SQL

Introducción

El lenguaje SQL (*Structured Query Language*) es el lenguaje estándar para trabajo con bases de datos relacionales.
Permite la definición, acceso y control de datos en una base de datos relacional.
Está basado principalmente en el álgebra relacional.
Componentes:

- Lenguaje para definición de datos. Permite la definición de esquemas, borrado de relaciones, creación de índices y modificación de esquemas.
- Control. Permite definir vistas, especificar derechos de acceso a relaciones y especificar restricciones de integridad.
- Lenguaje para manipulación de datos. Instrucciones para insertar, borrar y modificar tuplas así como para consultar tablas.
- Control de transacciones. Permiten especificar los límites de una transacción así como bloques explícitos de datos para controlar la concurrencia.

Uso de SQL

Uso directo, interactivo.

```
SELECT nombre
FROM   alumno
WHERE  calificacion > 8 ;
```

Uso desde un programa de aplicación (JAVA):

```
Statement s = c.createStatement();
for (int i = 0; i < cuentas.length; i++)
  s.executeUpdate(" UPDATE cuentas "+
                  "SET    balance = "+ cuentas[i].getBalance()
                  "WHERE  id = " + cuentas[i].getId());
```

o bien

```
ResulSet r =s.executeQuery("SELECT nombre FROM alumno" +
                           "WHERE  id = " + cuentas[i].getId()
//Imprime resultados
```

Esquema de una base de datos

```
Sucursal (nombreSucursal, ciudad, activo)
Cliente  (nombreCliente, calle, ciudad)
Prestamo (nombreSucursal, numPrestamo, importe)
Prestatario (nombreCliente, numPrestamo)
Cuenta   (nombreSucursal, numCuenta, saldo)
CtaCliente (nombreCliente, numCuenta)
```

Consulta de Datos

Para consultar una base de datos se usa la instrucción

`SELECT` $A_1, A_2, ...A_n$
`FROM` $R_1, R_2, ..., R_m$
`WHERE` condición;

- La cláusula `FROM` indica las relaciones que serán consultadas.
- La cláusula `WHERE` especifica la condición que deben satisfacer las tuplas para ser seleccionadas.
- La cláusula `SELECT` se utiliza para describir los atributos que se desea formen parte de la respuesta.

Esta consulta es equivalente a la expresión algebráica:

$$\pi_{A_1,A_2,...,A_n}(\sigma_{condicion}(R_1 \times R_2 \times ... \times R_m))$$

Selección

`WHERE` *expresión*.
Expresión:

- Operandos: Constantes, y atributos de las relaciones mencionadas en la cláusula `FROM`.
- Operadores: `=`, `<>`, `>`, `<`, `<=`, `>=`, `AND`, `OR`, `NOT`.

Ejemplo: Obtener todos los préstamos hechos en la sucursal San Ángel y cuyo importe sea mayor que $24,000.

```
SELECT *
FROM prestamo
WHERE nombreSucursal = 'San Angel' AND importe > 24000;
```

Es posible aplicar operadores aritméticos usuales a valores numéricos antes de compararlos. Por ejemplo: `(año - 1930) < 100`.

Rangos

Obtener la información de los préstamos cuyo importe está entre $6,000 y $30,000.

```
SELECT *
FROM   prestamo
WHERE  importe >= 6000 AND importe <= 30000;
```

esta consulta se puede escribir como sigue

```
SELECT *
FROM   prestamo
WHERE  importe BETWEEN 6000 AND 30000;
```

También se puede usar la comparación `NOT BETWEEN`

Comparación de cadenas

Es posible comparar cadenas aunque estas sean de diferente tipo (VARCHAR o CHAR).
La comparación se hace usando el orden lexicográfico.
Búsqueda de patrones, implica usar el operador `LIKE` una cadena y un patrón. `s LIKE p`.
La cadena con el uso opcional de los caracteres especiales `%`, `_`).

- `%` indica que `p` puede coincidir con cualquier subcadena en `s`.
- `_` coincide con cualquier caracter en s.

El valor de esta expresión es verdadero si y sólo si la cadena s, coincide con p.
`s NOT LIKE p` es verdadera si y sólo si, la cadena s no coincide con el patrón p.

Ejemplos

Si recordamos que el teléfono empieza con 5606

```
SELECT *
FROM   clientes
WHERE  telefono LIKE '5606____' ;
```

Todas las sucursales que empiecen con San

```
SELECT *
FROM   sucursal
WHERE  nombreSucursal LIKE 'San%' ;
```

Los patrones al igual que las cadenas deben ir entre apóstrofes.

Caracteres de escape

- Cualquier cadena que contenga un apostrófe
  `nombre LIKE '%''%'`
  '' implica búsqueda de un caracter no fin de cadena.
- SQL permite al usuario definir su propio caracter de escape:
  Cualquier cadena que comienza y termina con %
  `s LIKE 'x%%x%' ESCAPE 'x'`
- Operador de concatenación son dos líneas verticales. ||.

Duda ¿`SELECT * FROM R;` ?

Fechas y horas

- Para SQL una fecha constante se representa por la palabra `DATE` seguida de una fecha entre apóstrofes en formato *yyyy-mm-dd*.
  Ejemplo: `DATE '1810-09-15'`.
- Una hora constante es uan cadena entre apóstrofes, en formato *hh:mm:ss* precedida de la palabra `TIME`.
  Ejemplos: `TIME '18:15:00'` o bien `TIME '10:05:10.5'`
- Para combinar las fechas con las horas se utiliza la palabra `TIMESTAMP`.
  Ejemplo: `TIMESTAMP '1992-04-14 07:50:00'`

Se comparan estos tipos de datos con los operadores de relación utilizados con cadenas y números.

Proyección

Para eliminar atributos de las tuplas elegidas se puede proyectar la relación producida por una consulta SQL sobre algunos atributos.

```
SELECT nombreSucursal
FROM   prestamo;
```

Para asegurar que no haya duplicados se debe usar la palabra `DISTINCT`

```
SELECT DISTINCT nombreSucursal
FROM   prestamo;
```

Es posible cambiar de nombre a un atributo en la salida:

```
SELECT DISTINCT nombreSucursal AS nSucursal
FROM   prestamo;
```

Fórmula en lugar de un atributo:

```
SELECT nombreSucursal, numPrestamo, importe * 100
FROM   prestamo;
```

...Proyección

Constantes

```
SELECT nombreSucursal, numPrestamo, importe, 'dolares'
FROM   prestamo;
```

Operador de concatenación

```
SELECT 'Sr. '|| nombreCliente
FROM   cliente
```

Combinación de selección, proyección y búsqueda de cadenas

```
SELECT titulo
FROM   pelicula
WHERE  titulo LIKE '%''s%';
```

Ordenando la presentación del resultado

Resultado presentado en orden ascendente (`ASC`) se debe agregar a la instrucción `SELECT-FROM-WHERE` la clausula `ORDER BY <lista de atributos>`
Obtener una lista ordena de los clientes que viven en Cuautla.

```
SELECT DISTINCT nombreCliente
FROM   cliente
WHERE  ciudad = 'Cuautla'
ORDER BY nombreCliente;
```

Es posible ordenar de acuerdo a más de un atributo.

```
SELECT *
FROM   prestamo
ORDER BY importe DESC, numPrestamo ASC ;
```

Productos y Joins

Para especificar más de una relación en una consulta basta con listar cada relación en la clausula FROM y los atributos de las otras dos clausulas hacer referencia a las relaciones.
Nombre de todos los clientes que tienen un préstamo y el importe del mismo.

```
SELECT nombreCliente, importe
FROM   prestatario, prestamo
WHERE  prestatario.numPrestamo = prestamo.numPrestamo ;
```

Nombre de todos los clientes que tienen un préstamo en la sucursal San Angel y el número de préstamo.

```
SELECT nombreCliente, prestamo.numPrestamo
FROM   prestatario, prestamo
WHERE  prestatario.numPrestamo = prestamo.numPrestamo
       AND nombreSucursal = 'San Angel';
```

Variables de tupla

Para evitar ambigüedades se precede el nombre del atributo con el nombre de la relación.
Otra posibilidad es usar la palabra AS como sigue:

```
SELECT nombreCliente, T.numPrestamo
FROM   prestatario AS T , prestamo AS S
WHERE  T.numPrestamo = S.numPrestamo
       AND nombreSucursal = 'San Angel';
```

con lo cual se define un alias o **variable de tupla**.
Ejemplo: Obtener el nombre de todas las sucursales con un activo mayor que al menos una sucursal situada en Cuernavaca.

```
SELECT DISTINCT T.nombreSucursal
FROM   sucursal AS T, sucursal as S
WHERE  T.activo > S.activo AND S.ciudad = 'Cuernavaca' ;
```

Operaciones de conjuntos

SQL proporciona los operadores `UNION`, `INTERSECT` y `EXCEPT` para trabajar con relaciones compatibles, es decir que tengan el mismo conjunto de atributos.
Encontrar el nombre de los clientes que tienen un préstamo, una cuenta o ambas:

```
(SELECT nombreCliente  FROM   ctaCliente )
    UNION
(SELECT nombreCliente  FROM   prestatario );
```

Si en alguno de los `SELECT` se tuviera nombre de atributos diferente se deben renombrar con la palabra `AS`.
Clientes que tienen abierta una cuenta pero no tienen ningún crédito.

```
(SELECT DISTINCT nombreCliente  FROM   ctaCliente )
    EXCEPT
(SELECT DISTINCT nombreCliente  FROM   prestatario ) ;
```

... Operaciones de conjuntos

Todos los clientes que tienen un préstamo y una cuenta.

```
(SELECT DISTINCT nombreCliente  FROM   ctaCliente )
    INTERSECT
(SELECT DISTINCT nombreCliente  FROM   prestatario ) ;
```

A diferencia del `SELECT`, las operaciones para manejo de conjuntos eliminan duplicados automáticamente.
Para conservar los duplicados se debe utilizar `UNION ALL`, `EXCEPT ALL` o `INTERSECT ALL` según sea el caso.

Operadores de agregación

Toman una colección de valores y producen un único valor de salida.
Los operadores de agregación son:

1. `SUM`, suma los valores en la columna indicada.
2. `AVG`, promedia los valores en la columna indicada.
3. `MIN`, el menor de los valores en la columna indicada.
4. `MAX`, el mayor de los valores en la columna indicada.
5. `COUNT`, la cantidad de los valores en la columna indicada.

Los dos primeros operadores trabajan sobre números, los otros pueden operar con tipos no-numéricos.
Estos valores se aplican típicamente en la columna `SELECT`.

Ejemplos

Obtener el saldo promedio de todas las cuentas

```
SELECT AVG(sueldo)
FROM   cuenta;
```

Determinar la cantidad de tuplas de la relación cliente.

```
SELECT COUNT(*)
FROM   cliente;
```

Obtener el saldo promedio de las cuentas de la sucursal San Ángel.

```
SELECT AVG(saldo)
FROM   cuenta;
WHERE nombreSucursal = 'San Angel';
```

Agrupaciones

Con frecuencia se requiere agrupar las tuplas antes de aplicar un operador de agregación `GROUP BY` *atributos*.
Obtener el saldo promedio de las cuentas de cada sucursal.

```
SELECT nombreSucursal, AVG(saldo)
FROM   cuenta
GROUP BY nombreSucursal;
```

La claúsula `SELECT` tiene dos tipos de términos:

- Funciones de agregación.
- Atributos que aparecen en la clausula `GROUP BY`.

... Agrupaciones

Es posible usar `GROUP BY` en consultas que trabajan con más de una relación.
Obtener la cantidad de prestatarios de cada sucursal

```
SELECT nombreSucursal, COUNT (DISTINCT nombreCliente)
FROM   prestatario AS A , prestamo AS B
WHERE  A.numPrestamo = B.numPrestamo
GROUP BY nombreSucursal;
```

Para cada sucursal con saldo promedio de sus cuentas superior a $100,000 obtener su nombre y el saldo promedio.

```
SELECT nombreSucursal, AVG(saldo)
FROM   cuenta
GROUP BY nombreSucursal
HAVING AVG(saldo) > 100000;
```

`HAVING` se utiliza para restringir las tuplas agrupadas. Su sintaxis es la palabra `HAVING` seguida de una condición acerca del grupo.

... Agrupaciones

Obtener el saldo promedio de cada cliente que vive en Mazatlán y que tiene como mínimo tres cuentas.

```
SELECT cc.nombreCliente AVG (saldo)
FROM   ctaCliente as CC, cliente, cuenta
WHERE  cc.numCuenta = cuenta.numCuenta AND
       cc.nombreCliente = cliente.nombreCliente AND
       ciudad = 'Mazatlan'
GROUP BY cc.nombreCliente
HAVING COUNT(DISTINCT cc.numCuenta) >= 3;
```

Si hay `WHERE`, `HAVING` y `GROUP BY`:

- Se aplica predicado del `WHERE`.
- Las tuplas seleccionadas se agrupan por `GROUP BY`.
- Se aplica la clausula `HAVING` a cada grupo.
  Los grupos que no satisfagan esta clausula se eliminan.
- La clausula `SELECT` utiliza los grupos restantes para generar las tuplas

Subconsultas

Una **subconsulta** es una consulta que está incluida en otra.
Formas de uso para las subconsultas:

1. En consultas con operadores de conjuntos:
```
(SELECT DISTINCT nombreCliente FROM ctaCliente )
      INTERSECT
(SELECT DISTINCT nombreCliente FROM prestatario ) ;
```
2. En la clausula `WHERE`, si regresa un valor escalar, para comparar contra algún otro valor.
3. En la clausula `WHERE`, si regresa una relación, para comparar contra un conjunto de valores (relación).
4. En la clausula `FROM`, si devuelve una relación, como la relación sobre la que realizará la consulta.

Subconsultas que devuelven un valor escalar

Si se tienen las relaciones:

```
Pelicula(titulo, anio, duracion, nombreEstudio, idProductor)
Ejecutivo(nombre, direccion, idEjecutivo)
```

Se desea saber el nombre del productor de “Lo que el viento se llevo”.

```
SELECT nombre
FROM   ejecutivo, pelicula
WHERE  titulo = 'Lo que el viento se llevo' AND idProductor =
```

o bien

```
SELECT nombre
FROM   ejecutivo
WHERE  idEjecutivo =
            (SELECT idProductor
             FROM   pelicula
             WHERE  titulo = 'Lo que el viento se llevo'
            );
```

Operadores para producir un valor Booleano

Sea s un valor o una tupla, los operadores que pueden aplicarse al resultado de una subconsulta y producir un resultado Booleano son:

1. `EXISTS` R devuelve verdadero si y sólo si R no está vacía.
2. s `IN` R, devuelve verdadero si y sólo si s es igual a alguno de los valores de R.
3. $s >$ `ALL` R, devuelve verdadero si y sólo si s es mayor que todos los valores en la relación R. (El signo de mayor puede sustituirse por cualquier operador de comparación). R relación unaria.
4. $s >$ `ANY` R, devuelve verdadero si y sólo si s es mayor que al menos un valor en la relación R. (El signo de mayor puede sustituirse por cualquier operador de comparación). Se puede usar `SOME` como sinónimo.

Los operadores pueden ser negados precediéndolos de la palabra `NOT`.

```
NOT EXISTS R, NOT s > ALL R, NOT s > ANY R, s NOT IN R
```

Ejemplos

Encontrar todos los clientes que tienen un préstamo y una cuenta.

```
SELECT DISTINCT nombreCliente
FROM   prestatario
WHERE  nombreCliente IN (SELECT nombreCliente FROM CtaCliente)
```

Listar los clientes que tienen un préstamo en el banco y que no se "llaman" ni Gómez ni Santos.

```
SELECT DISTINCT nombreCliente
FROM   prestatario
WHERE  nombre-cliente NOT IN ('Santos', 'Gomez');
```

...Ejemplos

Listar los clientes que tienen tanto una cuenta como un préstamo en la sucursal San Ángel.

```
SELECT nombreCliente
FROM   prestatario, prestamo
WHERE  prestatario.numPrestamo = prestamo.num_prestamo AND
       nombreSucursal = 'San Angel' AND
       (nombreSucursal, nombreCliente) IN
           (SELECT nombreSucursal, nombreCliente
            FROM   ctaCliente, cuenta
            WHERE  ctaCliente.numCuenta = cuenta.numCuenta);
```

...Ejemplos

Listar los clientes que tienen un préstamo y no tienen una cuenta en el banco.

```
SELECT DISTINCT nombreCliente
FROM   prestatario
WHERE  nombreCliente NOT IN (SELECT nombreCliente
                             FROM   ctaCliente);
```

Obtener el nombre de las sucursales que poseen un activo mayor que al menos una sucursal situada en Ensenada.

```
SELECT nombreSucursal
FROM   sucursal
WHERE  activo > ANY (SELECT activo
                     FROM   sucursal
                     WHERE  ciudad = 'Ensenada');
```

...Ejemplos

Obtener los clientes que tienen tanto una cuenta como un préstamo.

```
SELECT nombreCliente
FROM   prestatario
WHERE  EXISTS (SELECT *
               FROM   ctaCliente
               WHERE  ctaCliente.nombreCliente =
                      prestatario.nombreCliente);
```

Encontrar la sucursal que tiene el mayor saldo promedio.

```
SELECT nombreSucursal
FROM   cuenta
GROUP BY nombreSucursal
HAVING AVG (saldo) >= ALL (SELECT AVG (saldo)
                           FROM cuenta
                           GROUP BY nombreSucursal);
```

Prueba de ausencia de duplicados

El operador `UNIQUE` devuelve `true` si la subconsulta no contiene tuplas duplicadas.
Encontrar los clientes que tienen sólo una cuenta en la sucursal San Ángel.

```
SELECT T.nombreCliente
FROM   ctaCliente AS T
WHERE  UNIQUE (SELECT R.nombreCliente
               FROM   cuenta, ctaCliente AS R
               WHERE  T.nombreCliente = R.nombreCliente AND
                      R.numCuenta = cuenta.numCuenta AND
                      cuenta.nombreSucursal = 'San Angel');
```

Prueba de presencia de duplicados

Encontrar los clientes que tienen al menos dos cuentas en la sucursal Tlalpan.

```
SELECT T.nombreCliente
FROM   ctaCliente AS T
WHERE  NOT UNIQUE (SELECT R.nombreCliente
                   FROM   cuenta, ctaCliente AS R
                   WHERE  T.nombreCliente = R.nombreCliente AN
                          R.numCuenta = cuenta.numCuenta AND
                          cuenta.nombreSucursal = 'Tlalpan');
```

Subconsultas como relaciones

Está permitido usar una subconsulta en la clausula `FROM`, en cuyo caso es necesario dar nombre a la relación resultante de la subconsulta y posiblemente renombrar los atributos.
Ejemplo: Para cada sucursal con saldo promedio superior a $100,000 obtener el nombre y saldo promedio.

```
SELECT nombreSucursal, saldoProm
FROM   (SELECT nombreSucursal, AVG(saldo)
        FROM   cuenta
        GROUP BY nombreSucursal)
       AS resultado (nombreSucursal, saldoProm)
WHERE  saldoProm > 10000;
```

... Subconsultas como relaciones

Dadas las relaciones:

```
Pelicula(titulo, anio, duracion, nombreEstudio, idProductor)
Actores(tituloPelicula, anioPelicula, nombreEstrella)
Ejecutivo(nombre,direccion,idEjecutivo)
```

encontrar los productores de las películas donde participa Harrison Ford.

```
SELECT nombre
FROM   ejecutivo, (SELECT idProductor
                   FROM   pelicula, actores
                   WHERE  titulo = tiluloPelicula AND
                          anio = anioPelicula AND
                          nombreEstrella = 'Harrison Ford'
                  ) AS Prod
WHERE idEjecutivo = Prod.idProductor ;
```

Consultas correlacionadas

En ocasiones es necesario que cada subconsulta se evalúe más de una vez, porque se requiere un valor de la consulta externa.
Sea `Película (título, año, duración, estudio, productor)` y se desea conocer el título de películas que se han usado más de una vez.

```
SELECT DISTINCT titulo
FROM   pelicula AS P
WHERE  anio < ANY (SELECT anio
                   FROM   pelicula
                   WHERE  titulo = P.titulo);
```

...Ejemplos

Obtener el nombre de los clientes que tienen una cuenta en cada una de las sucursales de Cuernavaca.

```
SELECT S.nombreCliente
FROM   ctaCliente AS S
WHERE  NOT EXISTS ((SELECT nombreSucursal
                    FROM sucursal
                    WHERE ciudad = 'Cuernavaca')
                           EXCEPT
                    (SELECT R.nombreSucursal
                    FROM ctaCliente AS T, cuenta AS R
                    WHERE T.numCuenta = R.numCuenta AND
                          S.nombreCliente = T.nombreCliente));
```

El alcance de las variables de tupla es dentro de la instrucción que se definió.

Expresiones para Joins en SQL

Estas expresiones pueden usarse como consultas como una alternativa a la instrucción `SELECT-FROM-WHERE`, o bien como subconsultas en clausulas `FROM`.

La forma más sencilla es el *Cross Join* o producto cartesiano (R $\times$ S): R `CROSS JOIN` S.

Sean R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10

```
SELECT R.*, S.*
FROM   R CROSS JOIN S;
```

R.A	R.B	R.C	S.B	S.C	S.D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	2	3	4

Theta Join

$(R \bowtie_C S)$: R `JOIN` S `ON` condición

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10

```
SELECT R.*, S.*
FROM   R JOIN S ON A < D;
```

A	R.B	R.C	S.B	S.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Join Natural

$(R \bowtie S)$: `R NATURAL JOIN S`

```
SELECT *
FROM R NATURAL JOIN S;
```

```
SELECT A, R.B AS B, S.C AS C, D
FROM   R JOIN S ON R.C = S.C AND R.B = S.B;
```

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10

$R \bowtie S =$

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

Join externo

Agrega al resultado cada tupla que no se une con ninguna y completa con nulos los otros atributos. Existen tres variedades:

1. Join externo completo. (R _ ⋈ _ S): R `NATURAL FULL OUTER JOIN` S

R

A	B	C
1	2	3
6	7	18
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10
2	5	15

R _ ⋈ _ S =

A	B	C	D
1	2	3	4
1	2	3	5
9	7	8	10
6	7	18	NULL
NULL	2	5	15

Join externo por la izquierda

$(R \_\bowtie S)$: R `NATURAL LEFT OUTER JOIN` S

R

A	B	C
1	2	3
6	7	18
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10
2	5	15

$R \_\bowtie S =$

A	B	C	D
1	2	3	4
1	2	3	5
9	7	8	10
NULL	2	5	15

Join externo por la derecha

Por la derecha. (R ⋈ _ S): R `NATURAL RIGHT OUTER JOIN` S

R

A	B	C
1	2	3
6	7	18
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10
2	5	15

R ⋈ _ S =

A	B	C	D
1	2	3	4
1	2	3	5
9	7	8	10
6	7	18	NULL

En lugar de usar el Join natural puede especificarse cualquier condición:

- R `FULL OUTER JOIN` S `ON` condición
- R `LEFT OUTER JOIN` S `ON` condición
- R `RIGHT OUTER JOIN` S `ON` condición