




Bases de Datos

Gabriela Arévalo

garevalo@unq.edu.ar



SQL: Structured Query Language

- ✓ SQL es el lenguaje comercial más aceptado
- ✓ Standard aprobado por ANSI e ISO en 1986
- ✓ Es un lenguaje no procedural cuyo objetivo es el de soportar la:
 - ✓ Definición
 - ✓ Manipulación
 - ✓ Control de Datos
- ✓ Su poder expresivo incluye al AR y lo extiende



Características

- ✓ Opera sobre “conjuntos de tuplas”
 - ✓ Incluso para las operaciones de inserción, borrado y actualización
- ✓ No elimina automáticamente tuplas repetidas
- ✓ El mismo lenguaje puede ser usado en forma:
 - ✓ Interactiva
 - ✓ Embebida en un lenguaje de programación



Sublenguajes

- ✓ DDL (Data Definition Language)
 - ✓ CREATE
 - ✓ ALTER
 - ✓ DROP
- ✓ DML (Data Manipulation Language)
 - ✓ INSERT
 - ✓ UPDATE
 - ✓ DELETE
 - ✓ SELECT => es el único comando que permite recuperar tuplas almacenadas en la base



Mecanismos

- ✓ Seguridad
 - ✓ A veces se lo llama DCL (Data Control Language)
 - ✓ Grant (Permisos sobre objetos y logins)
- ✓ Concurrencia y Recuperación
 - ✓ Commit work
 - ✓ Rollback work
- ✓ Diccionario de Datos
 - ✓ Organizado como conjunto de tablas pudiendo ser accedido con la misma sintaxis, por los usuarios que tienen permiso



DDL: Data Definition Language

- ✓ La instrucción CREATE TABLE:
 - ✓ Crea una tabla vacía
 - ✓ Genera información en el Diccionario de Datos
 - ✓ Algunos elementos de la instrucción CREATE TABLE:
 - ✓ Nombre de la tabla
 - ✓ Nombre de cada atributo
 - ✓ Tipo de datos para cada atributo
 - ✓ Restricciones de integridad: not null, primary key, foreign key



DDL

- ✓ Comencemos con el ejemplo de las facultades y estudiantes que pertenecen a clubes.
- ✓ Cómo definimos el esquema de nuestra BD usando SQL?



Ejemplo: Definición de Tablas

```
CREATE TABLE department
```

```
( code    varchar(3)  PRIMARY KEY,  
  name    varchar(40) NOT NULL,  
  chair   varchar(11) );
```

```
CREATE TABLE person
```

```
( pId      varchar(11) PRIMARY KEY,  
  dob      date NOT NULL,  
  firstName varchar(20) NOT NULL,  
  lastName  varchar(20) NOT NULL);
```

```
CREATE TABLE faculty
```

```
( pId      varchar(11) PRIMARY KEY,  
  rank     varchar(10) NOT NULL,  
  dept     varchar(3) NOT NULL,  
  CONSTRAINT rankValue CHECK (rank IN ('Assistant', 'Associate', 'Full',  
  'Emeritus')),  
  CONSTRAINT facultyPIdFk FOREIGN KEY (pId) REFERENCES person (pId),  
  CONSTRAINT facultyDeptFk FOREIGN KEY (dept) REFERENCES department  
  (code));
```

```
ALTER TABLE department
```

```
ADD CONSTRAINT departmentChairFk FOREIGN KEY(chair)  
REFERENCES faculty(pId) ON DELETE SET NULL;
```


Ejemplo: Definición de Tablas

```
CREATE TABLE student
(  pId      varchar(11)    PRIMARY KEY,
   status  varchar(10)    NOT NULL,
   major   varchar(3)     NOT NULL,
   CONSTRAINT statusValue CHECK (status IN ('Freshman', 'Sophomore',
   'Junior', 'Senior', 'Graduate')),
   CONSTRAINT studentPidFk FOREIGN KEY (pId) REFERENCES person(pId),
   CONSTRAINT studentMajorFk FOREIGN KEY (major) REFERENCES department
   (code));

CREATE TABLE campusclub
(  cId      varchar(10)    PRIMARY KEY,
   name     varchar(50)    NOT NULL,
   phone    varchar(12),
   location varchar(40),
   advisor  varchar(11),
   CONSTRAINT campusclubAdvisor Fk FOREIGN KEY(advisor) REFERENCES faculty
   (pId) ON DELETE SET NULL);

CREATE TABLE clubs
(  pId      varchar(11)    NOT NULL,
   cId      varchar(10)    NOT NULL,
   CONSTRAINT clubsPk PRIMARY KEY(pId, cId),
   CONSTRAINT clubsPidFk FOREIGN KEY (pId) REFERENCES student(pId) ON
   DELETE CASCADE,
   CONSTRAINT clubsCidFk FOREIGN KEY (cId) REFERENCES campusclubs (cId)
   ON DELETE CASCADE);
```



Tipos de Datos

```
CREATE TABLE person
( pId          varchar(11) PRIMARY KEY,
  dob          date NOT NULL,
  firstName    varchar(20) NOT NULL,
  lastName     varchar(20) NOT NULL);
```



Tipos de Datos

Category

Data types

Character
String

char, varchar, nchar, nchar varying, clob (character large object), nclob

Number

int, integer, smallint, numeric, decimal, float, real, double precision

Temporal

date, time, timestamp, interval

Boolean

boolean (true/false/unknown)

Constructed

User-defined types (UDTs)

<reference type>: ref udt_name

<row type>: row (one or more (Field, <data_type>) pairs)

<collection type>: array[]

Binary String

blob (binary large object)

Bit String

bit, bit varying



Restricciones de Integridad

```
CREATE TABLE person
( pId          varchar(11) PRIMARY KEY,
  dob          date NOT NULL,
  firstName    varchar(20) NOT NULL,
  lastName     varchar(20) NOT NULL);
```

```
CREATE TABLE clubs
( pId          varchar(11) NOT NULL,
  cId          varchar(10) NOT NULL,
  CONSTRAINT clubsPk PRIMARY KEY(pId, cId),
  ... );
```



Restricciones de Integridad

- ✓ **NOT NULL:** especifica que el valor de la columna nunca puede ser nulo.
- ✓ **PRIMARY KEY:** especifica el atributo (o conjunto de atributos) que definen la clave primaria en una tabla.
- ✓ **UNIQUE:** define que el valor de un atributo (o conjunto de atributos) que es único, es decir dos tuplas no tienen el mismo valor en la tabla.



Restricciones de Integridad

```
CREATE TABLE faculty
( pId          varchar(11) PRIMARY KEY,
  rank         varchar(10) NOT NULL,
  dept         varchar(3)  NOT NULL,
  CONSTRAINT rankValue CHECK (rank IN
    ('Assistant', 'Associate', 'Full',
    'Emeritus')),
  ...);
```



Restricciones de Integridad

- ✓ CHECK: define restricciones conocidas como restricciones de dominio sobre el valor de la columna.



Restricciones de Integridad

```
CREATE TABLE faculty
( pId    varchar(11) PRIMARY KEY,
  rank   varchar(10) NOT NULL,
  dept   varchar(3)  NOT NULL,
  CONSTRAINT rankValue CHECK (rank IN
    ('Assistant', 'Associate', 'Full',
    'Emeritus')),
  CONSTRAINT facultyPIdFk FOREIGN KEY (pId)
  REFERENCES person (pId),
  CONSTRAINT facultyDeptFk FOREIGN KEY
  (dept) REFERENCES department (code));
```




Restricciones de Integridad

- ✓ FOREIGN KEY: define la noción de integridad referencial entre una clave foránea y una clave primaria en una tabla separada. Ambas claves deben tener el mismo tipo de datos.



DDL

- ✓ Otras instrucciones sobre tablas:
 - ✓ ALTER TABLE: permite modificar la definición de la tabla
 - ✓ cambiar nombre de atributos
 - ✓ cambiar tipos de atributos
 - ✓ agregar o quitar atributos
 - ✓ agregar o quitar restricciones de integridad
 - ✓ etc ...
 - ✓ DROP TABLE: permite eliminar una tabla



Ejemplo: ALTER

```
ALTER TABLE student  
  ADD gradePointAverage numeric  
  ALTER status SET DEFAULT =  
    'Freshman' ;
```

‘ Agrega el nuevo atributo (o columna) *gradePointAverage* a la tabla student y modifica el valor por defecto del atributo *status* a Freshman’

Ejemplo: ALTER

```
CREATE TABLE department
(
    code      varchar(3)  PRIMARY KEY,
    name      varchar(40) NOT NULL,
    chair     varchar(11) );
```

```
CREATE TABLE person
(
    pId       varchar(11) PRIMARY KEY,
    dob       date NOT NULL,
    firstName varchar(20) NOT NULL,
    lastName  varchar(20) NOT NULL);
```

```
CREATE TABLE faculty
(
    pId       varchar(11) PRIMARY KEY,
    rank      varchar(10) NOT NULL,
    dept      varchar(3)  NOT NULL,
    CONSTRAINT rankValue CHECK (rank IN ('Assistant', 'Associate', 'Full',
    'Emeritus')),
    CONSTRAINT facultyPidFk FOREIGN KEY (pId) REFERENCES person (pId),
    CONSTRAINT facultyDeptFk FOREIGN KEY (dept) REFERENCES department
    (code));
```

```
ALTER TABLE department
ADD CONSTRAINT departmentChairFk FOREIGN KEY (chair)
REFERENCES faculty(pId) ON DELETE SET NULL;
```



Ejemplo: DROP

```
DROP TABLE student RESTRICT;
```

```
DROP TABLE student CASCADE;
```

- ✓ **RESTRICT:** cuando una columna se elimina, esta opción se usa para indicar que la columna se pueda eliminar si no hay otro componente del esquema que se refiera a esa columna.
- ✓ **CASCADE:** con esta opción, se elimina todos los componentes del esquema que hace referencia a la columna eliminada.



DML: Data Manipulation Language

- ✓ Modificación de los datos
 - ✓ INSERT: permite agregar tuplas a una tabla
 - ✓ UPDATE: permite cambiar tuplas de una tabla
 - ✓ DELETE: permite borrar tuplas de una tabla



Ejemplo: INSERT

```
INSERT INTO department  
VALUES ('cse', 'Computer Science and  
Engineering', '111-11');
```

- ✓ Un valor se especifica para cada columna de la tabla en el orden de las columnas de la tabla. Se puede usar un valor NULL para columnas con valores desconocidos.

```
INSERT INTO department(code, name, chair)  
VALUES ('cse', 'Computer Science and  
Engineering', '111-11');
```

- ✓ Se especifica sólo las columnas a los cuales se les asignará los valores especificados. Las columnas no especificadas toman el valor NULL salvo cuando se haya dado un valor por defecto.



Ejemplo: DELETE y UPDATE

```
DELETE FROM student  
WHERE status = 'Junior';
```

- ✓ Se borran de la tabla todas las tuplas que cumplan con la condición especificada en el *where*. En este caso, que el status del estudiante sea 'Junior'

```
UPDATE faculty  
    SET rank = 'full professor'  
    WHERE name = 'Joseph';
```

- ✓ Se actualiza todas las tuplas con el rango de 'full professor' a todos los miembros de la facultad con nombre Joseph.



Ejemplo: Restricciones de Integridad

```
CREATE TABLE campusclub
( cId varchar(10) PRIMARY KEY,
  name varchar(50) NOT NULL,
  phone varchar(12),
  location varchar(40),
  advisor varchar(11),
  CONSTRAINT campusclubAdvisor_Fk FOREIGN KEY(advisor)
    REFERENCES faculty(pId) ON DELETE SET NULL);
```

```
CREATE TABLE clubs
( pId varchar(11) NOT NULL,
  cId varchar(10) NOT NULL,
  CONSTRAINT clubsPk PRIMARY KEY(pId, cId),
  CONSTRAINT clubsPIdFk FOREIGN KEY (pId) REFERENCES
  student(pId) ON DELETE CASCADE,
  CONSTRAINT clubsCIIdFk FOREIGN KEY (cId) REFERENCES
  campusclubs(cId) ON DELETE CASCADE);
```



Restricciones de Integridad

- ✓ La restricción de la foreign key incluye las acciones que deben realizarse para asegurarse la preservación de la integridad referencial cuando se actualizan o eliminan tuplas.
- ✓ Las acciones son:
 - ✓ NO ACTION
 - ✓ CASCADE
 - ✓ SET NULL
 - ✓ SET DEFAULT



Restricciones de Integridad

- ✓ **NO ACTION:** Es por defecto. Se evitará un update or delete si la integridad referencial se viola.
- ✓ **CASCADE:** Un update o delete de una tupla en la tabla referenciada hará que la base de datos actualice o borre todas las filas que “matchean” en la tabla que referencia.



Restricciones de Integridad

- ✓ **SET NULL:** Un update o delete de una tupla en la tabla referenciada hará que la base de datos anote un valor nulo sobre la foreign key en todas las filas que “matchean” en la tabla que referencia. Esta acción no puede usarse con la restricción de not null en la foreign key.
- ✓ **SET DEFAULT:** Un update o delete de una tupla en la tabla referenciada hará que la base de datos anote el valor por defecto sobre la foreign key en todas las filas que “matchean” en la tabla que referencia.



DML: Recuperación de datos

✓ SELECT:

- ✓ Devuelve tuplas de la base

```
SELECT a1, ..., an
```

```
FROM r1, ..., rm
```

```
WHERE p
```

donde:

a1, ..., an son nombres de atributos (puede usarse *)

r1, ..., rm son nombres de tablas

p es una condición

- ✓ Expresa en álgebra relacional:

$$\pi_{A_1, \dots, A_n} (\sigma_p (R_1 \times \dots \times R_m))$$



DML: Ejemplo

- ✓ Comencemos con un ejemplo
 - ✓ Sea la BD:
 - ✓ `fabs (#fab, nombre, dirección)`
 - ✓ `prods (#prod, descripción)`
 - ✓ `ventas (#fab, #prod, precio)`
- ✓ Cómo definimos el esquema de nuestra BD usando SQL?



Ejemplo

```
CREATE TABLE fabs
  (#fab number(5) NOT NULL,
  nombre char(20),
  direccion char(40));
```

```
CREATE TABLE prods
  (#prod number(5) NOT NULL,
  descripcion char(20));
```

```
CREATE TABLE ventas
  (#fab number(5) NOT NULL,
  #prod number(5) NOT NULL,
  precio float);
```



Ejemplos

- ✓ Dar el precio al que vende el producto 7 el fabricante 2

```
SELECT precio
FROM ventas
WHERE #fab = 2 AND #prod = 7
```

- ✓ Dar el número de los fabricantes que vendieron el producto 4 a menos de \$ 100

```
SELECT #fab
FROM ventas
WHERE precio < 100 AND #prod = 4
```




DML: Ordenamiento de tuplas

- ✓ Dar los #prod de los productos fabricados, pero ordenados en forma ascendente

```
SELECT #prod
FROM ventas
ORDER BY #prod asc;
```

- ✓ La cláusula ORDER BY permite indicar el campo por el cual se ordena el resultado.
- ✓ Se indica el orden (asc o desc)



DML: Filtrado de repetidos

- ✓ Dar los #prod de los productos fabricados, filtrando los repetidos

```
SELECT DISTINCT #prod
FROM ventas
ORDER BY #prod
```

- ✓ La cláusula **DISTINCT** filtra tuplas repetidas



DML: Join

- ✓ Dar los nombres de fabricantes y los #prod que venden

```
SELECT nombre, #prod
FROM ventas, fabs
WHERE ventas.#fab = fabs.#fab;
```

- ✓ Dar los nombres de fabricantes y los #prod que venden, tales que vendió el producto a \$100.

```
SELECT nombre, #prod
FROM ventas, fabs
WHERE ventas.#fab = fabs.#fab and precio
= 100;
```



DML: Union

- ✓ Dar los #fab que viven en “d1” o que venden algún producto a precio mayor que \$ 500.

```
SELECT #fab FROM fabs
```

```
WHERE direc = “d1”
```

```
UNION
```

```
SELECT #fab FROM ventas
```

```
WHERE precio > 500;
```

- ✓ La unión elimina tuplas repetidas



DML: Resta

- ✓ Dar los #fab que no venden ningún producto

```
SELECT #fab FROM fabs  
MINUS  
SELECT #fab FROM ventas;
```

O también:

```
SELECT #fab FROM fabs  
WHERE #fab NOT IN  
    (SELECT #fab  
     FROM ventas)
```



DML: Funciones de Suma

- ✓ Las consultas anteriores se podían resolver en álgebra relacional
- ✓ El SQL incluye además funciones para contar tuplas, sumar cantidades, etc.
- ✓ Ejemplo:

- ✓ Decir cuántos fabricantes hay

```
SELECT count (#fab)  
FROM fabs;
```

- ✓ Dar el precio máximo de venta

```
SELECT max (precio)  
FROM ventas;
```



DML: Operadores aritméticos

- ✓ Para cada producto dar su número y su peso en gramos (supongamos que el peso está en kg.)

```
SELECT #prod, peso * 1.000  
FROM prods
```

- ✓ Observaciones:
 - ✓ Expresiones en la cláusula select.
 - ✓ Se pueden utilizar expresiones aritméticas en lugar de nombres de atributos



DML: Operadores aritméticos

- ✓ Queremos que el resultado de peso * 1.000 sea redondeado a 2 cifras decimales.

```
SELECT #prod, round(peso * 1000, 2)  
FROM prods;
```

- ✓ Existe un conjunto de funciones para el tipo de datos number como por ej. abs(n), mod(m,n), sign(n), etc.
- ✓ Observemos que estas funciones toman el valor de UNA tupla por vez y devuelven un valor para cada una de ellas



DML: Operadores aritméticos

- ✓ Queremos poner un título en la columna:

```
SELECT #prod, round(peso * 1000,2) as  
    "Peso en gr."  
FROM prods
```

- ✓ Se conoce que el precio de la manteca es \$ 25 el gramo, queremos listar los productos de nombre manteca y su costo total.

```
SELECT #prod, round(peso * 1000,2) * 25  
    as "Precio"  
FROM prods  
WHERE nombre = "manteca";
```



DML: Operadores aritméticos

- ✓ Dar los nombres de los productos que pesan 3 veces más que su volumen.

```
SELECT nombre  
FROM prods  
WHERE peso = 3 * volumen
```

- ✓ Podemos usar expresiones aritméticas como condiciones en el WHERE