SQL Intermedio

Funciones de agregación

Contenido

- Operaciones de conjuntos
- Valores nulos
- Funciones de agregación
- Subconsultas anidadas
- DML y otras operaciones CRUD

Operaciones de conjuntos

UNION, INTERSECT y EXCEPT

Condiciones de compatibilidad:

- R y S deben tener la misma aridad (aridad = número de atributos)
- El dominio del *i*-ésimo atributo de R debe corresponder con el dominio del *i*-ésimo atributo de S

UNION: La unión de dos tablas A y B es la relación A ∪ B que contiene todas las tuplas de A, todas las de B y las que tienen en común A y B.

INTERSECT: La intersección de dos tablas A y B es el la relación A ∩ B que contiene todas las tuplas que tienen en común A y B.

EXCEPT: La diferencia entre dos tablas A y B es la relación A \ B que contiene todas las tuplas de A que no pertenecen a B.

Valores nulos

Definición

Un valor nulo (null) es un valor desconocido, por lo que no se puede comparar con un string con = o <>

Para comparar con nulo se usa las palabras clave

- IS NULL
- IS NOT NULL

```
SELECT *
FROM country
WHERE life_expectancy IS NULL;
```

Resultados importantes con NULL

- Expresión aritmética con NULL resulta en NULL

```
SELECT x.value_1 +
FROM tabla1 AS x;
```

- Cualquier comparación que compare un valor contra NULL, retorna un tipo *unknown*

```
SELECT 1 < NULL;
```

- Los valores booleanos saben cómo comportarse con valores *unknown*
  - true AND unknown is unknown, false AND unknown is false, unknown AND unknown is unknown .
  - true OR unknown is true, false OR unknown is unknown, unknown OR unknown is unknown
  - NOT unknown is unknown.

Funciones de agregación

Funciones de agregación/resumen

El estándar ISO define cinco funciones:

- COUNT: devuelve el número de valores en una columna especificada
- SUM: devuelve la suma de los valores en una columna específica
- AVG: devuelve el promedio de los valores en una columna específica
- MIN: devuelve el valor más pequeño en una columna específica
- MAX: devuelve el mayor valor en una columna específica

Síntaxis

- ¿Cuántas ciudades tiene Colombia?

```
SELECT COUNT(*)
FROM city
WHERE country_code = 'COL';
```

- ¿Cuánto es el Producto Interno Bruto (GNP) promedio?

```
SELECT AVG(GNP)
FROM country;
```

	avg numeric
1	122823.882427

- ¿Cuántos países superan el promedio de Producto Interno Bruto (GNP)?

```
SELECT COUNT(*)
FROM country
WHERE GNP > 122823.882427;
```

```
SELECT COUNT(*)
FROM country
WHERE GNP > (SELECT AVG(GNP
             FROM country);
```

Síntaxis

- ¿Cuáles son los nombres de los continentes sin repetir?

```
SELECT DISTINCT continent
FROM country;
```

- ¿Cuántos continentes diferentes hay?

```
SELECT COUNT(DISTINCT continent)
FROM country;
```

- Varias agregaciones en simultáneo

```
SELECT
   MIN(population) AS min_poblacion,
   AVG(population) AS prom_poblacion,
   MAX(population) AS max_poblacion
FROM city WHERE country_code = 'COL';
```

Algunas características importantes

Las funciones de agregación

- Operan en una sola columna de la tabla
- Retornan un solo valor
- COUNT, MIN, MAX funciona con campos numéricos y no numéricos
- MIN, MAX funcionan con campos numéricos y de texto
- Las funciones eliminan los nulos antes de operar (menos COUNT(*))
- COUNT(*) es especial, cuenta todas las columnas de la tabla
- Se pueden eliminar duplicados antes de contar usando DISTINCT (no funciona para MIN ni MAX)
- Se pueden usar solo el la lista del SELECT o en la clausula HAVING

Agregación con agrupación

Las agregaciones son como filas de totales

Podemos hacer subtotales, agrupar (GROUP BY) por alguna de las columnas seleccionadas en el SELECT

```
SELECT AVG(length)
FROM film;
```

avg
115.272

```
SELECT rating,
       AVG(length)
FROM film
GROUP BY rating;
```

rating	avg
PG-13	120.443…
PG	112.005…
NC-17	113.228…
G	111.050…
R	118.661…

Agregación con agrupación

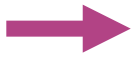
En el SELECT solo pueden haber nombres de columnas, agregaciones, o constantes

Columnas de agrupación
Toda columna del SELECT debe estar acá, a menos que sea agregación

```
SELECT rating,
       AVG(length)
FROM film
GROUP BY rating;
```

Agregación con agrupación


```sql
SELECT rating,
       ROUND(AVG(length),2) AS mean_length
FROM film
WHERE rental_rate BETWEEN 3 AND 5
GROUP BY rating
ORDER BY AVG(length) DESC;
```



rating	mean_length
PG-13	123.36
R	115.89
NC-17	114.46
G	112.96
PG	110.96

Filtro HAVING

Filtro WHERE afecta a las tuplas

Filtro HAVING permite filtrar agrupaciones y no a las tuplas

```
SELECT
rating,
ROUND(AVG(length),2) AS mean_length
FROM film
GROUP BY rating;
```

rating	mean_length
PG-13	120.44
R	118.66
NC-17	113.23
PG	112.01
G	111.05

```
SELECT
rating,
ROUND(AVG(length),2) AS mean_length
FROM film
GROUP BY rating
HAVING AVG(length) > 115;
```

rating	mean_length
PG-13	120.44
R	118.66

Subconsultas anidadas

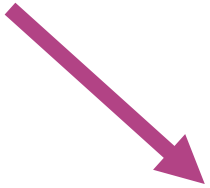
Subconsulta

- Una subconsulta es una expresión SELECT-FROM-WHERE que está anidada dentro de otra consulta
- Usos comunes
  - Pruebas de pertenencia
  - Comparaciones de conjuntos
  - Pruebas de relaciones vacías
  - Prueba de ausencia de duplicados
  - Subconsultas en el FROM
  - CTE (Common Table Expressions)
  - Subconsultas de escalares

Pruebas de pertenencia

Ejemplo: Consulte el código (film_id) de las películas en las que han actuado al tiempo el actor con id 177 y el actor con código 176

```
(SELECT film_id
FROM film_actor
WHERE actor_id = 176);
```



```
SELECT film_id
FROM film_actor
WHERE actor_id = 177 AND film_id IN
    (SELECT film_id
    FROM film_actor
    WHERE actor_id = 176);
```

Pruebas de pertenencia

Se pueden hacer con conjuntos enumerados

```
SELECT * FROM film WHERE rating IN ('PG','PG-13');
```

Se pueden hacer con múltiples atributos

```
SELECT DISTINCT staff_id
FROM staff
WHERE (store_id, staff_id) IN (SELECT store_id, manager_staff_id
                               FROM store);
```

Comparación de conjuntos

Consulte el listado de películas (nombre y duración) cuya duración es mayor que por lo menos una película de la categoría **R**

```
SELECT DISTINCT x.title, x.length
FROM film AS x, film AS y
WHERE x.length > y.length AND y.rating = 'R';
```

ANY/SOME

```
SELECT title, length
FROM film
WHERE length > ANY (SELECT length
                    FROM film
                    WHERE rating = 'R');
```

ALL

```
SELECT title, length
FROM film
WHERE length > ALL (SELECT length
                    FROM film
                    WHERE rating = 'NC-17');
```

Comparación de conjuntos

Consulte la categoría que tiene el promedio de duración más grande de todas las categorías

```
SELECT rating, AVG(length)
FROM film
GROUP BY rating
HAVING AVG(length) >= ALL (SELECT AVG(length)
                           FROM film
                           GROUP BY rating);
```

Pruebas de relaciones vacías

```
SELECT film_id
FROM film_actor
WHERE actor_id = 177 AND film_id IN
    (SELECT film_id
    FROM film_actor
    WHERE actor_id = 176);
```

```
SELECT fa_outer.film_id
FROM film_actor AS fa_outer
WHERE fa_outer.actor_id = 177 AND EXISTS
    (SELECT *
     FROM film_actor AS fa_inner
     WHERE
       fa_inner.actor_id = 176 AND
       fa_inner.film_id = fa_outer.film_id);
```

Retorna TRUE si la subconsulta que entra como argumento retorna una tupla o más

Pruebas de relaciones vacías

```
SELECT fa_outer.film_id
FROM film_actor AS fa_outer
WHERE fa_outer.actor_id = 177 AND EXISTS
    (SELECT *
     FROM film_actor AS fa_inner
     WHERE
     fa_inner.actor_id = 176 AND
     fa_inner.film_id = fa_outer.film_id);
```

Subconsulta correlacionada : Subconsulta que usa un nombre de correlación de una consulta externa

Prueba de relaciones vacías

¿Qué hace esta consulta?

Ejecutar por **partes**

```
SELECT
    DISTINCT f_bystore.film_id, f_bystore.title
FROM
(
    SELECT
        s.store_id,
        f.film_id,
        f.title,
        COUNT(f.film_id)
    FROM filmf JOIN inventory i ON f.film_id = i.film_id
               JOIN store s ON i.store_id = s.store_id
    GROUP BY s.store_id, f.film_id, f.title
) AS f_bystore
GROUP BY f_bystore.film_id, f_bystore.title
HAVING COUNT(*) = (SELECT COUNT(*) FROM store);
```

Prueba de relaciones vacías

Forma alterna usando pruebas de relaciones vacías

```
SELECT DISTINCT f.film_id, f.title
FROM filmf
WHERE NOT EXISTS ((SELECT store_id
    FROM store
    ) EXCEPT (SELECT i.store_id FROM inventory i  WHERE f.film_id = i.film_id));
```

Prueba de ausencia de tuplas duplicadas

UNIQUE retorna TRUE si en la subconsulta en el argumento no retorna tuplas duplicadas

Actores que han actuado por mucho en una película (No implementado en PostgreSQL)

```sql
SELECT *
FROM actor AS a
WHERE UNIQUE (
    SELECT film_id
    FROM film_actor
    WHERE actor_id = a.actor_id);
```

```sql
SELECT *
FROM actor AS a
WHERE 1 >= (
    SELECT COUNT(fa.film_id)
    FROM film_actor fa
    WHERE fa.actor_id = a.actor_id);
```

Subconsultas en el FROM

Subconsultas anidadas en el FROM no permiten subconsultas correlacionadas

```
SELECT MAX(total_length)
FROM (SELECT rating, SUM(length)
        FROM film
        GROUP BY rating) AS sumratings (cat, total_length);
```

Solucionado desde la versión 2013 con la palabra LATERAL

```
SELECT fi_1.title, fi_1.length, fi_1.rating, fi_2.sum_rating
FROM filmfi_1, LATERAL (SELECT SUM(length) AS sum_rating
                        FROM film
                        WHERE fi_1.rating = rating) AS fi_2;
```

Common Table Expressions (CTE)

Resultados de consultas reusables en otras partes de la consulta

SÍNTAXIS

```
WITH <subquery name> AS (<subquery code>) [, ...]
SELECT <Select list> FROM <subquery name>;
```

Common Table Expressions (CTE)

Ejemplo

```
WITH activos AS
    (
    SELECT *
    FROM   payment pay
             JOIN   customer cus
               ON     pay.customer_id = cus.customer_id
    WHERE  active = 0)
SELECT    country,
          SUM(amount)
FROM      activos cus
            JOIN     address adr
              ON       cus.address_id=adr.address_id
            JOIN     city cty
              ON       adr.city_id=cty.city_id
            JOIN     country ctr
              ON       cty.country_id = ctr.country_id
GROUP BY country
ORDER BY SUM(amount) DESC LIMIT 10;
```

Subconsulta de escalares


```
SELECT store_id, (SELECT COUNT(*)
                  FROM inventory
                  WHERE store.store_id = inventory.store_id)
FROM store;
```

DML y otros operadores CRUD

Actualizar la base de datos con INSERT

Para insertar una columna se usa el commando INSERT INTO, de acuerdo a la siguiente síntaxis:

```
INSERT INTO nombreTabla (col1, col2, ...)
VALUES (valor1, valor2,...);
```

- nombreTabla: Tabla donde se agregará el registro
- col1, col2, ... : Nombre de las columnas de la table
- valor1, valor2, ... : Valores que irán en la nueva tupla

Actualizar la base de datos con INSERT

INSERT sencillo

```
INSERT INTO orders
    (order_id, order_date, amount, customer_id)
VALUES (34, '03/14/1760', 45.6, 1);
```

INSERT con tabla como parámetro

```
INSERT INTO orders
    (order_id,
     order_date,
     amount,
     customer_id)
VALUES
        (SELECT * FROM orders WHERE order_id = 1);
```

Stackoverflow

El lenguaje de consulta estructurado (SQL) es un lenguaje para consultar bases de datos.

Las preguntas deben incluir:

- ejemplos de código,
- estructura de tabla,
- datos de muestra y
- una etiqueta para la implementación de DBMS (por ejemplo, MySQL, PostgreSQL, Oracle, MS SQL Server, IBM DB2, etc.) que se utiliza.

Si su pregunta se relaciona únicamente con un DBMS específico (usa extensiones / características específicas), use la etiqueta de ese DBMS en su lugar. Las respuestas a las preguntas etiquetadas con SQL deben usar SQL estándar ISO / IEC.

Actividad

Cree una tabla de prueba e inserte una fila usando SQL Fiddle

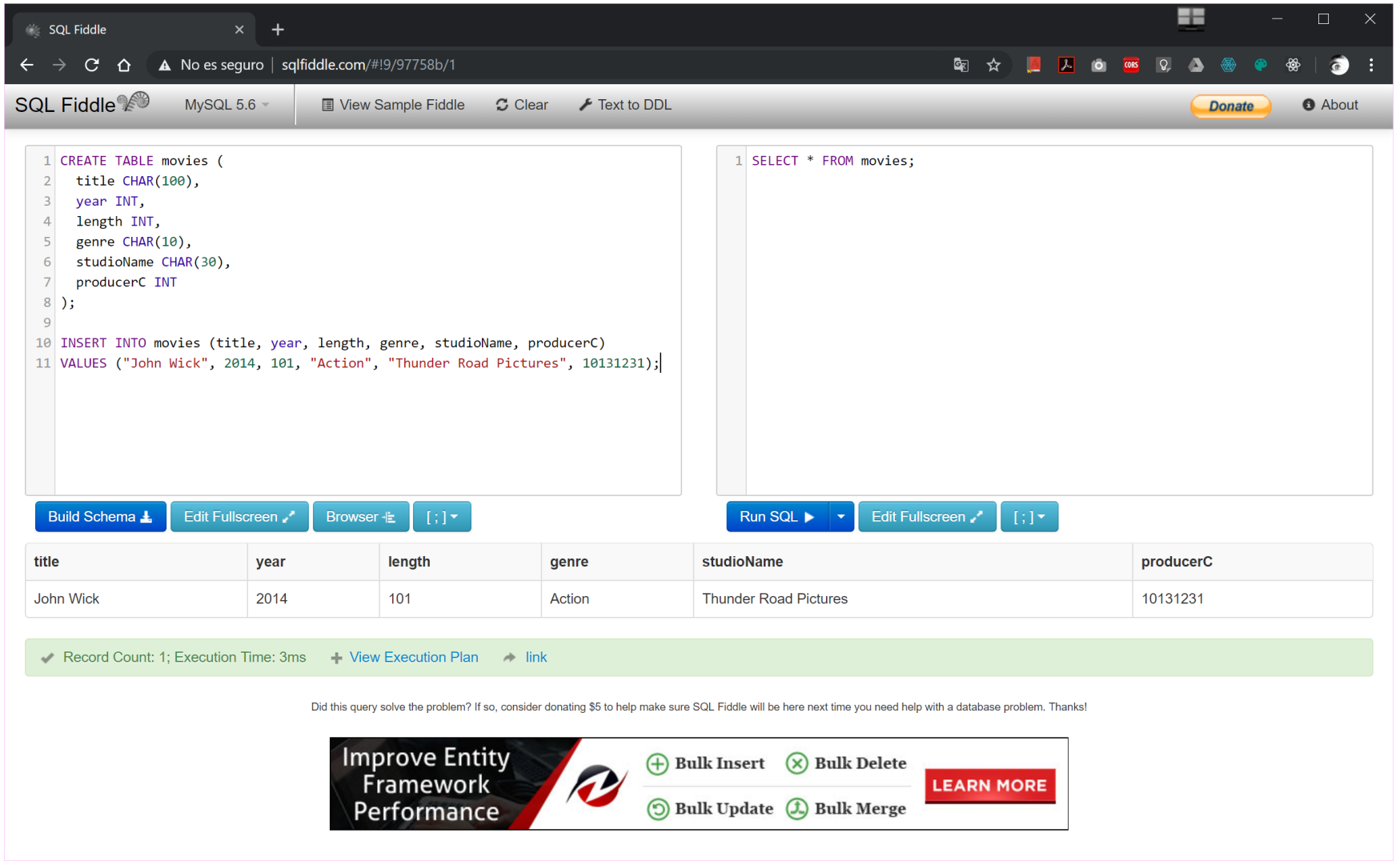
```
CREATE TABLE movies (
  title CHAR(100),
  year INT,
  length INT,
  genre CHAR(10),
  studioName CHAR(30),
  producerC INT
);
```

DDL

DML

```
INSERT INTO movies (title, year, length, genre, studioName, producerC)
VALUES ("John Wick", 2014, 101, "Action", "Thunder Road Pictures", 10131231);
```

SQL Fiddle


SQL Fiddle
No es seguro | sqlfiddle.com/#!9/97758b/1
SQL Fiddle
MySQL 5.6
View Sample Fiddle
Clear
Text to DDL
Donate
About
1 CREATE TABLE movies (
2 title CHAR(100),
3 year INT,
4 length INT,
5 genre CHAR(10),
6 studioName CHAR(30),
7 producerC INT
8);
9
10 INSERT INTO movies (title, year, length, genre, studioName, producerC)
11 VALUES ("John Wick", 2014, 101, "Action", "Thunder Road Pictures", 10131231);
1 SELECT * FROM movies;
Build Schema
Edit Fullscreen
Browser
[;]
Run SQL
Edit Fullscreen
[;]
title
year
length
genre
studioName
producerC
John Wick
2014
101
Action
Thunder Road Pictures
10131231
Record Count: 1; Execution Time: 3ms
View Execution Plan
link
Did this query solve the problem? If so, consider donating $5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!
Improve Entity Framework Performance
Bulk Insert
Bulk Delete
Bulk Update
Bulk Merge
LEARN MORE

La sentencia UPDATE

- Actualizar todas las filas

```
UPDATE orders
SET amount = amount * 1.10;
```

- Actualizar todas las filas que correspondan con la clausula WHERE

```
UPDATE orders
SET amount = amount * 1.10
WHERE customer_id = 1;
```

- Actualizar columnas múltiples

```
UPDATE orders
SET amount = amount * 1.10, order_date = REPLACE(order_date,'/','-')
WHERE customer_id = 1;
```

La sentencia DELETE

- Borra todas las filas

```
DELETE FROM orders;
```

- Borra todas las filas que correspondan con la clausula WHERE

```
DELETE FROM orders
WHERE customer_id = 1;
```