Optimización de consultas relacionales mediante el álgebra relacional

Dània Perpiñà Martí

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Director del trabajo: Jorge Lloret Gazo
13 de junio de 2023

Prólogo

Actualmente, las empresas recogen una gran cantidad de datos procedentes de muchas fuentes distintas. Es necesario analizar los datos para obtener información de calidad que pueda utilizarse para tomar las mejores deciciones empresariales.

El modelado de datos da la oportunidad de entender los datos y tomar las decisiones tecnológicas correctas para almacenarlos y gestionarlos. El modelo relacional se asemeja a una biblioteca bien organizada. En una biblioteca, los libros están clasificados y se pueden acceder fácilmente a través de su sistema de catalogación. Del mismo modo, en el modelo relacional, los datos se organizan en relaciones representadas como tablas, las cuales se pueden consultar y relacionar de manera eficiente.

En este documento se describe una introducción al álgebra relacional, el lenguaje para expresar consultas del modelo relacional, y la optimización de dichas consultas mediante árboles de consulta.

En concreto, en el Capítulo 1 se introduce el modelo relacional y se explican sus conceptos básicos. A continuación, en el Capítulo II se explican las operaciones más importantes del álgebra relacional. Con estas operaciones podremos realizar consultas en nuestra base de datos. Posteriormente, en el Capítulo III se desarrolla la idea principal del trabajo, la optimización de consultas. Mediante un algoritmo de optimización y varios ejemplos, se exponen varias formas de escribir la misma consulta hasta llegar a la más óptima.

Resumen

Currently, the relational model is very important in the data management of any company. It offers a robust framework for structuring data, establishing relationships between different entities, and performing complex queries and analyses.

This document is an introduction to relational algebra, the language that the relational model uses to express queries.

In Chapter I, I define the basic concepts of the relational model:

- **Atributte:** Relationship property.
- **Domain:** Set of atomic values that an attribute can be.
- **Relational schema:** Description of the relationship formed by a name and a list of attributes.
- **Relational schema database:** set of relationship schemas.
- **Tuple:** Ordered set of values that corresponds to the attributes or columns of the table.
- **Relation:** Set of tuples containing values for each attribute defined in the relation's schema.
- **State of relational database:** Set of all the relations in the relational database that comply the integrity constraints.

In the Chapter II, I describe the basic operations of relational algebra:

- **SELECT:** this provides a new relation with the tuples that satisfy the condition.
- **PROJECT:** this provides a new relation with the attributes in the attribute list.
- **JOIN:** joins tuples from two different relations that share the same value in different attributes.
- **RENAME:** this changes the result's name, or the atributes name, or both.
- **CARTESIAN PRODUCT:** creates a new relation with all possible combinations of the tuples of both relations.

Finally, the Chapter III is an introduction to query trees and their optimization. From an SQL query, translating it into relational algebra, we can create a query tree. But this tree won't be optimal. In this chapter I introduced some transformation rules and an optimization algorithm. With this, we will transform the initial query tree into an optimized tree. This optimized tree is more efficient. We will execute this tree to obtain the result of the query.

Índice general

Capítulo 1

Presentación del modelo

El modelo relacional usa el concepto matemático de esquema de relación como bloque de construcción básico y representa las bases de datos como un conjunto de esquemas de relaciones. Además, su base teorica es la teoría de conjuntos y la lógica de predicados de primero orden.

El modelo relacional tiene asociado el álgebra relacional, que tiene dos características fundamentales:

- Es la base del lenguaje SQL.
- Se utiliza en muchas implementaciones de bases de datos para el procesamiento y la optimización de consultas.

El modelo relacional fue presentado por primera vez por Ted Codd de IBM Research en 1970 en un artículo llamado "*A Relational Model of Data for Large Shared Data Banks*". En esta publicación creó el concepto de las bases de datos relacionales. Esto dió lugar a las 13 reglas de Codd, que establecen conceptos que hoy en día se utilizan a la hora de trabajar con bases de datos.

Inicialmente esto no fue suficiente para llamar la atención de IBM, ya que estaba centrado en bases de datos jerárquicas. Fue entonces cuando Larry Ellison, Bob Miner y Ed Oates fundaron la empresa Software Development Laboratories (SDI) en 1977. Fue ahí cuando nació Oracle, que es un sistema de administración de base de datos que destaca por sus transacciones, estabilidad, escalabilidad y multiplataforma.

En 1974, viendo el potencial del modelo, IBM creó un lenguaje para los sistemas de gestión de bases de datos relacionales basado en el trabajo de Codd. Ese lenguaje primero se llamó SEQUEL, siglas de *Structured English Query Language* y después de varias implementaciones y revisiones, pasó a llamarse SQL, *Structured Query Language* .

Actualmente, el modelado de datos se ha convertido en una herramienta muy importante en el crecimiento de las empresas, independientemente de su tamaño o sector de negocio. Las empresas suelen hacer uso de este tipo de base de datos para modelar sus datos y garantizar la simplicidad del sistema.

Las ventajas de este tipo de base de datos es que son fáciles de crear y ampliar, es decir, se pueden añadir nuevos datos sin necesidad de modificar las aplicaciones.

En conclusión, una base de datos relacional es un tipo de base de datos en la que los datos se definen de tal manera que se pueden reorganizar y acceder a ellos mediante distintas sentencias SQL. Los usuarios pueden interactuar con la base de datos para realizar operaciones como la consulta, creación, actualización o eliminación de datos de la relación.

Las bases de datos relacionales se utilizan para hacer un seguimiento de los inventarios, procesar transacciones de comercio electrónico o gestionar grandes cantidades de información.

Un ejemplo concreto es una empresa que decide almacenar todos sus datos importartantes en una base de datos relacional. Esta base de datos consta de varias relaciones, representadas como tablas. Cada tabla está compuesta por registros y campos. Los registros son las filas de la tabla y lo que formalmente se denomina como tupla. Los campos, son las columnas de la tabla y lo que formalmente se denomina como atributo. Así, recorriendo los distintos valores de la tupla obtendremos todos los datos que recoge la tabla sobre esa instancia en concreto. En el siguiente apartado ilustraremos el ejemplo anterior.

1.1. Conceptos básicos del modelo relacional

En esta sección se definen los conceptos más importantes del modelo relacional. Primero enunciaremos los conceptos relacionados con la estrucutura de las bases de datos, y luego, los conceptos relacionados con los datos.

Para la introducción de estos nuevos conceptos utilizaremos la siguiente relación de ejemplo:

EMPLEADO

Nombre	Apellido1	Apellido2	DNI	FechaNac	Dirección	Sexo	Salario	DNISup	Dnum
María	García	Pérez	18273082Q	01/01/1997	Calle Ávila, 34	M	32000	54121335J	1
José	David	Hoyos	29102495H	10/01/1975	Calle Alcañiz, 65	H	37000	64705566M	1
María	Cinta	Vives	60025189J	29/10/1982	Calle Celsa, 43	M	42000	55125799G	2
Martina	Villa	Vázquez	65202738O	11/10/1989	Vía Hispanidad, 32	M	45000	55125799G	5
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1
León	Nieto	Cadenas	72163703N	04/11/1975	Calle San Agustín, 27	H	27000	54121335J	3
Diego	Zamora	Navarro	87203689M	09/10/1987	Calle San Pablo, 56	H	51000	54121335J	1
Jordán	Cabo	Quintanilla	18952434R	07/09/1987	Calle Sarrión, 18	H	31000	55125799G	4
Sara	Plaza	Arenas	85961791B	03/02/1981	Calle Zafiro, 41	M	29000	55125799G	3
Marta	Castillo	Pérez	24946017H	27/05/1968	Calle Alfonso, 9	M	32000	55125799G	5

Figura 1.1: Esquema de relación Empleado junto con sus datos

A continuación vamos a introducir los conceptos relacionados con la estructura de la base de datos:

- **Atributo**

  Un atributo representa una propiedad de la relación. Además, cada atributo tiene un tipo de datos específico, que viene determinado por su dominio.

  En el ejemplo de la relación empleado, los atributos son Nombre, Apellido1, Apellido2, DNI, FechaNac, Dirección, Sexo, Salario, DNISup y Dnum.

  Cabe destacar que dos atributos pueden representar el mismo concepto, como por ejemplo DNI y DNISup. Ambos atributos son DNI, pero uno corresponde al DNI del empleado y el otro al DNI del superior. Para poder diferenciarse el uno del otro, estamos obligados a darle un nombre distinto a cada atributo.

- **Dominio $\mathscr{D}$**

  Un dominio $\mathscr{D}$ es un conjunto de valores atómicos. Por atómico entendemos que cada valor en el dominio es indivisible en lo que respecta al modelo relacional.

  El dominio de los atributos se construye con una definición lógica, el tipo de dato e información opcional.

  En el ejemplo de la relación empleado, la parte de defincón lógica y especificación del tipo de dato del dominio vendría dado por:

  - *Nombre*: nombre del empleado. De 1 a 30 carácteres.
  - *Apellido1*: primer apellido del empleado. De 1 a 30 carácteres.
  - *Apellido2*: segundo apellido del empleado. De 1 a 30 carácteres.
  - *DNI*: combinación de ocho números más una letra válidos correspondientes al número de DNI.
  - *FechaNac*: fecha de nacimiento del empleado, formato *dd/mm/aaaa*.
  - *Direción*: dirección del empleado. De 1 a 50 carácteres.
  - *Sexo*: Sexo del empleado, femenino o masculino (F/M).
  - *Salario*: salario del empleado, número entero de 0 a 100.000 (con la restricción de que no puede ser negativo).
  - *DNISuper*: número de la seguridad social del supervisor del empleado, el dominio de este atributo coincide con el dominio de Ssn.
  - *Dnum*: departamento donde trabaja el empleado, número entero de 1 a 5.

- **Esquema de relación**

  Un esquema de relación está formado un nombre y una lista de atributos:

  $$\mathscr{R} = \{A_1, A_2, ..., A_n\}$$

  El esquema de relación de Empleado es:

  Empleado = { Nombre, Apellido1, Apellido2, DNI, FechaNac, Dirección, Sexo, Salario, DNISup, Dnum }

  El grado de un esquema de relación es el número de atributos de su esquema de relación.
  El grado de la relación de Empleado es 10.

- **Esquema de base de datos S**

  Un esquema de base de datos es un conjunto de esquemas de relación $S = \{R_1, R_2, ..., R_n\}$ y un conjunto de restricciones de integridad IC.

  Se puede interpretar como una declaración o un tipo de afirmación, es decir, la información que nos proporciona un esquema de relación es que una entidad posee un valor para cada uno de los atributos del esquema.

  Otro aspecto importante son las claves primarias. Son un atributo para el que no hay dos instancias con el mismo valor, además de que no puede ser NULL. En este ejemplo, la clave primaria es DNI, porque este es único para cada persona, ya que no pueden existir dos empleados con el mismo DNI. Sin embargo, pueden existir dos empleados con el mismo nombre o con la misma fecha de nacimiento, pero siempre se pueden diferenciar por su DNI.

El esquema de relación Empleado es solo uno de los esquemas donde la empresa almacena información. Existen otros esquemas relacionadas con el esquema Empleado donde la empresa guarda información sobre distintas entidades como departamentos o proyectos. Un ejemplo de esquema de base de datos es:

EMPRESA = { EMPLEADO, DEPARTAMENTO, HORARIO, PROYECTO }

El diagrama del esquema de base de datos correspondiente a este ejemplo es:

Figura 1.2: Diagrama del esquema de relación

Una vez definidos los conceptos más importantes de la estructura de una base de datos, vamos a definir los conceptos relacionados con los datos:

- **Tupla**

  Una tupla representa una instancia única de los atributos de un esquema de relación.

  Por ejemplo, en el caso de la relación Empleado, cada tupla recoge características sobre una instancia de la entidad empleado. Es decir, cada tupla representa un empleado. Para cada empleado se registra su nombre, primer apellido, segundo apellido, DNI, fecha de nacimiento, dirección, sexo, salario, DNI del supervisor y departamento.

  Varios ejemplos de tuplas de la relación Empleado son:

  - (María, García, Pérez, 18273082Q, 01/01/1997, Calle Ávila, 34, M, 32000, 54121335J, 1)
  - (José, David, Hoyos, 29102495H, 10/01/1975, Calle Alzañiz, 65, H, 37000, 64705566M, 1)

- **Relación r($\mathscr{R}$)**

  Dado un esquema de relación $\mathscr{R}(A_1, A_2, ..., A_n)$, una relación denotada como r($\mathscr{R}$) es un conjunto de m-tuplas $r = \{t_1, t_2, ..., t_m\}$. Cada tupla es una lista ordenada de n-valores, $t =< v_1, v_2, ...v_n >$, donde cada valor v_i para $1 \leq i \leq n$ es un elemento de $dom(A_i)$ o es un valor NULL, es decir, cada valor v_i corresponde al valor que toma la entidad para el atributo i-ésimo de la relación.

  Las tuplas de una relación no están ordenadas. Sin embargo, una tupla es una lista ordenada de n-valores, las cuales siguen el orden de los atributos del esquema relacional.

Además, cada valor de una tupla es atómico, es decir, no se permiten atributos multivalorados, ya que éstos deberían estar representados por relaciones separadas.

También podríamos definirlo desde un punto de vista conjuntista: una relación $r(\mathscr{R})$ es un subconjunto del producto cartesiano de los dominios que definen $\mathscr{R}$:

$$r(\mathscr{R}) \subset (dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$$

donde el producto cartesiano especifica todas las posibles combinaciones de valores de los dominios en cuestión.

En el caso del esquema de relación Empleado, una relación son las tuplas correspondientes a la Figura 1.1.

- **Estado de base de datos relacional *DB* de una base de datos**

  Un estado es una colección de relaciones, una para cada esquema de base de datos, que satisfacen las restricciones de integridad. Se denota $DB = \{r_1, r_2, ..., r_n\}$ donde cada r_i es un estado de R_i.

  Un estado de base de datos que no obedece todas las restricciones de integridad es llamado no válido. Si, por el contrario, satisface todas las restricciones de integridad, es llamado estado válido.

  En el caso de de la base de datos empresa, un estado de esa base de datos puede verse en la Figura 1.3 a 1.6.

EMPLEADO

Nombre	Apellido1	Apellido2	DNI	FechaNac	Dirección	Sexo	Salario	DNISup	Dnum
María	García	Pérez	18273082Q	01/01/1997	Calle Ávila, 34	M	32000	54121335J	1
José	David	Hoyos	29102495H	10/01/1975	Calle Alcañiz, 65	H	37000	64705566M	1
María	Cinta	Vives	60025189J	29/10/1982	Calle Celsa, 43	M	42000	55125799G	2
Martina	Villa	Vázquez	65202738O	11/10/1989	Vía Hispanidad, 32	M	45000	55125799G	5
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1
León	Nieto	Cadenas	72163703N	04/11/1975	Calle San Agustín, 27	H	27000	54121335J	3
Diego	Zamora	Navarro	87203689M	09/10/1987	Calle San Pablo, 56	H	51000	54121335J	1
Jordán	Cabo	Quintanilla	18952434R	07/09/1987	Calle Sarrión, 18	H	31000	55125799G	4
Sara	Plaza	Arenas	85961791B	03/02/1981	Calle Zafiro, 41	M	29000	55125799G	3
Marta	Castillo	Pérez	24946017H	27/05/1968	Calle Alfonso, 9	M	32000	55125799G	5

Figura 1.3: Esquema de relación Empleado junto con sus datos

PROYECTO

Pname	Pnumero	Plocalizacion	Dnum
ProyectoA	1	Zaragoza	3
ProyectoB	2	Madrid	4
ProyectoC	3	Barcelona	5
ProyectoD	4	Zaragoza	4
ProyectoE	10	Málaga	5

Figura 1.4: Relación Proyecto

HORARIO

EDNI	PNUM	Horas
18273082Q	1	40
29102495H	1	35
29102495H	2	5
60025189J	1	40
65202738O	2	40
12457418P	2	15
12457418P	3	15
12457418P	4	10
72163703N	4	40
87203689M	4	40
18952434R	4	5
18952434R	10	35
85961791B	10	40
24946017H	10	40

Figura 1.5: Relación Horario

DEPARTAMENTO

Dnombre	Dnum	FechaCreac
Dirección general	1	01/05/1992
Recursos humanos	2	01/08/1993
Marketing	3	01/09/1998
Comercial	4	01/11/2000
Logística	5	01/12/2001

Figura 1.6: Relación Departamento

Capítulo 2

Álgebra relacional

El álgebra relacional es una estructura matemática que posee una colección de operadores que actuan para modelar datos y definir consultas sobre ellos. Es el lenguaje de consulta del modelo relacional, proporcionándole una base formal para las operaciones. Su objetivo principal es transformar una o más operaciones de entrada en una relación de salida, la cual representará el resultado de consulta.

Sus operadores se dividen en dos grupos:

- **Operaciones específicas para bases de datos**

  Son las operaciones SELECT, PROJECT y JOIN.

- **Operaciones basadas en la teoría de conjuntos matemática**

  Son las operaciones UNION, INTERSECCIÓN, SET DIFERENCE y CARTESIAN PRODUCT (CROSS PRODUCT).

Aparte de estas, hay algunas operaciones adicionales que permiten resumir los datos de las relaciones, y otras que son operaciones derivadas de los tipos anteriores, por ejemplo OUTER JOIN.

2.1. Operación SELECT

La operación SELECT nos permite filtrar tuplas de una relación mediante una condición. En el caso de la relación Empleado, nos permite filtrar empleados, diferenciando entre los que cumplen la condición y los que no. Esta propiedad recibe el nombre de selectividad de la condición.

SELECT nos proporciona una nueva relación con los empleados que cumplen la condición. Esta condición puede ser desde una condición simple como que el salario es igual a una cantidad de dinero, o una condición más compleja, como una combinación de varias condiciones.

Es como hacer una partición horizontal, donde distinguimos entre dos tipos de tupla: las que cumplen la condición y las que no.

La notación que se utiliza para la operación SELECT es:

$$\sigma_{\text{condición de selección}}(\mathscr{R})$$

donde la condición de selección es una expresión booleana.

Ejemplo

Queremos seleccionar las tuplas de la relación Empleado cuyo departamento es el número 1. La operación SELECT se define como:

$$\sigma_{\text{Dno=1}}\,(EMPLEADO)$$

Esta operación nos daría como resultado la siguente relación:

Nombre	Apellido1	Apellido2	DNI	FechaNac	Dirección	Sexo	Salario	DNISup	Dnum
María	García	Pérez	18273082Q	01/01/1997	Calle Ávila, 34	M	32000	54121335J	1
José	David	Hoyos	29102495H	10/01/1975	Calle Alcañiz, 65	H	37000	64705566M	1
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1
Diego	Zamora	Navarro	87203689M	09/10/1987	Calle San Pablo, 56	H	51000	54121335J	1

Figura 2.1: Resultado operación SELECT

En la condición se puede utilizar cualquiera de los siguientes operadores:

$$\{=, <, >, \leq, \geq, \neq\}$$

También se puede combinar condiciones:

- Condición1 AND Condición2 : la operación SELECT seleccionará la tupla si cumple las dos condiciones.
- Condición1 OR Condición2: la operación SELECT seleccionará la tupla si cumple almenos una de las condiciones.
- NOT Condición: la operación SELECT seleccionará la tupla si no cumple la condición.

Ejemplo

Queremos seleccionar las tuplas cuyo salario es mayor de 35000 y que trabajan en el departamento número 1. La operación SELECT se define como:

$$\sigma_{\text{(Dno=1) AND (Salario>35000)}}\,(EMPLEADO)$$

Esta operación nos daría como resultado la siguiente relación:

Nombre	Apellido1	Apellido2	DNI	FechaNac	Dirección	Sexo	Salario	DNISup	Dnum
José	David	Hoyos	29102495H	10/01/1975	Calle Alcañiz, 65	H	37000	64705566M	1
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1
Diego	Zamora	Navarro	87203689M	09/10/1987	Calle San Pablo, 56	H	51000	54121335J	1

Figura 2.2: Resultado operación SELECT

2.2. Operación PROJECT

La operación PROJECT nos permite obtener una nueva relación con todos los valores que toma un atributo. Es decir, a diferencia de la operación SELECT, que eliminaba tuplas según si cumplía o no la condición, el concepto de esta nueva operación es parecido pero con las atributos.

La operación PROJECT proyecta la relación sobre los atributos. En este caso se trata de una partición vertical de la tabla, donde la operación descarta o no un atributo dependiendo de la lista de atributos introducida en la operación.

La forma general de la operación PROJECT es:

$$\pi_{\text{Lista de atributos}}(\mathscr{R})$$

donde la lista de atributos es una sublista de los atributos del esquema de relación $\mathscr{R}$.

Ejemplo

Queremos obtener los nombres, apellidos y DNI de todos los empleados del esquema de relación Empleado. Para esta constulta, la operación PROJECT se define como:

$$\pi_{\text{Nombre, Apellido1, Apellido2, DNI}}(\text{Empleado})$$

La relación resultante es:

Nombre	Apellido1	Apellido2	DNI
María	García	Pérez	18273082Q
José	David	Hoyos	29102495H
María	Cinta	Vives	60025189J
Martina	Villa	Vázquez	65202738O
Alfredo	Francisco	Castilla	12457418P
León	Nieto	Cadenas	72163703N
Diego	Zamora	Navarro	87203689M
Jordán	Cabo	Quintanilla	18952434R
Sara	Plaza	Arenas	85961791B
Marta	Castillo	Pérez	24946017H

Figura 2.3: Resultado operación PROJECT

La clave primaria de este esquema de relación es el atributo DNI. En este caso, estaba incluido en la lista de atributos de la operación PROJECT que se ha realizado. Por lo tanto, la relación resultante no va a tener ninguna tupla repetida, ya que para cada tupla el valor de DNI va a ser distinto.

Sin embargo, si en la lista de atributos de la operación PROJECT no incluimos la clave primaria del esquema de relación, la relación resultante puede contener tuplas repetidas.

2.3. Operación JOIN

Esta operación es muy importante para cualquier base de datos con más de un esquema de relación porque nos permite procesar dos o más esquemas de relación. El resultado de esta operación es una

relación que usa la información de varios esquemas.

La operación JOIN une tuplas de dos relaciones distintas que comparten el mismo valor en atributos distintos. El resultado de la consulta es una relación con tuplas de mayor longitud, ya que contendrá los atributos de ambas relaciones iniciales.

Para describir esta operación se utiliza la siguiente notación:

$$\mathscr{S} \bowtie_{A_i=B_j} \mathscr{R}$$

donde $\mathscr{S}$ y $\mathscr{R}$ son los dos esquemas de relación, y A_i, B_j son los atributos de unión.

Ejemplo

Queremos todos los datos de empleado, los proyectos en los que participa y el número de horas que dedica a cada uno.

La operación que describe la relación anterior es:

$$\text{Empleado} \bowtie_{DNI=EDNI} \text{Horario}$$

Y la relación resultante es:

Nombre	Apellido1	Apellido2	DNI	FechaNac	Dirección	Sexo	Salario	DNISup	Dnum	EDNI	PNUM	Horas
María	García	Pérez	18273082Q	01/01/1997	Calle Ávila, 34	M	32000	54121335J	1	18273082Q	1	40
José	David	Hoyos	29102495H	10/01/1975	Calle Alcañiz, 65	H	37000	64705566M	1	29102495H	1	35
José	David	Hoyos	29102495H	10/01/1975	Calle Alcañiz, 65	H	37000	64705566M	1	29102495H	2	5
María	Cinta	Vives	60025189J	29/10/1982	Calle Celsa, 43	M	42000	55125799G	2	60025189J	1	40
Martina	Villa	Vázquez	65202738O	11/10/1989	Vía Hispanidad, 32	M	45000	55125799G	5	65202738O	2	40
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1	12457418P	2	15
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1	12457418P	3	15
Alfredo	Francisco	Castilla	12457418P	26/07/1992	Calle Juan Padilla, 65	H	49000	54121335J	1	12457418P	4	10
León	Nieto	Cadenas	72163703N	04/11/1975	Calle San Agustín, 27	H	27000	54121335J	3	72163703N	4	40
Diego	Zamora	Navarro	87203689M	09/10/1987	Calle San Pablo, 56	H	51000	54121335J	1	87203689M	4	40
Jordán	Cabo	Quintanilla	18952434R	07/09/1987	Calle Sarrión, 18	H	31000	55125799G	4	18952434R	4	5
Jordán	Cabo	Quintanilla	18952434R	07/09/1987	Calle Sarrión, 18	H	31000	55125799G	4	18952434R	10	35
Sara	Plaza	Arenas	85961791B	03/02/1981	Calle Zafiro, 41	M	29000	55125799G	3	85961791B	10	40
Marta	Castillo	Pérez	24946017H	27/05/1968	Calle Alfonso, 9	M	32000	55125799G	5	24946017H	10	40

Figura 2.4: Resultado operacion JOIN

Podemos observar como la relación resultante es una combinación de las tuplas de la relación Empleado y la relación Horario, cuyo nexo de unión es el atributo DNI.

2.4. Operación RENAME

Cuando tenemos que realizar una secuencia de operaciones tenemos la opción de desglosar esa secuencia en operaciones más sencillas. Para eso utilizaremos la operación RENAME.

La operación RENAME puede cambiar el nombre del resultado de una operación o los nombres de los atributos, o ambos.

La operación RENAME se aplica a un esquema de relación $\mathscr{R}$ de grado n y se denota como:

$$\rho_{\mathscr{S}(B_1,B_2,...B_n)}(\mathscr{R})$$

Donde $\mathscr{S}$ es el nombre del esquema de relación que se genera y $B_1, B_2, ..., B_n$ son los nuevos nombres de los atributos.

Ejemplo

Queremos el nombre, apellidos y salarios de los empleados que trabajan en el departamento 1.

Esta consulta está formada por dos operaciones, primero una operación SELECT donde se seleccionan todas las tuplas que cumple la condición de que Dnum = 1, y luego, sobre la relación resultante, una operación PROJECT donde se seleccionan los atributos Nombre, Apellido1 y Apellido2.
La consulta se puede plantear como una composición de operaciones:

$$\pi_{\text{Nombre, Apellido1, Apellido2}}(\sigma_{\text{Dno = 1}}(\text{Empleado}))$$

En cambio, utilizando la operación RENAME:

$$\text{DEP\_EMPLEADOS} \leftarrow \sigma_{\text{Dno = 1}}(\text{Empleado})$$

$$\pi_{\text{Nombre, Apellido1, Apellido2}}(\text{DEP\_EMPLEADOS})$$

Esta consulta solo constaba de dos operaciones. En consultas más complejas con más operaciones, el uso de RENAME simplifica la estructura. Además, facilita la lectura si se utilizan nombres representativos para las consultas intermedias.

2.5. Operación CARTESIAN PRODUCT

La operación CARTESIAN PRODUCT es una operación binaria que crea una nueva relación con todas las posibles combinaciones de las tuplas de ambas relaciones.

Aplicar la operación de CARTESIAN PRODUCT a dos esquemas de relación $\mathscr{R}$ y $\mathscr{S}$ se denota como:

$$\mathscr{R}(A_1,...,A_n) \times \mathscr{S}(B_1,...,B_m) = \mathscr{Q}(A_1,...,A_n,B_1,...,B_m)$$

donde $\mathscr{Q}$ es el nuevo esquema de relación, cuyas tuplas son todas las posibles combinaciones de las tuplas de $\mathscr{R}$ y $\mathscr{S}$.

Ejemplo

Realizamos la siguiente consulta:

$$\text{Departamento(Dnombre, Dnum, FechaCreac, DNISup)} \times \text{Proyecto(Pnombre, Pnum, Plocalizacion, Dnum)}$$

donde para cada tupla de la relación Departamento emparejará con cada una de las tuplas de la relación Proyecto.

En la Figura 2.7 se muestra la relación resultante de la operación.

Dnombre	Dnum	FechaCreac	DNISup	Pnombre	Pnum	Plocalizacion	Dnum
Dirección general	1	01/05/1992	54121335J	ProyectoA	1	Zaragoza	3
Dirección general	1	01/05/1992	54121335J	ProyectoB	2	Madrid	4
Dirección general	1	01/05/1992	54121335J	ProyectoC	3	Barcelona	5
Dirección general	1	01/05/1992	54121335J	ProyectoD	4	Zaragoza	4
Dirección general	1	01/05/1992	54121335J	ProyectoE	10	Málaga	5
Recursos humanos	2	01/08/1993	64705566M	ProyectoA	1	Zaragoza	3
Recursos humanos	2	01/08/1993	64705566M	ProyectoB	2	Madrid	4
Recursos humanos	2	01/08/1993	64705566M	ProyectoC	3	Barcelona	5
Recursos humanos	2	01/08/1993	64705566M	ProyectoD	4	Zaragoza	4
Recursos humanos	2	01/08/1993	64705566M	ProyectoE	10	Málaga	5
Marketing	3	01/09/1998	55125799G	ProyectoA	1	Zaragoza	3
Marketing	3	01/09/1998	55125799G	ProyectoB	2	Madrid	4
Marketing	3	01/09/1998	55125799G	ProyectoC	3	Barcelona	5
Marketing	3	01/09/1998	55125799G	ProyectoD	4	Zaragoza	4
Marketing	3	01/09/1998	55125799G	ProyectoE	10	Málaga	5
Comercial	4	01/11/2000	54121335J	ProyectoA	1	Zaragoza	3
Comercial	4	01/11/2000	54121335J	ProyectoB	2	Madrid	4
Comercial	4	01/11/2000	54121335J	ProyectoC	3	Barcelona	5
Comercial	4	01/11/2000	54121335J	ProyectoD	4	Zaragoza	4
Comercial	4	01/11/2000	54121335J	ProyectoE	10	Málaga	5
Logística	5	01/12/2001	55125799G	ProyectoA	1	Zaragoza	3
Logística	5	01/12/2001	55125799G	ProyectoB	2	Madrid	4
Logística	5	01/12/2001	55125799G	ProyectoC	3	Barcelona	5
Logística	5	01/12/2001	55125799G	ProyectoD	4	Zaragoza	4
Logística	5	01/12/2001	55125799G	ProyectoE	10	Málaga	5

Figura 2.5: Relación resultante del PRODUCT CARTESIAN

2.6. Ejemplo completo 1

Al realizar una consulta, podemos utilizar varias operaciones a la vez. Este ejemplo y el siguiente ilustran esta situación.

Queremos el nombre y el DNI de los empleados que cobran más de 30.000 y que trabajan en el departamento de Dirección General o de Comercial. Esta información corresponde a la siguiente consulta:

$$\text{Empleados30} \leftarrow \sigma_{\text{Salario>30.000}}(\text{Empleado})$$

$$\text{DirecGeneral\_o\_comercial} \leftarrow \sigma_{\text{Dnombre = 'Dirección general' OR Dnombre ='Comercial'}}(\text{Departamento})$$

$$\pi_{\text{Nombre, DNI}}((\text{Empleados30}) \bowtie_{\text{D.Dnum = E.Dnum}} (\text{DirecGeneral\_o\_comercial}))$$

El resultado de esta consulta es:

Nombre	DNI
María	18273082Q
José	29102495H
Alfredo	12457418P
Diego	87203689M

Figura 2.6: Relación resultante

2.7. Ejemplo completo 2

Queremos el nombre completo de los empleados cuya fecha de nacimiento es posterior a 03/09/1950 y que trabajan en el proyectoA. Para obtener esta información, escribimos la siguiente consulta:

$$\text{Empleado\_Proyecto} \leftarrow (\sigma_{\text{FechaNac>'03/09/1950'}}(\text{Empleado})) \bowtie_{\text{DNI = EDNI}} (\sigma_{\text{Nombre = 'ProyectoA'}}(\text{Proyecto}))$$

$$\pi_{\text{Nombre, Apellido1, Apellido2}} ((\text{Empleado\_Proyecto}) \bowtie_{\text{P.Pnum = H.Pro}} (\text{Horario}))$$

El resultado de la consulta anterior es:

Nombre	Apellido1	Apellido2
María	García	Pérez
José	David	Hoyos
María	Cinta	Vives

Figura 2.7: Relación resultante

Capítulo 3

Optimización de consultas

En esta sección veremos técnicas de optimización que aplican reglas heurísticas para modificar la representación interna de una consulta y así mejorar su rendimiento. La representación interna de una consulta es un árbol de consultas o un gráfico de consultas.

El objetivo de la optimización de consultas es seleccionar la mejor estrategia posible para la evaluación de consultas, es decir, la que genera menos tiempo de respuesta. El término optimización es un nombre inapropiado para este tipo de operaciones ya que el plan de ejecución elegido puede no ser siempre el plan más óptimo posible. Esto es debido a que el término óptimo incluye otro factores a parte del tiempo de respuesta, como podrían ser la eficiencia, la rentabilidad y el tiempo.

3.1. Arquitectura de un gestor de bases de datos

En este apartado vamos a introducir las técnicas internas utilizadas por gestores de bases de datos (DBMS) para procesar las consultas SQL.

Un sistema de gestión de bases de datos (DBMS) es un software de sistema para crear y administrar bases de datos a través de consultas SQL.

Los pasos típicos al procesar una consulta SQL son:

1. **Escaneo, análisis y validación**

   Una vez introducida la consulta se somete a un escaneo, donde se identifican las palabras clave SQL, los nombres de los atributos y los nombres de los esquemas de relación.

   Seguidamente, durante el proceso análisis, se comprueba si la sintaxis de la consulta es correcta según las reglas de sintaxis del lenguaje de la consulta.

   Y por último, en el proceso de validación, donde se comprueba que los nombres de los atributos y esquemas de relación sean válidos y semánticamente significativos en el esquema de la base de datos.

   Tras los procesos anteriores, se elabora una representación de la consulta en forma de árbol. Este árbol se llama árbol inicial de la consulta.

2. **Optimizador de consultas**

   El DBMS diseña una estrategia de ejecución para obtener el árbol optimizado equivalente al árbol inicial.

   Una consulta tiene muchas estrategias de ejecución posibles que proporcionan el mismo resultado final. Un DBMS debe evaluar sistemáticamente todas las alternativas de ejecución de la consulta

y elegir la más adecuada para procesarla. Este proceso se conoce como optimización de consultas. En este punto se sitúa la parte principal de nuestro trabajo, que se explicará con más detalle en los siguientes apartados.

3. **Generador de código de consulta**

   Se genera el código correspondiente al árbol optimizado para ejecutarlo.

4. **Procesador de la base de datos entiempo de ejecución**

   Se ejecuta el código obtenido en el paso anterior obteniendo el resultado de la consulta.

   Si se produce un error en tiempo de ejecución, el procesador de la base de datos en tiempo de ejecución genera un mensaje de error.

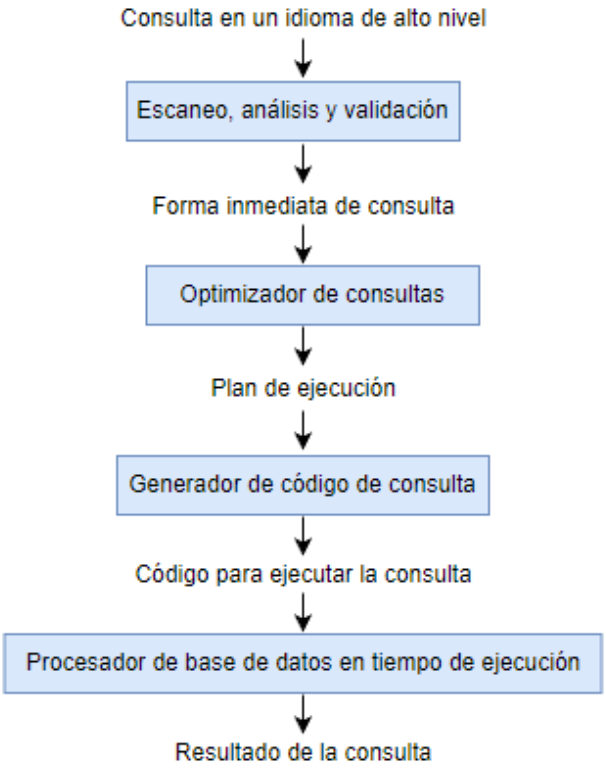

Figura 3.1: Proceso de una consulta

3.2. Notación para árboles de consulta

Como hemos explicado anteriormente, un árbol de consultas es la representación interna de una consulta. En este apartado vamos a explicar como se construyen los árboles de consulta.

Un árbol de consultas es una estructura de datos que corresponde a una expresión de álgebra relacional. El árbol de consutlas representa los esquemas de relación de entrada de la consulta como nodos externos, y las operaciones del álgebra relacional como nodos internos del árbol.

Ejemplo 1

Para ilustrar el concepto de árbol inicial, vamos a construir el árbol inicial de consultas del ejemplo del Apartado 2.6.

Consulta: Queremos saber el nombre y el DNI de los empleados que cobran más de 30.000 euros y que trabajan en el departamento de Dirección General o de Comercial.

Para realizar esta consulta introducimos en nuestro gestor de bases de datos el siguiente código SQL:

```
   SELECT Nombre, DNI
FROM Empleado E, Departamento D
WHERE Salario>30.000 AND (Dnombre='Dirección General' OR Dnombre='Comercial') AND
D.Dnum = E.Dnum
```

La consulta en forma de álgebra relacional es:

$$\text{Empleados30} \leftarrow \sigma_{\text{Salario>30.000}}(\text{Empleado})$$

$$\text{DirecGeneral\_o\_comercial} \leftarrow \sigma_{\text{Dnombre = 'Dirección general' OR Dnombre ='Comercial'}}(\text{Departamento})$$

$$\pi_{\text{Nombre, DNI}}((\text{Empleados30}) \bowtie_{\text{D.Dnum = E.Dnum}} (\text{DirecGeneral\_o\_comercial}))$$

Escaneo, análisis y vadilación: Si supera satisfactoriamente estos procesos, se crea el siguiente árbol de consultas inicial basado la consulta del álgebra relacional que acabamos de mostrar:

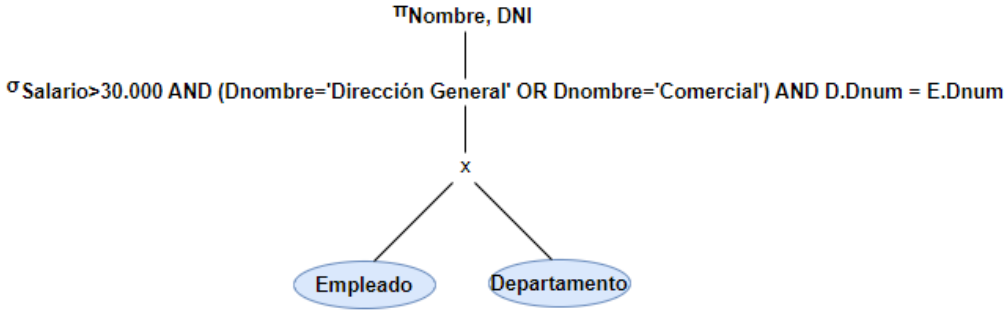

Figura 3.2: Árbol inicial de la consulta del Apartado 2.6

donde los nodos Empleado y Departamento representan esquemas de relación. En cambio, los nodos internos representan las operaciones del álgebra relacional que se ejecutarán sobre estos esquemas de relación.

Este árbol no se ejecuta nunca, sino que se envía al optimizador de consultas para optimizarlo.

Ejemplo 2

A continuación vamos a construir el árbol inicial de consultas del Apartado 2.7.

Consulta: Queremos el nombre completo de los empleados cuya fecha de nacimiento es posterior a 03/09/1950 y que trabajan en el proyectoA.

Para realizar esta consulta introducimos en nuestro gestor de base de datos el siguiente código SQL:

```
   SELECT E.Nombre, E.Apellido1, E.Apellido2
FROM Empleado E, Horario H, Proyecto P
WHERE P.Pnombre='ProyectoA' AND E.DNI=P.EDNI AND P.Pnum=H.Pno AND FechaNac>'03/09/1950'
```

La consulta en álgebra relacional es:

$$\text{Empleado\_Proyecto} \leftarrow (\sigma_{\text{FechaNac>'03/09/1950'}}(\text{Empleado})) \bowtie_{\text{DNI = EDNI}} (\sigma_{\text{Nombre = 'ProyectoA'}}(\text{Proyecto}))$$

$$\pi_{\text{Nombre, Apellido1, Apellido2}}((\text{Empleado\_Proyecto}) \bowtie_{\text{P.Pnum = H.Pro}} (\text{Horario}))$$

Escaneo, análisis y validación: Si supera satisfactoriamente estos tres procesos, se generará el siguiente árbol inicial basado la consulta relacional que acabamos de mostrar:

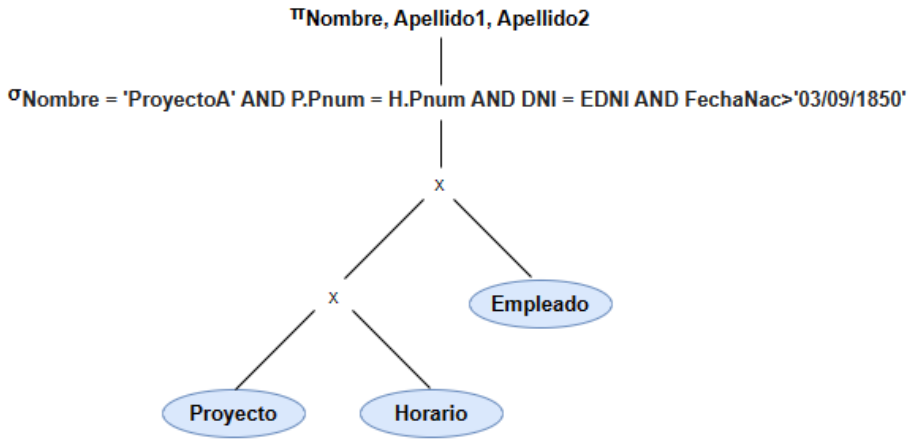

Figura 3.3: Árbol inicial de la consulta

donde los nodos Proyecto, Horario y Empleado representan las relaciones correspondientes a cada uno. En cambio, los nodos del árbol internos representan las operaciones del álgebra relacional.

Este árbol es el que se envía al optimizador de consultas para encontrar un árbol optimizado equivalente al árbol inicial.

3.3. Reescritura de un árbol de consultas

Una ejecución del árbol de consultas consiste en ejecutar una operación de nodo interno y luego reemplazar ese nodo interno por la relación que resulta de la ejecución de la operación. El orden de ejecución de las operaciones comienza en los nodos de hoja, que representan las relaciones de entrada de la base de datos y termina en el nodo raíz, que representa la operación final de la consulta. La ejecución termina cuando se ejecuta la operación del nodo raíz y se produce la relación resultado de la consulta.

Existen varias operaciones que resultan muy ineficientes durante la ejecución de una consulta. Este es el caso de la operación PRODUCT CARTESIAN, que puede llegar a aumentar considerablemente el tamaño de los archivos con los que trabajamos.

Por ejemplo, imaginemos que una empresa tiene 10.000 empleados y 500 proyectos. Si necesitaramos ejecutar la operación PRODUCT CARTESIAN entre estas dos relaciones, obtendríamos una relación de 5.000.000 tuplas, es decir, pasaríamos de trabajar con una relación de 10.000 tuplas a una de 5.000.000. Por ello esta operación puede llegar a ser muy costosa respecto al almacenamiento. En los próximos apartados explicaremos mejoras para ejecutar una consulta sin tener que utilizar esta operación.

Ejemplo 1

La ejecución del árbol de la Figura 3.2 empieza en los últimos nodos hasta llegar al nodo raíz, es decir, aplicado a este ejemplo, primero se ejecuta el PRODUCT CARTESIAN de las relaciones que aparecen en la cláusula FROM de la consulta. Luego se aplica la operación SELECT, la cual elimina todas las

tuplas que no cumplan la condición. Finalmente, se ejecuta la operación PROJECT, que selecciona todos los valores que toman los atributos de su lista.

Sin embargo, este árbol de consultas resulta muy ineficiente si se ejecuta debido a que la operación PRODUCT CARTESIAN puede mutiplicar considerablemente el número de tuplas de la relación.

Por esa razón, este árbol inicial nunca se llegará a ejecutar. El optimizador de consultas heurísticas transformará este árbol de consultas inicial en un árbol de consultas final equivalente, que será más eficiente a la hora de ejecutarse.

Ejemplo 2

En la ejecución de la consulta de la Figura 3.3 primero se aplica el PRODUCT CARTESIAN de las relaciones que aparecen en la cláusula FROM de la consulta. Luego se aplican las condiciones de selección y unión de la cláusula WHERE, seguidas de las condiciones de proyección sobre los atributos de la cláusula SELECT.

Igual que en el ejemplo anterior, este árbol de consultas resulta muy ineficiente si se ejecuta.

3.4. Reglas de transformación para operaciones del álgebra relacional

En esta sección enunciaremos las principales reglas que se utilizan para transformar el árbol inicial de la consulta en un árbol optimizado equivalente al inicial.

Un concepto importante es el de equivalencia de consultas. Dos consultas son equivalentes si su ejecución siempre proporciona la misma relación.

Vamos a relacionar este nuevo concepto con la optimización de consultas. Primero, nosotros introduciremos una consulta en nuestro gestor de bases de datos. Una vez esta consulta ha superado satisfactoriamente los procesos de escaneo, análisis y validación, se genera un árbol inicial de consultas. Pero este árbol inicial se puede mejorar con las reglas que daremos a continuación dando lugar a un árbol de consultas optimizado. Este árbol representará la misma consulta que el árbol inicial, pero más eficiente. Es decir, si ejecutasemos el árbol inicial y el árbol optimizado se obtendría exactamente la misma relación. Por eso decimos que el árbol de consultas inicial es equivalente al árbol de consultas optimizado, se obtiene el mismo resultado pero de forma más óptima.

A continuación, veamos las reglas de transformación para operaciones del álgebra relacional:

1. **Cascada de σ**

   Una condición de selección conjuntiva se puede dividir en una cascada (es decir, una secuencia) de operaciones individuales:

   $$\sigma_{C_1 \text{ AND } C_2 \text{ AND ... AND } C_N}(\mathscr{R}) \equiv \sigma_{C_1}(\sigma_{C_2}(...(\sigma_{C_N}(\mathscr{R}))))$$

   Demostración. *Para esta demostración y las siguentes vamos a utilizar la siguiente notación: Sea $\mathscr{R}(A_1, A_2, ..., A_n)$ un esquema de relación, denotamos $X_i \equiv dom(A_i)$. Luego, $\mathscr{R} \subseteq X_1 \times X_2 \times ... \times X_n$. Para demostrar este punto es suficiente probar el resultado para una cascada de dos operaciones, ya que el enunciado es una generalización de este caso. Sea t una tupla cualquiera, $A = \{t \in \mathscr{R} :$ t cumple C_1 $\}$ y $B = \{t \in \mathscr{R} :$ t cumple C_2 $\}$. Entonces:*

   $$\sigma_{C1}(\sigma_{C2}(\mathscr{R})) = \{t \in B : t \text{ cumple } C_1\} = \{t \in \mathscr{R} : t \text{ cumple } C_1 \text{ y } C_2\} = \sigma_{C_1 \text{ AND } C_2}(\mathscr{R})$$

2. **Conmutatividad de σ**

   La operación σ es conmutativa:

   $$\sigma_{C_1}(\sigma_{C_2}(\mathscr{R})) \equiv \sigma_{C_2}(\sigma_{C_1}(\mathscr{R}))$$

   Demostración. *Con el razonamiento de la demostración anterior:*

   $$\sigma_{C1}(\sigma_{C2}(\mathscr{R})) = \{t \in B : t \text{ cumple } C_1\} = \{t \in A : t \text{ cumple } C_2\} = \sigma_{C2}(\sigma_{C1}(\mathscr{R}))$$

3. **Cascada de π**

   En una secuencia de π, todas menos la última pueden ser ignoradas:

   $$\pi_{Lista_1}(\pi_{Lista_2}(...(\pi_{Lista_N}(\mathscr{R})))) \equiv \pi_{Lista_1}(\mathscr{R})$$

   Suponiendo que $Lista_1 \subset Lista_2 \subset ... \subset Lista_n$.

   Demostración. *Sean I=$\{i_1,...,i_k\}$, J=$\{j_1,...,j_l\}$ conjuntos de subíndices que corresponden a los atributos de $Lista_1$ y $Lista_2$, con $Lista_1 \subset Lista_2$. Denotamos $X_I = X_{i_1} \times ... \times X_{i_k}$ y $X_J = X_{j_1} \times ... \times X_{j_l}$. Entonces:*

   $$\pi_{Lista_1}(\pi_{Lista_2}(\mathscr{R})) = (\mathscr{R}|_{X_J})|_{X_I} = \pi_{Lista_1}(\mathscr{R})$$

4. **Conmutar σ con π**

   Si la condición c solo involucra los atributos $A_1, A_2, ..., A_N$ en la lista de la operación π, las dos operaciones conmutan:

   $$\pi_{A_1,A_2,...,A_N}(\sigma_C(\mathscr{R})) \equiv \sigma_C(\pi_{A_1,A_2,...,A_N}(\mathscr{R}))$$

   Demostración *Sea X_I correspondiente a la lista de atributos $A_1,...,A_n$. La condición C solo afecta a los atributos $A_1,...,A_n$:*

   $$\pi_{A_1,...,A_n}(\sigma_C(\mathscr{R})) = \{(t_1,...,t_n) \in \mathscr{R} : (t_1,...,t_n) \text{ cumple } C\}|_{X_I} =$$

   $$= \{(t_1,...,t_n) \in \mathscr{R}|_{X_I} : (t_1,...,t_n) \text{ cumple } C\} = \sigma_C(\pi_{A_1,...,A_n}(\mathscr{R}))$$

5. **Conmutatividad de $\bowtie$ y de $\times$**

   Las operaciones $\bowtie$ y $\times$ son conmutativas:

   $$\mathscr{R} \bowtie_C \mathscr{S} \equiv \mathscr{S} \bowtie_C \mathscr{R}$$

   $$\mathscr{R} \times_C \mathscr{S} \equiv \mathscr{S} \times_C \mathscr{R}$$

   Demostración *Vamos a demostrar el caso del JOIN. Podemos suponer que las relaciones que intervienen en la relación binaria no tienen atributos en común. Sea $\mathscr{R} \subseteq X_1 \times ... \times X_n$ y $\mathscr{S} \subseteq X_{n+1} \times ... \times X_{n+k}$:*

   $$\mathscr{R} \times \mathscr{S} = \{t = (t_1,...,t_{n+k}) : (t_1,...,t_n) \in \mathscr{R}, (t_{n+1},...,t_{n+k}) \in \mathscr{S}\} =$$

   $$= \{t = (t_1,...,t_{n+k}) : (t_{n+1},...,t_{n+k}) \in \mathscr{S}, (t_1,...,t_n) \in \mathscr{R}\} = \mathscr{S} \times \mathscr{R}$$

6. **Conmutar σ con $\bowtie$ o $\times$**

   Si todos los atributos en la condición C_1 involucran solo los atributos de una de las relaciones:

   $$\sigma_{C_1}(\mathscr{R} \bowtie_C \mathscr{S}) \equiv (\sigma_{C_1}(\mathscr{R})) \bowtie_C \mathscr{S}$$

Por otro lado, si la condición de selección $C_1 \cup C_2$, donde C_1 involucra solo los atributos de $\mathscr{R}$ y C_2 involucra solo los atributos de $\mathscr{S}$:

$$\sigma_{C_1 \cup C_2}(\mathscr{R} \bowtie_C \mathscr{S}) \equiv (\sigma_{C_1}(\mathscr{R})) \bowtie_C (\sigma_{C_2}(\mathscr{S}))$$

Análogo para el producto cartesiano $\times$.

Demostración: *Vamos a demostrar solo el segundo caso enunciado, ya que el primero es un caso particular de este. Sea* $\mathscr{R} \subseteq X_1 \times ... \times X_n$ *y* $\mathscr{S} \subseteq X_{n+1} \times ... \times X_{n+k}$. *Además,* C_1 *y* C_2 *afectan a* $\mathscr{R}$ *y* $\mathscr{S}$ *respectivamente. Entonces:*

$$\sigma_{C_1 \cup C_2}(\mathscr{R} \bowtie_C \mathscr{S}) = \{t \in \mathscr{R} \bowtie \mathscr{S} : \ t \text{ cumple } C_1 \text{ y } C_2, t \text{ cumple } C\} =$$

$$= \{t = (t_1, ..., t_{n+k}) : (t_1, ..., t_n) \in \mathscr{R} \text{ cumple } C_1 (t_{n+1}, ...t_{n+k}) \in \mathscr{S} \text{ cumple } C_2, t \text{ cumple } C\} =$$

$$= \{t = (t_1, ..., t_{n+k}) : (t_1, ..., t_n) \in \sigma_{C_1}(\mathscr{R}), (t_{n+1}, ...t_{n+k}) \in \sigma_{C_2}(\mathscr{S}), t \text{ cumple } C\} = \sigma_{C_1}(\mathscr{R}) \bowtie_C \sigma_{C_2}(\mathscr{S})$$

El caso de la operación PRODUCT CARTESIAN es un caso particular de esta demostración.

7. **Conmutar π con $\bowtie$ o $\times$**

   Supongamos que la lista de proyección es $L = \{A_1, ..., A_n, B_1, ..., B_m\}$ donde $A_1, ..., A_n$ son atributos de $\mathscr{R}$ y $B_1, ..., B_m$ son atributos de $\mathscr{S}$. Si la condicion c solo involucra atributos en L, las dos operaciones se pueden conmutar:

$$\pi_L(\mathscr{R} \bowtie_C \mathscr{S}) \equiv (\pi_{A_1, ..., A_n}(\mathscr{R})) \bowtie_C (\pi_{B_1, ..., B_m}(\mathscr{S}))$$

8. **Asociatividad de $\bowtie$ y $\times$**

$$(\mathscr{R} \circ \mathscr{S}) \circ \mathscr{T} \equiv \mathscr{R} \circ (\mathscr{S} \circ \mathscr{T})$$

   donde $\circ = \{\bowtie, \times\}$.

9. **Convertir una secuencia $(\sigma, \times)$ en $\bowtie$**

$$\sigma_C(\mathscr{R} \times \mathscr{S}) \equiv (\mathscr{R} \bowtie_C \mathscr{S})$$

   Demostración: *Por la definición de JOIN:*

$$\sigma_C(\mathscr{R} \times \mathscr{S}) = \{t \in \mathscr{R} \times \mathscr{S} : \ t \text{ cumple } C\} = \mathscr{R} \bowtie_C \mathscr{S}$$

10. **Transformaciones triviales**

    - Si toda tupla de $\mathscr{R}$ cumple C, entonces $\sigma_C(\mathscr{R}) = \mathscr{R}$.
    - NOT(C_1 AND C_2) $\equiv$ (NOT C_1) OR (NOT C_2).
    - NOT(C_1 OR C_2) $\equiv$ (NOT C_1) AND (NOT C_2).

3.5. Esquema del algoritmo de optimización algebraica heurística

A continuación se describen los pasos que sigue el algoritmo de optimización heurística con el objetivo de transformar un árbol inicial en un árbol final más eficiente de ejecutar.

1. Si en el árbol inicial aparece una operación SELECT con condiciones conjuntivas, se aplica la Regla 1 descrita anteriormente, que transforma la condición conjuntiva en una cascada de operaciones SELECT.

2. Como la operación SELECT reduce el tamaño del archivo resultante, nos interesa que se ejecute cuanto antes para así trabajar con archivos menos pesados. Por ello, las operaciones SELECT es mejor que estén lo más alejadas del nodo raíz. Para conseguirlo, se utilizan las Reglas 2, 4 y 6, referentes a la conmutatividad de la operación SELECT con otras operaciones.

   Si la condición de selección solo implica atributos de una tabla, la operación SELECT se situará en el nodo de hoja más cercano al nodo que representa esa relación.

   Por otro lado, si la condición de selección implica atributos de dos tablas, la operación SELECT se situará después de que se combinen esas dos tablas.

3. Este paso consiste en una reorganización de los nodos de hoja del árbol. Para ello utilizaremos las Reglas 5 y 8 sobre conmutatividad y asociatividad de las operaciones. Además, tendremos en cuenta estos criterios:

   - Se colocan las relaciones del nodo de hoja con las operaciones SELECT más restrictivas (que producen una relación con menor número de tuplas) para que se ejecuten primero en la representación del árbol de consultas.
   - Asegurarse que el orden de los nodos de la hoja no cause operaciones de producto cartesiano. Por ejemplo, si las dos relaciones con el SELECT más restrictivo no tienen una condición de unión directa entre ellas, puede ser deseable cambiar el orden de los nodos de hoja para evitar los productos cartesianos.

4. Tal y como dice la regla 9, la combinación un producto cartesiano con una operación SELECT es equivalente a una operación JOIN.

5. Usando las Reglas 3, 4 y 7 relativas a la cascada de PROJECT y a su conmutatividad con otras operaciones, vamos situar las operaciones PROJECT tan lejos como sea posible del nodo raíz.

En resumen, es mejor aplicar primero las operaciones que reducen el tamaño de los resultados intermedios, es decir, es mejor aplicar las operaciones SELECT y PROJECT antes de aplicar el JOIN u otras operaciones binarias. El tamaño del archivo resultante de una operación binaria, como JOIN, suele ser una función multiplicativa de los tamaños de los archivos de entrada. En cambio, las operaciones SELECT y PROJECT reducen el tamaño de un archivo y, por lo tanto, deben aplicarse antes de un JOIN u otra operación binaria.

Ejemplo 1

Vamos a aplicar el algoritmo de optimización heurístico paso a paso al ejemplo introducido en la sección 3.1. Su árbol inicial era el que aparece en la Figura 3.2. Este ejemplo es un ejemplo sencillo de como aplicar el algoritmo, donde no será necesario aplicar todos los pasos.

Paso 1

Consiste en intercambiar la condición de selección por una cascada de condiciones de selección. Es decir, en el árbol inicial de consultas hay una operación SELECT cuya condición implica 3 subcondiciones. Aplicando la regla 1 de transformación, consiste en cambiar esa operación por tres operaciones SELECT con cada una de las subconsultas. Es decir, cambiamos la operación SELECT inicial por tres operaciones SELECT: una que será D.Dnum = E.Dnum, otra con Salario>30.000, y otra con Dnombre='Dirección General' OR Dnombre = 'Comercial'.

Paso 2

Consiste en acercar todo lo posible esas operaciones SELECT a sus respectivas relaciones para que se apliquen cuanto antes.

Por ejemplo, la operación SELECT con la condición de Salario>30.000 la pondremos justo encima del nodo que representa la relación Empleado, para que así se ejecute cuando antes. Lo mismo con la operación SELECT con la condición Dnombre='Dirección General' OR Dnombre = 'Comercial' y su respectivo nodo Departamento. Sin embargo, no podemos mejorar la situación de la operación SELECT con la condición de D.Dnum = E.Dnum, porque como involucra atributos de dos relaciones no podemos ejecutarla antes de que se fusionen esas relaciones.

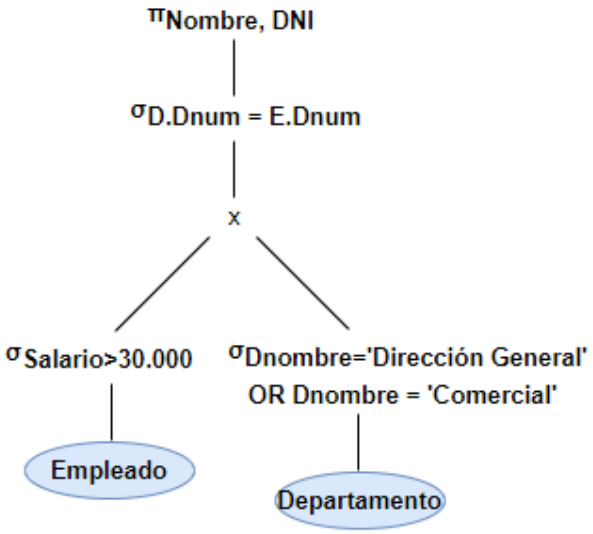

Figura 3.4: Árbol actualizado con el paso 1 y 2

Paso 3

En este caso, no tiene sentido la aplicación del paso 3, debido a que solo estamos trabajando con dos relaciones, por tanto no existe una forma mejor de posicionar los nodos.

Paso 4

Cambiamos la combinación de un producto cartesiano con una operación SELECT por una operación JOIN. El árbol de consultas actualizado con el paso 4 se representa en la Figura 3.5.

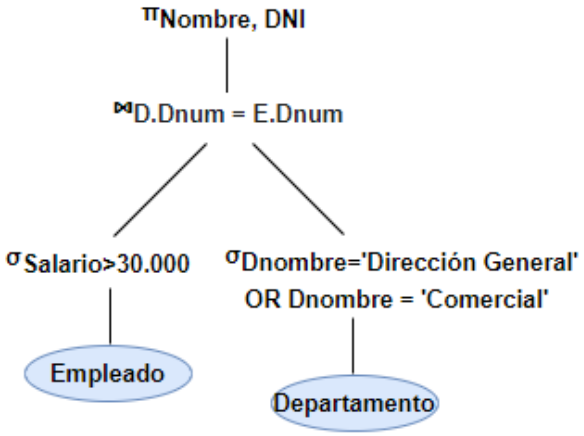

Figura 3.5: Árbol actualizado con el paso 4

Paso 5

Vamos a introducir nuevas operaciones PROJECT para solo trabajar con los atributos necesarios. Para decidir los atributos que son necesarios debemos fijarnos en la lista de atributos de la operación PROJECT y en los atributos que relacionen las operaciones JOIN. En este caso, la lista de atributos de la operación PROJECT está formada por los atributos Nombre y DNI. Además, tenemos una operación JOIN que involucra los atributos Dnum. Esos van a ser los atributos que queremos que permanezcan a lo largo de la ejecución de la consulta. El árbol final optimizado es el siguiente:

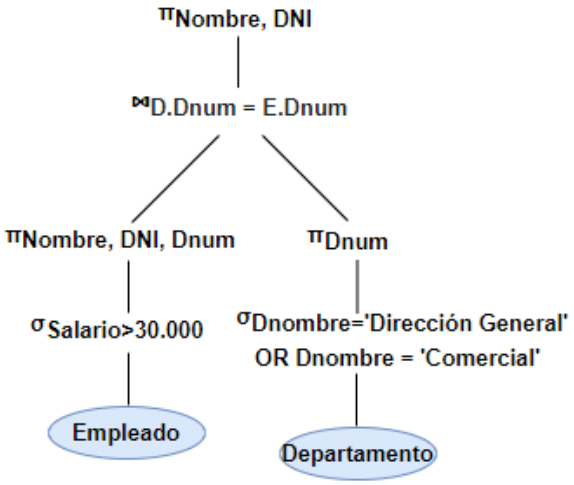

Figura 3.6: Árbol final optimizado

Ejemplo 2

Repetimos los pasos para el árbol inicial de la Figura 3.2. Esta vez es más complejo, ya que consta de más operaciones.

Paso 1

Descomponemos las operaciones SELECT con condiciones de selección complejas en una cascada de operaciones SELECT con condiciones de selección simples. En este caso la operación SELECT consta de 4 subcondiciones: FechaNac>'03/09/1950', Nombre = 'ProeyctoA', DNI=EDNI y P.Pnum = H.Pnum.

Paso 2

Reorganizamos estas operaciones para que las operaciones SELECT se ejecuten cuanto antes, es decir, se colocan lo más alejadas posible del nodo raíz posible.

En el caso de la operación SELECT con la condición FechaNac>'03/09/1950', la situamos justo encima del nodo Empleado, para que se ejecute cuando antes. Lo mismo con la operación SELECT con la condición Nombre = 'ProeyctoA' y su nodo Proyecto. Sin embargo, las otras dos operaciones SELECT involucran atributos de dos esquemas de relación distintos. Por ello, las pondremos justo después de hacer el producto cartesiano entre esas dos relaciones. Por ejemplo, DNI=EDNI involucra un atributo de Empleado y otro de Horario, por tanto vamos a situar esta operación SELECT justo después de hacer el producto cartesiano entre estas dos relaciones. Análogo para P.Pnum = H.Pnum.

Una vez aplicados el paso 1 y 2, el árbol de consultas inicial es:

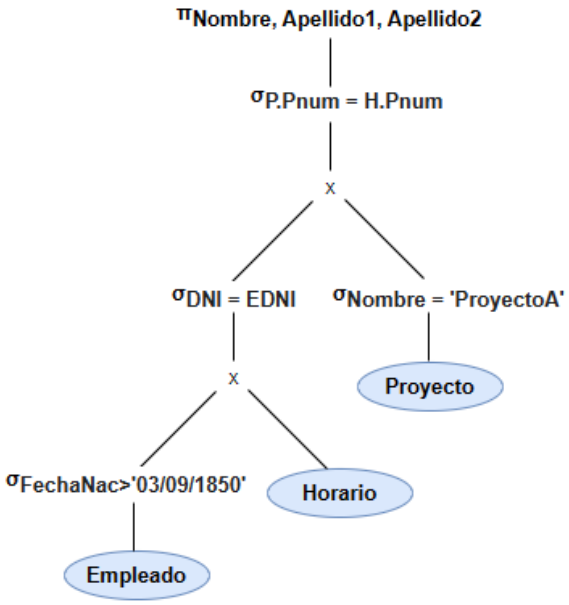

Figura 3.7: Árbol actualizado con los pasos 1 y 2

Paso 3

Reorganizamos los nodos hoja y los nodos internos. Situamos la operación PRODUCT CARTESIAN entre los esquemas de relación Proyecto y Horario, en vez de Empleado y Horario. Esto es debido a que la operación PRODUCT CARTESIAN entre los esquemas de relación Proyecto y Horario genera un número menor de tuplas que la operación PRODUCT CARTESIAN entre Empleado y Horario. Las operaciones SELECT más restrictivas, las hemos situado lo más lejos posible del nodo raíz, ya que así se ejecutarán primero. Aplicando el paso 3, el árbol actualizado es:

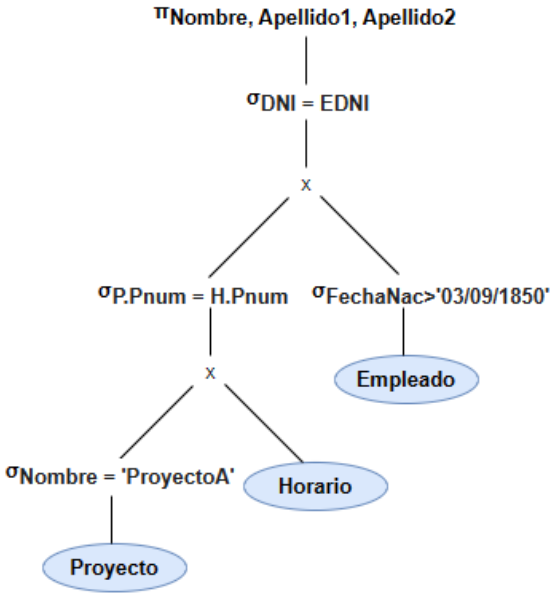

Figura 3.8: Árbol actualizado con el paso 3

Paso 4

Reemplazamos cualquier operación de CARTESIAN PRODUCT que vaya seguida de una operación SELECT por su correspondiente operación JOIN:

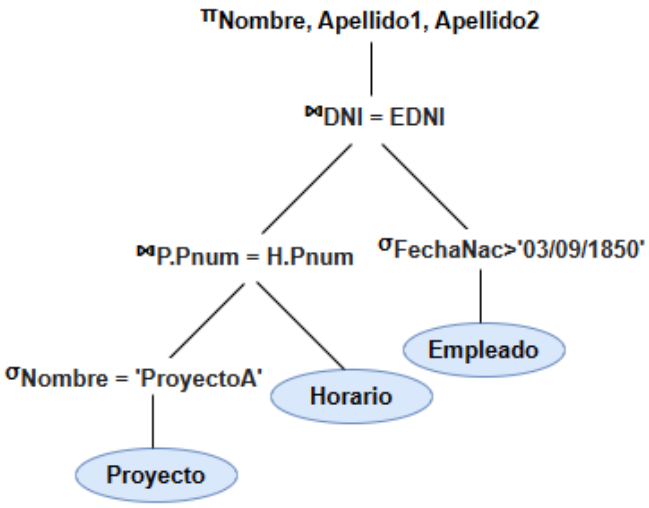

Figura 3.9: Árbol actualizado con el paso 4

Paso 5

Solo mantenemos los atributos necesarios para las operaciones posteriores. Es decir, de del esquema de relación Proyecto solo necesitamos el atributo Pnum, entonces vamos a introducir una operación PROJECT para solo quedarnos con ese atributo justo después de realizar la operación SELECT con la condición Nombre = 'ProyectoA'. Pasa algo parecido para la relación Horario, solo necesitamos los atributos EDNI y PNum, por eso introducimos una operación PROJECT con esos atributos. Para la relación Empleado solo necesitamos el atributo EDNI, pero como la operación PROJECT final es con la lista de atributos Nombre, Apellido1 y Apellido2, debemos mantener esoso también.

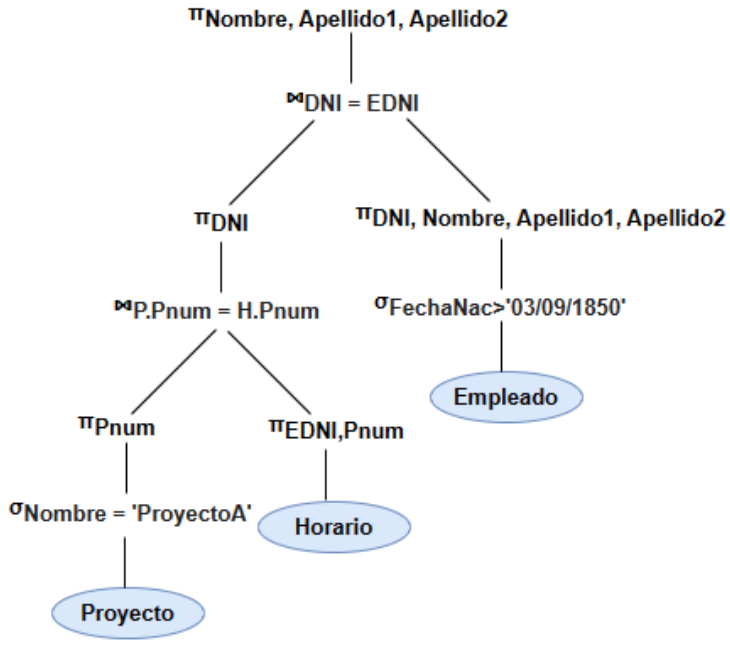

Figura 3.10: Árbol final optimizado

Bibliografía

[1] RAMEZ ELMASRI Y SHAMKANT B. NAVATHE, *Fundamentals of database systems*, Pearson Education, 7ª ed., 2015.

[2] E. F. CODD, *A Relational Model of Data for Large Shared Data Banks*, Commun ACM 13(6), 377-387 (1970).

[3] C. J. DATE, H. DARWEN, *A guide to the SQL standard*, 4ª ed., Addison-Wesley, 1997.