Álgebra Relacional - Operaciones avanzadas

El álgebra relacional es el marco teórico que permite describir cómo operar sobre las relaciones (tablas) en una base de datos relacional. Incluye las operaciones básicas (selección, proyección, unión, diferencia, y producto cartesiano) y operaciones derivadas (join, división, operadores de conjunto).

Ejemplo 1:

Consulta: "Encuentra los nombres de los empleados que trabajan en el departamento de 'Ventas' y ganan más de 50,000."

1. **Selección:** `σSalario > 50000 (Empleados)`
2. **Selección por departamento:** `σDepto = 'Ventas' (Empleados)`
3. **Intersección de ambos criterios:** `σSalario > 50000 ∧ Depto = 'Ventas' (Empleados)`

Álgebra relacional:

$$\pi_{\text{Nombre}}(\sigma_{\text{Salario} > 50000 \wedge \text{Depto} = \text{'Ventas'}}(\text{Empleados}))$$

Ejemplo 2:

Consulta: "Encuentra los productos que han sido pedidos por al menos un cliente y los que tienen un precio mayor a 100."

1. **Selección de productos caros:** `σPrecio > 100 (Productos)`
2. **Proyección de productos en pedidos:** `πProductoID (Pedidos)`
3. **Intersección de ambos conjuntos:**

Álgebra relacional:

$$\pi_{\text{ProductoID}}(\sigma_{\text{Precio} > 100}(\text{Productos})) \cap \pi_{\text{ProductoID}}(\text{Pedidos})$$

Ejemplo 3:

Consulta: "Muestra el nombre y salario de los empleados que no trabajan en el departamento de 'Recursos Humanos'."

1. **Proyección de empleados fuera de Recursos Humanos:** `σDepto ≠ 'Recursos Humanos' (Empleados)`
2. **Proyección de nombre y salario:**

Álgebra relacional:

$$\pi_{\text{Nombre, Salario}}(\sigma_{\text{Depto} \neq \text{'Recursos Humanos'}}(\text{Empleados}))$$

Marco teórico para consultas avanzadas:

- **Unión (∪)**: Combina los resultados de dos consultas, eliminando duplicados.
- **Intersección (∩)**: Devuelve solo las tuplas que aparecen en ambas relaciones.
- **Diferencia (-)**: Devuelve las tuplas que aparecen en la primera relación, pero no en la segunda.
- **Join (⋈)**: Permite combinar tuplas de dos relaciones basándose en una condición específica.

Estas operaciones pueden combinarse para resolver consultas más complejas. Si tienes acceso a un ejemplar del libro de Elmasri, el capítulo 6 profundiza en estos conceptos con muchos más ejemplos.

Consulta: "Encuentra los empleados que no trabajan en el proyecto 'X'."

- **Operación**: `Empleados - πID_Empleado (Trabaja_en ⋈ σNombre_Proyecto = 'X' (Proyectos))`
  - **Explicación**: La consulta realiza una diferencia entre la lista de todos los empleados y aquellos que están asociados con el proyecto 'X'.

Ejemplo 2:

Consulta: "Encuentra los productos que han sido pedidos por todos los clientes."

- **Operación**: `Pedidos ÷ πCliente (Clientes)`
  - **Explicación**: Usamos la operación de división para encontrar los productos que han sido pedidos por todos los clientes.

Guía de Conceptos de Álgebra Relacional y SQL

1. Álgebra Relacional

El álgebra relacional es una teoría matemática que describe un conjunto de operaciones para manipular conjuntos de datos (relaciones o tablas). Es fundamental para entender cómo funcionan las bases de datos relacionales y sus sistemas de gestión.

Operaciones en Álgebra Relacional:

- **Selección (σ)**: Filtra las filas de una relación según una condición específica.
  - Ejemplo: `σSalario > 50000 (Empleados)` devuelve todos los empleados con salario mayor a 50,000.
- **Proyección (π)**: Selecciona columnas específicas de una relación.
  - Ejemplo: `πNombre, Apellido (Empleados)` devuelve solo los nombres y apellidos de los empleados.
- **Unión (∪)**: Combina dos relaciones y elimina duplicados.
  - Ejemplo: `Empleados ∪ ExEmpleados` devuelve todas las tuplas que aparecen en ambas relaciones.
- **Intersección (∩)**: Devuelve las tuplas que están presentes en ambas relaciones.

  - Ejemplo: `Empleados ∩ ExEmpleados` devuelve los empleados que también son ex empleados.
- **Diferencia (−)**: Devuelve las tuplas que están en la primera relación pero no en la segunda.
  - Ejemplo: `Empleados − ExEmpleados` devuelve empleados que no son ex empleados.
- **Producto Cartesiano (×)**: Combina todas las tuplas de dos relaciones.
  - Ejemplo: `Empleados × Departamentos` genera todas las combinaciones posibles de empleados y departamentos.
- **Join (⋈)**: Une dos relaciones basándose en una condición específica.
  - Ejemplo: `Empleados ⋈ Departamentos` une empleados con su departamento basándose en el ID de departamento.

Propiedades del Álgebra Relacional:

- **Cierre relacional**: Las operaciones sobre relaciones siempre devuelven relaciones. Esto permite la composición de operaciones para crear consultas complejas.
- **Declarativa**: El álgebra relacional describe **qué** datos se necesitan, no **cómo** obtenerlos.

2. Varrel (Variable de Relación)

- Un **varrel** es una variable que puede contener una relación. Representa una tabla en un sistema de bases de datos y puede cambiar su contenido a lo largo del tiempo. Las tuplas (filas) de la tabla pueden modificarse mediante operaciones como inserciones, actualizaciones y eliminaciones en lenguajes como SQL.

Diferencia entre Relación y Varrel:

- **Relación**: Es un conjunto inmutable de tuplas. En álgebra relacional, las relaciones son estáticas y no cambian.
- **Varrel**: Es una variable que puede contener diferentes valores de relaciones en diferentes momentos. En SQL, las tablas son varrels, porque su contenido puede cambiar mediante operaciones.

3. Operaciones de Modificación de Datos

En el álgebra relacional pura, no se incluyen operaciones para modificar directamente los datos, como inserciones o eliminaciones. Estas operaciones se manejan en sistemas de bases de datos como **SQL**.

Operaciones de Modificación en SQL:

- **INSERT**: Agrega nuevas tuplas a una relación (tabla).

```
INSERT INTO Empleados (Nombre, Apellido, Salario)
VALUES ('Juan', 'Pérez', 60000);
```

- **UPDATE**: Modifica tuplas existentes en una relación.

```
UPDATE Empleados
```

```
SET Salario = 65000
WHERE Nombre = 'Juan';
```

- **DELETE**: Elimina tuplas de una relación.

```
DELETE FROM Empleados
WHERE Nombre = 'Juan';
```

4. DEE y DUM

- **DEE (relación universal)**: Es una relación especial que contiene exactamente una tupla vacía. Representa la verdad en álgebra relacional. Cuando se usa en una operación, siempre devuelve un resultado.
- **DUM (relación vacía)**: Es una relación especial que no contiene ninguna tupla. Representa la falsedad. Se usa para indicar que no hay resultados válidos en una operación.

5. SQL vs. Álgebra Relacional

- **SQL** es el lenguaje de consulta usado en sistemas de bases de datos relacionales. Implementa muchas de las operaciones del álgebra relacional, como selecciones, proyecciones, uniones y joins.
- **Diferencias clave**:
  - El **álgebra relacional** es teórica, mientras que **SQL** es práctico y se usa en bases de datos reales.
  - **SQL** incluye operaciones de manipulación de datos como **inserción**, **actualización**, y **eliminación**, que no están presentes en el álgebra relacional pura.

1. Varrel (Variable de Relación)

Un **varrel** es una variable que representa una relación (tabla) en la base de datos y puede cambiar a lo largo del tiempo.

Ejemplo de Varrel:

- Supongamos que tenemos una tabla de empleados llamada **Empleados**. La tabla puede cambiar a lo largo del tiempo cuando se agregan, eliminan o actualizan tuplas.
- **Estado inicial** del varrel:

```
Empleados:
| ID | Nombre  | Salario |
|----|---------|---------|
| 1  | Juan    | 50000   |
| 2  | María   | 60000   |
| 3  | Pedro   | 55000   |
```

- **Después de una inserción**:

```
INSERT INTO Empleados (ID, Nombre, Salario)
VALUES (4, 'Ana', 58000);
```

El **varrel Empleados** cambia:

```
Empleados:
| ID | Nombre  | Salario |
|----|---------|---------|
| 1  | Juan    | 50000   |
| 2  | María   | 60000   |
| 3  | Pedro   | 55000   |
| 4  | Ana     | 58000   |
```

Un varrel cambia dinámicamente conforme se realizan operaciones sobre él.

---

2. DEE (Relación Verdadera)

DEE es una relación especial que contiene exactamente una tupla vacía. Representa la verdad o un conjunto universal en álgebra relacional.

Ejemplo de DEE:

- **DEE** es una relación con **cero atributos** y exactamente **una tupla vacía**:

```
DEE: {()}
```

- En consultas de álgebra relacional, **DEE** siempre devuelve un resultado. Es útil en operaciones que no dependen de ninguna condición, por ejemplo:

```
SELECT * FROM DEE;
```

Este **SELECT** devolverá una tupla vacía, representando "verdad" o "universalidad".

---

3. DUM (Relación Falsa)

DUM es lo opuesto a **DEE**. Es una relación que no contiene ninguna tupla, lo que significa que representa la falsedad o el conjunto vacío.

Ejemplo de DUM:

- **DUM** es una relación con **cero atributos** y **cero tuplas**:

```
DUM: {}
```

- En consultas, **DUM** no devuelve ningún resultado. Si intentas consultar datos de una tabla que no contiene tuplas que cumplan las condiciones, obtienes **DUM** como respuesta. Por ejemplo, si buscas empleados con un salario mayor a 1,000,000 y no existe ninguno, obtienes **DUM**.

  ```
  SELECT * FROM Empleados WHERE Salario > 1000000;
  ```

  Como no hay empleados con salario mayor a 1,000,000, la consulta devuelve un conjunto vacío, representando **DUM**.

Las **restricciones de varrel** son restricciones impuestas a una varrel (variable de relación), expresadas solo en términos de esa varrel en particular. Estas restricciones son reglas que determinan qué valores pueden aparecer en la varrel y aseguran que los datos mantengan su consistencia. Aquí algunos ejemplos de restricciones de varrel tomadas del libro de C.J. Date:

1. **Restricción de varrel sobre proveedores en Londres**:
   - **Restricción**: "Los proveedores en Londres deben tener un status de 20".
   - **Expresión**:

     ```
     CONSTRAINT R5V
     ISEMPTY (V WHERE CIUDAD = 'Londres' AND STATUS != 20);
     ```

2. **Restricción sobre el almacenamiento de partes rojas**:
   - **Restricción**: "Las partes rojas deben almacenarse en Londres".
   - **Expresión**:

     ```
     CONSTRAINT R4P
     IS_EMPTY (P WHERE COLOR != 'Rojo' AND CIUDAD !=
     'Londres');
     ```

3. **Restricción de unicidad sobre números de proveedor**:
   - **Restricción**: "Los números de proveedor son únicos".
   - **Expresión**:

     ```
     CONSTRAINT RKV
     COUNT (V) = COUNT (V {V#});
     ```

4. **Restricción sobre la existencia de partes rojas**:
   - **Restricción**: "Si existen partes, por lo menos una de ellas debe ser roja".
   - **Expresión**:

     ```
     CONSTRAINT R7P
     IF NOT (IS_EMPTY (P)) THEN
     COUNT (P WHERE COLOR = 'Rojo') > 0
     END IF;
     ```

Comportamiento de las restricciones de varrel:

- Estas restricciones se verifican inmediatamente, como parte de la ejecución de cualquier instrucción que pueda violarlas. Si una operación intenta asignar un valor que infringe una restricción, la operación será rechazada y no se aplicará.

Las restricciones de varrel son esenciales para garantizar la **integridad de los datos** en las bases de datos, impidiendo que se ingresen o modifiquen datos incorrectos en una tabla individual .

Varrel (o **relvar**, que es la abreviatura de **relation variable** o **variable de relación**) es un término que se utiliza en el contexto de las bases de datos relacionales. Una **varrel** es una **variable** que representa una **relación** (tabla) en una base de datos relacional, y su contenido puede cambiar a lo largo del tiempo.

Conceptos clave sobre varrel:

- **Varrel (variable de relación)**: Es una tabla cuyos datos (tuplas o filas) pueden ser modificados mediante operaciones como **inserciones**, **eliminaciones** o **actualizaciones**.
- En teoría relacional, una **relación** es un conjunto inmutable de tuplas, pero en un sistema de base de datos real, las tablas no son inmutables; pueden cambiar, y por eso se les llama **varrel**, ya que son variables que pueden contener relaciones diferentes en distintos momentos.