Lenguaje SQL: Parte I

Base de Datos

Mónica Caniupán
mcaniupan@ubiobio.cl

Universidad del Bío-Bío

2020

Lenguaje SQL

- SQL: Structured Query Language, originalmente diseñado por la IBM
- Standard ANSI/ISO SQL:1999
- SQL es el lenguaje más usado en los SGBD relacionales y tiene varias funcionalidades

Funcionalidades del Lenguaje SQL

- Lenguaje de manipulación de datos(LMD). Este subconjunto del SQL sirve para realizar consultas, insertar, eliminar y modificar tuplas
- Lenguaje de definición de datos (LDD). Este subconjunto del SQL sirve para crear, eliminar y modificar las definiciones de tablas y *vistas*
- Triggers y restricciones de integridad. Permite que el SGBD ejecute acciones cuando cambios cumplen ciertas condiciones
- SQL embebido y SQL dinámico.
  - El SQL embebido permite insertar código SQL en un programa escrito en otro lenguaje tal como: C, JAVA, etc.
  - El SQL dinámico permite construir y ejecutar una consulta SQL en tiempo de ejecución

Funcionalidades del Lenguaje SQL

- Ejecución cliente-servidor y acceso remoto a BDs. SQL provee comandos para que las aplicaciones clientes se conecten a un servidor SQL remoto, o accedan datos a través de una red
- Manejo de transacciones. Comandos de SQL permiten al usuario controlar la ejecución de una transacción
- Seguridad. SQL provee mecanismos para controlar el acceso de usuarios a las tablas y vistas
- Características avanzadas. El SQL:1999 incluye aspectos de OO (orientación a objetos), consultas recursivas (con restricciones), apoyo a las consultas de toma de decisiones, data mining, datos espaciales, XML, etc.

Sintaxis de Consultas en SQL

- La forma básica de una consulta SQL es la siguiente:

```
SELECT [DISTINCT] lista-selección
FROM lista de tablas (vistas)
WHERE condición
```

- Las cláusulas `SELECT` y `FROM` son obligatorias
- La cláusula `WHERE` es opcional

Ejemplo: Consulta SQL

- Dada la relación ALUMNOS:

ALUMNOS		
ID	NOMBRE	EDAD
11	*pedro*	21
12	*luis*	22
13	*juan*	20

- La consulta "Encontrar los alumnos con edad mayor a 20 años" se expresa por:

```
SELECT ID, NOMBRE, EDAD
FROM ALUMNOS
WHERE EDAD > 20
```

```
SELECT *
FROM ALUMNOS
WHERE EDAD > 20
```

- La respuesta a la consulta es:

ID	NOMBRE	EDAD
11	*pedro*	21
12	*luis*	22

Ejemplo: Consulta SQL

- Consideremos la siguiente instancia de la relación INS:

INS		
ID	IDC	NOTA
10	1	7
10	2	6.5
11	2	5
11	3	7

- Y la consulta:

```
SELECT ID
FROM INS
```

- La respuesta a la consulta es:

ID
10
10
11
11

- Para evitar las tuplas repetidas usamos `DISTINCT` en la consulta

Ejemplo: Consulta SQL

- Relación INS:

INS		
ID	IDC	NOTA
10	1	7
10	2	6.5
11	2	5
11	3	7

- La nueva consulta es:

```
SELECT DISTINCT ID
FROM INS
```

- La respuesta a la consulta es:

ID
10
11

Sintaxis de Consultas en SQL

- La *lista de tablas* en `FROM` es una lista de nombres de tablas (vistas)
  - El nombre de cada tabla puede ir seguido de una variable de rango (alias)
  - Los alias son útiles cuando la tabla aparece más de una vez en la consulta o cuando varias tablas comparten el nombre de los atributos
- La *lista de selección* en `SELECT` es una lista de expresiones que implican a nombres de columnas de tablas de la lista `FROM`
  - Estas columnas deben aparecer en las tablas/vistas de la lista `FROM`
- La *condición* en `WHERE` es una combinación booleana (expresión que emplea conectivos lógicos `AND, OR, NOT`) de condiciones de la forma *expresion op expresion* donde
  - Expresión es un nombre de columna, una constante o una expresión aritmética o cadena de caracteres
  - *op* es uno de $\{<, \leq, >, \geq, =, \neq\}$

Respuestas a Consultas SQL

- La respuesta a una consulta SQL es una relación cuyos atributos corresponden a los atributos en la cláusula `SELECT`
- La estrategia de evaluación de consultas (no óptima) es:
  1. Calcular el producto cartesiano de las tablas en la lista `FROM`
  2. Eliminar las tuplas del producto cartesiano que no cumplen las condiciones especificadas en la cláusula `WHERE`
  3. Eliminar las columnas que no aparecen en la lista de selección de `SELECT`
  4. Si se especifica `DISTINCT` entonces eliminar las tuplas repetidas

Ejemplo: Consultas SQL

- Consideremos el siguiente esquema e instancia de BD:

CURSO	
IDC	NOMBREC
1	*BD1*
2	*BD2*

INS		
ID	IDC	NOTA
10	1	7
10	2	5
11	2	7

ALUMNOS			
ID	NOMBRE	EDAD	CIUDAD
10	*luis*	20	*concepcion*
11	*pedro*	21	*chillan*
12	*antonio*	23	*concepcion*

Ejemplo: Consultas SQL

- Esquema:
  - *ALUMNOS(ID, NOMBRE,EDAD,CIUDAD)*
  - *CUR(IDC, NOMBREC)*
  - *INS(ID,IDC, NOTA)*
- Q_1: Encontrar el nombre y la edad de los alumnos que viven en concepcion

```
SELECT NOMBRE, EDAD
FROM ALUMNOS
WHERE CIUDAD = 'concepcion'
```

- La respuesta a Q_1 es:

NOMBRE	EDAD
luis	20
antonio	23

Ejemplo: Consultas SQL

- Esquema:
  - *ALUMNOS(ID, NOMBRE,EDAD,CIUDAD)*
  - *CUR(IDC, NOMBREC)*
  - *INS(ID,IDC, NOTA)*
- Q_2: Encontrar el nombre de los alumnos, id del curso y nota obtenida por los alumnos (uso de alias):

```
SELECT A.NOMBRE, I.IDC, I.NOTA
FROM ALUMNOS A, INS I
WHERE A.ID = I.ID
```

- La respuesta a Q_2 es:

NOMBRE	IDC	NOTA
luis	1	7
luis	2	5
pedro	2	7

Ejemplo: Consultas SQL

- Esquema:
  - *ALUMNOS(ID, NOMBRE,EDAD,CIUDAD)*
  - *CUR(IDC, NOMBREC)*
  - *INS(ID,IDC, NOTA)*
- Q_3: Encontrar el nombre de los alumnos que inscribieron el curso con $IDC = 1$

```
SELECT A.NOMBRE
FROM ALUMNOS A, INS I
WHERE A.ID = I.ID AND I.IDC= 1
```

- La respuesta a Q_3 es:

NOMBRE
luis

Ejemplo: Consultas SQL

- Esquema:
  - *ALUMNOS(ID, NOMBRE,EDAD,CIUDAD)*
  - *CUR(IDC, NOMBREC)*
  - *INS(ID,IDC, NOTA)*
- Q_4: Encontrar el nombre de los alumnos que inscribieron al menos dos cursos

```
SELECT A.NOMBRE
FROM ALUMNOS A, INS I1, INS I2
WHERE A.ID = I1.ID AND A.ID = I2.ID AND I1.IDC ≠ I2.IDC
```

- La respuesta a Q_4 es:

NOMBRE
luis

Ejemplo: Consultas SQL

- Esquema:
  - *ALUMNOS(ID, NOMBRE,EDAD,CIUDAD)*
  - *CUR(IDC, NOMBREC)*
  - *INS(ID,IDC, NOTA)*
- Q_5: Encontrar el nombre de los alumnos que inscribieron todos los cursos

```
SELECT A.NOMBRE
FROM ALUMNOS A
WHERE NOT EXISTS (
                  SELECT C.IDC
                  FROM  CURSOS C
                  WHERE NOT EXISTS (
                        SELECT I.IDC
                        FROM  INS I
                        WHERE I.IDC=C.IDC AND I.ID=A.ID))
```

- La respuesta a Q_5 es:

NOMBRE
luis

Vistas

- Una vista es una tabla cuyas filas (tuplas) no están almacenadas explícitamente en la BD
- Ejemplo: Consideremos las siguientes relaciones:

ALUMNOS			
ID	NOMBRE	EDAD	CIUDAD
10	*luis*	20	*concepcion*
11	*pedro*	21	*chillan*

INS		
ID	IDC	NOTA
10	1	3
11	2	5

- La siguiente vista obtiene el nombre y la nota de los alumnos que reprobaron el curso con $IDC = 1$

```
CREATE VIEW REP(NOMBRE,CALIFICACION)
AS   (SELECT A.NOMBRE, I.NOTA
        FROM ALUMNOS A, INS I
        WHERE A.ID=I.ID AND I.IDC=1 AND I.NOTA< 4)
```

Vistas

```
CREATE VIEW REP(NOMBRE,CALIFICACION)
AS   (SELECT A.NOMBRE, I.NOTA
        FROM ALUMNOS A, INS I
        WHERE A.ID=I.ID AND I.IDC=1 AND I.NOTA< 4)
```

- La vista *REP* tiene dos atributos *NOMBRE* y *CALIFICACION*
- Estos atributos tienen el mismo dominio que los atributos *NOMBRE* de la tabla *ALUMNOS* y *NOTA* de *INS*
- Si estos nombres se omiten en la cláusula `CREATE VIEW`, se heredan los atributos de la cláusula `SELECT`
- Las vistas se evalúan sobre una instancia de base de datos
- En nuestro ejemplo, al evaluar la vista sobre la instancia de la BD obtenemos las siguientes tuplas para *REP*:

NOMBRE	CALIFICACION
luis	3

Vistas

- Una vista puede ser usada como tabla base, i.e., puede aparecer en consultas SQL, o en la definición de otras vistas
  - Cada vez que *REP* aparece en una consulta, se evalúa la definición de la vista en la instancia de BD
  - Luego se evalúa el resto de la consulta tratando a *REP* como cualquier otra relación
- Ejemplo: La siguiente consulta hace uso de la vista *REP*:

```
SELECT R.NOMBRE, R.CALIFICACION, C.NOMBREC
FROM REP R, CURSOS C
WHERE C.IDC =1
```

- La respuesta a la consulta es:

NOMBRE	CALIFICACION	NOMBREC
luis	3	*BD1*

Expresiones y cadenas de caracteres en `SELECT`

- Cada elemento en la cláusula `SELECT` puede ser de la forma:

  expresion AS nombre_columna

  donde:

  - *expresion* es cualquier expresión aritmética o cadena de caracteres para los nombres de columnas y constantes
  - *nombre_columna* es un nombre nuevo para esa columna en el resultado de la consulta

Ejemplo: Consulta SQL

- Dada la siguiente instancia de BD:

ALUMNOS		
ID	NOMBRE	CATEGORIA
10	*luis*	3
11	*pedro*	7
12	*juan*	4

- La consulta "Calcular el incremento de la categoría de los alumnos que inscribieron al menos dos cursos" se expresa por:

  SELECT A.NOMBRE, A.CATEGORIA+ 1 as ICATEGORIA
  FROM ALUMNOS A, INS I_1, INS I_2
  WHERE A.ID = I_1.ID AND A.ID = I_2.ID AND I_1.IDC $\neq$ I_2.IDC

- La respuesta a la consulta es:

NOMBRE	ICATEGORIA
luis	4

Ejemplo: Consulta SQL

- Cada elemento de la condición en la cláusula `WHERE` puede ser tan general como por ejemplo:

  `SELECT` A1.NOMBRE AS NOMBRE_1, A2.NOMBRE AS NOMBRE_2
  `FROM` ALUMNOS A_1, ALUMNOS A_2
  `WHERE` $2 * A_1$.CATEGORIA=A_2.CATEGORIA-1

- La respuesta a la consulta es:

ALUMNOS		
ID	NOMBRE	CATEGORIA
10	*luis*	3
11	*pedro*	7
12	*juan*	4

NOMBRE_1	NOMBRE_2
luis	pedro

Unión, Intersección y Diferencia en SQL

- SQL permite el empleo de operaciones como la unión, intersección y diferencia, tal como lo hace el algebra relacional
  - UNION
  - INTERSECT
  - EXCEPT
- Sin embargo, muchos SGBD sólo soportan UNION
- Otros aceptan MINUS para referirse a EXCEPT

Ejemplo: Operador UNION

- Consideremos la siguiente instancia de ALUMNOS:

ALUMNOS		
ID	NOMBRE	CIUDAD
11	*pedro*	*concepcion*
12	*luis*	*chillan*
13	*juan*	*concepcion*
14	*domingo*	*concepcion*
15	*maria*	*santiago*
16	*anita*	*santiago*

- Q_1: Encontrar el nombre de los alumnos que viven en concepcion o chillan

```
SELECT A.NOMBRE
FROM ALUMNOS A
WHERE (A.CIUDAD = 'concepcion' OR
A.CIUDAD= 'chillan')
```

NOMBRE
pedro
luis
juan
domingo

Ejemplo: Operador UNION

- La siguiente consulta obtiene el mismo conjunto respuesta:

```
SELECT NOMBRE
FROM ALUMNOS
WHERE CIUDAD= 'concepcion'
UNION
SELECT NOMBRE
FROM ALUMNOS
WHERE CIUDAD = 'chillan'
```

- La respuesta a la consulta es:

NOMBRE
pedro
luis
juan
domingo

Ejemplo: Consulta SQL

- Consideremos el siguiente esquema e instancia de BD:

NAVEGANTES			
IDN	NOMBRE	CATEGORIA	EDAD
22	*pedro*	7	45
23	*andres*	1	35
24	*juan*	10	30
33	*loreto*	8	31
29	*natalia*	7	40
30	*esteban*	9	50

RESERVAS		
IDN	IDB	FECHA
22	101	10.10.98
23	102	10.11.00
29	103	09.12.00
30	104	05.11.99
22	102	03.11.99
22	103	05.12.00
22	104	05.01.01
33	101	05.01.02

BOTES		
IDB	NOMBREB	COLOR
101	*marino*	azul
102	*inter-lagos*	rojo
103	*clipper*	verde
104	*inter-lagos*	rojo

Ejemplo: Operador INTERSECT

- Esquema: *NAVEGANTES*(*IDN*, *NOMBRE*, *CATEGORIA*, *EDAD*), *RESERVAS*(*IDN*, *IDB*, *FECHA*), *BOTES*(*IDB*, *NOMBREB*, *COLOR*)
- Q_2: Encontrar el IDN de los navegantes que han reservado botes rojos y botes verdes

```
SELECT N.IDN
FROM NAVEGANTES N, RESERVAS R, BOTES B
WHERE N.IDN = R.IDN AND R.IDB = B.IDB AND B.COLOR = 'rojo'
INTERSECT
SELECT N2.IDN
FROM NAVEGANTES N2, RESERVAS R2, BOTES B2
WHERE N2.IDN = R2.IDN AND R2.IDB = B2.IDB AND B2.COLOR= 'verde'
```

Ejemplo: Operador INTERSECT

- Sin INTERSECT:

```
SELECT N.IDN
FROM NAVEGANTES N, RESERVAS R1, BOTES B1,
        RESERVAS R2, BOTES B2
WHERE N.IDN = R1.IDN AND R1.IDB = B1.IDB AND
        N.IDN = R2.IDN AND R2.IDB = B2.IDB AND
        B1.COLOR= 'rojo' AND B2.COLOR= 'verde'
```

Ejemplo: Operador EXCEPT

- Esquema: *NAVEGANTES*(*IDN*, *NOMBRE*, *CATEGORIA*, *EDAD*), *RESERVAS*(*IDN*, *IDB*, *FECHA*), *BOTES*(*IDB*, *NOMBREB*, *COLOR*)
- Q_2: Encontrar el identificador de los navegantes que han reservado botes rojos pero no botes verdes

```
SELECT N.IDN
FROM NAVEGANTES N, RESERVAS R, BOTES B
WHERE N.IDN = R.IDN AND R.IDB = B.IDB AND B.COLOR = 'rojo'
EXCEPT
SELECT N2.IDN
FROM NAVEGANTES N2, RESERVAS R2, BOTES B2
WHERE N2.IDN = R2.IDN AND R2.IDB = B2.IDB AND B2.COLOR = 'verde'
```

Algunas Consideraciones

- `UNION, INTERSECT, EXCEPT` se pueden emplear sobre tablas que sean unión compatibles
- La siguiente consulta es válida:

```
SELECT IDN
FROM NAVEGANTES
WHERE CATEGORIA =10
UNION
SELECT IDN
FROM RESERVAS
WHERE IDB =104
```

- Con `UNION` (`INTERSECT` y `EXCEPT`) se eliminan las tuplas duplicadas, para retenerlas se debe usar `UNION ALL`

Consultas Anidadas

- Una consulta *anidada* es una consulta que tiene otra consulta en su interior la cual se denomina sub-consulta
- La sub-consulta puede, a su vez, contener otra sub-consulta
- Las sub-consultas suelen aparecer:
  - en la cláusula `WHERE`,
  - en la cláusula `FROM` o
  - en la cláusula `HAVING` (que veremos más adelante)

Ejemplo: Consultas Anidadas

- Consideremos el esquema:

  NAVEGANTES (**IDN**, *NOMBRE*, *CATEGORIA*, *EDAD*),
  RESERVAS (**IDN,IDB,FECHA**)
  BOTES(**IDB**, *NOMBREB*, *COLOR*)

- La siguiente consulta obtiene el nombre de los navegantes que han reservado el bote con $IDB = 103$

```
SELECT NOMBRE
FROM NAVEGANTES
WHERE IDN IN (SELECT IDN
              FROM RESERVAS          ⟵ sub-consulta
              WHERE IDB =103)
```

Ejemplo: Consultas Anidadas

- La siguiente consulta obtiene el nombre de los navegantes que NO han reservado el bote con $IDB = 103$

```
SELECT NOMBRE
FROM NAVEGANTES
WHERE IDN NOT IN (SELECT IDN
                  FROM RESERVAS          ⟵ sub-consulta
                  WHERE IDB =103)
```

Ejemplo: Consultas Anidadas

- La siguiente consulta obtiene el nombre de los navegantes que han reservado botes rojos

```
SELECT NOMBRE
FROM NAVEGANTES
WHERE IDN IN (SELECT IDN
              FROM RESERVAS
              WHERE IDB IN (SELECT IDB
                            FROM BOTES
                            WHERE COLOR = 'rojo'))
```

Ejemplo: Consultas Anidadas

- La siguiente consulta obtiene el nombre de los navegantes que no han reservado botes rojos

```
SELECT N.NOMBRE
FROM NAVEGANTES N
WHERE N.IDN NOT IN (SELECT R.IDN
                    FROM RESERVAS R
                    WHERE R.IDB IN (SELECT B.IDB
                                    FROM BOTES B
                                    WHERE B.COLOR = 'rojo'))
```

Ejemplo: Consultas Anidadas

- ¿Que computan las siguientes consultas?

```
SELECT N.NOMBRE
FROM NAVEGANTES N
WHERE N.IDN IN (SELECT R.IDN
                FROM RESERVAS R
                WHERE R.IDB NOT IN (SELECT B.IDB
                                    FROM BOTES B
                                    WHERE B.COLOR = 'rojo'))

SELECT N.NOMBRE
FROM NAVEGANTES N
WHERE N.IDN NOT IN (SELECT R.IDN
                FROM RESERVAS R
                WHERE R.IDB NOT IN (SELECT B.IDB
                                    FROM BOTES B
                                    WHERE B.COLOR = 'rojo'))
```

Consultas Anidadas Correlacionadas

- En las consultas anidadas vistas hasta el momento la sub-consulta interior ha sido completamente independiente de la consulta exterior

```
SELECT NOMBRE
FROM NAVEGANTES
WHERE IDN IN (SELECT IDN
              FROM RESERVAS
              WHERE IDB IN (SELECT IDB
                            FROM BOTES
                            WHERE COLOR = 'rojo'))
```

- La sub-consulta interior puede depender de la fila (tupla) que se está examinando en cada momento en la consulta exterior

Consultas Anidadas Correlacionadas

- Ejemplo: Encontrar el nombre de los navegantes que han reservado el bote con $IDB = 103$

```
SELECT N.NOMBRE
FROM NAVEGANTES N
WHERE EXISTS (SELECT * FROM RESERVAS R
              WHERE R.IDN =N.IDN AND R.IDB=103)
```

  - `EXITS` permite comprobar si un conjunto es vacío o no
  - Para cada fila N de *NAVEGANTES* se comprueba si el conjunto de filas de *RESERVAS* R tal que $R.IDN = N.IDN$ `AND` $R.IDB = 103$ no está vacío (que exista)
  - Si no lo está (existe), el navegante *N* ha reservado el bote con $IDB = 103$ y se despliega su nombre
- La aparición de N de *NAVEGANTES* en la sub-consulta ($N.IDN$) se denomina *correlación* y estas consultas se denominan *consultas correlacionadas*

Ejemplo: Consultas Anidadas Correlacionadas

- La siguiente consulta correlacionada obtiene el nombre de los navegantes que no reservaron botes con $IDB = 103$

```
SELECT N.NOMBRE
FROM NAVEGANTES N
WHERE NOT EXISTS (SELECT *
                  FROM RESERVAS R
                  WHERE R.IDN =N.IDN AND R.IDB=103)
```

Ejemplo: Consultas Anidadas Correlacionadas

- La siguiente consulta obtiene el nombre de los navegantes que han reservaron todos los botes

```
SELECT N.NOMBRE
FROM NAVEGANTES N
WHERE NOT EXISTS (
                SELECT B.IDB
                FROM  BOTES B
                WHERE NOT EXISTS (
                      SELECT R.IDB
                      FROM  RESERVAS R
                      WHERE R.IDB=B.IDB AND R.IDN=N.IDN))
```