Universidad Tecnológica De Panamá
Facultad de Ingeniería en Sistemas Computacionales
2do Semestre

Profesor: Ronald Ponce

Integrantes: Jean Meléndez 8-985-955

Kevin Valdés 8-1021-301

Daniel Gonzales 8-1022-1099

Martin Liao 1-757-1706

Ricardo Rose

Asignatura: Base de datos

Investigación No.1: El álgebra relacional y su relación con las bases de datos

Año Lectivo: 2024

1. las bases de datos relacionales, la Teoría de Conjuntos juega un papel crucial al proporcionar un marco matemático para las operaciones que pueden realizarse sobre los datos. Muchas de las operaciones fundamentales de la teoría de conjuntos tienen sus equivalentes directos en las bases de datos relacionales, en particular en el álgebra relacional, que es el fundamento teórico de las operaciones en SQL (Structured Query Language). A continuación, se describen las principales operaciones de la teoría de conjuntos y su correspondencia en las bases de datos relacionales:

1. Unión (Union)

- Teoría de Conjuntos: La unión de dos conjuntos AAA y BBB es el conjunto de elementos que pertenecen a AAA, BBB, o a ambos.
- Base de Datos Relacional: En SQL, la operación UNION se utiliza para combinar los resultados de dos o más consultas. Las tablas involucradas deben tener la misma estructura (número y tipo de columnas).

- Ejemplo:

```
SELECT nombre, edad FROM Clientes
UNION
SELECT nombre, edad FROM Empleados;
```

2. Intersección (Intersection)

- Teoría de Conjuntos: La intersección de dos conjuntos AAA y BBB es el conjunto de elementos que pertenecen a ambos conjuntos.
- Base de Datos Relacional: En SQL, la operación INTERSECT devuelve las filas que son comunes entre dos conjuntos de resultados.
- Ejemplo:

```
SELECT nombre FROM Clientes
INTERSECT
SELECT nombre FROM Empleados;
```

3. Diferencia (Difference)

- Teoría de Conjuntos: La diferencia de conjuntos A−BA - BA−B es el conjunto de elementos que están en AAA pero no en BBB.
- Base de Datos Relacional: En SQL, la operación EXCEPT o MINUS se utiliza para devolver las filas que están en el primer conjunto de resultados pero no en el segundo.
- Ejemplo:

```
SELECT nombre FROM Clientes
EXCEPT
SELECT nombre FROM Empleados;
```

4. Producto Cartesiano (Cartesian Product)

- Teoría de Conjuntos: El producto cartesiano de dos conjuntos AAA y BBB es el conjunto de todos los pares posibles (a,b)(a, b)(a,b), donde a∈Aa \in Aa∈A y b∈Bb \in Bb∈B.
- Base de Datos Relacional: En SQL, el CROSS JOIN genera el producto cartesiano entre dos tablas. Se combinan todas las filas de la primera tabla con todas las filas de la segunda tabla.
- Ejemplo:

```
SELECT * FROM Clientes
CROSS JOIN Empleados;
```

5. Proyección (Projection)

- Teoría de Conjuntos: La proyección es una operación que devuelve un subconjunto de los atributos de un conjunto.
- Base de Datos Relacional: En SQL, la proyección se implementa seleccionando columnas específicas mediante el comando SELECT. Esta operación reduce el número de columnas en el conjunto de resultados.
- Ejemplo:

```
SELECT nombre, edad FROM Clientes;
```

6. Selección (Selection)

- Teoría de Conjuntos: La selección devuelve los elementos de un conjunto que cumplen una condición específica.
- Base de Datos Relacional: En SQL, la selección se implementa utilizando la cláusula WHERE. Esta operación reduce el número de filas que satisfacen ciertas condiciones.
- Ejemplo:

```
SELECT * FROM Clientes WHERE edad > 30;
```

7. División (Division)

- Teoría de Conjuntos: La división es una operación que busca elementos de un conjunto que están relacionados con todos los elementos de otro conjunto.
- Base de Datos Relacional: Esta operación no tiene un operador específico en SQL, pero puede lograrse combinando otras operaciones como JOIN y NOT EXISTS o EXCEPT.
- Ejemplo: Supongamos que queremos encontrar a todos los clientes que han comprado todos los productos disponibles:

```
SELECT nombre FROM Clientes c
WHERE NOT EXISTS (
  SELECT * FROM Productos p
  WHERE NOT EXISTS (
    SELECT * FROM Compras co
    WHERE co.cliente_id = c.id AND co.producto_id = p.id
  )
);
```

3. ¿Qué Relación existe entre álgebra relacional y SQL como lenguaje de consulta a fin de entender las equivalencias entre conceptos matemáticos y su implementación en Bases de Datos?

El álgebra relacional y SQL están estrechamente relacionados, ya que el álgebra relacional sirve como la base teórica para el funcionamiento de SQL como lenguaje de consulta en bases de datos relacionales. El álgebra relacional proporciona un conjunto de operaciones formales que definen cómo se manipulan y consultan los datos en una base de datos relacional, mientras que SQL implementa esas operaciones de manera más accesible y práctica. Aquí están las equivalencias clave entre los conceptos matemáticos del álgebra relacional y su implementación en SQL:

1. Selección (σ)

- Álgebra Relacional: La operación de selección extrae tuplas que cumplen con una condición dada.
- SQL: Esta operación corresponde a la cláusula WHERE en SQL, que filtra filas según condiciones.
- Ejemplo:
  - Álgebra Relacional: σ_{edad > 30} (Personas)
  - SQL: SELECT * FROM Personas WHERE edad > 30;

2. Proyección (π)

- Álgebra Relacional: La proyección selecciona columnas específicas de una relación.
- SQL: Corresponde a la lista de columnas que se seleccionan después del SELECT.
- Ejemplo:
  - Álgebra Relacional: π_{nombre, edad} (Personas)
  - SQL: SELECT nombre, edad FROM Personas;

3. Unión (∪)

- Álgebra Relacional: La unión combina todas las tuplas de dos relaciones que tienen el mismo esquema.
- SQL: Se implementa con UNION, que combina los resultados de dos consultas.
- Ejemplo:
  - Álgebra Relacional: Personas ∪ Empleados
  - SQL: SELECT * FROM Personas UNION SELECT * FROM Empleados;

4. Intersección (∩)

- Álgebra Relacional: La intersección devuelve las tuplas comunes entre dos relaciones.
- SQL: Se implementa con INTERSECT.
- Ejemplo:
  - Álgebra Relacional: Personas ∩ Empleados
  - SQL: SELECT * FROM Personas INTERSECT SELECT * FROM Empleados;

5. Diferencia (−)

- Álgebra Relacional: La diferencia devuelve las tuplas que están en una relación pero no en la otra.
- SQL: Se implementa con EXCEPT o MINUS (dependiendo del dialecto).
- Ejemplo:
  - Álgebra Relacional: Personas − Empleados
  - SQL: SELECT * FROM Personas EXCEPT SELECT * FROM Empleados;

6. Producto Cartesiano (×)

- Álgebra Relacional: El producto cartesiano combina todas las tuplas posibles de dos relaciones.
- SQL: En SQL, esto ocurre cuando se seleccionan tablas sin una condición de unión explícita.
- Ejemplo:
  - Álgebra Relacional: Personas × Direcciones
  - SQL: SELECT * FROM Personas, Direcciones;

7. Join o Unión Natural (⋈)

- Álgebra Relacional: El join combina tuplas de dos relaciones basadas en una condición.
- SQL: Corresponde a la cláusula JOIN en SQL.
- Ejemplo:
  - Álgebra Relacional: Personas ⋈ Direcciones
  - SQL: SELECT * FROM Personas INNER JOIN Direcciones ON Personas.id = Direcciones.persona_id;

8. Renombramiento (ρ)

- Álgebra Relacional: Permite renombrar una relación o atributos.
- SQL: Se utiliza el AS para cambiar el nombre de una tabla o columna en una consulta.
- Ejemplo:
  - Álgebra Relacional: ρ_{P} (Personas)
  - SQL: SELECT nombre AS nom FROM Personas;

Conclusión

El álgebra relacional proporciona una base formal para el procesamiento de consultas en bases de datos, definiendo un conjunto preciso de operaciones que se pueden aplicar a los datos. SQL, por su parte, es un lenguaje de alto nivel que implementa esas operaciones, permitiendo a los usuarios interactuar con la base de datos de manera más comprensible y declarativa, sin preocuparse tanto por los detalles de la implementación. Así, entender el álgebra relacional facilita la comprensión profunda de cómo funcionan internamente las consultas SQL.

4. Construya un cuadro comparativo entre la formulación matemática del álgebra relacional de las operaciones de selección, proyección, unión, intersección, diferencia y producto cartesiano, división, y como se traducen en instrucciones de Base de Datos.

Operación	Formulación Matemática	Instrucción SQL
Selección (σ)	σ_{condición}(R)	`SELECT * FROM R WHERE condición;`
Proyección (π)	π_{atributos}(R)	`SELECT atributos FROM R;`
Unión (∪)	R ∪ S	`SELECT * FROM R UNION SELECT * FROM S;`
Intersección (∩)	R ∩ S	`SELECT * FROM R INTERSECT SELECT * FROM S;`
Diferencia (-)	R - S	`SELECT * FROM R EXCEPT SELECT * FROM S;`
Producto cartesiano (×)	R × S	`SELECT * FROM R, S;`
División (÷)	R ÷ S (con S como subconjunto de atributos)	`SELECT R.atributos FROM R WHERE R.atributos IN (SELECT S.atributos FROM S);`

Este cuadro resume cómo las operaciones del álgebra relacional se traducen a instrucciones SQL, facilitando la comprensión de ambos lenguajes.

5. Correspondencia entre operaciones del álgebra relacional y SQL:

El álgebra relacional es un lenguaje de consulta compuesto por varios operadores, cada uno de los cuales toma relaciones como argumentos y devuelve una única relación como resultado.

Los siguientes son los principales operadores en álgebra relacional:

Intersección: La intersección de dos relaciones A y B, denotada A ∩ B, es el
conjunto de puntos que pertenecen tanto a A como a B. El operador de intersección se puede aplicar solo a operandos que tienen el mismo conjunto y orden de atributos.

Unión: La unión de dos relaciones A y B, denotada A ∪ B, es el conjunto de puntos que pertenecen a A o B o a ambos. El operador de unión se puede aplicar solo a operandos que tienen el mismo conjunto y orden de atributos.

Diferencia: La diferencia de dos relaciones A y B, denotada A \ B es el conjunto de puntos que pertenecen a A pero no pertenecen a B. El operador de diferencia se puede aplicar solo a operandos que tienen el mismo conjunto y orden de atributos.

Producto: El operador de producto aplicado a una relación A de n dimensiones y una
relación B de m dimensiones, denotada A × B, devuelve una relación que contiene todos los puntos (n + m)-dimensionales cuyos primeros n componentes pertenecen a A y los últimos m componentes pertenecen a B. El operador de producto se puede aplicar solo a operandos que no tienen atributos comunes.

Proyecto: Este operador se utiliza para reordenar las columnas de una relación o para eliminar
algunas columnas de una relación. El operador de proyecto de una relación A se denota L A, donde L es una lista [l1,...,lk] que especifica un ordenamiento de un subconjunto de los atributos de A.
El operador de proyecto crea una nueva relación que contiene en su i-ésima columna la columna de A correspondiente al atributo li .

Selección: El operador de selección se utiliza para seleccionar de una relación A aquellos puntos que satisfagan una determinada fórmula lógica F. El operador de selección tiene la forma σF A.

Renombrar: El operador de cambio de nombre, ρB(X1/Y1,...,Xn /Yn)A para cualquier n ≥ 1, cambia el
nombre de la relación A a B y cambia el atributo Xi a Yi . Si solo queremos cambiar el nombre de la relación, entonces se puede utilizar la forma ρB A.

Unión natural: El operador de unión natural aplicado a una relación n-dimensional A y una relación m-dimensional B que tienen k atributos en común se denota A B.

El operador de unión natural devuelve una relación que contiene todos los puntos (n + m − k)-
dimensionales cuya proyección sobre los atributos de A pertenece a A y cuya proyección sobre los atributos de B pertenece a B.

6. Que son las Relaciones de Conjuntos y su relación con las Base de Datos.

Para saber qué son las relaciones en las bases de datos es necesario hablar del tipo de base de datos relacionales. En estas bases de datos, la información se almacena en diferentes tablas, distribuida en filas y columnas.

La relación de una base de datos es el vínculo que se establece entre distintos elementos de las tablas que la conforman. En este tipo de relaciones es fundamental el uso de los campos de llave primaria (primary key) que son los que se relacionan con otros registros de otras tablas.

Es importante destacar que, a la hora de definir las relaciones entre los campos de distintas tablas en una base de datos, los nombres de los mismos no tienen por qué ser iguales. Sin embargo, sí es necesario a la hora de establecer estas relaciones, que el tipo de datos de los campos enlazados sea el mismo.