Fundamentos de Bases de Datos

Facultad de Ciencias UNAM

M.I. Gerardo Avilés Rosas
gar@ciencias.unam.mx

Resumen **El Lenguaje de Consulta Estructurado**

1. El lenguaje de Consulta Estructurado

SQL (*Structured Query Language*; **Lenguaje de Consulta Estructurado**) es un lenguaje de consulta para bases de datos, siendo adoptado como estándar (**ANSI/ISO**) de la industria en **1986**. Se trata de un lenguaje para *definición, manipulación y control* de bases de **datos relacionales**. Es un lenguaje declarativo (sólo hay que indicar **qué** se quiere hacer). SQL es un lenguaje muy parecido al lenguaje natural; concretamente, se parece al inglés, y es muy expresivo. Los componentes del **SQL** son:

- ✓ **DDL** (*Data Definition Language*; **Lenguaje para Definición de Datos**)
- ✓ **DML** (*Data Manipulation Language*; **Lenguaje para Manipulación de Datos**)
- ✓ **DCL** (*Data Control Language*; **Lenguaje para Control de Datos**

2. Lenguaje para Manipulación de Datos

Consulta de datos

Para consultar una base de datos se usa la instrucción:

```
select A1,A2,...,An
from R1,R2,...,Rn
where <condición>;
```

- La cláusula **SELECT** se utiliza para describir los atributos que se desea formen parte de la respuesta.
- La cláusula **FROM** indica las relaciones (tablas) que serán consultadas.
- La cláusula **WHERE** especifica la condición que deben satisfacer las tuplas para ser seleccionadas.
- **<condición>**:
  - ✓ **Operandos**. Constantes y atributos de las relaciones mencionadas en la cláusula **FROM**.
  - ✓ **Operadores:** =, <>, >, <, <=, >=, AND, OR, NOT

Comparación de cadenas

Es posible comparar cadenas, aunque estas sean de diferente tipo (**VARCHAR** o **CHAR**). La comparación se hace usando el orden **lexicográfico**.

La 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**.
- _ (guion bajo) coincide con cualquier carácter 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**. Los patrones al igual que las cadenas deben ir entre **apóstrofes**.

Caracteres de escape

- Cualquier cadena que contenga un apóstrofe **nombre LIKE '%''%'**, **''** implica la búsqueda de un caracter no fin de cadena.
- SQL permite al usuario definir su propio caracter de escape, por ejemplo, cualquier cadena que comienza y termina con **%**:
- **s LIKE 'x%%x%' ESCAPE 'x'**
- El operador de concatenación es **+**.

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**.
- Una hora constante es una cadena entre apóstrofes, en formato **hh:mm:ss** precedida de la palabra **TIME**.
- Para combinar las fechas con las horas se utiliza la palabra **TIMESTAMP**.

Proyección

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

```
select a1,a2
from nombreTabla;
```

- Para asegurar que no haya duplicados se debe usar la palabra **DISTINCT**.

```
select distinct a1,a2
from nombreTabla;
```

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

```
select a1 as otroNombre
from nombreTabla;
```

- Formula en lugar de atributo:

```
select a1,a2,a3*valor
from nombreTabla;
```

- Constantes:

```
select a1,a2,'cadena constante'
from nombreTabla;
```

- Operador de concatenación:

```
select 'Cadena ' + atributo
from nombreTabla;
```

- Combinación de selección, proyección y búsqueda de cadenas:

```
select a1,a2,a3
from nombreTabla
where a1 like 'patrón';
```

Ordenando la presentación del resultado

Para presentar el resultado de forma ascendente (**ASC**) o descendente (**DESC**) se agrega a la instrucción **select-from-where** la cláusula **order by <lista de atributos>**:

```
select a1,a2,a3
from nombreTabla
where <condición>
order by <lista-de-atributos>;
```

Productos y Joins

Para especificar más de una relación en una consulta basta con listar cada relación en la cláusula **FROM** y los atributos de las otras dos cláusulas hacer referencia a las relaciones.

```
select a1,a2,a3
from tabla1,tabla2,...,tablaN
where <condición>;
```

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 a1,S.a2
from tabla1 as R, tabla2 as S
where R.a2 = S.a2 and <condición>;
```

con lo cual se define un **alias** o **variable de tupla**.

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.

```
(select atributo1 from tabla1)
UNION | INTERSECT | EXCEPT
(select atributo2 from tabla2);
```

Si en alguno de los **SELECT** se tuviera nombre de atributos diferentes se deben renombrar con la palabra **AS**.

A diferencia del **SELECT**, las operaciones para manejo de conjuntos eliminan duplicados automáticamente. En SQL Server solo es posible utilizar **UNION ALL** para conservar los duplicados.

Operadores de agregación

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

- **SUM**, suma los valores en la columna indicada
- **AVG**, promedia los valores en la columna indicada
- **MIN**, el menor de los valores en la columna indicada
- **MAX**, el mayor de los valores en la columna indicada
- **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**. Ninguno de ellos considera a los valores nulos (**NULL**).

Agrupaciones

Con frecuencia se requiere agrupar las tuplas antes de aplicar un operador de agregación, esto se hace a través de la instrucción: **GROUP BY atributos**.

```
select a1,a2,...,agregación1,agregación2,...
from tabla1,tabla2,...
where <condicion>
group by <lista-de-atributos>;
```

La cláusula **SELECT** tiene dos tipos de términos: **funciones de agregación** y atributos que aparecen en la cláusula **GROUP BY**. Es posible usar **GROUP BY** en consultas que trabajan con más de una relación.

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

```
select a1,a2,...,agregación1,agregación2,...
from tabla1,tabla2,...
where <condicion>
group by <lista-de-atributos>
having <condición>;
```

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

- ✓ Se aplica el predicado del **WHERE**
- ✓ Las *tuplas* seleccionadas se agrupan por **GROUP BY**
- ✓ Se aplica la cláusula **HAVING** a cada grupo
  Los grupos que no satisfagan esta cláusula, se eliminan
- ✓ La cláusula **SELECT** utiliza los grupos restantes para generar las tuplas resultado de la consulta.

Subconsultas

Una **subconsulta** es una consulta que está incluida en otra, las formas de uso para las subconsultas son:

1. En consultas con operadores de conjuntos:
2. En la cláusula **WHERE**, si regresa un valor escalar, para comparar contra algún otro valor.
3. En la cláusula **WHERE**, si regresa una relación, para comparar contra un conjunto de valores (relación).
4. En la cláusula **FROM**, si devuelve una relación, como la relación sobre la que realizará la consulta.

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 es una 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.**

Subconsultas como relaciones

Está permitido usar una **subconsulta** en la cláusula **FROM**, en cuyo caso es necesario dar nombre a la relación resultante de la **subconsulta** y posiblemente renombrar los atributos.

```
select a1,a2,a3,...
from (subconsulta) as resultado
where <condicion>;
```

3. Vistas en SQL

Una vista es una **relación virtual** creada a partir de relaciones base:

```
CREATE VIEW nombreVista AS consulta_SQL;
```

Las modificaciones en los datos se ven reflejadas en la vista y cualquier cambio en la vista se verá reflejado en la relación base. Con la vista, solo se muestra una porción de la base de datos que van a ver los usuarios (con lo único que tienen permitido trabajar). Una vez definida la vista puede emplearse como relación real y aplicar las mismas operaciones que se vieron con anterioridad.

Cambio de nombre a los atributos

Es necesario escribir los nuevos nombres separados por comas y todo entre paréntesis después del nombre de la vista. Solo se utiliza cuando es obligatorio un nuevo nombre: cuando la columna se deriva de una expresión aritmética o si hay ambigüedad en el nombre de una columna (tablas con el mismo nombre).

Modificaciones al contenido de las vistas

Una vista es actualizable si es posible traducir la modificación de la vista en una modificación equivalente sobre una tabla base.

Borrado de tuplas

De una vista se pueden suprimir tuplas, estos cambios se reflejan tanto en la vista como en las relaciones subyacentes.

Inserción de tuplas

Una vista NO es actualizable si en su creación aparece:

- **DISTINCT** en la lista de selección.
- Alguna columna calculada o agregada en la lista de selección.
- Referencia a más de una tabla, ya sea en el **FROM**, en una subconsulta.
- Una cláusula **GROUP BY** o **HAVING**.
- Referencia a alguna vista que no es actualizable.

Actualización y eliminación de tuplas

La actualización de tuplas es una vista actualizable implica actualizar todas las tuplas de la relación base. Por su parte, la eliminación de vista se realiza con independencia de si la vista es actualizable o no:

Importancia de las vistas

- Proporcionan independencia lógica

- Proporcionan relaciones personalizadas
- Facilitan consultas
- Razones de seguridad

Referencias

[1] Beaulieu, A. *Learning SQL*. O'Reilly Media, Second Edition, 2009

[2] Limeback, R. *Simply SQL*. SitePoint, First Edition, 2008

[3] Rockoff, L. *The Language of SQL*, Course Technology (Cengage Learning), 2011

[4] Ullman, J. D. and Widom, J. *A First Course in Database Systems*. Prentice Hall, 1997