Uso del álgebra relacional para mejorar la velocidad de respuesta a consultas SQL

Iván Perales Boal

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Director del trabajo: Jorge Lloret Gazo
24 de junio de 2022

Resumen

The relational database model allows databases to be represented independently of the machine representation and on a mathematical basis. It was first described by Ted Codd in 1970, and a few years later it was implemented in relational database managers.

A relational database is made up of relation schemas. Each relation schema is composed of a relation name and a number of attributes that represent characteristics of the relation. In a relational database, relation schemas are connected through common attributes. Each attribute has a domain that represents the set of values that it can take, and that can be finite or infinite. On the other hand, we denote a tuple as an ordered list of atomic values of a relation, where each value represents a different attribute. Finally, we define a relation state as a set of tuples of a relation schema, although we will usually refer to it as a relation.

At a conceptual level, the relational model is flexible with respect to the ordering of tuples or the ordering of values within a tuple. However, at the physical level it is often important to follow some order in the tuples, and even for optimizations of relational algebra operations, as will be seen in the work. On the other hand, there is a special value that attributes can take, which is the NULL value. It means that a tuple has an empty value for a certain attribute and it can be for various reasons.

One of the central aspects of the project is to explain the main operations of relational algebra, which is a set of operations that allow modifying a database to retrieve information in the form of queries. Depending on whether we apply the operation to one or two relations, we distinguish between unary and binary operations. Also, the result of applying an operation is always another relation.

The unary operations to be explained are selection, projection, and rename. The selection applied to a relation consists of choosing the tuples that meet one or several conditions. Conditions are made up of attribute values or constants and logical operators. The projection chooses a list of attributes contained in the attributes that make up the relationship schema and creates a new relationship but only with those attributes. Finally, the rename operation names the result of an algebraic expression. This allows you to decompose an expression into several intermediate operations and access them.

The binary operations to be explained are the operations of set theory, the cartesian product and the join. Set-theoretic operations can only be applied to two relations that are join-compatible, that is, that have the same attributes. The three operations covered are union, intersection, and difference, which work exactly the same as in set theory.

The cartesian product creates a new relation with the attributes of the two initial relations. All the tuples of both are combined, which causes relationship states to be created long and make no sense in real life. The solution to this problem is to use the join operation. The join consists of defining a product followed by a selection as a single operation. Due to the feature that in a relational database the relation schemas are connected by common attributes, it is very common to replace the cartesian product with a join in which the union condition is equality between two matching attributes.

The most important section of the paper deals with query optimization. The language used by query managers is SQL. When the manager receives an SQL query, some processes occur until the result appears on the screen. One of them is to transform an SQL query into a relational algebra expression. An SQL query consists of a SELECT-FROM-WHERE block, where there may also be GROUP BY and HAVING clauses.

The process that is most explained in the work is optimization, which consists of choosing the most efficient execution strategy. Before explaining the implementations of operations, we will define some important concepts at the physical level such as record, field and index. Many operation implementations require prior sorting algorithms. One of the most used is merge sort.

To implement the select operation efficiently, it is necessary to apply a sorting algorithm or set up an index structure, to avoid unnecessary condition evaluations. Similarly, a join whose condition is equality has various forms in the implementation. Having the files ordered or having an index structure allows this operation to be applied in the most efficient way. The same goes for projection, product, and set theoretic operations.

The main goal of explaining relational algebra operations is to see how they can be used to optimize queries. One of the steps from when an SQL query is received until the result appears is the creation of an internal representation in the form of a query tree. A query tree represents a specific order in the execution of the operations of a relational algebra expression. It is made up of leaf nodes with the initial relations, and internal nodes that contain the operations. The goal is to replace the generated query tree with a more efficient one that is equivalent, that is, that it represents the same query and therefore has the same result.

There are heuristic rules that are used for this purpose. The types of rules that will be explained are commutativity, associativity, converting relations to others, and creating intermediate relations. In this work a possible demonstration of them using arguments from set theory will be given. Finally, a possible algorithm that uses all the rules to optimize the queries in several steps will be explained. An example will be given to explain how the internal representation changes at each step.

An implementation made with the Python programming language is explained in the annex. It is an application that allows you to load and create relationships, write and execute queries. In addition, it contains some of the programmed heuristic rules that allow you to optimize the query and see how it changes. The objective of the program is mainly educational, because it is not possible to load large databases, but it serves to understand how relational managers work internally.

Índice general

Capítulo 1

Introducción

El modelo relacional de bases de datos fue descrito y presentado por primera vez en junio de 1970, en un artículo titulado *"A Relational Model of Data for Large Shared Data Banks"*, publicado en la revista *"Communications of the ACM"*.

El autor del conocido artículo es el científico informático inglés Edgar Frank Codd (19 de agosto de 1923 - 18 de abril de 2003), más conocido como Ted Codd. Su publicación fue una revolución para la materia debido a la claridad y simplicidad del modelo, además de estar apoyado en las matemáticas. Aunque la compañía en la que trabajaba, IBM, inicialmente no apostó por su implementación, finalmente en 1974 la empresa puso en marcha su primer gestor relacional de bases de datos, el *System R*. El lenguaje de consulta utilizado se denominó SQL (*Structured Query Language*). El éxito del modelo hizo que Codd recibiera el Premio Turing de las Ciencias de Computación en 1981.

La principal diferencia con los modelos previos que menciona Codd en su artículo de modelo relacional es que lo importante es saber qué consultar en una base de datos y no preocuparse de cómo está almacenado. Con una base teórica en la teoría de conjuntos, el modelo proporciona una estructura natural para describir los datos, y hay independencia entre el lenguaje utilizado y la organización de datos y representación de la máquina.

El artículo de Codd está dividido en dos capítulos. En el primero presenta los conceptos importantes del modelo y su implementación en una base de datos. En la segunda parte presenta las operaciones posibles en una base de datos, pertenecientes a la teoría de conjuntos. Esta nueva rama de estudio de las operaciones para computar una respuesta en una base de datos se denomina álgebra relacional. En la siguiente sección se presentará el modelo relacional y las definiciones principales, acompañado de ejemplos.

Capítulo 2

El modelo relacional

El modelo relacional proporciona las herramientas para modelar una base de datos. Para ello, propone una estructura lógica y apoyada en la teoría de conjuntos de las matemáticas. Así, una base de datos es una colección de relaciones representada como una tabla en la que cada fila son una serie de valores relacionados. En la figura 1 se muestra un ejemplo de base de datos relacional de cinco esquemas de relación y sus correspondientes relaciones para ilustrar las definiciones y conceptos del modelo.

Figura 1: Base de datos relacional EMPRESA

ESQUEMA DE RELACIÓN: EMPLEADO

Nombre	Apellido	DNI	FechaNac	Dirección	Sexo	Sueldo	SuperDNI	Dno
José	Pérez	123456789	01-09-1965	Eloy I, 98	H	30000	333445555	5
Alberto	Campos	333445555	08-12-1955	Avda. Ríos	H	40000	888665555	5
Alicia	Jiménez	999887777	12-05-1968	Gran Vía, 38	M	25000	987654321	4
Juana	Sainz	987654321	20-06-1941	Cerquillas, 67	M	43000	888665555	4
Fernando	Ojeda	666884444	15-09-1962	Portillo, s/n	H	38000	333445555	5
Aurora	Oliva	453453453	31-07-1972	Antón, 6	M	25000	333445555	5
Luis	Pajares	987987987	29-03-1969	Enebros, 90	H	25000	987654321	4
Eduardo	Ochoa	888665555	10-11-1937	Las Peñas, 1	H	55000	NULL	1

ESQUEMA DE RELACIÓN: DEPARTAMENTO

NombreDpto	NumeroDpto	DniDirector	FechaIngresoDirector
Investigación	5	333445555	22-05-1988
Administración	4	987654321	01-01-1995
Sede central	1	888665555	19-06-1981

ESQUEMA DE RELACIÓN: PROYECTO

NombreProyecto	NumProyecto	UbicacionProyecto	NumDptoProyecto
ProyectoX	1	Sevilla	5
ProyectoY	2	Valencia	5
ProyectoZ	3	Madrid	5
Computcación	10	Gijón	4
Reorganización	20	Madrid	1
Comunicaciones	30	Gijón	4

ESQUEMA DE RELACIÓN: TRABAJA_EN

DniEmpleado	NumProy	Horas
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

ESQUEMA DE RELACIÓN: SUBORDINADO

DniEmpleado	NombSubordinado	Sexo	FechaNac	Relacion
333445555	Ana	M	05-04-1986	Hija
333445555	Teodoro	H	25-10-1983	Hijo
333445555	Luisa	M	03-05-1958	Esposa
987654321	Alfonso	H	28-02-1942	Esposo
123456789	Miguel	H	04-01-1988	Hijo
123456789	Alicia	M	30-12-1988	Hija
123456789	Elisa	M	05-05-1967	Esposa

Por ejemplo, el esquema de relación EMPLEADO esta formado por atributos que caracterizan los empleados, que son el nombre, apellido, DNI, fecha de nacimiento, dirección, sexo, sueldo, DNI del supervisor y el número del departamento. La primera fila de la relación EMPLEADO indica que se trata de un empleado llamado José Pérez, con DNI 123456789, que nació el 1 de septiembre de 1965, vive en la calle Eloy I 98, es varón, cobra 30000 euros, su supervisor tiene como DNI 333445555 y trabaja en el departamento 5. Cada fila representa a un empleado distinto, aunque haya valores de atributos que se puedan repetir. Puede ocurrir que dos empleados distintos se llamen o se apelliden igual.

2.1. Conceptos del modelo relacional

Definición. El dominio de un atributo es el conjunto de valores que puede tomar un atributo y son válidos dentro de sus restricciones, sin contar el valor NULL, el cual se explicará más adelante. El dominio de un atributo A_i se expresa como $dom(A_i)$.

Ejemplo 1. En el atributo edad el dominio son los números naturales, y en el caso de números de teléfono serían los números naturales de nueve cifras. Así, para el atributo sexo, $dom(Sexo) = \{H, M\}$

Definición. Un atributo de una relación es una característica de ella. Cada atributo tiene un dominio.

Ejemplo 2. Los atributos del esquema de relación EMPLEADO son:

Nombre Apellido DNI FechaNac Dirección Sexo Sueldo SuperDNI Dno

Definición. Definimos tupla como una lista ordenada de valores atómicos de los atributos, es decir, que contienen un único valor para cada atributo. El término atómico indica que no se puede subdividir.

Ejemplo 3. El término atómico en el esquema de relación EMPLEADO significa que cada empleado tiene un solo nombre, dirección, etc. En ese esquema de relación, una tupla es:

José Pérez 123456789 01-09-1965 Eloy I, 98 H 30000 333445555 5

Cada tupla t es una lista de valores de dimensión n, siendo n el número de atributos del esquema de relación, donde $t[A_i] \in dom(A_i), \forall i = 1,\ldots,n$, excepto cuando tiene el valor NULL. En una representación del modelo en forma de tabla, las tuplas corresponden a las filas, y los atributos a las columnas.

Definición. Un esquema de relación de una base de datos tiene la forma $R(A_1,...,A_n)$, donde R es el nombre del esquema de relación y $A_1,...,A_n$, sus atributos. El grado de un esquema de relación es el número de atributos, que se denota con la letra n.

Ejemplo 4. Esquema de relación EMPLEADO junto con sus atributos.

Ejemplo: EMPLEADO(Nombre,Apellido,DNI,FechaNac,Dirección,Sexo,Sueldo,SuperDNI,Dno)

Definición. Un estado de relación $r(R)$ es un conjunto de tuplas, $r = \{t_1,\ldots,t_m\}$. Si denotamos D el producto cartesiano de los dominios, $D := dom(A_1) \times \ldots \times dom(A_n)$, se tiene que $r(R) \subseteq D$.

Ejemplo 5. En la Figura 1, cada tabla representa un estado de relación de sus respectivos esquemas de relación. Abreviadamente, el estado de relación se expresa como relación. Por tanto si conocemos el cardinal de cada dominio, el número total de tuplas posibles en R es:

$$| D | = | dom(A_1) | \ \ldots \ | dom(A_n) |.$$

En la base de datos relacional EMPRESA, ese cardinal es el infinito contable. También es importante decir que los dominios no son fijos y que representan una realidad particular. A medida que cambia el mundo real cambia el estado de relación y los valores de los atributos. Debido a esto una operación importante sobre una base de datos es renombrar o actualizar una tupla o un valor.

Definición: Se define base de datos relacional como un conjunto de esquemas de relación conectados entre sí mediante atributos comunes.

Ejemplo 6. Tanto en el esquema de relación EMPLEADO como en TRABAJA_EN, una fila hace referencia a un empleado, y existen dos atributos que las conectan que son Dni y DniEmpleado.

2.2. Propiedades

Ordenación de tuplas: Matemáticamente, al estar hablando de un conjunto de tuplas, la relación no cambia con la acción de reordenarlas. A efectos de modelo, dos colecciones de tuplas con las mismas tuplas en distinto orden son idénticas. Sin embargo, al plasmar la relación en una tabla o fichero, lo usual es seguir un orden lógico. En las bases de datos en las que figuran nombres de personas, lo más normal es ordenar las tuplas por el orden alfabético del apellido, del nombre o incluso por edad.

Ejemplo 7. Los siguientes dos estados de relación tienen las tuplas en distinto orden, pero respresentan la misma información.

NombreDpto	NumeroDpto	DniDirector	FechaIngresoDirector
Administración	4	987654321	01-01-1995
Sede central	1	888665555	19-06-1981
Investigación	5	333445555	22-05-1988

NombreDpto	NumeroDpto	DniDirector	FechaIngresoDirector
Investigación	5	333445555	22-05-1988
Administración	4	987654321	01-01-1995
Sede central	1	888665555	19-06-1981

Ordenación de los valores de una tupla: Según la definición que se ha dado de tupla y relación, una tupla es una lista ordenada de elementos, por lo que cobra importancia el orden de los elementos. Considerar esta definición es lo más usual, debido a que simplifica la notación.
Sin embargo, si definimos un estado de relación r como un conjunto finito de tuplas, donde cada una asigna a cada atributo un valor, no importaría tanto el orden como la asignación de cada valor al atributo que corresponde. Veamos un ejemplo para clarificarlo.

Ejemplo 8. Utilizando una definición de esquema de relación donde no importe el orden dentro de una tupla, lo importante es saber a que atributo pertenece cada valor de una tupla. Por ejemplo, teniendo que 987654321 corresponde al atributo DNI, esté en el orden que esté. Los siguientes dos estados de relación representan la misma información.

NombreDpto	NumeroDpto	DniDirector	FechaIngresoDirector
Administración	4	987654321	01-01-1995
Sede central	1	888665555	19-06-1981
Investigación	5	333445555	22-05-1988

DniDirector	FechaIngresoDirector	NombreDpto	NumeroDpto
987654321	01-01-1995	Administración	4
888665555	19-06-1981	Sede central	1
333445555	22-05-1988	Investigación	5

El valor NULL: Un valor nulo, denotado NULL, significa valor no conocido, no disponible o no aplicable a una tupla. Por lo general se intenta evitar, debido a que su implementación en las operaciones relacionales que se verán posteriormente es complejo y produce ambigüedad, porque dos tuplas con valor NULL en un atributo no significa que tengan el mismo valor.

Ejemplo 9. En el esquema de relación EMPLEADO hay un valor NULL en el atributo *SuperDNI*. Esto puede ser debido a que no hayan facilitado el DNI del supervisor, que se desconozca, o por el contrario que ese empleado no tiene supervisor en la empresa.

2.3. Notación del modelo

- Esquema de relación R de grado n: $R(A_1,\ldots,A_n)$.
- Las letras R,Q,S denotan nombres de relaciones y r,q,s nombre de estados de relaciones.
- Atributos en una relación: $A_i,\forall i=1,\ldots,n$. También R.A, para indicar que es un atributo especifico para R.
- Tuplas: Letras t,u,v. Se accede al valor del i-ésimo atributo de una tupla t con las notación $t[A_i]$. Además $t[A_k,\ldots,A_m]$, con $1\leq k\leq m\leq n$ es una subtupla.

Capítulo 3

Álgebra relacional

El álgebra relacional es el conjunto de operaciones básicas en un modelo relacional de bases de datos. Su función es permitir al usuario manipular la bases de datos y especificar perfectamente una consulta de recuperación de información. El resultado de una consulta es una nueva relación formada al aplicar a la relación inicial una manipulación algebraica, a la que a su vez se le puede aplicar más operaciones. Una expresión de álgebra relacional es una secuencia de operaciones algebraicas cuyo resultado es también una relación. Las operaciones del álgebra relacional se dividen en dos grupos. Dependiendo en si actúan sobre una sola relación o sobre dos, se distinguen las operaciones unarias y binarias. Dentro de las operaciones unarias se distinguen la selección, la proyección y la operación renombrar. Dentro de las binarias, las más importantes son el producto cartesiano y el join.

3.1. Operaciones unarias

3.1.1. Selección

La operación SELECCIÓN (σ) se usa para escoger un subconjunto de tuplas que cumplan una condición. Considerando una relación en forma de tabla, SELECCIÓN es una partición horizontal entre las filas que cumplen la condición y las que no, por lo que el resultado posee el mismo número de atributos que la original. La notación de esta operación es la siguiente:

$$\sigma_{<condición\,de\,selección>}(\mathrm{R})$$

Donde R denota el esquema de relación y $<condición\,de\,selección>$ es una condición sobre los atributos de R formada por cláusulas de la siguiente forma:

$<nombre\,del\,atributo><operador\,de\,comparación><valor\,constante>$ ó
$<nombre\,del\,atributo><operador\,de\,comparación><nombre\,de\,atributo>$

Ejemplo 10. Un ejemplo de cada tipo sería:

$$Horas \leq 30$$

$$Dni = DniEmpleado$$

En la expresión anterior el <*operador de comparación*> en la mayoría de los casos esta entre los siguientes: {=, $>$, $<$, $\leq$, $\geq$, $\neq$ }. Notar que los operadores lógicos de comparación solo pueden utilizarse en dominios de valores ordenados, como números o cadenas de carácteres con una regla de ordenación, como por ejemplo el orden alfabético. En otro caso, el dominio se denomina desordenado, y solo podrán utilizarse $=, \neq$. El valor constante es un valor que se encuentra en el dominio del atributo al que se compara. La condicción de selección se aplica a cada tupla de la relación, y puede evaluarse TRUE si es cierta, FALSE si no lo es, y UNKNOWN si la condición se aplica en algún valor NULL de esa tupla.

Únicamente se seleccionan las filas que se evalúan como TRUE. Además, las clausulas también pueden estar relacionadas entre ellas mediante operadores lógicos como AND, OR ó NOT, cuyo significado es:

- (condición1 AND condición2): Se evalúa TRUE solamente si las dos condiciones son TRUE.
- (condición1 OR condición2): Se evalúa TRUE si alguna de las dos condiciones es TRUE, o las dos.
- (NOT condición1): Se evalúa TRUE si la condición se evalúa como FALSE.

Notar que el número de tuplas de la relación resultante es siempre menor o igual que de la que parte, es decir, el número de tuplas de la relación R.

$$| \sigma_c(R) | \leq | R |$$

. Además, es una operación conmutativa, luego combinar varias operaciones SELECCIÓN seguidas es equivalente a utilizar AND:

$$\sigma_{condición1}(\sigma_{condición2}(\ldots(\sigma_{condiciónK}(R))\ldots)) = \sigma_{condición1}\,AND_{condición2}\,AND\ldots AND_{condiciónK}(R)$$

Ejemplo 11. Para ilustrar las operaciones usaremos ejemplos de consultas con la base de datos relacional EMPRESA, de la figura 1. Supongamos que se quiere seleccionar los empleados que trabajan en el despacho 4 y cobran más de 25000 euros o en el departamento 5 y cobran más de 30000 euros. La operación es la siguiente:

Consulta 1: $\sigma_{(Dno=4\ AND\ Sueldo>25000)OR(Dno=5\ AND\ Sueldo>30000)}(EMPLEADO)$
El resultado de esta consulta es:

Nombre	Apellido	DNI	FechaNac	Dirección	Sexo	Sueldo	SuperDNI	Dno
Alberto	Campos	333445555	08-12-1955	Avda. Ríos	H	40000	888665555	5
Juana	Sainz	987654321	20-06-1941	Cerquillas 67	M	43000	888665555	4
Fernando	Ojeda	666884444	15-09-1962	Portillo, s/n	H	38000	333445555	5

Ejemplo 12. Empleados que trabajan en los proyectos 1 ó 10 de la relación TRABAJA_EN

Consulta 2: $\sigma_{(NumProy=1)OR(NumProy=10)}(TRABAJA\_EN)$
El resultado de esta consulta es:

DniEmpleado	NumProy	Horas
123456789	1	32.5
453453453	1	20.0
333445555	10	10.0
999887777	10	10.0
987987987	10	35.0

3.1.2. Proyección

La operación PROYECCIÓN sirve para escoger un conjunto de atributos de un esquema de relación y obtener subtuplas que correspondan a esos atributos. Observando la relación como una tabla, al contrario que SELECCIÓN, que elige un conjunto de filas y descarta la demás, PROYECCIÓN escoge una serie de columnas y descarta el resto. Por lo tanto puede considerarse una partición vertical en dos. La notación utilizada es la siguiente:

$$\pi_{<lista\,de\,atributos>}(R)$$

La letra π denota el nombre de la operación PROYECCIÓN. El resultado es otra relación cuyos únicos atributos son los de *<lista de atributos>* en el orden en el que se escriben. Por tanto, el grado de la relación resultante es el número elementos de *<lista de atributos>*.

Ejemplo 13. PROYECCIÓN del esquema de relación EMPLEADO sobre los atributos: Nombre, Apellido y Sueldo.

Consulta 3: $\pi_{Nombre,Apellido,Sueldo}(EMPLEADO)$

Nombre	Apellido	Sueldo
José	Pérez	30000
Alberto	Campos	40000
Alicia	Jiménez	25000
Juana	Sainz	43000
Fernando	Ojeda	38000
Aurora	Oliva	25000
Luis	Pajares	25000
Eduardo	Ochoa	55000

Existe la posibilidad de que en el resultado pudiera haber tuplas duplicadas. Sin embargo, por convenio la operación las elimina para que la nueva relación sea válida. Este hecho se denomina eliminación de duplicados. En las implementaciones de bases de datos existe la opción de no eliminar los registros duplicados.

Ejemplo 14. PROYECCIÓN de EMPLEADO sobre Sexo y Sueldo.

Consulta 4: $\pi_{Sexo,Sueldo}(EMPLEADO)$

Sexo	Sueldo
H	30000
H	40000
M	25000
M	43000
H	38000
H	25000
H	55000

Se puede observar que la relación pasa de tener 8 tuplas a 7, debido a la eliminación de duplicados.

3.1.3. Renombrar

En una expresión de álgebra relacional con varias operaciones, puede ser que al usuario le interese dar nombres a las relaciones intermedias que se van creando. Una forma de dar un nuevo nombre es con la siguiente notación:

$$\text{NUEVO\_NOMBRE} \longleftarrow R_1$$
$$\text{NUEVO\_NOMBRE(nuevo\_atributo1, nuevo\_atributo2)} \longleftarrow \pi_{atributo1,atributo2}(R)$$

El NUEVO_NOMBRE se le asigna a una relación que puede ser un resultado de una operación. Para renombrar los atributos, basta con escribir entre paréntesis los nuevos nombres, en el orden en el que aparecen en la original.

Otra notación de RENOMBRAR como operador unario es:

$$\rho_S(B_1,\ldots,B_n)$$

La letra ρ denota el nombre de la operación, S el nuevo nombre de la relación y $B_1,\ldots,B_n$ los nuevos nombres de los atributos.

Ejemplo 15. Queremos recuperar el nombre, apellido y sueldo de los empleados de los empleados que trabajan en el departamento 5. El problema se puede resolver con y sin relaciones intermedias.

Consulta 5: (Sin relaciones intermedias) $\pi_{Nombre,Apellido,Sueldo}(\sigma_{Dno=5}(EMPLEADO))$

Nombre	Apellido	Sueldo
José	Pérez	30000
Alberto	Campos	40000
Fernando	Ojeda	38000
Aurora	Oliva	25000

Consulta 6: (Con relaciones intermedias)
TEMP $\longleftarrow \sigma_{Dno=5}(EMPLEADO)$
R(NuevoNombre, NuevoApellido, NuevoSueldo) $\longleftarrow \pi_{Nombre,Apellido,Sueldo}(TEMP)$

Nombre	Apellido	DNI	FechaNac	Dirección	Sexo	Sueldo	SuperDNI	Dno
José	Pérez	123456789	01-09-1965	Eloy I, 98	H	30000	333445555	5
Alberto	Campos	333445555	08-12-1955	Avda. Ríos	H	40000	888665555	5
Fernando	Ojeda	666884444	15-09-1962	Portillo, s/n	H	38000	333445555	5
Aurora	Oliva	453453453	31-07-1972	Antón, 6	M	25000	333445555	5

NuevoNombre	NuevoApellido	NuevoSueldo
José	Pérez	30000
Alberto	Campos	40000
Fernando	Ojeda	38000
Aurora	Oliva	25000

En la consulta 6, hemos denotado TEMP al resultado de la SELECCIÓN, y posteriormente le aplicamos la PROYECCIÓN para obtener el mismo resultado en los dos casos.

3.2. Operaciones binarias

En esta sección se incluyen las operaciones en las que intervienen dos relaciones. En particular, las operaciones de la teoría de conjuntos, el PRODUCTO CARTESIANO y el JOIN.

3.2.1. Unión, intersección y diferencia de conjuntos

Una de las visiones posibles de una relación es como conjunto de tuplas. Por lo tanto, parece lógico pensar en aplicar operaciones de la teoría de conjuntos. Las operaciones de conjuntos que incluimos son unión, intersección y diferencia. Para poder aplicar estas operaciones es necesario que las dos sean de unión compatible.

Definición. Dos relaciones $R(A_1,\ldots,A_n)$ y $S(B_1,\ldots,B_n)$ son de unión compatible si tienen el mismo grado y además $dom(A_i) = dom(B_i), i = 1,\ldots,n$.

La operación binaria UNIÓN entre dos conjuntos de tuplas selecciona las que aparecen en alguna de ellas o en las dos a la vez, pero eliminando las repetidas para que cada tupla aparezca una sola vez. En el caso de INTERSECCIÓN selecciona las tuplas que aparecen en ambas relaciones, escribiéndolas solo una vez. Estas dos operaciones cumplen que son conmutativas y asociativas. Por último, la operación no conmutativa DIFERENCIA DE CONJUNTOS o MINUS incluye las tuplas que están en el primer conjunto pero no en el segundo. La notación de estas operaciones es:

Unión: $R \cup S$ Intersección: $R \cap S$ Diferencia: $R - S$

Por convenio, los nombres de los atributos de la relación resultante son los de R.

Ejemplo 16. Sean R y S las relaciones que contienen los DNI de los empleados que trabajan en el departamento 5 y de los supervisores de esos empleados, respectivamente. Podemos calcular la UNIÓN, INTERSECCIÓN Y DIFERENCIA de estas relaciones.

Consulta 7: T $\longleftarrow \sigma_{Dno=5}(EMPLEADO)$
R $\longleftarrow \pi_{Dni}(T)$
S(Dni) $\longleftarrow \pi_{SuperDni}(T)$

R

DNI
123456789
333445555
666884444
453453453

S

DNI
333445555
888665555

Consulta 8: $R \cup S$

DNI
123456789
333445555
666884444
453453453
888665555

Consulta 9: $R \cap S$

DNI
333445555

Consulta 10: $R - S$

DNI
123456789
666884444
453453453

3.2.2. Producto cartesiano

El PRODUCTO CARTESIANO es una operación binaria entre dos conjuntos que no tienen unión compatible necesariamente. También se denota como PRODUCTO CRUZADO o CONCATENACIÓN CRUZADA. El resultado al aplicarla sobre dos relaciones es una nueva relación cuyo grado es la suma de grados de ambas, en la que se combina cada tupla de la primera con todos los de la segunda. Siguiendo con la notación de antes, se escribe:

$$R(A_1,\ldots,A_n) \times S(B_1,\ldots,B_m) = Q(A_1,\ldots,A_n,B_1,\ldots,B_m)$$

Si denotamos el número de tuplas de R y S como n_R y n_S, respectivamente, entonces $n_Q = n_R * n_S$. Debido a que produce múltiples combinaciones con todas las tuplas de R y S, su aplicación suele ir seguida de una SELECCIÓN para obviar las combinaciones que no tengan sentido.

Ejemplo 17. Recupera una lista de los nombres y apellidos de las empleadas femeninas que tienen subordinados junto con el nombre de subordinado.

Consulta 11: EMP_FEM $\longleftarrow \sigma_{Sexo=M}(EMPLEADO)$
NOMB_EMP $\longleftarrow \pi_{Nombre,Apellido,Dni}(EMP\_FEM)$
EMP_SUB $\longleftarrow$ NOMB_EMP $\times$ SUBORDINADO

Nombre	Apellido	DNI	DniEmpleado	NombreSubordinado	Sexo	FechaNac	Relación
...	...	...	...	...	...	...	...
Alicia	Jiménez	999887777	123456789	Alicia	M	30-12-1988	Hija
Alicia	Jiménez	999887777	123456789	Elisa	M	05-05-1967	Esposa
Juana	Sainz	987654321	333445555	Ana	M	05-04-1986	Hija
Juana	Sainz	987654321	333445555	Teodoro	H	25-10-1983	Hijo
Juana	Sainz	987654321	333445555	Luisa	M	03-05-1958	Esposa
Juana	Sainz	987654321	987654321	Alfonso	H	28-02-1942	Esposo
Juana	Sainz	987654321	123456789	Miguel	H	04-01-1988	Hijo
Juana	Sainz	987654321	123456789	Alicia	M	30-12-1988	Hija
Juana	Sainz	987654321	123456789	Elisa	M	05-05-1967	Esposa
Aurora	Oliva	453453453	333445555	Ana	M	05-04-1986	Hija
Aurora	Oliva	453453453	333445555	Teodoro	H	25-10-1983	Hijo
...	...	...	...	...	...	...	...

Las columnas Dni y DniEmpleado nos dan los DNI de las empleadas femeninas y los DNI de los empleados que tienen subordinados. Por lo tanto, como únicamente queremos las tuplas en las que coincida ese DNI, tenemos una serie de tuplas que no tienen sentido real.

REALES $\longleftarrow \sigma_{Dni=DniEmpleado}$(EMP_SUB)

Nombre	Apellido	DNI	DniEmpleado	NombreSubordinado	Sexo	FechaNac	Relación
Juana	Sainz	987654321	987654321	Alfonso	H	28-02-1942	Esposo

La solución de las consulta del ejemplo queda así:
RESULTADO $\longleftarrow \pi_{Nombre,Apellido,NombreSubordinado}(REALES)$

Nombre	Apellido	NombreSubordinado
Juana	Sainz	Alfonso

Debido a que utilizando esta operación se crean tuplas que no corresponden a la realidad, existe una operación que permite hacer un producto cruzado y una SELECCIÓN en un solo paso, como veremos a continuación.

3.2.3. Join

La operación binaria JOIN, o concatenación, combina tuplas de distintas relaciones en una tabla de forma filtrada. Dadas $R(A_1,...,A_n)$ y $S(B_1,...,B_m)$ dos relaciones, la notación que sigue es la siguiente:

$$R \bowtie_{condición\,de\,conexión} S$$

Al igual que en el producto cartesiano, el resultado de JOIN es una nueva relación con $n+m$ atributos. Además cada tupla de la nueva relación es una combinación de las tuplas de R y S que cumplen la condición de conexión. Por lo tanto la operación JOIN no es más que un PRODUCTO CARTESIANO seguido de una SELECCIÓN.

Las condiciones de conexión se relacionan con los operadores lógicos vistos en la SELECCIÓN, y cada condición es de la forma $A_i\,\theta\,B_j$, siendo $1 \leq i \leq n$, $1 \leq j \leq m$ y siendo $\theta \in \{=, \neq, <, >, \leq, \geq\}$. Tal como en la operación SELECCIÓN, puede haber más de una condición separada por AND, OR y NOT.

Como se ha explicado en la sección anterior, una condición de selección muy utilizada es la igualdad entre dos atributos que representan lo mismo, lo que permite relacionar tablas con atributos comunes. Veamos un ejemplo.

Ejemplo 18. Respuesta al ejemplo 10 usando la operación JOIN. Para ello, en la solución del ejercicio 2.8, es equivalente realizar este paso intermedio de estas dos formas:

Consulta 12: EMP_SUB $\longleftarrow$ NOMB_EMP $\times$ SUBORDINADO
REALES $\longleftarrow$ $\sigma_{Dni=DniEmpleado}$(EMP_SUB)

Consulta 13: REALES $\longleftarrow$ NOMB_EMP $\bowtie_{Dni=DniEmpleado}$ SUBORDINADO
Notar que el atributo DNI de EMPLEADO y el atributo DniEmpleado de SUBORDINADO representan lo mismo, y por lo tanto deben coincidir. Con este paso intermedio evitamos crear una relación intermedia que no tiene sentido, obteniendo el mismo resultado.

Ejemplo 19. Se quiere recuperar el nombre y apellido de los trabajadores que han trabajado en el Proyecto X y que cobran más de 25000. Tenemos condiciones de SELECCIÓN que afectan a los esquemas de relación EMPLEADO y PROYECTO.

Se pueden relacionar estos dos esquemas mediante el esquema TRABAJA_EN. Para que la relación tenga sentido real, deben coincidir los atributos DNI y DniEmpleado de EMPLEADO Y TRABAJA_EN, respectivamente. De la misma forma con los atributos NumProy y NumProyecto pertenecientes a TRABAJA_EN y PROYECTO, respectivamente.

Por lo tanto, una posible solución de la consulta en dos pasos es:

Consulta 14:

$$R \longleftarrow EMPLEADO \bowtie_{Dni=DniEmpleado} (TRABAJA\_EN \bowtie_{NumProy=NumProyecto} PROYECTO)$$

$$RESULTADO \longleftarrow \pi_{Nombre,Apellido}(\sigma_{Sueldo>25000\ AND\ NombreProyecto='ProyectoX'}(R))$$

Capítulo 4

Algoritmos para procesamiento y optimización de consultas

Como se ha explicado en la introducción, el primer gestor de bases de datos relacionales de IBM utilizaba el SQL como lenguaje para representar las consultas. Actualmente, el SQL sigue siendo el lenguaje a través del cual se manejan las bases de datos relacionales, y también ha sido implementado por otras grandes empresas como Oracle.

Desde que se escribe una consulta en un lenguaje de alto nivel como SQL hasta que obtenemos el resultado de dicha consulta, son necesarios una serie de pasos para procesarla. Uno de ellos corresponde a la optimización de consultas. Optimizar una consulta consiste en escoger la estrategia de ejecución mas eficiente posible, es decir, que el tiempo de respuesta al ejecutar una consulta sea el menor posible. Por ejemplo, en caso de una SELECCIÓN, interesa que se realicen el menor número de evaluaciones de la condición posibles, evitando realizarlas en el mayor número de tuplas que no se seleccionen.

4.1. Procesamiento de consultas SQL

El procesamiento de una consulta SQL sigue una secuencia de procesos desde que se recibe hasta que se ejecuta, entre los que se encuentra la optimización de dicha consulta.

- En primer lugar, se realiza un análisis léxico, que identifica nombres de relaciones, atributos y palabras reservadas.

- A partir del análisis léxico, se produce un análisis sintáctico, el cual se comprueba que los nombres son válidos y tienen un significado dentro del esquema.

- Posteriormente, se transforma la consulta al lenguaje de consultas del álgebra relacional, y se crea una representación interna en forma de árbol de consultas, de lo cual se hablará en una sección más adelante.

- Debido a que una misma consulta puede tener muchas estrategias de ejecución, se realiza una optimización que consiste en escoger el plan de ejecución más adecuado.
  En particular, en este trabajo se tratarán las formas óptimas de aplicar cada operación del álgebra relacional individualmente, y el uso de reglas heurísticas para optimizar la consulta globalmente.

- En el último paso, se genera un código de ejecución y el procesador de la base de datos en tiempo de ejecución produce el resultado de la consulta, o bien un mensaje de error en caso de haberlo.

4.2. Traducción de consultas SQL al álgebra relacional

Desde la perspectiva del SQL, se usa el término de tabla en vez de relación, y nos referimos a filas y columnas en lugar de tuplas y atributos. Veamos como se traduce una consulta SQL al álgebra relacional sin entrar en muchos detalles de la sintaxis del SQL, ya que no es el objetivo del trabajo. Las consultas SQL se están formadas por bloques SELECT-FROM-WHERE, pudiendo tener cláusulas llamadas GROUP BY y HAVING. Además, una consulta puede estar configurada por varios bloques de consultas anidados, pero a efectos de optimización, podemos considerarlos bloques independientes.

- SELECT: Se escriben los nombres de las columnas que se quieren recuperar. Al escribir '*' seleccionamos todas las columnas. Esta sentencia es lo equivalente a la PROYECCIÓN, y siempre aparece en una consulta SQL.

- FROM: En esta clausula obligatoria se escribe el nombre de las tablas de la que queremos extraer la información. También puede ser la respuesta a otra consulta, con lo que tendríamos una consulta anidada, o un PRODUCTO o JOIN entre dos tablas. En el último caso, también se especifica la condición de unión después de la clausula ON.

- WHERE: Establece las condiciones que deben cumplir las filas para seleccionarse, y es una cláusula opcional. Es el equivalente a una SELECCIÓN.

- GROUP BY y HAVING: No tienen transformación directa con las operaciones que se han visto del álgebra relacional. Debido a que por defecto en lenguaje SQL no hay eliminación de duplicados en el resultado de una consulta, la cláusula GROUP BY sirve para ello, especificando por qué columnas queremos agrupar. La cláusula HAVING impone una condición para la selección de grupos. La diferencia que tiene con el WHERE es que las condiciones afectan a un conjunto de filas agrupadas por atributos en común, y no a cada fila por separado.

Se transforma cada bloque a una expresión de álgebra relacional, y entonces el optimizador de consultas escoge el plan de ejecución más adecuado en cada caso.

Ejemplo 20. Suponer que se quiere recuperar el nombre de los empleados que trabajan en cada proyecto junto con las horas que trabajan en dicho proyecto. Veamos como conseguir este resultado mediante una consulta SQL y su consulta equivalente en álgebra relacional.

Consulta 15 (SQL):
SELECT Nombre, NumProy, Horas
FROM EMPLEADO JOIN TRABAJA_EN ON (DNI=DniEmpleado)
WHERE Sueldo=25000

Consulta 16: Su consulta equivalente de álgebra relacional es:

$$\pi_{Nombre,NumProy,Horas}(\sigma_{Sueldo=25000}(EMPLEADO \bowtie_{DNI=DniEmpleado} TRABAJA\_EN))$$

Ejemplo 21. Veamos la consulta que usaremos más adelante con los árboles de consultas. Suponer que se quiere recuperar el nombre y apellido de los empleados que trabajan en el proyecto 'ProyectoX' y que cobran 25000 euros. Veamos su consulta SQL y su transformación.

Consulta 17 (SQL):
SELECT Nombre, Apellido
FROM EMPLEADO JOIN TRABAJA_EN ON (DNI=DniEmpleado)
JOIN PROYECTO ON (NumProy=NumProyecto)
WHERE Sueldo=25000 AND NombreProyecto='ProyectoX'

Consulta 18: Su consulta equivalente de álgebra relacional es:

$$\pi_{Nombre,Apellido}(\sigma_{Sueldo>25000\ AND\ NombreProyecto='ProyectoX'}(EMPLEADO \bowtie_{Dni=DniEmpleado} (TRABAJA\_EN \bowtie_{NumProy=NumProyecto} PROYECTO)))$$

4.3. Algoritmos para las operaciones del álgebra relacional

A la hora de escoger el plan de ejecución que mejore el tiempo de respuesta a una consulta SQL, se puede optimizar tanto las consulta en su totalidad, como cada operación por separado. En esta sección se verán las distintas formas que tiene un gestor de bases de datos relacional de implementar las operaciones, y cuales son más eficientes. Veamos una serie de conceptos relacionados con el nivel físico de las bases de datos relacionales. Las bases a nivel físico se organizan en ficheros donde los datos se almacenan en forma de registros:

Definición. Los campos son los tipos de datos que almacena una base de datos física. En muchas ocasiones se corresponden directamente con los atributos de las relaciones dentro de la base de datos relacional. Pueden tener una longitud fija, como el atributo DNI o los atributos fecha, o longitud variable, como atributos que hagan referencia a nombres.

Definición. Un fichero es una secuencia de registros. Dentro de un fichero los registros no tiene por qué ser del mismo tipo, aunque lo usual es que sí lo sean.

Definición. Un registro es una colección de valores de datos relacionados. Están formados por campos. Se almacenan en la memoria del ordenador de forma que se pueda acceder a ellos cuando sea necesario. El tipo de un registro lo determina la disposición de los campos y la longitud, que puede ser fija o variable dependiendo de los campos que contenga.

Una vez visto lo que son los ficheros y registros, veamos el concepto de índice en una base de datos.

Definición. Una clave principal es un atributo o un grupo de atributos para los cuales no existen dos tuplas con el mismo valor y que no toma valor NULL en ninguno de los atributos que lo componen.

Definición. Un índice en una base de datos es una estructura auxiliar que proporciona una ruta de acceso a los registros. En una base de datos con varios campos o atributos, se escoge normalmente uno de ellos y se define una estructura de índice, lo que se conoce como campo de indexación. Si el ese atributo es una clave principal, se tiene un índice principal.

Ejemplo 22. Una clave principal en una base de datos sobre personas sería el DNI, que no puede repetirse en distintas tuplas. Un ejemplo de campo de indexación habitual en bases de datos que hacen referencia a personas es el conjunto de campos Nombre y Apellidos, ya que es muy poco común que los tres valores coincidan y permite ordenar los registros.

4.3.1. Algoritmos de ordenación

Un paso previo a hablar de los algoritmos de optimización de operaciones es hablar de los algoritmos de ordenación. La ordenación es uno de los algoritmos más importantes a la hora de procesar una consulta. Aplicar una ordenación en ciertos atributos previamente permite en muchas ocasiones ejecutar una operación del álgebra relacional de una forma mas eficiente. Es evitable si se dispone de un índice adecuado que permite acceder con facilidad a los valores de un atributo, pero no siempre es posible.

Los ficheros de bases de datos de tamaño reducido, que pueden almacenarse por completo en la memoria principal del ordenador, se ordenan siguiendo los algoritmos de **ordenación interna**. Sin embargo, la mayoría de ficheros de bases de datos son de un tamaño muy grande y no es posible almacenarlos en la memoria principal del ordenador. Por lo tanto lo más habitual es recurrir a los llamados algoritmos de **ordenación externa**.

Dentro de ese grupo, el utilizado con más frecuencia es el algoritmo de **ordenación-mezcla**. Consiste en la división del fichero en subficheros llamados **porciones** del fichero principal, y su ordenación interna. Posteriormente se mezclan con otras porciones formando subficheros ordenados del fichero mayor mediante el siguiente proceso. Vamos a suponer que los registros solo tienen valores de un campo para simplificar:

- Se parte de un número n de porciones ordenadas internamente y se reserva espacio para crear un subfichero rellenarlo con los valores ordenados de las porciones.
- Se compara el primer elemento de todas las porciones y se rellena el subfichero con el primer valor en el orden establecido, perteneciente a una de las porciones.
- Se comparan el segundo valor de esa porción y los n-1 primeros valores de las porciones restantes y se rellena el subfichero con el siguiente valor.
- Se repite hasta completar el subfichero, teniendo en cuenta de que cuando una porción ha aportado todos sus valores, ya no se tiene en cuenta.

Hay muchas variaciones dentro de los algoritmos de ordenación mezcla, y dependen de factores como la capacidad del ordenador, el tamaño de las porciones y el numero de porciones que se toman para mezclar. Este tipo de algoritmos están implementados dentro de los gestores de bases de datos, sobre todo en aquellos que manejan ficheros de gran tamaño.

4.3.2. Implementación de la operación SELECCIÓN

Para explicar esta implementación vamos a suponer que una base de datos está almacenada en un fichero. Según las condiciones de selección la operación SELECT se puede ejecutar de diversas maneras. **Métodos de búsqueda en una selección simple.** Los siguientes algoritmos de búsqueda se conocen como exploraciones del fichero, o exploración indexada en caso de que requiera la utilización de índices:

- **Búsqueda lineal.** Comprobar registro a registro que se cumple la condición de selección en sus atributos. Es el equivalente a la fuerza bruta.
- **Búsqueda binaria.** Se utiliza en casos donde la condición de selección es una igualdad sobre una clave principal. Primero se aplica un algoritmo de ordenación-mezcla sobre la clave principal. La exploración consiste en evaluar la condición de selección en la tupla que ocupa el lugar del medio en la ordenación. En caso de que no se cumpla la igualdad, se repite el proceso para una de las dos mitades que restan según si en la evaluación de la condición el valor de la clave principal en esa tupla está antes o después en el orden establecido para el campo. Por tanto, lo que es más eficiente que la búsqueda lineal.
- **Utilización de índices principales.** Si la condición de selección es una igualdad con un atributo clave, se utiliza el índice para encontrar el único registro que la cumple. En caso de ser una comparación ($<$,$<$,$\leq$,$\geq$), en primer lugar se encuentra el único registro que cumple la igualdad, y posteriormente se extraen los registros posteriores o anteriores.

4.3.3. Implementación de la operación JOIN

Una operación JOIN puede involucrar a dos ficheros, siendo de dos vías, o a un número mayor de dos, lo que se denomina multivía. En este trabajo únicamente se estudiarán las diversas formas de implementar un JOIN de dos vías.

Métodos de implementación de JOIN. Operaciones de la forma: $R \bowtie_{A=B} S$

- **JOIN de bucle anidado.** Para cada registro t de R, se toman todos los registros de s de S y se comprueba cuales cumplen la condición, $t[A] = s[B]$. El equivalente a la fuerza bruta.
- **JOIN de bucle simple.** Si existe una estructura de índice para A o B (pongamos B), se toman todos los registros t de R, y luego se utiliza el índice para escoger únicamente los que cumplen la condición $s[B] = t[A]$, del mismo modo que se hace con una selección.
- **JOIN de ordenación-mezcla.** La operación se ejecuta de la forma más eficiente si los registros están ordenados por el valor de los atributos del JOIN. Si los ficheros no se encuentran ordenados, una opción es utilizar un algoritmo de ordenación-mezcla. Mediante los índices de ordenación se pueden explorar los dos archivos simultáneamente, emparejando las parejas de registros que tienen el mismo valor para el atributo de ordenación.

Ejemplo 23. Veamos un ejemplo del JOIN de ordenación-mezcla. Tenemos el siguiente JOIN de dos relaciones de la base de datos EMPRESA:

Consulta 19: $EMPLEADO \bowtie_{DNI=DniEmpleado} TRABAJA\_EN$

Suponer que los atributos DNI y DniEmpleado han sido ordenados como números naturales. Notar que en la relación empleado, el atributo DNI es una clave principal, y al ordenarlos quedan

$$\{123456789, 333445555, 453453453, 666884444, 888665555, 987654321, 987987987, 999887777\}$$

De la misma forma se procede con el atributo DniEmpleado de TRABAJA_EN:

$$\{123456789, 123456789, 333445555, 333445555, 333445555, 333445555, \dots\}$$

Se comienza en orden, $123456789 = 123456789$ se cumple tomando las dos primeras tuplas ordenadas de TRABAJA_EN. Con la tercera no se cumple así que se pasa a comprobar la igualdad tomando la segunda tupla de EMPLEADO. Sin embargo, al estar ordenadas, no es necesario comprobar las anteriores y directamente se verifica que $333445555 = 333445555$. Por lo tanto, ahorra comprobaciones respecto a los otros métodos y es más eficiente. El proceso continúa análogamente con el resto de tuplas.

4.3.4. Implementación de la operación PROYECCIÓN

La proyección se aplica de forma rápida en los casos en los que la lista de atributos que seleccionamos tienen una clave primaria en la relación, puesto que no se formarían tuplas duplicadas en la solución. La eliminación de duplicados, en los casos donde se requiera, es lo que motiva a buscar formas eficientes de aplicar esta operación. Hay dos algoritmos principales, que son aplicar una ordenación a los resultados para proceder a la eliminación, y aplicar dispersión antes de insertar un registro en un fichero.

4.3.5. Implementación de las operaciones conjuntistas y PRODUCTO CARTESIANO

Debido a que las operaciones UNION, INTERSECCIÓN y DIFERENCIA solo se aplican en relaciones compatibles, mediante un algoritmo de ordenación-mezcla que ordene las dos relaciones por los mismos atributos se puede conseguir que para realizar la operación solo sea necesaria una exploración. En el caso del producto cartesiano, al combinar una gran cantidad de registros, es una operación muy costosa que conviene optimizar o sustituir por otras consultas equivalentes.

4.4. Optimización utilizando reglas heurísticas

En esta sección se va a tratar el tema central del trabajo. Hasta ahora se ha hablado de la optimización de las implementaciones de cada operación por separado, pero también existen métodos de optimizar la ejecución de consultas.

En uno de esos métodos, las reglas heurísticas actúan modificando la representación interna inicial de una consulta generada por el analizador de consultas. Dada la flexibilidad que permite el álgebra relacional, es posible cambiar unas operaciones por otras o cambiar el orden de estas para convertir una consulta en otra que tendrá el mismo resultado, pero con una ejecución más optimizada. Un ejemplo de regla heurística sería aplicar en primer lugar las operaciones SELECT y PROJECT, para que al aplicar una operación binaria como un JOIN, el tamaño del fichero sea menor. La estructura interna de una expresión de álgebra relacional se describe como un árbol de consultas, lo que introduce un nuevo concepto.

4.4.1. Árboles de consultas

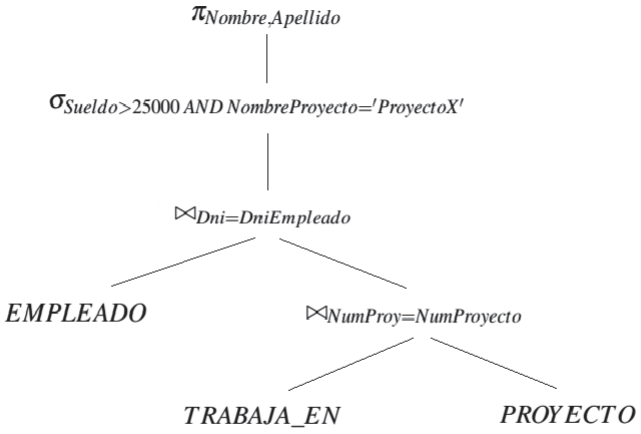
Un árbol de una consulta representa un orden determinado en la ejecución de las operaciones que la componen. Se componen de nodos hoja, que contienen las relaciones de partida, y nodos internos con relaciones resultantes de operaciones de álgebra relacional. El nodo raíz es el nodo interno que contiene el resultado a la última operación. La ejecución de un árbol se hace por orden comenzando por los nodos hoja, siempre y cuando los operandos estén disponibles. En el momento en el que llega a la raíz, la ejecución termina y se genera por pantalla el resultado.

Ejemplo 24. Recordemos la consulta 18 y planteemos el árbol de consultas asociado a ella.

Consulta 20:

$$\pi_{Nombre,Apellido}(\sigma_{Sueldo>25000\ AND\ NombreProyecto='ProyectoX'}(EMPLEADO \bowtie_{Dni=DniEmpleado} (TRABAJA\_EN \bowtie_{NumProy=NumProyecto} PROYECTO)))$$

Figura 2. Árbol asociado a la consulta anterior:



4.4.2. Optimización heurística

Definición. Dos árboles de consultas son equivalentes son la representación de la misma consulta.

Definición. Dos expresiones de álgebra relacional se dicen equivalentes si representan a misma consulta.

Definición. Dos relaciones se dicen equivalentes si tienen los mismos atributos pero en distinto orden.

A partir de una consulta SQL, introducidas en la sección 4.1, el analizador genera un árbol de consultas inicial y el optimizador heurístico lo transforma en uno eficiente, el cual es equivalente con el primero. Existen unas reglas generales de equivalencia que pueden aplicarse en la transformación del árbol de consultas.

Reglas generales de transformación para operaciones de álgebra relacional. En las siguientes reglas consideraremos que dos relaciones con la misma información pero los atributos en distinto orden son equivalentes. La validez de las reglas se puede demostrar haciendo uso de la teoría de conjuntos:

1. **Cascada de σ y π.** La operación σ con un número n de condiciones separadas por AND se puede escribir como una cascada de n selecciones anidadas de una condición. En caso de una cascada de la operación π, donde las listas de atributos están contenidas en cadena, es equivalente a proyectar la relación en la última lista de atributos.

$$A)\ \sigma_{C_1}(\dots(\sigma_{C_n}(R))\dots) \equiv \sigma_{C_1\ AND\dots AND\ C_n}(R)$$

$$B)\ \pi_{L_1}(\dots(\pi_{L_n}(R))\dots) \equiv \pi_{L_1}(R)$$

   Donde en la última estamos suponiendo $L_1 \subseteq \cdots \subseteq L_n$.

2. **Conmutatividad de σ y π.** La operación σ es conmutativa, pero π no lo es. Además, si la condición de selección solo involucra a atributos que pertenece a la lista de proyección, σ y π conmutan.

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

$$B)\ \pi_L(\sigma_C(R)) \equiv \sigma_C(\pi_L(R))$$

3. **Conmutatividad y asociatividad de $\bowtie$, $\times$, $\cup$ y $\cap$.** Estas operaciones son conmutativas, debido a la definición de relación equivalente. Se tiene:

$$A)\ R \cup S \equiv S \cup R$$

$$B)\ (R \cup S) \cup T \equiv R \cup (S \cup T)$$

$$C)\ R \times S \equiv S \times R$$

$$D)\ (R \times S) \times T \equiv R \times (S \times T)$$

   Las igualdades son las mismas con $\cap$ y $\bowtie_C$ haciendo el papel de $\cup$ y $\times$, respectivamente.

4. **Conmutatividad de σ y π con $\bowtie$ o equivalentemente con $\times$.** En el caso de σ, si la condición C se puede descomponer en C_1 AND C_2, afectando C_1 a atributos de R y C_2 a atributos de S únicamente, las operaciones conmutan de la siguiente manera:

$$A)\ \sigma_C(R \bowtie S) \equiv (\sigma_{C_1}(R)) \bowtie (\sigma_{C_2}(S))$$

   Si la lista de atributos de proyección se puede descomponer en $L = \{A_1,\dots,A_n,B_1,\dots,B_m\}$, donde los atributos $A_1,\dots,A_n$ pertenecen a R y los $B_1,\dots,B_m$ a S, y además la condición de $\bowtie$, C, contiene

atributos que no están en L, por ejemplo $A_{n+1},\ldots,A_{n+k},B_{m+1},\ldots,B_{m+l}$, en R y S respectivamente, se puede escribir la operación de este modo.

$$B)\ \pi_L(R \bowtie_C S) \equiv \pi_L((\pi_{A_1,\ldots,A_{n+k}}(R)) \bowtie_C (\pi_{B_1,\ldots,B_{m+l}}(S)))$$

En particular, si C afecta a atributos de L, se tiene:

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

La operación $-$ no es conmutativa ni asociativa.

5. **Conmutatividad de σ y π con las operaciones de conjunto.** Denotando $\theta = \{\cup,\cap,-\}$, se tiene:

$$A)\ \sigma_C(R\theta S) \equiv (\sigma_C(R))\theta(\sigma_C(S))$$

Por otro lado, π conmuta con $\cup$, pero no conmuta con $\cap$ y $-$.

$$B)\ \pi_L(R \cup S) = \pi_L(R) \cup \pi_L(S)$$

6. **Conversión de una secuencia de σ y $\times$ en $\bowtie$.** Por la propia forma en la que está definida $\bowtie$, se sigue:

$$\sigma_C(R \times S) \equiv R \bowtie_C S$$

Demostración. *A continuación vamos a demostrar cada una de las propiedades anteriores. Denotamos $X_i \equiv dom(A_i)$, $Y_i \equiv dom(B_i)$, que trataremos como conjuntos. Por lo tanto, si tenemos el esquema de relación $R(A_1,\ldots,A_n)$, podemos considerar la relación R como un conjunto contenido en $X_1 \times \cdots \times X_n$. La letra t se utilizará para denotar tuplas.*

- ***1 A) y 2 A):*** *Consideramos $R \subseteq X_1 \times \cdots \times X_n$. Sean $A = \{t \in R : t\ cumple\ C_1\}$, $B = \{t \in R : t\ cumple\ C_2\}$ Notar que es suficiente probar el resultado para una cascada de dos operaciones y se puede generalizar.*

$$\sigma_{C_1}(\sigma_{C_2}(R)) = \{t \in B : t\ cumple\ C_1\} \equiv \{t \in R : t\ cumple\ C_1\ y\ C_2\} = \sigma_{C_1\ AND\ C_2}(R) \equiv$$

$$\{t \in A : t\ cumple\ C_2\} = \sigma_{C_2}(\sigma_{C_1}(R))$$

- ***1 B):*** *Sean $I = \{i_1,\ldots,i_k\}$, $J = \{j_1,\ldots,j_l\}$ conjuntos de índices que corresponden a los atributos de L_1 y L_2, con $I \subseteq J$. Denotamos $X_I := X_{i_1} \times \cdots \times X_{i_k}$, $X_J := X_{j_1} \times \cdots \times X_{j_l}$.*

$$\pi_{L_1}(\pi_{L_2}(R)) = (R\big|_{X_J})\Big|_{X_I} \equiv R\big|_{X_I} = \pi_{L_1}(R)$$

- ***2 B):*** *Sea X_I correspondiente a la lista L. La condición C solo afecta a atributos de la lista.*

$$\sigma_C(\pi_L(R)) = \{(t_{i_1},\ldots,t_{i_k}) \in R\big|_{X_I} : (t_{i_1},\ldots,t_{i_k})\ cumple\ C\} \equiv$$

$$\{(t_1,\ldots,t_n) \in R : (t_{i_1},\ldots,t_{i_k})\ cumple\ C\}\big|_{X_I} = \pi_L(\sigma_C(R))$$

- ***3 A) y 3 B):*** *Se desprenden trivialmente de la teoría de conjuntos.*

- ***3 C) y 3 D):*** *Podemos suponer que las relaciones que intervienen en la relación binaria no tienen atributos coincidentes. Sean $R \subseteq X_1 \times \cdots \times X_n$, $S \subseteq X_{n+1} \times \cdots \times X_{n+k}$, $T \subseteq X_{n+k+1} \times \cdots \times X_{n+k+l}$. Debido a la definición de relación equivalente que estamos utilizando, se tiene:*

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

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

$$(R \times S) \times T = \{t = (t_1,\ldots,t_{n+k+l}) : (t_1,\ldots,t_{n+k}) \in R \times S, (t_{n+k+1},\ldots,t_{n+k+l}) \in T\} \equiv$$

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

Para el caso de $\bowtie_C$, basta probar las reglas 4 y 6.

- ***4 A):*** *Veamos, los casos para la operación JOIN, y para el PRODUCTO CARTESIANO es un caso particular. Sean* $R \subseteq X_1 \times \cdots \times X_n$ *y* $S \subseteq X_{n+1} \times \cdots \times X_{n+k}$*. La condiciones* C_1 *y* C_2 *afectan a R y S respectivamente. Se tiene:*

$$\sigma_{C_1\ AND\ C_2}(R \bowtie_C S) = \{t \in R \bowtie_C S : t\ cumple\ C_1\ y\ C_2\} \equiv$$

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

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

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

- ***4 B):*** *Sean ahora* $R \subseteq X_1 \times \cdots \times X_N$ *y* $S \subseteq Y_1 \times \cdots \times Y_M$*. Denotamos* $X_I \equiv X_1 \times \cdots \times X_{n+k}$, $n+k \leq N$, $Y_J \equiv Y_1 \times \cdots \times Y_{m+l}$, $m+l \leq M$ *y* $Z_K \equiv X_1 \times \cdots \times X_{n+k} \times Y_1 \times \cdots \times Y_{m+l}$.

$$\pi_L(R \bowtie_C S) = \{t \in X_1 \times \cdots \times X_N \times Y_1 \times \cdots \times Y_M : t\ cumple\ C\}\big|_{Z_K} \equiv$$

$$\{t \in X_1 \times \cdots \times X_{n+k} \times Y_1 \times \cdots \times Y_{m+l} : t\ cumple\ C\}\big|_{Z_K} = \pi_L((\pi_{A_1,\ldots,A_{n+k}}(R)) \bowtie_C (\pi_{B_1,\ldots,B_{m+l}}(S)))$$

- ***5):*** *Sean* $R,S \subseteq X_1,\ldots,X_n$.
  $A = \{t \in X_1 \times \cdots \times X_n : t\ cumple\ C\}$ *Si denotamos* $\theta = \{\cup,\cap,-\}$*, se tiene:*

$$\sigma_C(R\theta S) = A \cap (R\theta S) = (A \cap R)\theta(A \cap S) = \sigma_C(R)\theta\sigma_C(S)$$

  Sea X_I *correspondiente a L:*

$$\pi_L(R \cup S) = (R \cup S)\big|_{X_I} = \{(t_{i_1},\ldots,t_{i_k}) \in X_I | (t_1,\ldots,t_n) \in R \cup S\} \equiv$$

$$\{(t_{i_1},\ldots,t_{i_k}) \in X_I | (t_1,\ldots,t_n) \in R\} \cup \{(t_{i_1},\ldots,t_{i_k}) \in X_I | (t_1,\ldots,t_n) \in S\} = \pi_L(R) \cup \pi_L(S)$$

  Notar que para $\cap$ *y* $-$ *la igualdad no se cumple.*

- ***6):*** *Por la definición de JOIN es trivial:*

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

□

Descripción de un algoritmo. Utilizando las anteriores reglas, se puede transformar un árbol de consultas en su forma inicial a uno más eficiente de ejecutar. Consta de varios pasos:

1. Haciendo uso de la regla 1, descomponer las selecciones separadas por AND en una cascada de selecciones.

2. Utilizando la conmutatividad de la SELECCIÓN con el resto de operaciones dada por las reglas 2, 4 y 5, desplazar las selecciones hacia la parte más baja del árbol posible, es decir, que se ejecuten antes siempre y cuando la consulta lo permita.

3. Mediante las regla 3 de conmutatividad y asociatividad de operaciones binarias, se ordenan los nodos hoja de la siguiente forma. Primero ejecutar antes los nodos que tengan operaciones SELECCIÓN con el menor número de tuplas. Después, evitar los PRODUCTOS CARTESIANOS que no tengan una condición de JOIN entre las relaciones.

4. Convertir los PRODUCTOS CARTESIANOS en JOIN con la regla 6.

5. Por último, haciendo uso de las reglas de conmutatividad y cascada de la PROYECCIÓN, mover las proyecciones para que se ejecuten tan pronto como sea posible, llegando incluso a descomponer o crear proyecciones si es necesario.

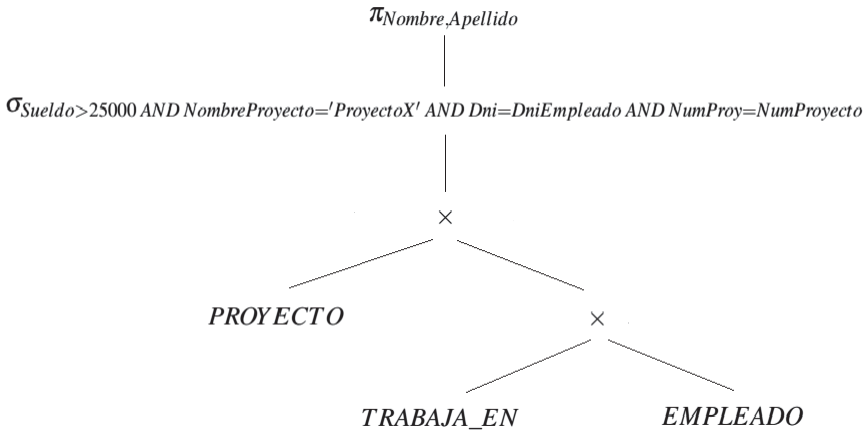
Ejemplo 25: Para clarificar el proceso, veamos como se optimiza el árbol de una que soluciona el ejemplo 21, que consitía en recuperar el nombre y apellido de los empleados que trabajan en el proyecto 'ProyectoX' y que cobran 25000 euros. Empezaremos con la representación en árbol de la forma menos eficiente y aplicaremos el algoritmo en los siguientes pasos.

Consulta 21:

$$\pi_{Nombre,Apellido}(\sigma_{Sueldo>25000\ AND\ NombreProyecto='ProyectoX'\ AND\ Dni=DniEmpleado\ AND\ NumProy=NumProyecto}$$

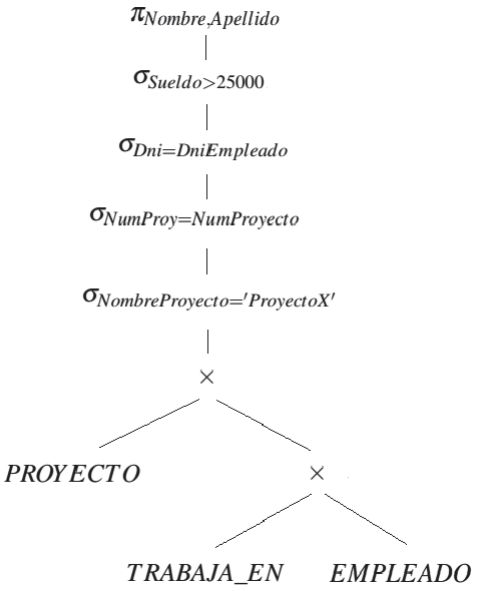
$$(PROYECTO \times (TRABAJA\_EN \times EMPLEADO)))$$

Figura 3:



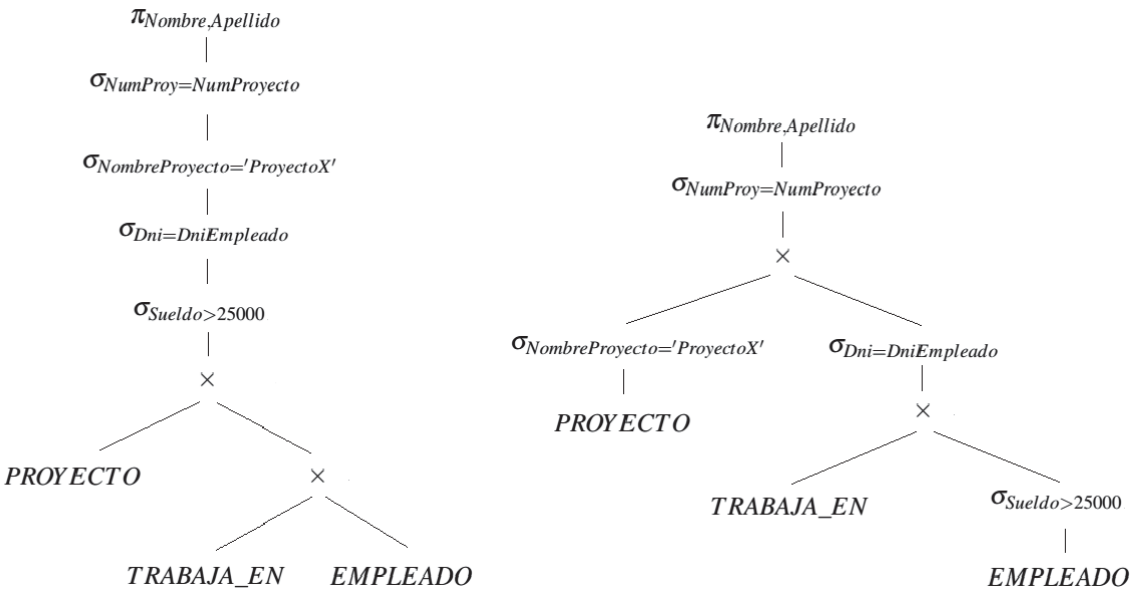
Paso 1. En primer lugar se separan las cascadas de selecciones. Mediante la regla 1A, la descomponemos en cuatro operaciones separadas, y obtenemos el siguiente árbol que es equivalente al inicial.

Figura 4:



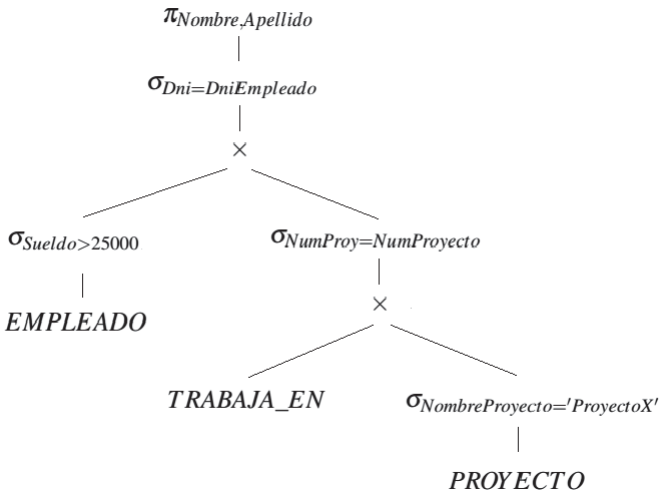
Paso 2. Cada una de las selecciones la desplazamos en la dirección de los nodos hoja tanto como se pueda. E primer lugar usamos la regla 2A para ordenar el orden de ejecución de las selecciones. Posteriormente mediante la regla 4A, conmutamos las selecciones con los productos, siempre y cuando la condición de selección pertenezca a una de las dos relaciones de partida.

Figura 5:



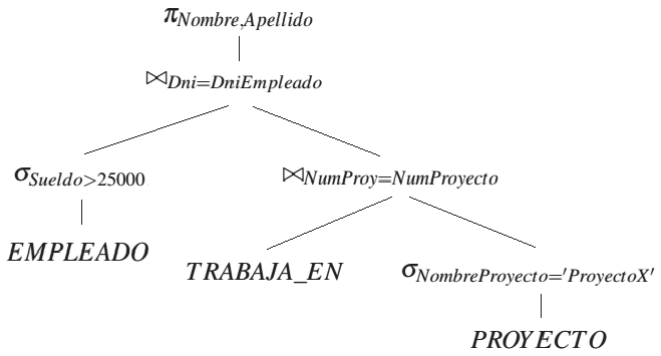
Paso 3. La regla 3C indica que los PRODUCTOS CARTESIANOS conmutan. Por tanto, ordenamos los nodos hoja de forma que se ejecute antes las selecciones que menos tuplas tenga. Notar que realizando primero el JOIN de PROYECTO y TRABAJA_EN con la condición de selección dada se obtienen menos tuplas que con el primer JOIN que se ejecuta en este árbol. Nos aseguramos además de que los JOIN tienen una selección posterior.

Figura 6:



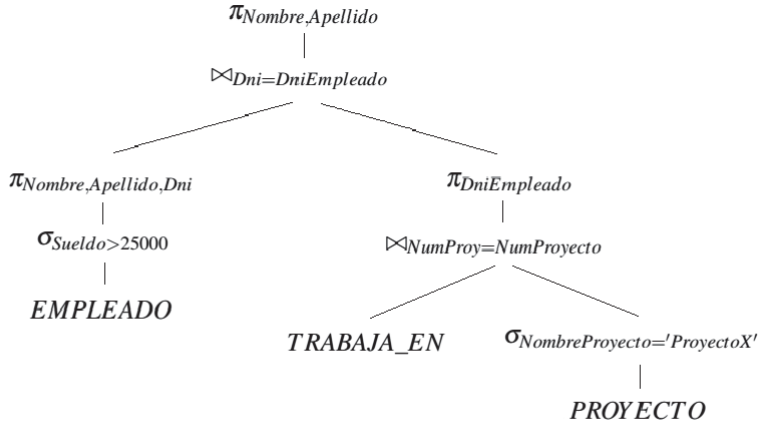
Paso 4. Al tener productos con una SELECCIÓN posterior, usamos la regla 6 para convertir los PRODUCTOS CARTESIANOS en JOIN. La condición de la SELECCIÓN posterior se convierte en la condición del JOIN.

Figura 7:



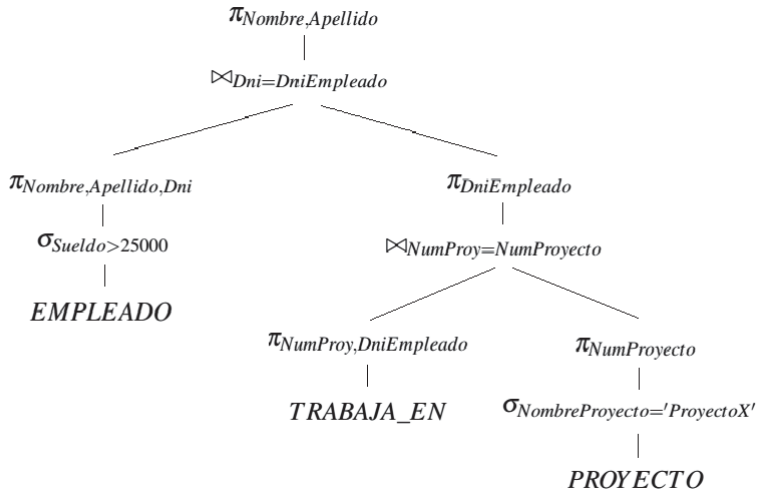
Paso 5. Por último, creamos las proyecciones pertinentes de modo que no se cree ninguna relación intermedia con un atributo que después no se vaya a usar, para que el coste computacional sea el menor posible. Para ello usamos las siguientes reglas. En primer lugar, por la regla 4B, se pueden crear dos proyecciones en los dos nodos que subyacen del nodo raíz. En ellas, nos quedamos tanto con los atributos respectivamente que se usan en la condición del JOIN como en los de la proyección del nodo raíz.

Figura 8:



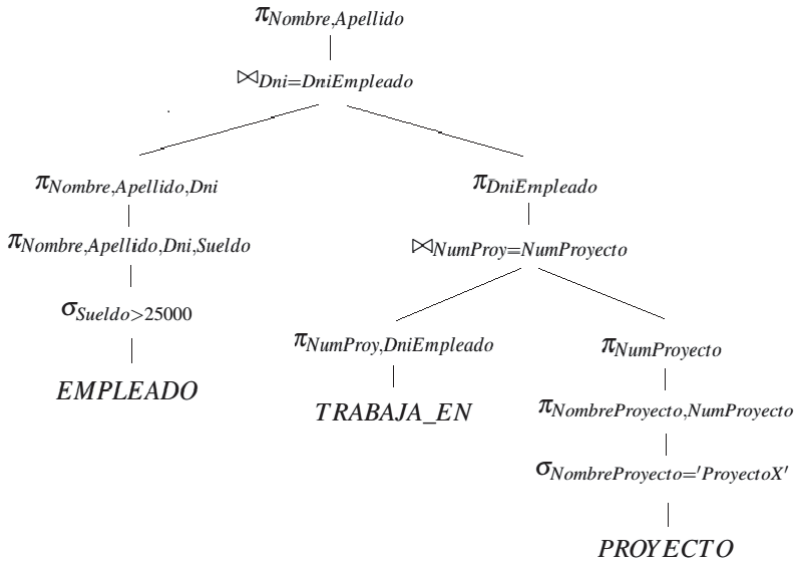
Aplicamos la misma regla análogamente con el nodo con la operación $\pi_{DniEmpleado}$. Obtenemos el siguiente árbol:

Figura 9:



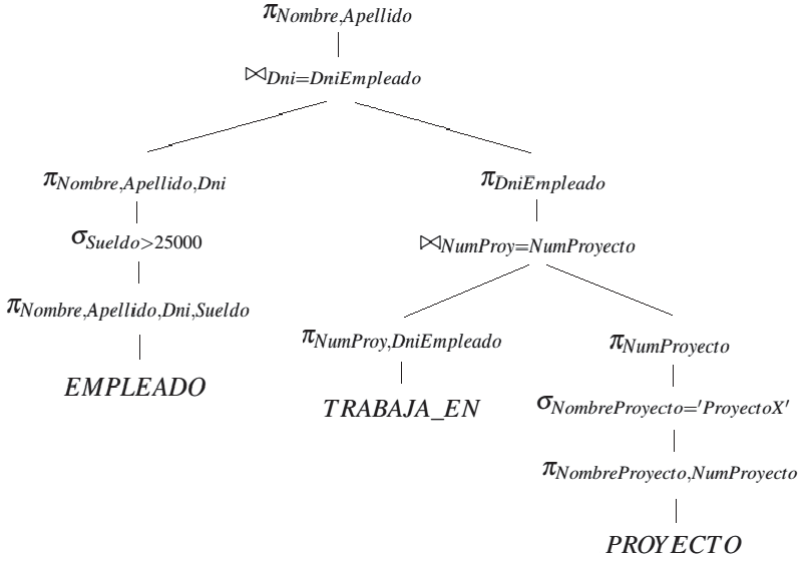
En el siguiente paso, notar que desde los nodos de las selecciones $\sigma_{Sueldo>25000}$ y $\sigma_{NombreProyecto='ProyectoX'}$ en adelante, no se necesitan todos los atributos de las relaciones EMPLEADO y PROYECTO. Entre cada uno de esos nodos y las proyecciones inmediatamente superiores, creamos una PROYECCIÓN con los atributos de la PROYECCIÓN de encima junto con los atributos que intervienen en la condición de selección de abajo. Notar que por la regla 1B para cascada de proyecciones, el nuevo árbol es equivalente.

Figura 10:



Se cumplen ahora las condiciones para aplicar la regla 2B, para desplazar la proyección a la parte baja del árbol.

Figura 11:



Este último árbol representa la ejecución optimizada de la consulta del ejemplo utilizando este algoritmo heurístico. Si convirtiéramos el último árbol en una consulta de álgebra relacional, tendríamos la consulta optimizada que devuelve el mismo valor que la inicial.

Anexos

Apéndice A

Ejemplo de implementación

A partir del modelo relacional de Codd, se ha extendido el uso del álgebra relacional y ha hecho que se hayan realizado implementaciones en diversos programas de algoritmos de optimización de consultas. En este apéndice se a mostrar una implementación del algoritmo que se ha visto en la sección anterior, en el lenguaje de alto nivel Python, así como un ejemplo real de una base de datos relacional. Esta herramienta puede descargarse y ejecutarse en [6].

La herramienta permite cargar archivos con bases de datos relacionales o crearlas el usuario. A estas relaciones se le pueden aplicar las operaciones del álgebra relacional en forma de consultas y posteriormente procede a optimizarla utilizando reglas del tipo heurístico. Los formatos que admite para cargar bases de datos son .txt, .csv y .JSON.

La aplicación es principalmente educativa. Permite expresar consultas del álgebra relacional y ver una forma de optimizarlas, pero en ningún caso permite trabajar con grandes bases de datos, las cuales podrían tener tablas con miles y miles de filas. Para este propósito se necesitaría hacer uso de programas como los de Oracle, los cuales ya tienen en su interior sus propios de algoritmos de optimización implementados que se aplican automáticamente, y pueden ser los tipos de algoritmos que se han visto, entre otros.

A.1. Sintaxis de las consultas

La sintaxis de las operaciones es similar al de las expresiones del álgebra relacional, con la diferencia de que no se utilizan subíndices:

- **Selección**: Tiene una sintaxis muy similar a la ya vista y es la siguiente: σ *condición* (R), donde la condición puede contener los siguientes símbolos o expresiones $\{+, -, *, /, <, >, \leq, \geq, ==, != , and, or, not\}$. Respecto a la notación vista en el trabajo, el único cambio es que la condición $=$ se expresa $==$.
- **Proyección**: π *lista* (R), donde la lista de atributos va separada por comas. Por ejemplo:

$$\pi\, Nombre, Apellido\ (EMPLEADO)$$

- **Renombrar atributos**: ρ *lista* (R), siendo una lista de atributos con nuevo nombre de la siguiente forma, separadas por comas: $antiguo1 \rightarrow nuevo1, antiguo2 \rightarrow nuevo2, \ldots$

- **Binarias**: Aquí entran tanto las operaciones de la teoría de conjuntos como PRODUCTO CARTESIANO y JOIN. La sintaxis es la siguiente:

$$R\theta S,\ \theta = \{\cup, \cap, -, *, \bowtie\}$$

'*' denota el PRODUCTO CARTESIANO. Además la operación JOIN está implementada de distinta forma. Esta operación no viene con una condición, y solo puede aplicarse con relaciones en las que hay atributos comunes, con el mismo nombre. Al aplicar el JOIN entre dos relaciones, únicamente escoge las tuplas en las que los atributos comunes coinciden. Por lo tanto, en caso de querer aplicar el JOIN con una condición distinta a la igualdad, solamente es posible con un PRODUCTO CARTESIANO seguido de una SELECCIÓN.

- **Nombrar consulta**: $nombre = consulta$.

- **Cadenas de caracteres:** En las condiciones de selección se pueden relacionar cadenas de caracteres con los operadores lógicos $==$. Se escriben de la forma 'x'.

- **Tipos de valores:** Se admiten los siguientes tipos de datos: String, int, float, none y Rdate. Donde los String son cadenas de caracteres, int son números enteros, float números reales, none son los valores NULL y Rdate las fechas. En particular, Rdate lee las fechas con la siguiente sintaxis: YYYY-MM-DD. Años, meses y días en ese orden. Si en la columna solo hay valores con ese formato, automáticamente los detecta como Rdate. Si en la columna hay un algún valor tipo String, automáticamente toda la columna la considera de ese tipo.

A.2. Optimización de las consultas

Al seleccionar la opción de optimizar en el programa, este asume que la consulta es correcta y nos realiza los cambios pertinentes, tras los cuales el resultado es el mismo que el original. Las reglas que sigue la herramienta son las siguientes, y descritas de la siguiente forma:

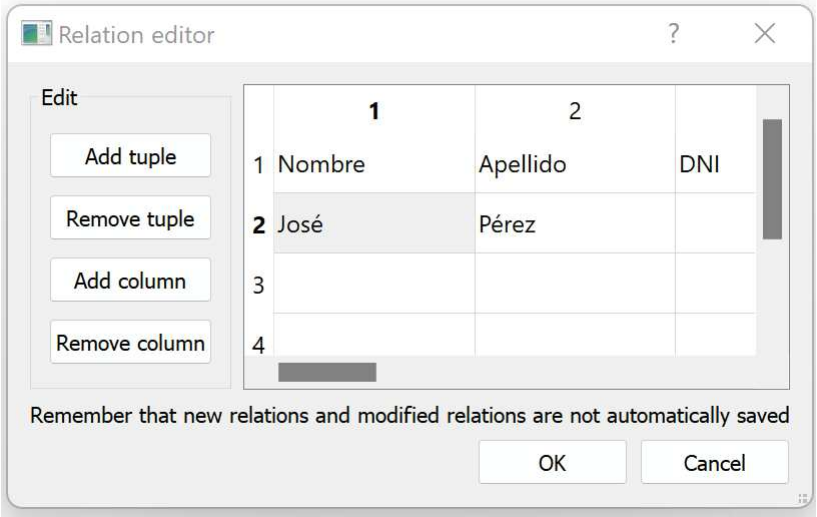
$$SinOptimizar \rightarrow Optimizado$$

1. $\sigma\, C_1\ (\sigma\, C_2\ (R)) \rightarrow \sigma\, C_1\ and\ C_2\ (R)$
2. $\sigma\, C\ (R\,\theta\, S)) \rightarrow \sigma\, C\ (R)\ \theta\ \sigma\, C\ (S)$, siendo $\theta = \{\cup, \cap, -\}$.
3. $\sigma\, C\ (\pi\, L\ (R)) \rightarrow \pi\, L\ (\sigma\, C\ (R))$. También, en caso de tener una proyección que no se utiliza, se elimina.
4. $\pi\, L_1\ (\pi\, L_2\ (R)) \rightarrow \pi\, L_1\ (R)$, siendo $L_1 \subseteq L_2$.
5. $R \times S \cup R * T \rightarrow R * (S \cup T)$
6. $\sigma\, C_1\ (R) \cup \sigma\, C_2\ (R) \rightarrow \sigma\, C_1\ or\ C_2\ (R)$
7. $\pi\, L\ (R) \cup \pi\, L\ (S) \rightarrow \pi\, L\ (A \cup B)$, siendo R y S compatibles.
8. $\sigma\, C\ (R * S) \rightarrow \sigma\, C_T\ (\sigma\, C_R\ (R) * \sigma\, C_S\ (S))$, donde C_R y C_S son las partes de la condición que contienen atributos de R y S respectivamente, y C_T de ambos.
9. Consultas triviales de diversos tipos. Ejemplo: $R \cup R \rightarrow R$. También se encuentran en este grupo las operaciones con ρ, el cual va en último lugar en la ejecución.

A.3. Aplicación a una consulta

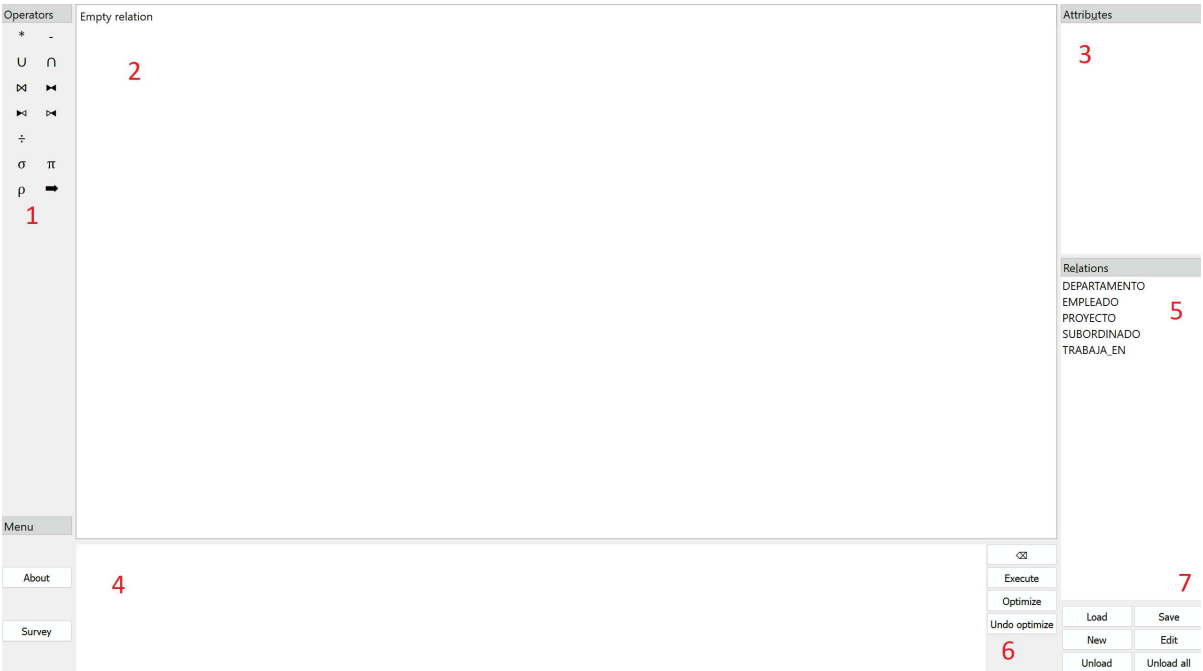
Veamos un ejemplo de una forma de cargar una base de datos, de las muchas que hay. Si se tiene un archivo en formato Excel, lo más sencillo es convertirlo a formato csv, del cual se puede importar directamente a la aplicación. De otra forma, si las tablas no son muy extensas, se puede crear una tabla directamente desde el programa introduciendo manualmente los datos, así como crear y eliminar filas y columnas. En la primera fila de esta opción del programa se escriben los nombres de los atributos.

Figura 12.



Ejemplo 26. De esta última forma descrita, cargamos la base de datos relacional EMPRESA, para ver un ejemplo. Una vez están las relaciones cargadas, tenemos las siguientes funcionalidades de la aplicación:

Figura 13.



1. Símbolos de las operaciones del álgebra relacional. Al seleccionar uno de ellos se escribe automáticamente en el espacio habilitado para escribir consultas.

2. Espacio donde aparecen los resultados de las consultas, o las relaciones que queramos visualiza para comprobar que no se han introducido datos erróneamente. La relación EMPLEADO tiene el siguiente aspecto.

Figura 14. Relación EMPLEADO en la aplicación:

Nombre	Apellido	DNI	FechaNac	Direccion	Sexo	Sueldo	SuperDni	Dno
Fernando	Ojeda	666884444	1962-09-15	Portillo, s/n	H	38000	333445555	5
Juana	Sainz	987654321	1941-06-20	Cerquillas, 67	M	43000	888665555	4
José	Pérez	123456789	1965-09-01	Eloy I, 98	H	30000	333445555	5
Aurora	Oliva	453453453	1972-07-31	Antón, 6	M	25000	333445555	5
Alicia	Jiménez	999887777	1968-05-12	Gran Vía, 38	M	25000	987654321	4
Alberto	Campos	333445555	1955-12-08	Avda. Ríos	H	40000	888665555	5
Luis	Pajares	987987987	1969-03-29	Enebros, 90	H	25000	987654321	4
Eduardo	Ochoa	888665555	1937-11-10	Las Peñas, 1	H	55000		1

3. Muestra la lista de atributos de la relación que está en el espacio 2.

4. Espacio para escribir las consultas, utilizando la sintaxis antes descrita. Escribimos la consulta 21.

Figura 15.

$$\pi\ \text{Nombre,Apellido}\ (\sigma\ \text{DNI==DniEmpleado and NumProy==NumProyecto and NombreProyecto=='ProyectoX' and Sueldo>25000}\ (\text{EMPLEADO*(TRABAJA\_EN*PROYECTO)}))$$

5. Nombres de las relaciones que tenemos cargadas en la aplicación. También aparecerán las nuevas relaciones a las que renombremos a partir de consultas.

6. Opciones para borrar la consulta, ejecutarla, optimizarla o anular la optimización y obtener la consulta inicial. Seleccionamos la opción 'Execute' y nos muestra por pantalla el resultado de la consulta.

Figura 16.

Nombre	Apellido
José	Pérez

7. Lugar para modificar las relaciones que están cargadas. Se puede crear relaciones, editar las existentes y borrarlas. En caso de crear, nos aparecerá la pantalla de la figura 12. También es posible cargar una relación con un archivo de los formatos que se han dicho y guardar en un archivo la alguna relación para poder borrar las relaciones y recuperarlas en otro momento.

Por último, seleccionamos la opción de optimizar, y nos muestra una optimización de la consulta aplicando las reglas descritas en la sección anterior. La consulta queda de la siguiente forma:

Figura 17.

optm_a = π Nombre,Apellido (σ DNI == DniEmpleado (σ Sueldo > 25000 (EMPLEADO)*
σ NumProy == NumProyecto (TRABAJA_EN*σ NombreProyecto == 'ProyectoX' (PROYECTO))))

La aplicación realiza las siguientes optimizaciones. Primero separa la única SELECCIÓN con muchas condiciones en cuatro distintas con una condición. Las tienen condiciones $DNI == DniEmpleado$ y $NumProy == NumProyecto$ las ejecuta justo después de los PRODUCTOS que unen las relaciones que tienen estos atributos, respectivamente. Lo que no hace la aplicación es convertir los PROCUCTOS con posterior SELECCIÓN en JOIN, como el algoritmo que se ve en la sección 4.4. Por otro lado, las SELECCIONES con condiciones $NombreProyecto ==' ProyectoX'$ y $Sueldo > 25000$ son las primeras operaciones que se ejecutan en en sus respectivas relaciones, PROYECTO y EMPLEADO.

Como se puede observar, el algoritmo que utiliza la implementación es menos completo que el visto en la sección 4.4. No se crean JOIN a partir de PRODUCTOS ni tampoco se crean proyecciones intermedias para eliminar atributos que no se utilicen en el futuro. Por tanto, se deduce que esta implementación puede ser mejorada mediante la programación de las reglas heurísticas restante que se explican en este trabajo.

Bibliografía

[1] RAMEZ ELMASRI y SHAMKANT B.NAVATHE, *Fundamentals of database systems*, quinta edición, Pearson, Georgia, 2007.

[2] TED CODD, *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, Volume 13, Number 6, Georgia, 1970.

[3] JAVIER QUIROZ, *El modelo relacional de bases de datos*, Boletín de política informática, Número 6, INEGI, México, 2003.

[4] EXTERNAL SORT-MERGE ALGORITHM, `https://www.javatpoint.com/external-sort-merge-algorithm`.

[5] CLASIFICACIÓN EXTERNA, `https://opendsa-server.cs.vt.edu/OpenDSA/Books/CS3/html/ExternalSort.html`.

[6] SALVO TOMASELLI, *Relational tool* `https://ltworf.github.io/relational/`.