

# BASES DE DATOS II

## Tema 3

### Lenguaje de Consulta SQL

Víctor Valenzuela Ruz  
INACAP Copiapó

### SQL (*Structured Query Language*)

- SQL se planteó en la década del '70 para su uso en una BD experimental (System R).
- Constituye un estándar:
  - SQL1 (1986)
  - SQL2 ó SQL '92
  - SQL3 ó SQL '99
- Tiene dos componentes principales:
  - **DDL - Definición de datos**
    - create
    - alter
    - drop
  - **DML - Manipulación de datos**
    - select

## SQL - Definición de datos

```

CREATE TABLE HOSPITAL
( HOSP_ID SMALLINT NOT NULL,
  NOMBRE CHAR(30),
  LOCALIDAD CHAR(30),
  DIRECTOR CHAR(50),
  FECHA_FUND DATE,
PRIMARY KEY (HOSP_ID),
FOREIGN KEY (DIRECTOR) REFERENCES DIR(DIRECTOR)
ON DELETE SET NULL ON UPDATE CASCADE );

ALTER TABLE HOSPITAL
  ADD ( CANT_EMPL SMALLINT );

ALTER TABLE HOSPITAL
  DROP FECHA_FUND CASCADE;

DROP TABLE HOSPITAL CASCADE;
  
```

06-04-2002

V́ctor Valenzuela Ruz

3

## SQL - Inserción y eliminación de datos

- Para insertar una tupla en una relación, por ejemplo en la tabla Hospital:

```

INSERT INTO HOSPITAL
VALUES ( 10, 'HOSP1', 'LOCAL1', 'DIR1', 20/02/76);
  
```

```

INSERT INTO HOSPITAL ( HOSP_ID, NOMBRE )
VALUES ( 10, 'HOSP1');
  
```

- Para eliminar tuplas de la relación:

```

DELETE FROM HOSPITAL
WHERE HOSP_ID = 10;
  
```

*elimina una tupla*

```

DELETE FROM HOSPITAL
WHERE FECHA_FUND < 01/01/1980;
  
```

*puede eliminar varias tuplas*

06-04-2002

V́ctor Valenzuela Ruz

4

## SQL - Consultas (1)

- Forma básica de la sentencia SELECT :

```
SELECT <lista de atributos>  
FROM <lista de tablas>  
WHERE <condición>
```

Ej: *Listar los Identificadores y nombre de los hospitales fundados*

En SQL: 

```
SELECT HOSP_ID, NOMBRE  
FROM HOSPITAL  
WHERE FECHA_FUND >= 01/01/1980  
AND FECHA_FUND <= 31/12/1989;
```

- La cláusula SELECT especifica los atributos de Proyección
- La cláusula WHERE especifica la condición de Selección
- En SQL se pueden generar tuplas repetidas

## SQL - Consultas (2)

- Sentencia SELECT con recuperación de varias tablas:

Ej: *Listar los Identificadores de hospitales, junto con sus directores y domicilios*

En SQL: 

```
SELECT HOSP_ID, DIRECTOR, DOMICILIO  
FROM HOSPITAL, DIR  
WHERE HOSPITAL.DIRECTOR = DIR.DIRECTOR ;
```

- En SQL el ensamble se realiza indicando explícitamente la condición de igualdad entre atributos.
- NOTA: cuando los atributos de diferentes tablas poseen el mismo nombre se los debe calificar con el nombre de la tabla o un alias.

## SQL - Consultas (3)

- Para recuperar todos los datos de las tablas en la cláusula FROM:

Ej: *Listar todos los datos de los hospitales*

```
SELECT * FROM HOSPITAL, DIR
WHERE HOSPITAL.DIRECTOR = DIR.DIRECTOR ;
```

- Para eliminar los duplicados:

Ej: *Listar las distintas localidades donde hay hospitales*

```
SELECT DISTINCT LOCALIDAD FROM HOSPITAL;
```

- Uso de IN y LIKE:

Ej: *Listar los hospitales de Tandil, Azul y Olavarría*

```
SELECT * FROM HOSPITAL
WHERE LOCALIDAD IN ('Tandil', 'Azul', 'Olavarría');
```

Ej: *Listar los datos de los directores cuyo apellido comience con 'C'*

```
SELECT * FROM DIR
WHERE DIRECTOR LIKE 'C%';
```

## SQL - Consultas (4)

- Funciones de Agregado: Resumen el resultado de una consulta:

- SUM() *da el total de la columna especificada*
- AVG() *da el promedio de la columna especificada*
- MAX() *da el valor máximo de la columna especificada*
- MIN() *da el valor mínimo de la columna especificada*
- COUNT(\*) *da la cantidad total de tuplas*

en todos los casos el resultado se calcula columnas de tipo numérico y sobre las tuplas que satisfacen la condición.

Ejemplos:

Ej: *Indicar la antigüedad promedio de los directores*

```
SELECT AVG (ANTIGÜEDAD) FROM DIR;
```

Ej: *Listar la cantidad total de hospitales de Tandil*

```
SELECT COUNT(*) FROM HOSPITAL WHERE LOCALIDAD = 'TANDIL';
```

## SQL - Consultas (5)

- Una consulta SELECT puede consistir de hasta 6 cláusulas, pero sólo las 2 primeras son obligatorias:

```
SELECT <lista de atributos>  
FROM <lista de tablas>  
[ WHERE <condición> ]  
[ GROUP BY <atributos de agrupamiento> ]  
[ HAVING <condición del grupo> ]  
[ ORDER BY <lista de atributos> ]
```

## SQL - Consultas continuación (1)

- Una consulta SELECT puede contener las siguientes cláusulas:

```
SELECT <lista de atributos>  
FROM <lista de tablas>  
[ WHERE <condición> ]  
[ GROUP BY <atributos de agrupamiento> ]  
[ HAVING <condición del grupo> ]  
[ ORDER BY <lista de atributos> ]
```

- Las cláusulas encerradas entre corchetes son opcionales.

## SQL - Consultas continuación (2)

```
CREATE TABLE DEPARTAMENTO (
    nro_departamento INTEGER NOT NULL,
    descripcion VARCHAR(20),
    area VARCHAR(20),
    PRIMARY KEY (nro_departamento)
);

CREATE TABLE EMPLEADO (
    nro_emp INTEGER NOT NULL,
    apellido VARCHAR(20),
    nombre VARCHAR(20),
    sueldo INTEGER,
    antiguedad INTEGER,
    PRIMARY KEY (nro_emp)
);

CREATE TABLE TRABAJA (
    nro_emp INTEGER NOT NULL,
    nro_departamento INTEGER NOT NULL,
    PRIMARY KEY (nro_emp, nro_departamento),
    FOREIGN KEY (nro_departamento)
        REFERENCES DEPARTAMENTO,
    FOREIGN KEY (nro_emp)
        REFERENCES EMPLEADO
);
```

06-04-2002

Victor Valenzuela Ruz

11

## SQL - Consultas continuación (3)

- Consultas agrupadas:
- Listar los departamentos que tienen mas de 5 empleados

```
SELECT descripcion
FROM departamento
GROUP BY nro_departamento
HAVING count(*) > 5
```

06-04-2002

Victor Valenzuela Ruz

12

## SQL - Consultas continuación (4)

- **Consultas y subconsultas**
- Seleccionar los empleados cuyo sueldo es mayor al promedio de los sueldos

```
SELECT nro_emp, apellido, nombre
FROM empleado
WHERE sueldo > (SELECT AVG(sueldo)
                FROM empleado)
```

## SQL - Consultas continuación (5)

- **Consultas con subconsultas**
- Seleccionar los empleados que trabajen en el departamento COMPRAS

```
SELECT nro_emp, apellido, nombre
FROM empleado E
WHERE nro_departamento IN (SELECT nro_departamento
                             FROM departamento D
                             WHERE D.descripcion = 'COMPRAS' )
```

## SQL - Consultas continuación (6)

- Otra solución:

```
SELECT nro_emp, apellido, nombre
FROM empleado E
WHERE EXISTS (SELECT nro_departamento
                FROM departamento D
                WHERE D.nro_departamento = E.nro_departamento
                AND D.descripcion = 'COMPRAS')
```

## SQL - Consultas continuación (7)

- Consultas con cuantificación universal
- Dar los empleados que trabajan en todos los departamentos
- Seleccionar los empleados tales que no exista un departamento en el que ellos no trabajen

```
SELECT nro_emp, apellido, nombre
FROM empleado E
WHERE NOT EXISTS
  (SELECT *
   FROM departamento D
   WHERE NOT EXISTS (
     SELECT *
     FROM trabaja T
     WHERE T.nro_emp = E.nro_em AND
           T.nro_departamento = D.nro_departamento))
```