

BASES DE DATOS DISTRIBUIDAS

Sistemas informáticos I

3.2 SQL *STRUCTURED QUERY LANGUAGE*

Bases de datos distribuidas

LENGUAJES DE SQL

- **DDL:** Lenguaje de definición de datos (*Data Definition Language*)
 - Definición de esquemas de relación
 - Borrado de relaciones
 - Creación de índices
 - Modificación de esquemas de relación
 - Órdenes para la definición de vistas
 - Órdenes para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos
 - Órdenes para especificar derechos de acceso para las relaciones y vistas
 - ...
- **DML:** Lenguaje de manipulación de datos (*Data Manipulation Language*)
 - Incluye un lenguaje de consultas, basado en el álgebra relacional (y en el cálculo de tuplas)
 - Incluye órdenes para insertar, borrar, modificar y seleccionar (CRUD) tuplas de la base de datos



ELEMENTOS QUE YA DEBES CONOCER DE SQL

- Tipos de datos
- DDL: Creación y destrucción de tablas
- DDL: Definición de restricciones
 - Check: Restricción arbitraria
 - Not Null: El atributo no acepta valores nulos
 - Unique: El atributo no acepta valores repetidos
 - Primary Key: El atributo es clave primaria
 - Foreign Key: El atributo es clave extranjera
- DDL: Modificación de tablas
- DML: Consultas CRUD sencillas
 - INSERT
 - SELECT
 - UPDATE
 - DELETE/TRUNCATE
- Aquí vamos centrarnos en elementos directa o indirectamente relacionados con el rendimiento de una base de datos SQL en el contexto de un sistema distribuido



TIPOS DE DATOS EN SQL

- **char(n)**. Cadena de longitud fija. La longitud es n caracteres
- **varchar(n)**. Cadena de longitud variable. La longitud máxima es n caracteres
- **int/integer**. Entero
- **smallint**. Entero corto
- **numeric(p,d)**. Número en formato de coma fija, con precisión de p dígitos, con d dígitos a la derecha de la coma decimal. (1-> 0.9999)
- **real, double precision**. Número en coma flotante y número en coma flotante con doble precisión
- **float(n)**. Número en coma flotante con una precisión no menor de n dígitos
- El valor **NULL** esta permitido para todos los atributos a menos que se prohíba explícitamente
- La construcción **create domain** en SQL-92 crea tipos de datos definidos por el usuario:

```
create domain nombre-persona char(20) not null
```



DÍA Y HORA EN SQL

- **date**: fecha (día del año), año (4 dígitos), mes y día
 - Ej. date '2001-7-27'
- **time**: hora del día, en horas, minutos y segundos.
 - Ej. time '09:00:30' time '09:00:30.75'
- **timestamp**: día y hora
 - Ej. timestamp '2001-7-27 09:00:30.75'
- **Interval**: periodo de tiempo
 - Ej. Interval '1' day
 - la diferencia entre date/time/timestamp da un interval
 - Interval se puede sumar a date/time/timestamp
- Se pueden extraer valores independientes de date/time/timestamp:
 - Ej. extract (year from r.comienzo)



CREACIÓN Y DESTRUCCIÓN DE TABLAS

```
CREATE TABLE tablita (  
    nombre1 char(20),  
    nombre2 integer);
```

```
DROP TABLE tablita;
```



RESTRICCIÓN CHECK

Permite especificar que los valores de una columna deben satisfacer una expresión

```
CREATE TABLE productos (  
    producto_no integer,  
    nombre text,  
    precio numeric(10,2) CHECK (precio > 0)  
);
```

```
CREATE TABLE productos (  
    producto_no integer,  
    nombre text,  
    precio numeric(10,2) CONSTRAINT precio_positivo CHECK (precio > 0)  
);
```



RESTRICCIÓN NOT NULL

Indica que el atributo no puede valer NULL

```
CREATE TABLE productos (  
    producto_no integer NOT NULL,  
    nombre text NOT NULL,  
    precio numeric(10,2)  
);
```

```
CREATE TABLE productos (  
    producto_no integer NOT NULL,  
    nombre text NOT NULL,  
    precio numeric(10,2) NOT NULL CHECK (precio > 0)  
);
```



RESTRICCIÓN UNIQUE

Asegura que un determinado valor no está repetido en una columna

```
CREATE TABLE productos (  
    producto_no integer UNIQUE,  
    nombre text,  
    precio numeric(10,2)  
);
```

```
CREATE TABLE productos (  
    producto_no integer,  
    nombre text,  
    precio numeric,  
    UNIQUE(producto_no,nombre)  
);
```



RESTRICCIÓN PRIMARY KEY

Asegura que un conjunto de campos son clave primaria

```
CREATE TABLE productos (  
    producto_no integer PRIMARY KEY,  
    nombre text,  
    precio numeric  
);
```

```
CREATE TABLE ejemplo (  
    a integer,  
    b integer,  
    c integer,  
    PRIMARY KEY (a, c)  
);
```



RESTRICCIÓN REFERENCES

Asegura que los valores de una determinada columna debe ser idénticos a los valores que aparecen en otra determinada columna que puede estar en otra tabla

```
CREATE TABLE productos (  
    producto_no integer PRIMARY KEY,  
    nombre text,  
    precio numeric  
);  
CREATE TABLE pedidos (  
    pedido_no integer PRIMARY KEY,  
    producto_no integer REFERENCES productos(producto_no),  
    cantidad integer  
);
```



MODIFICAR ESTRUCTURA DE TABLA

```
ALTER TABLE productos ADD COLUMN a integer;
ALTER TABLE productos DROP a integer;
ALTER TABLE productos ADD CHECK (nombre <> '');
ALTER TABLE productos ADD CONSTRAINT some_name UNIQUE (producto_no);
ALTER TABLE productos
    ADD FOREIGN KEY (producto_group_id) REFERENCES grupo_productos;
ALTER TABLE productos ALTER COLUMN producto_no SET NOT NULL;
ALTER TABLE productos ALTER COLUMN precio SET DEFAULT 7.77;
ALTER TABLE productos RENAME COLUMN producto_no TO product_number;
```



OPERACIONES CRUD BÁSICAS

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

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

DELETE FROM tabla [WHERE condicion];

TRUNCATE tabla;

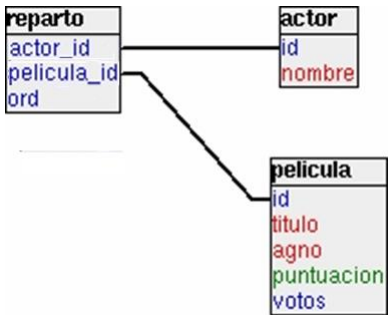
SELECT [DISTINCT] campo1, campo2, ... FROM tabla [WHERE condición];

SELECT * FROM tabla [WHERE condición];
```



BASE DE DATOS DE EJEMPLO

- Modelo de datos que utilizaremos para la mayoría de pruebas en clase
 - Puedes encontrar el script de creación y carga de la base de datos en Moodle



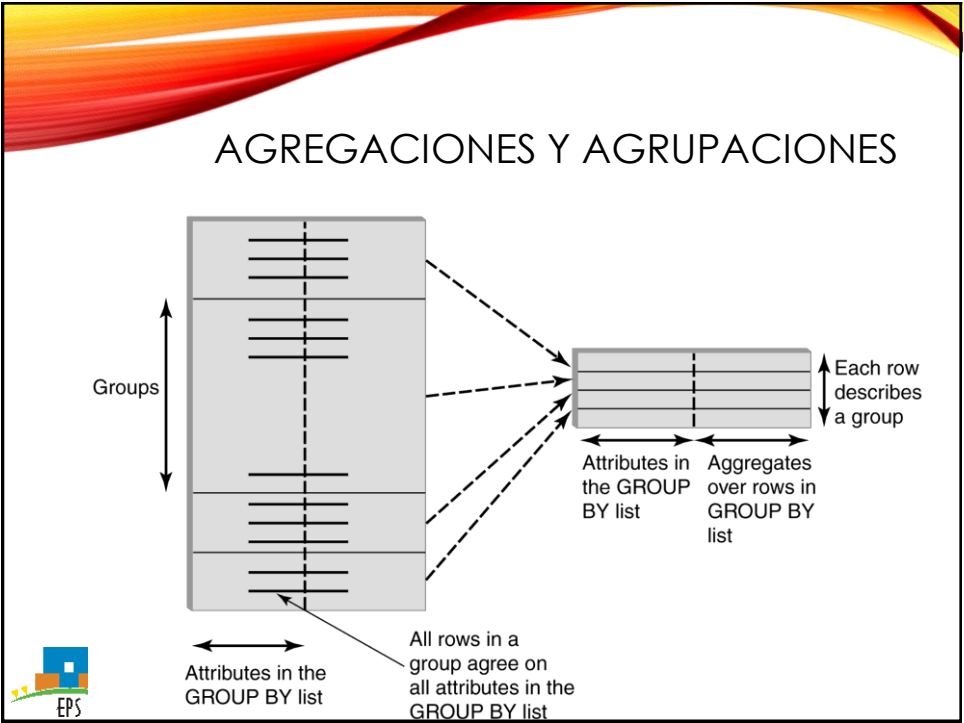
AGREGACIONES Y AGRUPACIONES

- Operadores que calculan un valor único a partir de una columna de valores: SUM, AVG, MIN, MAX, COUNT
 - SELECT COUNT(*) FROM tabla;
 - SELECT COUNT(campo) FROM tabla;
 - SELECT COUNT(DISTINCT campo) FROM tabla;
- Las agregaciones se puede realizar sobre todos los campos que cumplen los criterios de filtrado a agrupando registros según alguno de sus atributos
 - SELECT campo1, ..., campo_n SUM(campo) FROM table
 - GROUP BY campo1, ..., campo_n HAVING condicion

Si aparecen deben hacerlo en este orden

{	SELECT	}	obligatorios
	FROM		
	WHERE		
	GROUP BY		
	HAVING		
	ORDER BY		






COMPARACIÓN DE CARACTERES

- Con expresiones algebraicas: <, >, =, ...
- Con la instrucción like que permite búsquedas según un patrón (*wildcards*)
 - Comodines: `_` (un carácter) y `%`
 - Encontrar el título y puntuación de todas las películas que empiezan por 'Star'

```
SELECT titulo, puntuacion FROM pelicula
WHERE titulo LIKE 'Star%';
```

titulo	puntuacion
Star Wars	8.9
Star Trek: First Contact	8.2
Star Trek: The Wrath of Khan	7.5
Starship Troopers	7.1
Star Trek: Generations	7
Stargate	6.9
Star Trek IV: The Voyage Home	7.4
Star Trek: The Motion Picture	5.7
Star Trek III: The Search for Spock	6.2
Star Trek VI: The Undiscovered Country	7.3
Starman	6.9
Star Trek: The Next Generation - All Good Things...	8.9
Star Trek: The Next Generation - Encounter at Farpoint	7.5
Star Trek V: The Final Frontier	4.7

(14 rows)



CRUCES DE TABLAS

- Gran parte de la potencia de las bases relacionales se basa en la posibilidad de combinar/cruzar dos (o más) relaciones → producto (*join*)
- En un cruce se toman dos o mas relaciones y se obtiene otra relación, en principio con todos los atributos de las relaciones que se han cruzado
- El tipo de cruce más simple es el **producto cartesiano**: todos con todos
 - Enumerando cada relación en la clausula FROM
- Otros cruces tipos de cruces presentan sentencias específicas (depende del SGBD):
 - Condición en join: qué tuplas de las relaciones se corresponden
 - Where de join: qué atributos que estarán en la relación resultante
 - Tipo de Join: cómo son tratadas las tuplas de una relación que no tienen correspondencia en la otra relación (según la condición de join)



Tipo de Join
[inner] join left [outer] join right [outer] join full [outer] join

Condición en Join
natural on <predicado> using (A₁, A₂, ..., A_n)

PRODUCTO CARTESIANO

```
SELECT * FROM reparto WHERE pelicula_id=2;
```

pelicula_id	actor_id	ord
2	180	43
2	257	2
2	959	26
2	1033	12
2	1071	9
2	1099	8
2	1479	27
2	2196	33
2	2567	20
2	2581	14
2	2778	7
2	3308	45
2	4520	4
2	4612	21
2	4708	46
2	4789	36
2	4846	37
2	5161	3
2	5181	47
2	5328	39
2	5639	35
2	6702	11
2	6816	41
2	7006	44
2	7183	31
2	7502	30
2	7657	13
2	7810	5
2	8148	23
2	8483	1
2	8735	15
2	8767	38
2	8838	10
(33 rows)		

```
SELECT nombre FROM actor, reparto WHERE pelicula_id=2 AND actor_id=id;
```

nombre
Alexis Arquette
Amanda Plummer
Brenda Hillhouse
Bronagh Gallagher
Bruce Willis
Burr Steers
Christopher Walken
Don Blakely
Eric Clark
Eric Stoltz
Frank Whaley
Harvey Keitel
John Travolta
Josef Pilato
Julia Sweeney
Karen Maruyama
Kathy Griffin
Laura Lovelace
Lawrence Bender
Linda Kaye
Maria de Medeiros
Paul Calderon
Peter Greene (I)
Quentin Tarantino
Rich Turner
Robert Ruth
Rosanna Arquette
Samuel L. Jackson
Steve Buscemi
Tim Roth
Uma Thurman
Venessia Valentino
Ving Rhames
(33 rows)



MÁS SOBRE JOIN

- Relación *loan*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relación *borrower*

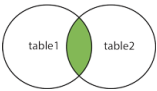
<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155



- No hay información en *borrower* para L-260 y no hay información en *loan* para L-155

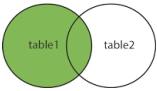
MÁS SOBRE JOIN

loan inner join borrower
on loan.loan-number = borrower.loan-number



<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>	<i>loan-number</i>
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230

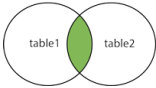
loan left outer join borrower
on loan.loan-number = borrower.loan-number



<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>	<i>loan-number</i>
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-260	Perryridge	1700	null	null

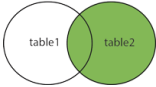


MÁS SOBRE JOIN




loan **natural inner join** borrower

loan-number	branch-name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

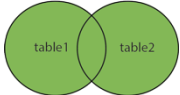


loan **natural right outer join** borrower

loan-number	branch-name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	null	null	Hayes




MÁS SOBRE JOIN



loan **full outer join** borrower using (loan-number)

loan-number	branch-name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	null
L-155	null	null	Hayes



COMBINACIÓN DE RELACIONES

- Para poder combinar dos o más relaciones deben ser compatibles:
 - **UNION**: unión de relaciones
 - **INTERSECT**: intersección de relaciones. Dependiente de SGBD
 - **EXCEPT**: resta relaciones . Dependiente de SGBD
- Estos operadores eliminan los duplicados
- Si se usa **ALL** los duplicados no se eliminan: p.ej., **UNION ALL**
- Ejemplo: Actores comunes a las películas Star Trek IV y Star Trek V

```
(SELECT nombre FROM pelicula,actor,reparto
WHERE titulo LIKE 'Star Trek V:%' AND
  pelicula_id=pelicula.id AND
  actor_id=actor.id)
INTERSECT (SELECT nombre FROM pelicula,actor,reparto
WHERE titulo LIKE 'Star Trek IV:%' AND
  pelicula_id=pelicula.id AND
  actor_id=actor.id );
```

```
nombre
-----
DeForest Kelley
George Takei
James Doohan
Leonard Nimoy
Nichelle Nichols
Walter Koenig
William Shatner
(7 rows)
```



SUBCONSULTAS

- Hasta ahora las condiciones en **WHERE** involucraban valores escalares, pero pueden aparecer subconsultas como parte de la condición descrita en una cláusula **WHERE**
 - Pueden devolver un valor (**S**)
 - **s op ALL R**, **op** = {<,>,<>,<=>}: cierto si se cumple para todos los valores de **R**
 - **s op ANY R**, **op** = {<,>,<>,<=>}: cierto si se cumple para al menos un valor de **R**
 - Pueden devolver una relación (**R**) que será procesada valor por valor:
 - **S IN R**: cierto si **S** está en **R**
 - **EXISTS R**: cierto si **R** es una relación no vacía
- También es posible usar una subconsulta en lugar de una relación ya almacenada en la cláusula **FROM**
 - Dependiendo del SGBD puede ser obligatorio proporcionar un alias para la subconsulta
 - La sintaxis para proporcionar el alias también puede variar ligeramente dependiendo del SGBD




SUBCONSULTAS

Reparto de 'Star Wars' que tiene id 1 (utilizamos dos tablas)

```
SELECT nombre FROM actor, reparto
WHERE pelicula_id=1 AND actor_id=actor.id;
```

nombre
Alec Guinness
Angus MacInnes
Anthony Daniels
Carrie Fisher
David Prowse
Denis Lawson
Don Henderson (II)
Garrick Hagon
Harrison Ford
Jack Purvis
Jeremy Sinden
Kenny Baker (I)
Leslie Schofield
Mark Hamill
Peter Mayhew (II)
Peter Cushing
Richard Le Parmentier
Shelagh Fraser
William Hootkins


(19 rows)



SUBCONSULTAS

- En lugar de realizar un producto escalar podemos hacer primero una "subconsulta" que devuelva los identificadores de los actores (utilizamos una tabla inicialmente)

```
SELECT nombre FROM actor
WHERE id IN
  (SELECT DISTINCT actor_id FROM reparto
   WHERE pelicula_id = 1);
```



ALGUNA CONSULTA COMPLEJA USANDO TODO LO VISTO

- Obtener todos los actores que han sido el actor principal en al menos 10 películas ordenado por numero de veces que han sido la estrella

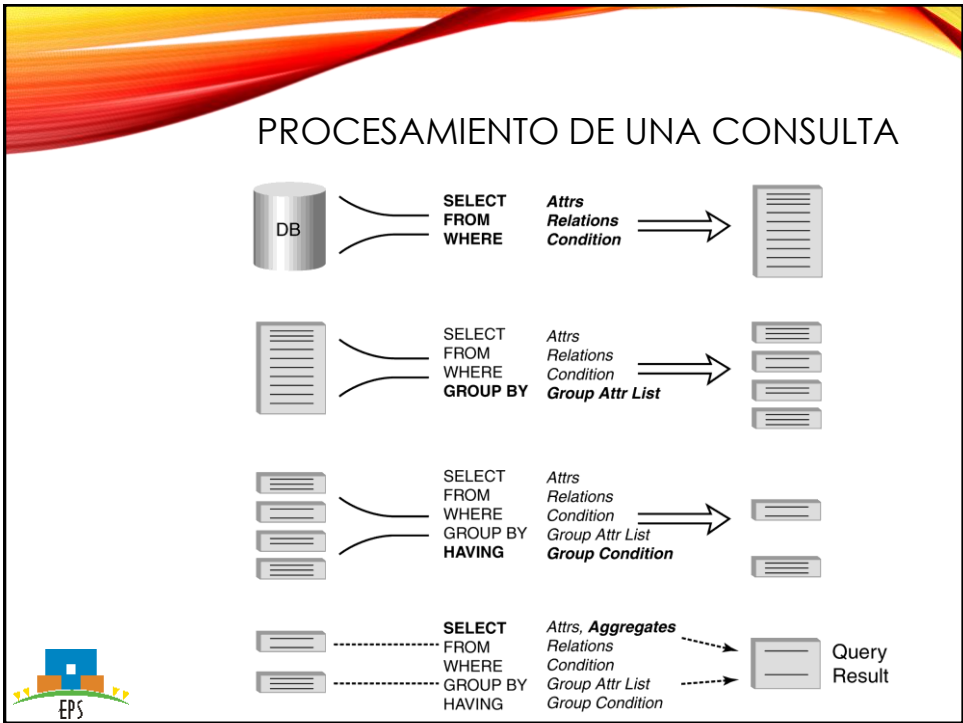
```
SELECT nombre, n_protas FROM
actor JOIN
  (SELECT actor_id, count(*) AS n_protas
   FROM reparto WHERE ord = 1
   GROUP BY actor_id HAVING count(*) >= 10) protas
ON actor.id = protas.actor_id
ORDER BY n_protas DESC, nombre;
```



PROCESAMIENTO DE UNA CONSULTA

1. Crear el producto descrito en FROM
2. Aplicar las restricciones descritas en WHERE
3. Si no hay GROUP BY, proyectar la relación (2) según describe SELECT → Fin
4. Agrupar las tuplas por valores tal y como especifica por GROUP BY
5. Aplicar HAVING
6. Aplicar SELECT → Fin





UN POCO SOBRE RENDIMIENTO


```
SELECT star1.nombre, star1.id FROM actor star1, actor star2
WHERE star1.nombre = star2.nombre AND star1.id < star2.id;

SELECT nombre, id FROM actor star1
WHERE nombre = ANY (SELECT nombre FROM actor
WHERE id < star1.id AND nombre = star1.nombre);

SELECT nombre, id FROM actor star1
WHERE EXISTS (SELECT NULL FROM actor
WHERE id < star1.id AND nombre = star1.nombre);

SELECT nombre, id FROM actor star1
WHERE nombre IN (SELECT nombre FROM actor
WHERE id < star1.id AND nombre = star1.nombre);
```

nombre	id
John Luxie	4432
Marc Lawrence (I)	5565



UN POCO SOBRE RENDIMIENTO

```
SELECT titulo
  FROM actor, pelicula, reparto
 WHERE nombre = 'Harrison Ford' AND
        actor.id=actor_id AND
        pelicula_id=pelicula.id;

SELECT titulo FROM pelicula
 WHERE id IN
    (SELECT pelicula_id FROM reparto
     WHERE (actor_id) IN
        (SELECT id FROM actor
         WHERE nombre = 'Harrison Ford'));
```

titulo
Star Wars
Blade Runner
Empire Strikes Back, The
Raiders of the Lost Ark
Apocalypse Now
Indiana Jones and the Last Crusade
Indiana Jones and the Temple of Doom
Witness
Air Force One
American Graffiti
Patriot Games
Working Girl
Sabrina
Presumed Innocent
Devil's Own, The
Conversation, The
Frantic
Mosquito Coast, The
Regarding Henry
Force 10 from Navarone

