Lenguaje SQL

Base de Datos

Mónica Caniupán
mcaniupan@ubiobio.cl

Universidad del Bío-Bío

2024

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. Permite insertar/generar código SQL en un programa escrito en un lenguaje tal como: C, JAVA, etc.
- Seguridad. SQL provee mecanismos para controlar el acceso de usuarios a las tablas y vistas.

Contenidos

- Consultas Básicas y Vistas
- Consultas Anidadas
- Operadores de Agregación
- Valores Nulos
- Reuniones Externas

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:

CURSOS	
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)*
  - *CURSOS(IDC, NOMBREC)*
  - *INS(ID,IDC, NOTA)*
- Q_1: Encontrar el nombre y la edad de los alumnos que viven en Concepción

```
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)*
  - *CURSOS(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)*
  - *CURSOS(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)*
  - *CURSOS(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)*
  - *CURSOS(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
10	*luis*	20
11	*pedro*	21

INS		
ID	**IDC**	**NOTA**
10	1	3
11	2	5

CURSOS	
IDC	NOMBREC
1	*BD1*
2	*BD2*

- La siguiente vista obtiene el código de curso, nombre de alumno y la nota de los alumnos que reprobaron cursos

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

Vistas

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

- La vista *REP* tiene tres atributos *CODIGO*, *NOMBRE* y *CALIFICACION*
- Estos atributos tienen el mismo dominio que los atributos de donde provienen
- 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*:

CODIGO	NOMBRE	CALIFICACION
1	*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 R.CODIGO=C.IDC
```

- La respuesta a la consulta es:

NOMBRE	CALIFICACION	NOMBREC
luis	3	*BD1*

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: 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 UNION

- Esquema: *NAVEGANTES*(*IDN*, *NOMBRE*, *CATEGORIA*, *EDAD*), *RESERVAS*(*IDN*, *IDB*, *FECHA*), *BOTES*(*IDB*, *NOMBREB*, *COLOR*)
- Q_1: Encontrar el nombre de los navegantes con categoria 7 o categoría 8

```
SELECT NOMBRE
FROM NAVEGANTES
WHERE CATEGORIA= 7
UNION
SELECT NOMBRE
FROM NAVEGANTES
WHERE CATEGORIA= 8
```

- La respuesta a la consulta es:

NOMBRE
pedro
loreto
natalia

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 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`

Contenidos

- √ Consultas Básicas y Vistas
- Consultas Anidadas
- Operadores de Agregación
- Valores Nulos
- Reuniones Externas

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 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'))
```

Consultas Anidadas

- 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 el bote 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 reservado botes rojos y botes verdes

```
SELECT N.NOMBRE
FROM NAVEGANTES N, RESERVAS R, BOTES B
WHERE N.IDN=R.IDN AND R.IDB=B.IDB AND B.COLOR =’rojo’
AND EXISTS (SELECT *
            FROM BOTES B2, RESERVAS R2
            WHERE N.IDN=R2.IDN AND R2.IDB =B2.IDB AND
            B2.COLOR=’verde’)
```

Contenidos

- √ Consultas Básicas y Vistas
- √ Consultas Anidadas
- Operadores de Agregación
- Valores Nulos
- Reuniones Externas

Operadores de Agregación

- SQL soporta cinco operaciones de agregación:
  - `COUNT ([DISTINCT] A)`: computa el número de valores (únicos) de la columna A
  - `SUM ([DISTINCT] A)`: computa la suma de todos los valores (únicos) de la columna A
  - `AVG ([DISTINCT] A)`: computa el promedio de todos los valores (únicos) de la columna A
  - `MAX (A)`: computa el valor máximo de la columna A
  - `MIN (A)`: computa el valor mínimo de la columna A

Ejemplo: Consultas de Agregación

- Encontrar el promedio de EDAD de todos los navegantes

```
SELECT AVG (EDAD)
FROM NAVEGANTES
```

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

- La respuesta es: $\langle 38,5 \rangle$

Ejemplo: Consultas de Agregación

- Encontrar el promedio de EDAD de los navegantes con categoría=7

```
SELECT AVG (EDAD)
FROM NAVEGANTES
WHERE CATEGORIA = 7
```

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

- La respuesta es: $\langle 42,5 \rangle$

Ejemplo: Consultas de Agregación

- Contar el número total de botes

```
SELECT COUNT(*)
FROM BOTES
```

- Contar el número de nombres (distintos) de navegantes

```
SELECT COUNT(DISTINCT NOMBRE)
FROM NAVEGANTES
```

Ejemplo: Consultas de Agregación

- Encontrar el nombre y edad del navegante con mayor edad

```
SELECT N.NOMBRE, N.EDAD
FROM NAVEGANTES N
WHERE N.EDAD = (SELECT MAX (N2.EDAD)
                FROM NAVEGANTES N2)
```

- La respuesta es: $\langle esteban, 50 \rangle$

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

- La siguiente no es una consulta legal en SQL:

```
SELECT NOMBRE, MAX(EDAD)
FROM NAVEGANTES
```

GROUP BY y HAVING

- Hasta ahora se ha aplicado las operaciones de agregación a todas las tuplas de la relación
- Sin embargo, a menudo se desea aplicar operaciones de agregación a cada uno de los **grupos** de filas de una relación
- Ejemplo: Consideremos la consulta: “Encontrar la edad del navegante más joven de cada categoría”
- Una forma de resolverlo sería:

```
SELECT MIN(EDAD)
FROM NAVEGANTES
WHERE CATEGORIA =i
```

  con $i = 1, 2, \ldots$
- Necesitariamos tantas consultas como categorías existen

GROUP BY y HAVING

- Necesitamos otra forma de hacerlo

```
SELECT CATEGORIA,
MIN(EDAD) AS MIN_EDAD
FROM NAVEGANTES
GROUP BY CATEGORIA
```

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

- La respuesta es:

CATEGORIA	MIN_EDAD
7	40
1	30
8	31

Consultas Generales

- Una consulta general en SQL tiene la siguiente forma:

```
SELECT [DISTINCT] lista-selección
FROM lista(tablas,vistas)
WHERE condición
GROUP BY lista-para-formar-grupos
HAVING condición-sobre-grupos
```

- La *lista-selección* en la cláusula `SELECT` consiste de:
  1. Una lista de nombres de atributos
  2. Una lista de términos de la forma **OPAGR***(nombre-columna)* **AS** *nuevo-nombre*
- Todos los atributos que aparecen en (1) deben aparecer en *lista-para-formar-grupos*

Consultas Generales

```
SELECT [DISTINCT] lista-selección
FROM lista(tablas,vistas)
WHERE condición
GROUP BY lista-para-formar-grupos
HAVING condición-sobre-grupos
```

- Cada fila del resultado de la consulta se corresponde con un grupo, que es un conjunto de filas que concuerdan con los valores para las columnas de *lista-para-formar-grupos*
- Las expresiones en *condición-sobre-grupos* de la cláusula `HAVING` deben tener un único valor por grupo
- Si se omite `GROUP BY`, se considera a toda la tabla como un solo grupo

Ejemplo: Consultas con GROUP BY y HAVING

- Mostrar el total de reservas realizadas de cada bote de color rojo, considerar solo aquellos botes que fueron reservados más de 4 veces

```
SELECT B.IDB, B.COLOR, COUNT (*) AS NUMRESERVAS
FROM BOTES B, RESERVAS R
WHERE B.IDB = R.IDB AND B.COLOR ='rojo'
GROUP BY B.IDB, B.COLOR
HAVING COUNT(*) > 4
```

Contenidos

- √ Consultas Básicas y Vistas
- √ Consultas Anidadas
- √ Operadores de Agregación
- Valores Nulos
- Reuniones Externas

Valores Nulos

- En la práctica los valores de las columnas pueden ser desconocidos
- Ejemplo: Consideremos la siguiente relación:

PERSONAS			
ID	NOMBRE	EDAD	NOMBRECONYUGE
11	*Pedro*	21	*Maria*
12	*Luis*	22	*Sandra*
13	*Juan*	20	*Paola*

- Se desea insertar una tupla en la relación pero:
  1. No conocemos la edad de una persona. ¿Qué valor le asignamos al atributo EDAD?
  2. ¿Qué valor le asignamos al atributo NOMBRECONYUGE si la nueva persona es soltera?

Valores Nulos

- SQL ofrece un valor especial para las columnas denominado *NULL* (nulo) para emplearlo en estas situaciones
- El valor *NULL* significa *desconocido* o *no aplicable*
- La siguiente operación es válida:
  INSERT INTO PERSONAS VALUES(14,'Enrique',NULL,NULL)
- Sin embargo, los valores nulos en las bases de datos producen un impacto en la evaluación de consultas

Comparaciones que Emplean Valores Nulos

- Consideremos una comparación como $Edad = 20$ evaluada sobre la siguiente relación:

PERSONAS			
ID	NOMBRE	EDAD	NOMBRECONYUGE
11	*Pedro*	21	*Maria*
12	*Luis*	22	*Sandra*
13	*Juan*	20	*Paola*
14	*Enrique*	*NULL*	*NULL*

- ¿Es la condición $Edad = 20$ verdadera o falsa en la fila de *Enrique*?

Comparaciones que Emplean Valores Nulos

- El resultado debería ser *desconocido*
- De hecho éste es el caso para cualquier comparación con operadores $\{<, >, =, \neq\}$, que involucre valores nulos
- Más aún, si comparamos dos valores nulos con $\{<, >, =, \neq\}$ el resultado siempre es *desconocido*
- SQL ofrece el operador de comparación especial IS NULL para verificar si el valor de un atributo es *NULL*
- Por ejemplo, EDAD IS NULL evaluado en la fila de *Enrique* es *verdadero*
- También se puede usar IS NOT NULL, e.g. EDAD IS NOT NULL que evaluado en la fila de *Enrique* es *falso*

Operaciones Booleanas con Nulos

- Consideremos la tupla con nulos PERSONA(14,Enrique,NULL,NULL)
  - $EDAD > 20$ OR $NOMBRE =' Enrique'$

    es *verdadero*, porque $NOMBRE =' Enrique'$ es verdad
  - $EDAD > 20$ OR $NOMBRE =' Pedro'$

    es *desconocido* porque la primera comparación es *desconocido* y la segunda es *falsa*
- En la presencia de valores nulos, hay que definir los operadores lógicos AND,OR y NOT mediante una lógica de tres valores en la que las expresiones toman el valor de verdadero, falso o desconocido

Operaciones Booleanas con Nulos

- NOT desconocido es desconocido
- OR de dos argumentos es:
  - verdadero si uno de los argumentos es verdadero
  - desconocido si uno de los argumentos es falso y el otro es desconocido
  - falso si los dos argumentos son falsos
- AND de dos argumentos es:
  - falso si uno de los argumentos es falso
  - desconocido si uno de los argumentos es desconocido y el otro es verdadero o desconocido
  - verdadero si los dos argumentos son verdaderos

Consecuencias para las Estructuras de SQL

- Duplicidad de tuplas: Dos tuplas de una relación se consideran *iguales* si sus atributos tienen el mismo valor o ambas contienen nulos
  - Sin embargo, si comparamos dos valores nulos usando el símbolo de igualdad obtenemos *desconocido* (*NULL* = *NULL* es siempre *desconocido*)
- Los operadores aritméticos $\{+, -, *, /\}$ retornan *NULL* si uno de los argumentos es *NULL*
- Comportamientos inesperados de las operaciones de agregación:
  - COUNT(*) maneja el valor *NULL* como cualquier otro valor
  - Todas las demás operaciones de agregación COUNT, SUM, AVG, MIN MAX y las variaciones usando DISTINCT descartan los valores nulos
  - Como caso especial, si uno de estos operadores, que no sea COUNT, se aplica sólo a valores nulos, el resultado es *NULL*

Ejemplo: Operaciones con NULL

- Consideremos la siguiente relación:

PERSONAS			
ID	NOMBRE	EDAD	NOMBRECONYUGE
11	*Pedro*	21	*Maria*
12	*Luis*	22	*Sandra*
13	*Juan*	20	*Paola*
14	*NULL*	*NULL*	*NULL*

- La respuesta a:
  SELECT COUNT (*)
  FROM PERSONAS
  es 4
- La respuesta a:
  SELECT SUM(EDAD)
  FROM PERSONAS
  es 63

Evitando los Valores Nulos

- SQL nos permite prohibir que ciertos atributos tomen valores nulos
- Esta restricción se especifica en la definición de atributos:

```
CREATE TABLE .....
EDAD INTEGER NOT NULL
...
```

- Para cada atributo clave existe una restricción NOT NULL implícita

Contenidos

- √ Consultas Básicas y Vistas
- √ Consultas Anidadas
- √ Operadores de Agregación
- √ Valores Nulos
- Reuniones Externas

Variantes de Joins

- SQL soporta algunas variedades interesantes de la operación join que aprovechan los valores nulos, las que se denominan Outer Joins
- Considere la siguiente operación: $NAVEGANTES \bowtie_{idn=idn} RESERVAS$

NAVEGANTES			
IDN	NOMBRE	EDAD	CATEGORIA
22	*pedro*	45	4
23	*andres*	35	6
33	*loreto*	31	6
29	*natalia*	40	7
30	*esteban*	50	8

RESERVAS		
IDN	**IDB**	**FECHA**
23	102	10.11.00
22	102	10.11.00
33	101	05.01.02

- El resultado es:

IDN	NOMBRE	EDAD	CATEGORIA	IDN	IDB	FECHA
22	*pedro*	45	4	22	102	10.11.00
23	*andres*	35	6	23	102	10.11.00
33	*loreto*	31	6	33	101	05.01.02

Variantes de Joins

- Sin embargo, podría ser interesante mantener las tuplas de navegantes que no tienen reservas en el resultado. Para esto usamos el Outer Join
- Con Outer Join, las tuplas que no tienen reservas aparecen en el resultado del join y los atributos correspondientes a reservas toman valores nulos
- Existen tres variantes de Outer Join:
  1. Left outer join
  2. Right outer join
  3. Full outer join

LEFT OUTER JOIN

- Consideremos las siguientes relaciones:

NAVEGANTES			
IDN	NOMBRE	EDAD	CATEGORIA
22	*pedro*	45	4
23	*andres*	35	6
33	*loreto*	31	6
29	*natalia*	40	7
30	*esteban*	50	8

RESERVAS		
IDN	**IDB**	**FECHA**
23	102	10.11.00
22	102	10.11.00
33	101	05.01.02

- SELECT *
  FROM NAVEGANTES NATURAL LEFT OUTER JOIN RESERVAS

IDN	NOMBRE	EDAD	CATEGORIA	IDN	IDB	FECHA
22	*pedro*	45	4	22	102	10.11.00
23	*andres*	35	6	23	102	10.11.00
33	*loreto*	31	6	33	101	05.01.02
29	*natalia*	40	7	NULL	NULL	NULL
30	*esteban*	50	8	NULL	NULL	NULL

RIGHT OUTER JOIN

- Consideremos las siguientes relaciones:

RESERVAS		
IDN	**IDB**	**FECHA**
23	102	10.11.00
22	102	10.11.00
33	101	05.01.02

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

- SELECT *
  FROM RESERVAS NATURAL RIGHT OUTER JOIN BOTES

IDN	IDB	FECHA	IDB	NOMBREB	COLOR
23	102	10.11.00	102	*inter-lagos*	rojo
22	102	10.11.00	102	*inter-lagos*	rojo
33	101	05.01.02	101	*marino*	azul
NULL	NULL	NULL	103	*clipper*	verde
NULL	NULL	NULL	104	*inter-lagos*	rojo

FULL OUTER JOIN

- Consideremos las siguientes relaciones:

RESERVAS		
IDN	**IDB**	**FECHA**
23	102	10.11.00
22	102	10.11.00
33	101	05.01.02
33	106	06.01.02

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

- SELECT *
  FROM RESERVAS NATURAL FULL OUTER JOIN BOTES

IDN	IDB	FECHA	IDB	NOMBREB	COLOR
23	102	10.11.00	102	*inter-lagos*	rojo
22	102	10.11.00	102	*inter-lagos*	rojo
33	101	05.01.02	101	*marino*	azul
33	106	06.01.02	NULL	NULL	NULL
NULL	NULL	NULL	103	*clipper*	verde
NULL	NULL	NULL	104	*inter-lagos*	rojo

Contenidos

- √ Consultas Básicas y Vistas
- √ Consultas Anidadas
- √ Operadores de Agregación
- √ Valores Nulos
- √ Reuniones Externas