SQL II

Base de Datos 2019

CONSULTAS EN SQL - OPERACIONES DE CONJUNTO

```
SELECT ...
UNION [ALL]
SELECT ...

SELECT ...
INTERSECT [ALL]
SELECT ...

SELECT ...
EXCEPT [ALL]
SELECT ...
```

SQL provee las siguientes operaciones de conjunto: **union, intersect y except.**

Todas:

- Operan sobre tablas.
- Eliminan automaticamente los duplicados.
- Para retener duplicados hay que usar **ALL**

CONSULTAS EN SQL - UNION

```
(
  SELECT course_id
  FROM section
  WHERE semester = 'Fall' AND year= 2009
)

 UNION

(
  SELECT course_id
  FROM section
  WHERE semester = 'Spring' AND year= 2010
)
```

course_id
CS-101
CS-347
PHY-101

course_id
CS-101
CS-315
CS-319
CS-319
FIN-201
HIS-351
MU-199

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

CONSULTAS EN SQL - INTERSECT

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

course_id
CS-101
CS-347
PHY-101

course_id
CS-101
CS-315
CS-319
CS-319
FIN-201
HIS-351
MU-199

course_id
CS-101

CONSULTAS EN SQL - EXCEPT

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

course_id
CS-101
CS-347
PHY-101

course_id
CS-101
CS-315
CS-319
CS-319
FIN-201
HIS-351
MU-199

course_id
CS-347
PHY-101

CONSULTAS EN SQL - NULL VALUES

- En una operación aritmética:

  NULL (+ | - | * | /) X = NULL

- En operaciones booleanas:

  NULL **AND** TRUE = NULL
  NULL **AND** FALSE = FALSE
  NULL **AND** NULL = NULL
  NULL **OR** TRUE = TRUE
  NULL **OR** FALSE = NULL
  NULL **OR** NULL = NULL
  NOT NULL = NULL

- Si el predicado de un WHERE evalúa a FALSE o NULL para una tupla, la misma no forma parte del resultado.
- Para testear si un valor es NULL:

```
WHERE salary IS null;

WHERE salary IS NOT null;
```

- Todas las funciones de agregación, excepto **COUNT**, ignoran los valores nulos.

CONSULTAS EN SQL - CONSULTAS ANIDADAS

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

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

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

- Una subquery es una consulta anida 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 (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.
- Mejoran la performance de la consulta *

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