupo resultante de aplicar una cláusula GROUP BY.
- Las funciones de agregado definidas en SQL92 son: **COUNT**, **MAX**, **MIN**, **SUM** y **AVG**.
- **Ejemplo:**

```
SELECT ciudad, COUNT(*) FROM alumno GROUP BY
ciudad;
```


Operación de consulta o recuperación

Orden de ejecución de cláusulas

- Se eligen las tuplas que cumplen las condiciones de la cláusula WHERE
- Se construyen los grupos según la cláusula GROUP BY
- Se calculan los resultados de las funciones de agregado para cada grupo, si las hay
- Se eliminan aquellos grupos que no cumplen las restricciones definidas en la cláusula HAVING
- Por último, se ordenan las tuplas de salida según las columnas indicadas en ORDER BY

SQL y el Álgebra Relacional

Extensión de Ejemplo

Grupo

cod_grupo	curso	Turno
G81	A	M
G82	B	T

Alumno

Num_mat	nombre	ciudad	Cod_grupo
334	Luis	Madrid	G81
335	Ana	Madrid	G81
336	Jorge	Lugo	G81
337	Jose	Burgos	G81

SQL y el Álgebra Relacional

Selección

$\pi_{num\_mat,\ nombre}\ (\sigma_{ciudad='Madrid'}\ (\textit{alumno}))$

```
SELECT num_mat,nombre
FROM alumno
WHERE (ciudad='Madrid');
```

Num_mat	nombre
334	Luis
335	Ana

SQL y el Álgebra Relacional

Proyección

$\pi_{nombre,ciudad}$ (***alumno***)

SELECT nombre, ciudad
FROM alumno;

Nombre	Ciudad
Luis	Madrid
Ana	Madrid
Jorge	Lugo
Jose	Burgos

El Lenguaje SQL

3. SQL como Lenguaje de Manipulación de Datos

SQL y el Álgebra Relacional

Proyección

π_{ciudad} (***alumno***)

SELECT ciudad
FROM alumno;

Ciudad
Madrid
Madrid
Lugo
Burgos

¿Proyección?

SQL y el Álgebra Relacional

Producto Cartesiano

alumno x ***grupo***

```
SELECT *
FROM alumno, grupo;
```

Cod_grupo	curso	turno	Num_mat	nombre	ciudad	Cod_grupo
G81	A	M	334	Luis	Madrid	G81
G81	A	M	335	Ana	Madrid	G81
G81	A	M	336	Jorge	Lugo	G81
G81	A	M	337	Jose	Burgos	G81
G82	B	T	334	Luis	Madrid	G81
G82	B	T	335	Ana	Madrid	G81
G82	B	T	336	Jorge	Lugo	G81
G82	B	T	337	Jose	Burgos	G81

SQL y el Álgebra Relacional

Combinación Natural

$\pi_{\text{cod\_grupo, curso, nombre}}$ (***alumno*** $*_{\text{cod\_grupo}}$ ***grupo)***

Cod_grupo	curso	nombre
G81	A	Luis
G81	A	Ana
G81	A	Jorge
G81	A	Jose

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

⇕

```
SELECT grupo.cod_grupo, curso,
nombre FROM alumno INNER JOIN
grupo ON
(alumno.cod_grupo=curso.cod_gru
po);
```

3. SQL como Lenguaje de Manipulación de Datos

SQL y el Álgebra Relacional

Combinación Externa

$\pi_{cod\_grupo,\ curso,\ nombre}$ (*alumno* * / $_{cod\_grupo}$ *grupo*)

Cod_grupo	curso	nombre
G81	A	Luis
G81	A	Ana
G81	A	Jorge
G81	A	Jose
G82	B	

```
SELECT grupo.cod_grupo, curso,
nombre FROM alumno RIGHT
OUTER JOIN grupo ON
(alumno.cod_grupo=curso.cod_gru
po);
```


4. SQL como Lenguaje de Control

- El creador de un objeto es siempre el propietario del mismo y tiene todos los privilegios sobre el mismo.

- Autorización de acceso:

<sentencia de concesión>::=**GRANT** <privilegios> **ON** <nombre de objeto> **TO** <usuario>[, <usuario>] **[WITH GRANT OPTION]**

Donde:

<privilegios>::= **ALL PRIVILEGES** | <lista de acción>

<lista de acción>::= **SELECT** | **DELETE** | **INSERT** | **UPDATE** | **REFERENCES** | **USAGE**

4. SQL como Lenguaje de Control

- Revocar privilegios:

<sentencia de revocación>::= **REVOKE [GRANT OPTION FOR]**
<privilegios> **ON** <nombre de objeto> **FROM**
<usuario>[, <usuario>] <comportamiento de borrado>

Donde:

<comportamiento de borrado>::= **CASCADE** | **RESTRICT**

- **Ejemplo:**
  - GRANT DELETE ON Alumno TO Jose;
  - REVOKE INSERT, UPDATE (precio) ON Alumno FROM Jose, Juan CASCADE

Bibliografía

1. Miguel, A. De, Piattini, M. **Fundamentos y modelos de Bases de Datos**, Ed. Rama 1999
2. Miguel, A. De, Piattini, M. y Marcos, E. **Diseño de Bases de Datos Relacionales**, Ed. Rama 1999
3. Miguel, A. De, Martínez, P., Castro, E., Cavero, J.M., Cuadra, D., Iglesias, A.M. y Nieto, C. **Diseño de Bases de Datos. Problemas Resueltos**, Ed. Rama, 2001
4. Oszu, M.T. y Valduriez, P., **Principles of Distributed database systems**, 2ª Edición, Prentice Hall, 1999