SQL III

Base de Datos 2024

Consultas Anidadas

CONSULTAS EN SQL - CONSULTAS ANIDADAS

```
SELECT ...
FROM ...
WHERE [SUBQUERY]

SELECT ...
FROM [SUBQUERY]
WHERE ...

SELECT ... , [SUBQUERY], ...
FROM ...
WHERE ...
```

- Una subquery es una consulta anidada en
  - Un **WHERE**
  - Un **FROM**
  - Una **COLUMNA**
- Una subquery anidada en una columna se denomina **subquery escalar**

CONSULTAS ANIDADAS - SET MEMBERSHIP

```
SELECT ...
FROM ...
WHERE (columns) [IN | NOT IN]
[SUBQUERY]

SELECT ...
FROM ...
WHERE (columns) [IN | NOT IN]
[ENUMERATION]
```

```
SELECT DISTINCT course_id
FROM section
WHERE semester = 'Fall' AND year= 2009 AND
      course_id IN (
            SELECT course_id
            FROM section
            WHERE semester = 'Spring'
               AND year= 2010);

SELECT name
FROM instructor
WHERE name NOT IN ('Mozart', 'Einstein');
```

CONSULTAS ANIDADAS - SET COMPARISON

```
SELECT ...
FROM ...
WHERE (columns)
comp SOME [SUBQUERY]

SELECT ...
FROM ...
WHERE (columns)
comp ALL [SUBQUERY]

comp := <,<=,>,>=, <>, =
```

```
SELECT name
FROM instructor
WHERE salary > ALL (
        SELECT salary
        FROM instructor
        WHERE dept_name = 'Biology');
```

- Para pensar:
  - = SOME == IN ??
  - <> SOME == NOT IN ??
  - <> ALL == NOT IN ??
  - = ALL == IN ??

CONSULTAS ANIDADAS - EMPTY RELATIONS

```
SELECT ...
FROM ...
WHERE EXISTS [SUBQUERY]

SELECT ...
FROM ...
WHERE NOT EXISTS [SUBQUERY]
```

```
SELECT course
FROM section AS S
WHERE semester = 'Fall'
  AND year= 2009
  AND EXISTS (
      SELECT *
      FROM section AS T
      WHERE semester = 'Spring'
        AND year= 2010
        AND S.course_id = T.course_id)
```

- "tabla A contiene a tabla B" == "not exists (B except A)."

CONSULTAS ANIDADAS - CORRELATED SUBQUERIES

- Ya vimos que se pueden renombrar las tablas en una consulta:

```
SELECT  t.ID, i.ID
FROM instructor AS i, teaches t
```

- El alias se denomina **nombre de correlación.**
- SQL permite referenciar un nombre de correlación introducido en una query externa en una subquery anidada en el **WHERE.**
- Una subquery que usa un nombre de correlación de una query externa es una **subquery correlacionada.**
- **Regla de Alcance:** en una subquery se pueden usar solo nombres de correlación definidos en la propia subquery o en cualquier query que la contenga.
- Si un nombre de correlación se define localmente en una subquery y globalmente en la query que la contiene, la definición local tiene precedencia.

CONSULTAS ANIDADAS - CORRELATED SUBQUERIES

```
SELECT ...
FROM [SUBQUERY]
WHERE ...
```

Concepto clave: Un query retorna una tabla y por lo tanto puede aparecer en otra query en cualquier lugar donde una tabla es esperada.

```
SELECT dept name, avg_salary
FROM (
    SELECT dept_name,
           avg (salary) as avg_salary
    FROM instructor
    GROUP BY dept_name
) WHERE avg_salary > 42000;
```

Subqueries en un **FROM** no pueden usar nombres de correlación de otras tablas en el **FROM**.

SQL:2003 introduce el keyword **LATERAL** para permitirlo.

CONSULTAS ANIDADAS - SCALAR SUBQUERIES

```
SELECT ... , [SUBQUERY], ...
FROM ...
WHERE ...
```

- Las consultas escalares pueden ocurrir en un SELECT, un WHERE, y un HAVING.

Concepto clave: Podemos utilizar una subconsulta donde se espera una expresión siempre que la consulta retorne una fila con una sola columna. Tales consultas se denominan **consultas escalares.**

```
SELECT d.dept_name,
    (SELECT count(*)
     FROM instructor i
     WHERE d.dept_name =  i.dept_name
    ) AS num_instructors
FROM department d;
```

Common Table Expressions

COMMON TABLE EXPRESSIONS (CTE)

```
WITH
    cte_name AS (SUBQUERY)
    [,cte_name AS (SUBQUERY)]...
QUERY
```

- **WITH** permite definir tablas temporarias que están disponibles para la query asociada al **WITH**.
- Mejoran la legibilidad de las consultas.

```
WITH
    max_budget (value) AS (
        SELECT max(budget)
        FROM department
    )
SELECT budget
FROM department, max_budget
WHERE department.budget =   max_budget.value
```

Agregaciones

CONSULTAS EN SQL - AGREGACIONES

- SQL provee un conjunto de funciones de agregación.
- Una función de agregación toma una colección de valores y retorna un solo valor.
- Algunas funciones de agregación:
  - **max(C_i)**
  - **min(C_i)**
  - **avg(C_i)**
  - **count(C_i) | count(distinct C_i)**
  - **sum(C_i)**
  - **Otras en MySQL**

- Ejemplo:

```
SELECT avg(salary) AS avg_salary,
       max(salary) AS max_salary,
       min(salary) AS min_salary
FROM instructor
WHERE dept_name = 'Music';
```

CONSULTAS EN SQL - GROUP BY

```
SELECT select_expr
FROM table_expr
[WHERE where_condition]
[GROUP BY {col|alias|pos},]
[ORDER BY order_expr]
```

- **GROUP BY** permite calcular las agrupaciones sobre un grupo de filas.
- Las columnas especificadas en el **GROUP BY** definen los grupos. A igual valor, mismo grupo.
- Si se omite el **GROUP BY**, todas las filas se tratan como parte de un solo grupo.
- Las columnas en el **SELECT** que no sean agregadas deben aparecer en el **GROUP BY**.

CONSULTAS EN SQL - GROUP BY

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

```
SELECT dept_name, avg(salary) AS salary
FROM instructor
GROUP BY dept_name;
```

dept_name	*salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

CONSULTAS EN SQL - HAVING

```
SELECT select_expr
FROM table_expr
[WHERE where_condition]
[GROUP BY {col|alias|pos},]
[HAVING where_condition]
[ORDER BY order_expr]
```

- **HAVING** permite filtrar grupos sobre los valores agregados.
- Importante:
  - El predicado de HAVING se chequea **luego** de que los grupos son computados.
  - El predicado del WHERE se chequea **antes** de que los grupos sean computados.

CONSULTAS EN SQL - HAVING

ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

```
SELECT dept_name, avg(salary) AS salary
FROM instructor
GROUP BY dept_name
HAVING avg(salary) > 42000;
```

dept_name	salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

dept_name	avg(avg_salary)
Physics	91000
Elec. Eng.	80000
Finance	85000
Comp. Sci.	77333
Biology	72000
History	61000