

Temario

- ◆ Introducción y fundamentos
- ◆ Introducción a SQL
- ◆ Modelo Entidad / Relación
- ◆ Modelo relacional
- ◆ Diseño relacional: formas normales
- ◆ Consultas
 - Cálculo relacional
 - Álgebra relacional
- ◆ Implementación de bases de datos
 - Estructura física: campos y registros
 - Indexación
 - Índices simples
 - Árboles B
 - Hashing
 - Compresión

Structured Query Language – SQL

- ◆ Lenguaje de “programación” para SGBDs
 - Data definition language: creación del modelo de datos (diseño de tablas)
 - Data manipulation language: inserción, modificación, eliminación de datos
 - Data query language: consultas
- ◆ Se ejecuta sobre un SGBD
- ◆ El estándar más utilizado
 - Creado en 1974 (D. D. Chamberlin & R. F. Boyce, IBM)
 - ANSI en 1986, ISO en 1987
 - Core (todos los SGBD) + packages (modulos opcionales)
- ◆ Versiones
 - SQL1 – SQL 86
 - SQL2 – SQL 92, SQL 99
 - SQL 3 – no plenamente soportado por la industria
- ◆ Limitaciones
 - No es puramente relacional (p.e. las *vistas* son *multiconjuntos* de tuplas)
 - Importantes divergencias entre implementaciones (no es directamente portable en general, incompletitudes, extensiones) –uno termina aprendiendo variantes de SQL

Elementos de una base de datos SQL

- ◆ Base de datos = conjunto de tablas
- ◆ Tabla (relación, entidad, esquema...) =
 - Estructura fija de campos (esquema)
 - Conjunto de registros con valores de campos
- ◆ Campo (atributo, propiedad, “columna”), tiene un tipo de dato
- ◆ Registro (tupla, “fila”)
- ◆ Clave primaria
- ◆ Clave externa

Estructura léxica del lenguaje

Operaciones SQL

- ◆ DDL – Creación, diseño, eliminación de tablas
- ◆ DML – Inserción, modificación, eliminación de registros
- ◆ DQL – Consulta

Estructura léxica de SQL

- ◆ Case-insensitive, insignificant whitespace
- ◆ Sentencias, expresiones, valores, tipos de datos
- ◆ Referencias
 - Elmasri cap. 8
 - PostgreSQL SQL ref: <http://www.postgresql.org/docs/9.1/static/sql.html>

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Data definition

CREATE TABLE *nombre* (

campo1 tipo1 [restricciones1],

campo2 tipo2 [restricciones2],

...,

[restricciones]

);

ALTER TABLE *nombre* ADD COLUMN *campo tipo [restricciones];*

ALTER TABLE *nombre* ADD *restricción;*

ALTER TABLE *nombre* DROP COLUMN *campo;*

DROP TABLE *nombre;*

DROP CONSTRAINT *nombre-restricción;*

Ejemplo

```
CREATE TABLE Artista (
```

```
    id            int            PRIMARY KEY,
```

```
    nombre       text           NOT NULL,
```

```
    nacionalidad text
```

```
);
```

```
CREATE TABLE Cancion (
```

```
    id            int            PRIMARY KEY,
```

```
    titulo       text           NOT NULL,
```

```
    genero       text,
```

```
    duracion     int,
```

```
    fecha       date,
```

```
    autor       int            NOT NULL REFERENCES Artista (id)
```

```
);
```

```
CREATE TABLE Usuario (  
    nick          varchar(30),  
    nombre       text          NOT NULL,  
    email        text          NOT NULL UNIQUE  
);
```

```
CREATE TABLE Contacto (  
    usuario1     varchar(30) REFERENCES Usuario (nick),  
    usuario2     varchar(30) REFERENCES Usuario (nick),  
    PRIMARY KEY (usuario1,usuario2)  
);
```

```
CREATE TABLE Escucha (  
    usuario      varchar(30),  
    cancion      int          REFERENCES Cancion (id),  
    instante     timestamp;  
    PRIMARY KEY (usuario,cancion,instante)  
);
```

```
ALTER TABLE Escucha DROP COLUMN instante;  
ALTER TABLE Escucha ADD instante timestamp; /* NULL's */
```

```
ALTER TABLE Escucha ADD FOREIGN KEY (usuario)  
    REFERENCES Usuario (nick);
```

```
ALTER TABLE Usuario ADD PRIMARY KEY (nick);
```

Tipos y expresiones

Tipos SQL

character (*n*) \equiv char(*n*), varchar(*n*), text

integer \equiv int, smallint

float, real, double precision

numeric(*precisión*, *escala*) \equiv decimal(*precisión*, *escala*)

date, time, timestamp

dígitos *decimales*
(por defecto 0)

Valores literales

Cadenas de caracteres entre '...'

Valores numéricos similar p.e. a C

date 'YYYY-MM-DD', time 'HH:MM:SS'

Expresiones

Se pueden utilizar en WHERE, SELECT, SET, DEFAULT, CHECK...

Operadores

+ - * / % ^

AND OR NOT

= < > <= >= LIKE ISNULL

operaciones con strings: contatenación, like, expresiones regulares ('%' '_')

Comentarios

--

/* ... */

Restricciones

En un campo

NOT NULL

UNIQUE

PRIMARY KEY

REFERENCES *tabla* (*clave*) [(ON DELETE | ON UPDATE)

(SET NULL | CASCADE | SET DEFAULT)]

DEFAULT *valor*

Con nombre

CONSTRAINT *nombre* *restricción*

En una tabla

PRIMARY KEY (campo1, campo2, ...)

FOREIGN KEY (campo1, campo2, ...)

REFERENCES *tabla* (*clave1*, *clave2*, ...)

UNIQUE (campo1, campo2, ...)

CHECK (*expresión*)

Claves primarias

- ◆ Designan un identificador único de las filas de una tabla
- ◆ Sólo puede haber una clave primaria por tabla, aunque puede incluir varios campos
- ◆ Es muy aconsejable que toda tabla tenga su clave primaria
- ◆ Técnicamente equivalen a UNIQUE más NOT NULL
- ◆ Pero juegan un papel diferente en indexación (lo veremos más adelante)
- ◆ Opción de diseño: selección de clave primaria entre varias posibles
 - Clave primaria natural (email, dominio web, DNI, ISBN, etc.)
 - Clave primaria artificial: p.e. un ID entero (típicamente autoincremental), una cadena de caracteres (códigos), etc.

Claves externas

- ◆ Conceptualmente son comparables a punteros
- ◆ Referencian campos únicos de otra tabla
- ◆ Normalmente el campo referenciado es una clave primaria
- ◆ Técnicamente no es imprescindible usarlas
- ◆ Pero ayuda a asegurar la consistencia en las referencias!
- ◆ Y permiten establecer qué se debe hacer cuando desaparece una clave referenciada

Data manipulation

```
INSERT INTO tabla [(campo1, campo2, ...)] VALUES  
    (valor11, valor12, ...),  
    (valor21, valor22, ...),  
    ...  
;
```

```
UPDATE tabla SET campo1 = valor1, campo2 = valor2, ... [WHERE ...];
```

```
DELETE FROM tabla [WHERE ...];
```

```
TRUNCATE tabla;
```

Ejemplo

```
INSERT INTO Artista VALUES (1, 'The Beatles', 'UK');
```

```
INSERT INTO Artista VALUES (2, 'The Rolling Stones', 'UK');
```

```
INSERT INTO Artista (id, nombre) VALUES (3, 'David Bowie');
```

```
INSERT INTO Cancion VALUES
```

```
(1, 'Norwegian wood', 'Pop', '125', '1965-03-12', 1),
```

```
(2, 'Here, there and everywhere', 'Pop', '145', '1966-08-05', 1),
```

```
(3, 'Jumping jack flash', 'Pop', '225', '1968-04-20', 2);
```

```
INSERT INTO Usuario VALUES
```

```
('lola', 'Dolores', 'lola@gmail.com'),
```

```
('pepe', 'José', 'jose@gmail.com'),
```

```
('chema', 'José María', 'chema@gmail.com'),
```

```
('charo', 'Rosario', 'rosario@gmail.com');
```

```
INSERT INTO Contacto VALUES
```

```
    ('pepe', 'lola'),
```

```
    ('charo', 'pepe'),
```

```
    ('chema', 'charo');
```

```
INSERT INTO Escucha VALUES
```

```
    ('charo', 2, '2011-09-09 16:57:54'),
```

```
    ('pepe', 3, '2011-09-12 21:15:30');
```

```
UPDATE Artista SET nacionalidad = 'UK' WHERE nombre = 'David Bowie';
```

```
UPDATE Album SET precio = precio * 1.2;
```

```
DELETE FROM Escucha WHERE instante < '2000-01-01 00:00:00';
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Data query

```
SELECT [DISTINCT] campos FROM tablas  
[WHERE condición];
```

Ejemplos:

```
SELECT titulo, genero FROM Cancion  
WHERE fecha > '1959-12-31' AND fecha < '1970-01-01';
```

```
SELECT DISTINCT nacionalidad FROM Artista;
```

```
/* Varias tablas */
```

```
SELECT * FROM Cancion, Artista
```

```
WHERE Cancion.autor = Artista.id AND Artista.nacionalidad = 'UK';
```

```
/* Expresiones */
```

```
SELECT dni, teoria * 0.6 + practicas * 0.4 FROM Notas;
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Join

```
SELECT campos  
FROM tabla1 JOIN tabla2 ON condición  
[WHERE condición];
```

Uso típico (pero no sólo) con claves externas: *ON externa = primaria*
Más eficiente que el producto cartesiano (i.e. juntar tablas sin más)

Ejemplos:

```
SELECT titulo FROM Cancion, Escucha  
WHERE usuario = 'lola' AND cancion = id;
```

```
SELECT titulo FROM Cancion JOIN Escucha ON cancion = id  
WHERE usuario = 'lola';
```

```
SELECT * FROM Contacto JOIN Usuario  
ON (usuario1 = nick OR usuario2 = nick)  
WHERE nombre = 'Rosario';
```

Tipos de join

- ◆ INNER Por defecto (no hace falta ponerlo)
- ◆ LEFT | RIGHT | FULL Se añaden también filas que no cumplen la condición (incompatible con INNER)
- ◆ NATURAL La condición consiste en igualdad entre los campos que se llamen igual

Tipos de join (cont)

Ejemplo

```
CREATE TABLE Alumno (  
    dni VARCHAR(12) PRIMARY KEY, nombre text);  
CREATE TABLE Asignatura (  
    codigo NUMERIC PRIMARY KEY, nombre text);  
CREATE TABLE Notas (  
    dni VARCHAR(12) REFERENCES Alumno(dni),  
    codigo NUMERIC REFERENCES Asignatura(codigo),  
    teoria NUMERIC (4,2), practicas NUMERIC (4,2),  
    PRIMARY KEY (dni, codigo));  
  
SELECT nombre, teoria FROM Notas NATURAL JOIN Asignatura;  
  
SELECT nombre, teoria FROM Notas JOIN Asignatura  
ON Notas.codigo = Asignatura.codigo;
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual que algún otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Alias

```
SELECT campos FROM tabla AS alias [(alias-campo1, alias-campo2, ...)]  
[WHERE condición];
```

```
SELECT campo AS alias FROM ...
```

Ejemplo:

```
SELECT u1.nombre FROM Usuario AS u1, Usuario AS u2  
WHERE u1.nombre = u2.nombre AND u1.nick < > u2.nick;
```

```
SELECT dni, teoria * 0.6 + practicas * 0.4 AS media FROM Notas;
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Consultas anidadas

SELECT *campos* FROM (SELECT ...) AS *alias* WHERE ...;

SELECT *campos* FROM *tabla*

WHERE *campo1, campo2, ...* **IN** (SELECT *campo1, campo2, ...*);

SELECT *campos* FROM *tabla*

WHERE *campo comparación* (**SOME** | **ALL**) (SELECT ...);

SELECT *campos* FROM *tabla*

WHERE [NOT] **EXISTS** (SELECT ...);

SELECT *campos* FROM *tabla*

WHERE (SELECT ...) **CONTAINS** (SOME | ALL) (SELECT ...);

Consultas anidadas (cont)

Ejemplos:

```
SELECT seguidor1.usuario1  
FROM Contacto AS seguidor1 JOIN  
(SELECT * FROM Contacto WHERE Contacto.usuario2 = 'lola')  
AS seguidor2 /* En FROM siempre con AS */  
ON seguidor1.usuario2 = seguidor2.usuario1
```

```
SELECT usuario1 FROM Contacto  
WHERE usuario2 IN (SELECT usuario1 FROM Contacto  
WHERE usuario2 = 'lola')
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Álgebra de conjuntos

consulta1 UNION *consulta2*

consulta1 INTERSECT *consulta2*

consulta1 EXCEPT *consulta2*

Tuplas homogéneas: los conjuntos de tuplas tienen que tener los mismos campos
Aplica un DISTINCT implícito (a menos que indiquemos ALL)

Ejemplo:

```
(SELECT usuario2 FROM Contacto WHERE usuario1 = 'charo'  
UNION
```

```
SELECT usuario1 FROM Contacto WHERE usuario2 = 'charo')  
INTERSECT
```

```
(SELECT usuario2 FROM Contacto WHERE usuario1 = 'lola'  
UNION
```

```
SELECT usuario1 FROM Contacto WHERE usuario2 = 'lola')
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertas en la BD

Canciones de artistas del reino unido

Título de las canciones escuchadas por un usuario dado

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual que otro

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios dados

Cuántas veces ha sido escuchada una canción dada

Cuántas veces ha sido escuchado cada artista

Usuarios con más de dos contactos

Usuarios ordenados por n° de contactos

Usuario con más contactos

Orden, agregación

SELECT COUNT ([DISTINCT] *campo*) FROM *tabla* ...
[**GROUP BY** *campo1*, *campo2*, ...];

SELECT SUM | MAX | MIN | AVG (*campo*) FROM *tabla* ...
[**GROUP BY** *campo1*, *campo2*, ...];

SELECT ...

[**ORDER BY** *campo1*, *campo2*, ... [DESC]];

¿Por qué?



En general si se usa **GROUP BY**, sólo se pueden usar esos campos en **SELECT** (aunque algunos SGBD lo toleran)

Orden, agregación (cont)

Ejemplos:

```
SELECT COUNT (*) FROM Escucha JOIN Cancion ON cancion = id  
WHERE titulo = 'Norwegian Wood';
```

```
SELECT autor, COUNT (*) FROM Escucha JOIN Cancion ON cancion = id  
GROUP BY autor;
```

```
SELECT * FROM Usuario  
WHERE (SELECT COUNT (*) FROM Contacto  
WHERE usuario1 = nick OR usuario2 = nick) > 2;
```

```
SELECT * FROM Usuario  
ORDER BY (SELECT COUNT (*) FROM Contacto  
WHERE usuario1 = nick OR usuario2 = nick);
```

Vistas

CREATE VIEW *nombre* **AS** SELECT...;

Dan un nombre a una consulta

Útil para reutilizar consultas y evitar ejecutarlas varias veces

Pueden configurarse para que se almacenen en disco

Ejemplos:

```
CREATE VIEW Contactos_Usuario AS
SELECT u1.nick, u2.nombre FROM Usuario AS u1, Usuario AS u2
WHERE (u1.nick, u2.nick) IN
    ((SELECT usuario1, usuario2 FROM Contacto)
     UNION (SELECT usuario2, usuario1 FROM Contacto));

SELECT nombre FROM Contactos_Usuario WHERE nick = 'pepe';
```

Otros elementos de SQL

- ◆ Esquemas
- ◆ Dominios
- ◆ Triggers
- ◆ Asserts
- ◆ Transacciones
- ◆ ...