Optimización de consultas

Bases de Datos 1

Optimización de consultas

- Se desea una estrategia eficiente para la evaluación de una expresión relacional.
- Es necesaria para que el SABD opere eficientemente (un acceso a los datos en el menor tiempo posible, utilizando una cantidad limitada de recursos y en forma transparente para el usuario).
- Las expresiones son dadas con un alto nivel semántico (SQL), por lo que la optimización es requerida.

Optimización de consultas

- El optimizador puede realizar un trabajo eficiente de optimización porque:
  - Tiene información de control disponible, (acerca de: cardinalidad de las tablas, estadísticas, índices, otros).
  - Puede evaluar muchas más alternativas de implementación que el programador.

Optimización de consultas

- El procesamiento de una consulta se divide en dos grandes funciones:
  - Evaluación y optimización.

Optimización de consultas

- Cuando un usuario efectúa una consulta a la base de datos, el SABD verifica la sintaxis y seguidamente traduce el estatuto SQL a una expresión algebraica equivalente, la cual puede ser representada como un árbol, en donde los nodos representan los operadores relacionales y las hojas representan las relaciones y vistas involucradas en dicha consulta.

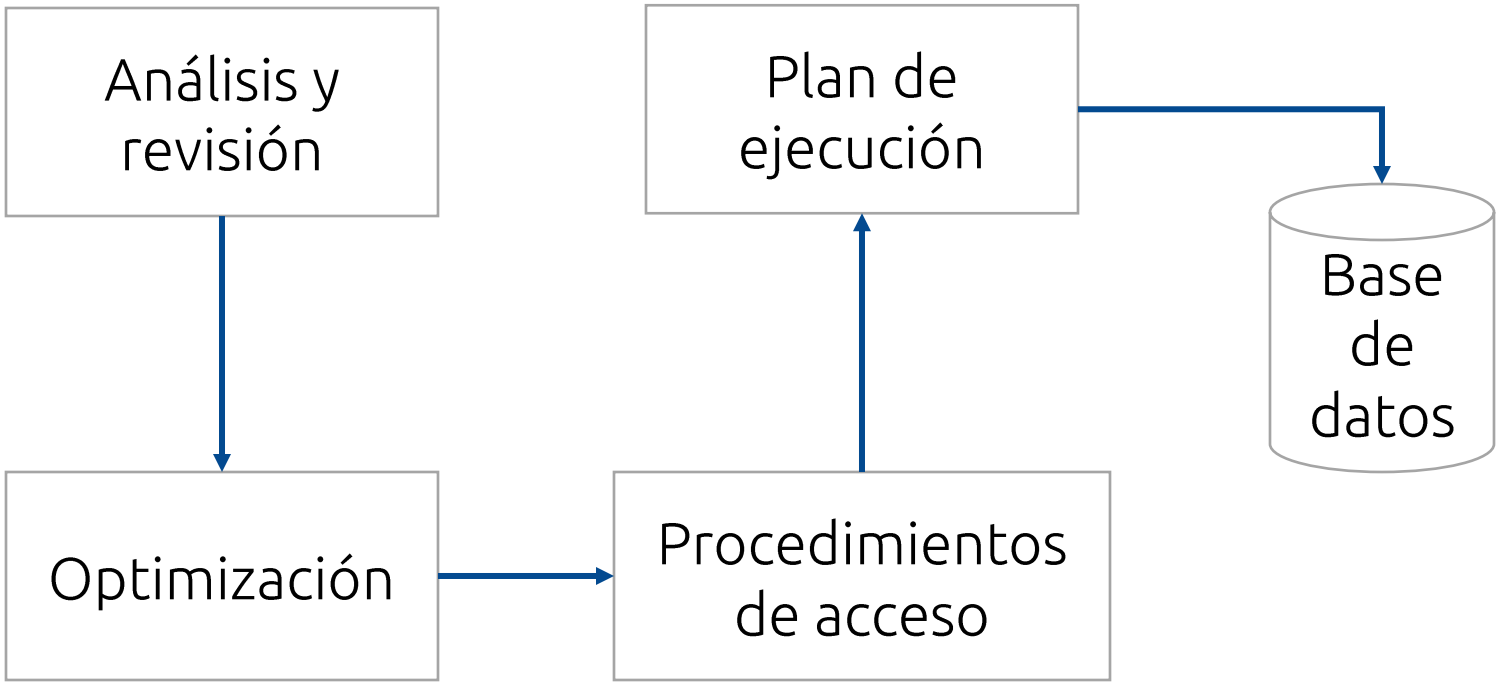
Optimización de consultas

- Posteriormente, este árbol se pasa al módulo optimizador que se encarga de escoger, entre un número importante de posibilidades, una en donde el orden en que se van a ejecutar las operaciones garantiza el acceso más rápido.
- La salida del optimizador es un plan de ejecución (QEP) de la consulta del usuario.

Optimización de consultas

- Finalmente cuando se ejecuta el código para responder la consulta se invocan servicios de otro módulo denominado procedimientos de acceso, los cuales acceden las tuplas de la base de datos.

Optimización de consultas



Optimización de consultas

- El proceso de optimización de una consulta, involucra la transformación de una expresión relacional en otra expresión lógicamente equivalente con el fin de alcanzar un rendimiento aceptable.
- Se desea transformar la expresión (de alto nivel) original en una segunda (optimizada) para lo cual un optimizador debe conocer un conjunto de leyes de transformación y aplicarlas.
- Las transformaciones se basan en la utilización de propiedades tales como la distributiva, asociativa, la conmutativa, la transitividad y su aplicación a los operadores relacionales, de comparación, lógicos y aritméticos.

Optimización de consultas

- El objetivo es aplicar las selecciones y proyecciones sobre las relaciones involucradas en la consulta lo más tempranamente posible para reducir el número de filas a ser examinadas en una próxima operación en la secuencia y probablemente también reducir el número de filas de salida de la próxima operación.
- El reducir el tiempo de respuesta de una consulta está condicionado a la forma en que se implementaron los operadores relacionales en el SABD, y a las técnicas que permiten la escogencia de los accesos óptimos a los datos.

Optimización de consultas

- El parámetro fundamental para medir la eficiencia de una consulta es el tiempo que transcurre entre el momento en que se hace y el momento en que se reciben los datos.
- El costo total que debe minimizarse es entonces la suma de los siguientes costos:
  - Costo de comunicación.
  - Costo de acceso a la memoria secundaria.
  - Costo de almacenamiento.
  - Costo de procesamiento.

Costo de comunicación

- Se refiere al costo de transmitir los datos del o de los sitios en donde se encuentran almacenados, al sitio o los sitios en donde se van a procesar.
- En un ambiente centralizado este factor es irrelevante.

Costo de acceso a la memoria secundaria

- Este costo se refiere al tiempo requerido para transmitir las páginas que se encuentran en la memoria secundaria, a los buffers del CPU.

Costo de almacenamiento

- Se refiere al costo del tiempo que transcurre para transmitir y confirmar las páginas de datos que se envían del CPU a la memoria secundaria.

Costo de procesamiento

- Es el costo referido al tiempo de utilización del CPU.

Ejemplo

- El costo se va a medir en función del número de tuplas accedidas.
- Para calcular el costo de procesar una consulta se podrían tomar en consideración otras variables como: la longitud de las tuplas de una relación, índices definidos, organización física de tuplas (número de accesos a disco).

Ejemplo

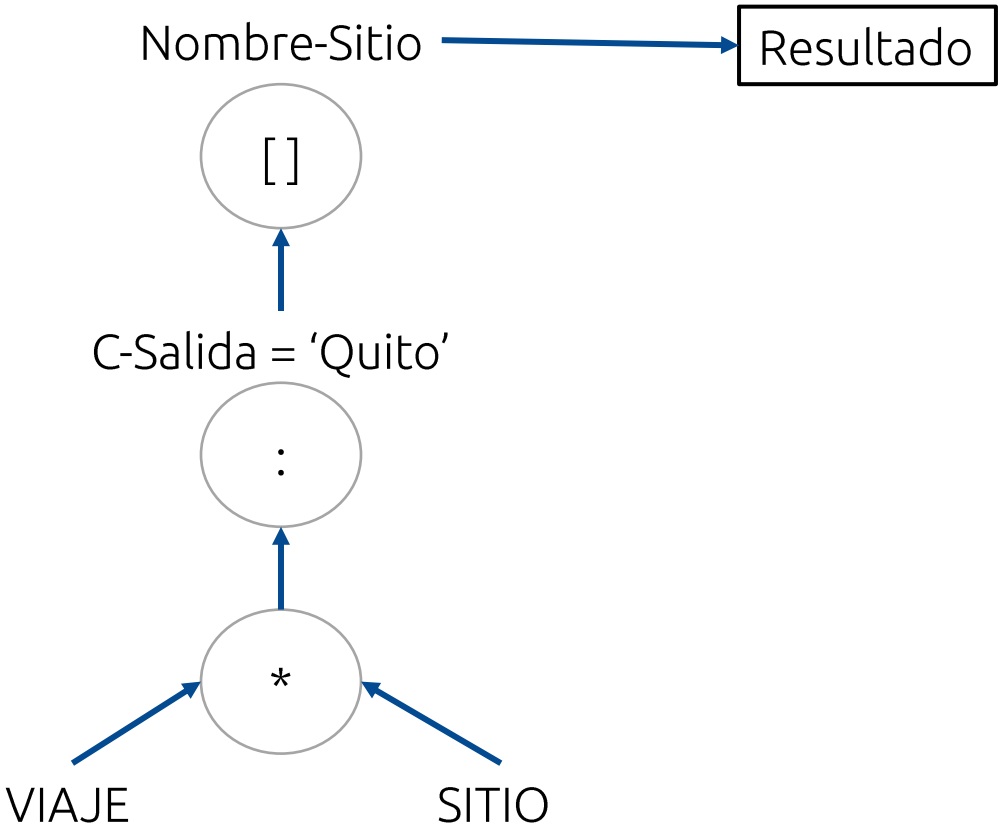
- Viaje = {Número_viaje, número_turista, número_sitio, fecha_salida, fecha_llegada, ciudad_salida}
  - Tuplas = 10,000 de las cuales 20 contienen a 'Quito' como ciudad de salida.
  - El número de viaje tiene una ciudad de salida, una fecha de salida y una de llegada. Un turista visita el número de sitio indicado.

- Sitio = {Número_sitio, nombre_sitio, tipo, continente}
  - Tuplas = 200.

- Considere la siguiente consulta: dar los nombres de los sitios visitados en viajes cuya ciudad de salida es Quito.

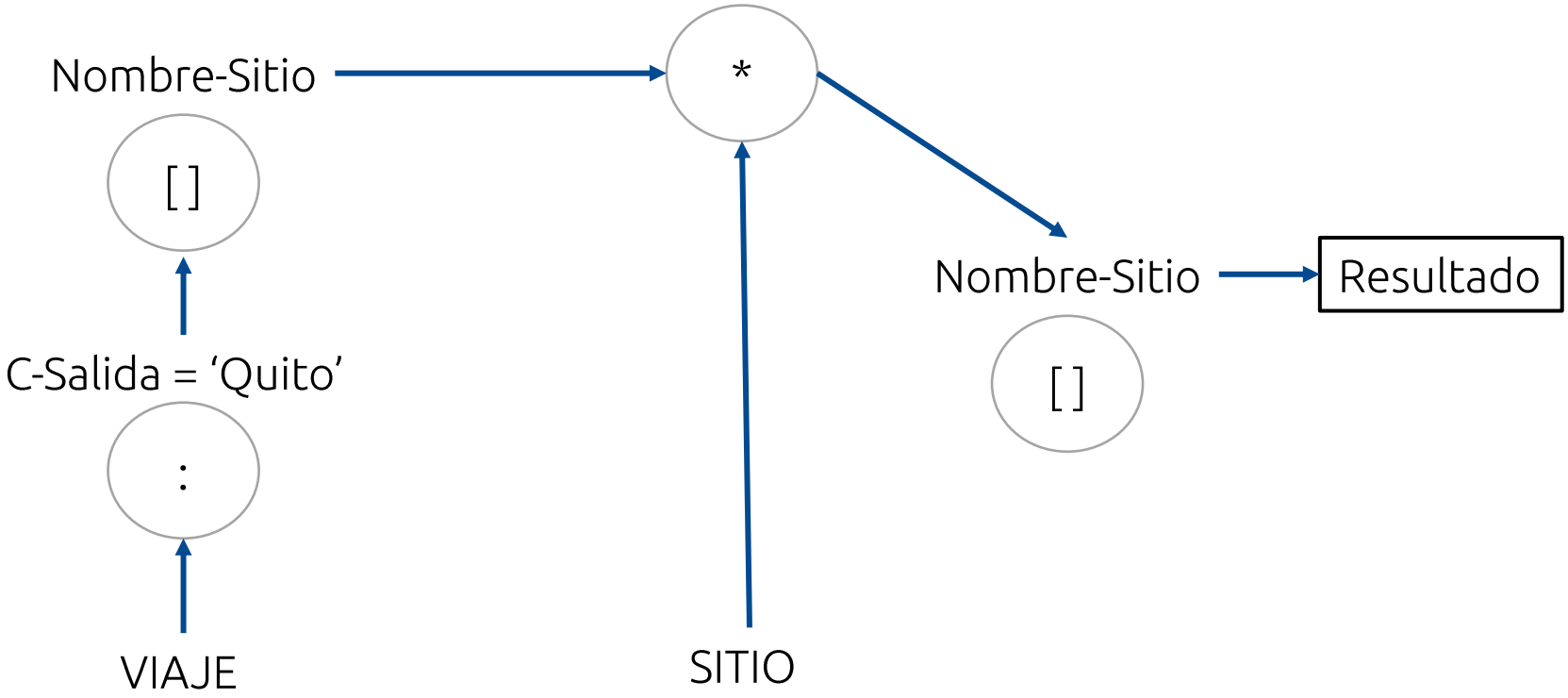
Ejemplo - Respuesta Número 1

- ((VIAJE * SITIO): C-Salida = ‘Quito’) [Nombre-Sitio].



Ejemplo - Respuesta Número 2

- (((VIAJE: C-Salida = ‘Quito’) [Numero-Sitio]) * SITIO) [Nombre-Sitio].



Ejemplo - Respuesta Número 1

- ((VIAJE * SITIO): C-Salida = ‘Quito’) [Nombre-Sitio].

Operación	Lecturas	Escrituras
Join	2,000,000	10,000
Selección	10,000	20
Proyección	20	20
Total	2,010,020	10,040

Ejemplo - Respuesta Número 2

- (((VIAJE: C-Salida = ‘Quito’) [Numero-Sitio]) * SITIO) [Nombre-Sitio].

Operación	Lecturas	Escrituras
Selección	10,000	20
Proyección	20	20
Join	4,000	20
Proyección	20	20
Total	14,040	80

¿Cuál es más eficiente?

Costo de una consulta

- El costo, en tiempo de ejecución, de una consulta a una base de datos depende de:
  - Tiempo de acceso al sistema de E/S (90%).
  - Tiempo de procesamiento de CPU (10%).
- El tiempo de acceso al sistema de E/S depende de:
  - Volumen de datos: número y tamaño de las tuplas, tanto de las relaciones involucradas como de los resultados intermedios
  - Organización física: índices, tablespaces y otros.
  - Tamaño de los buffers en memoria para almacenar los resultados intermedios.

Para reducir el costo de una consulta

- Reducir los accesos al sistema de E/S:
  - Reducir los resultados intermedios: menos accesos y más probabilidad de que quepan en los buffers en memoria.
  - Seleccionar por atributos indexados: accesos mucho más eficientes y búsquedas mucho más rápidas.
- Reducir el procesamiento de CPU:
  - Simplificar expresiones de selección: ahorran tiempo de procesamiento y a veces accesos al sistema.
- Si los resultados intermedios no caben en los buffers en memoria, deben almacenarse en disco.

Heurísticas de optimización de consultas

- Realizar las selecciones tan pronto como sea posible = mover las selecciones lo más abajo que se pueda en el árbol.
  - Reduce el número de tuplas de los resultados intermedios.
- Realizar las selecciones sobre atributos indexados antes que sobre los no indexados.
  - Reduce las operaciones de E/S al usar los índices y se ejecutan mucho más rápido. Aplicar la cascada de selecciones si es necesario.
- Realizar las proyecciones tan pronto como sea posible.
  - Reduce el número de atributos de los resultados intermedios.

Heurísticas de optimización de consultas

- Realizar las operaciones de selección y join más restrictivas primero.
  - Reduce el número de tuplas de los resultados intermedios.
- Eliminar proyecciones redundantes.
  - Reduce el número de atributos de los resultados intermedios. Aplicar la cascada de proyecciones.
- Usar DISTINCT sólo cuando sea imprescindible.
  - Evita tener que comparar resultados intermedios para detectar duplicados y eliminarlos del resultado.

Implementación de operadores relacionales

- Operador proyección:
  - Se procede al recorrido de la tabla base, y a la vez se genera el resultado.
- Operador selección:
  - Se pueden utilizar algoritmos clásicos de búsqueda en un archivo para ejecutar una selección sobre una relación. Por ejemplo: barrido secuencial o índices.

Implementación de operadores relacionales

- Operadores conjuntistas (unión, intersección, diferencia, producto cartesiano):
  - Se procede con el ordenamiento de los atributos de las dos relaciones involucradas, los cuales son compatibles. Posteriormente se recorren para dar el resultado.
- Los operadores binarios son costosos (join, división). Por esta razón, en cualquier técnica de optimización, esos operadores deben colocarse lo más cerca posible de la raíz del árbol de la expresión relacional y los operadores unarios cerca de las hojas.

Operador θ Join

- Sean R(A, A_2, ..., A_n) y S(B, B_2, ..., B_m) dos relaciones en donde los atributos A y B son compatibles.
- Además, sean t_1 y t_2 dos tuplas de R y S respectivamente y sea RESULTADO = R<A θ B>S.

Operador θ Join

- Entrada: Dos relaciones R y S.
- Salida: R<A θ B>S.
- Para cada tupla t_2 en R hacer:

  Para cada tupla t_1 en S hacer:

  Si (t_1[A] θ t_2[B]) entonces

  Poner en la relación RESULTADO la concatenación de t_1 y t_2.

- El costo del operador θ Join, es el costo de leer cada tupla de una relación multiplicado por el costo de leer todas las tuplas de la otra relación.

Ordenamiento y fusión

- Las relaciones se ordenan según los atributos involucrados en el Join. Así, las dos relaciones se barren en dicho orden y las tuplas que satisfacen la condición se fusionan para generar una nueva relación.

Ordenamiento y fusión

- Entrada: Dos relaciones R y S.
- Salida: R <A = B>S.
- Fase de ordenamiento:
  - Ordenar R sobre t_1[A].
  - Ordenar S sobre t_2[B].

Ordenamiento y fusión

- Fase de fusión:

  Leer primera tupla de S

  Para cada tupla t_1 de R hacer:

  Mientras $t_2[B] < t_1[A]$ hacer:

  Leer la siguiente tupla de S

  Si $(t_1[A] = t_2[B])$ entonces

  Poner en la relación RESULTADO la concatenación t_1 y t_2.

- En este caso el costo del algoritmo es la suma de los costos de leer cada una de las relaciones.

Hash

- Entrada: Dos relaciones R y S.
- Salida: R<A = B>S.
- Para cada tupla t_2 en S hacer:

  Haga un hash en los atributos $t_2[B]$,

  Ponga la referencia a la tupla en la tabla hash dependiendo del valor producido.

- Para cada tupla t_1 hacer:

  Haga un hash en los atributos $t_1[A]$,

  Si hash($t_1[A]$) cae en un lugar no vacío de la tabla entonces,

  Si ($t_1[A] = t_2[B]$) entonces

  Poner en la relación RESULTADO la concatenación de t_1 y t_2.

Optimizador de consultas

- Es un conjunto de programas que tienen como finalidad la transformación de una expresión algebraica en otra equivalente que es menos costosa que la primera.
- Maneja reglas de equivalencia.

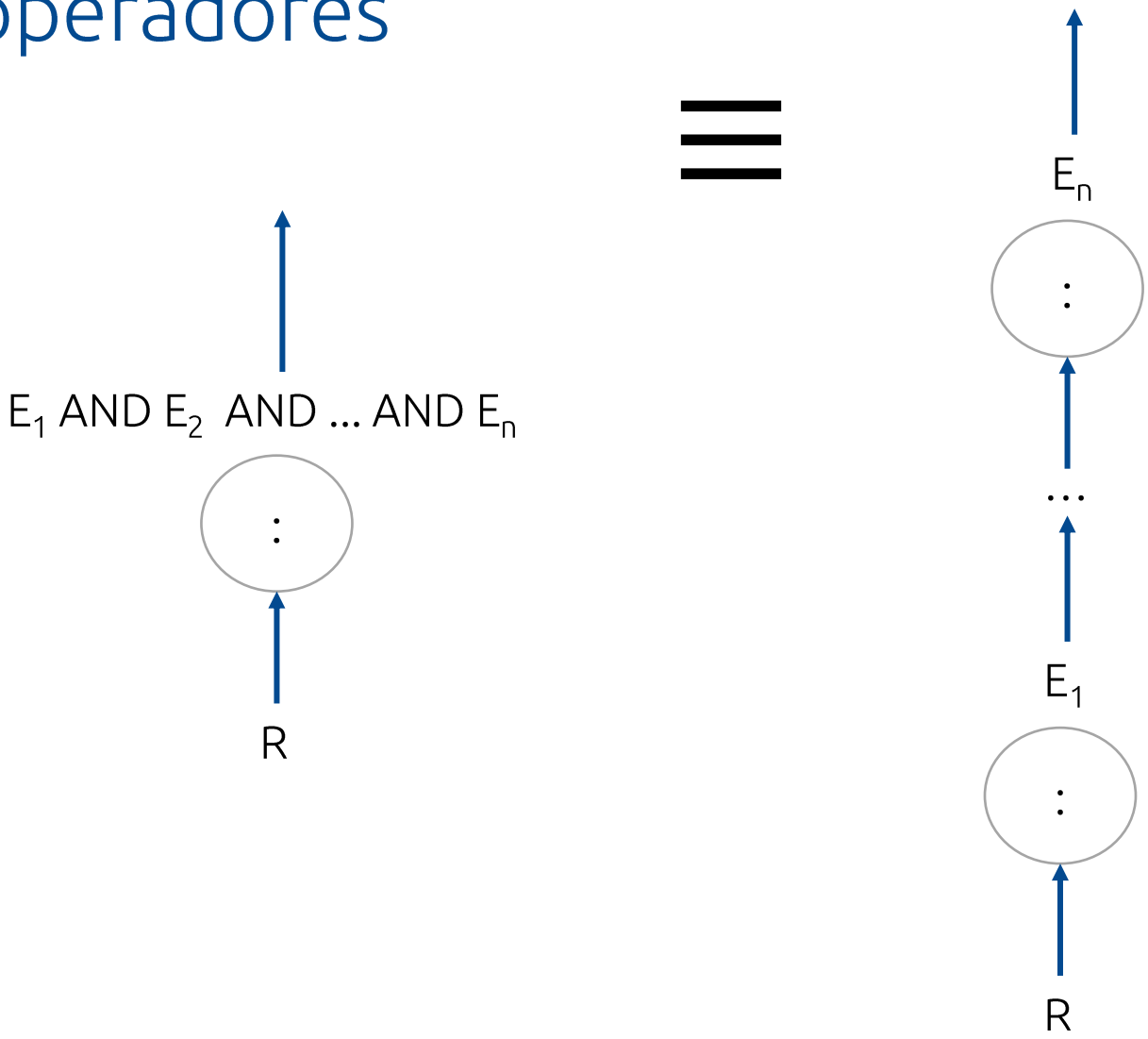
Reglas de equivalencia

- r_1: Explosión en cadena de los operadores
- r_2: Commutatividad de la selección
- r_3: Explosión en cadena de la proyección
- r_4: Transposición de la proyección con la selección
- r_5: Transposición de la selección con el producto cartesiano o con el Join.
- r_6: Transposición de la proyección con el producto cartesiano o con el Join.
- r_7: Conmutatividad de la unión y la intersección
- r_8: Asociatividad del Join
- r_9: Transposición de la selección con la unión, intersección o con la diferencia
- r_{10}: Transposición de la proyección con la unión

r_1: Explosión en cadena de los operadores

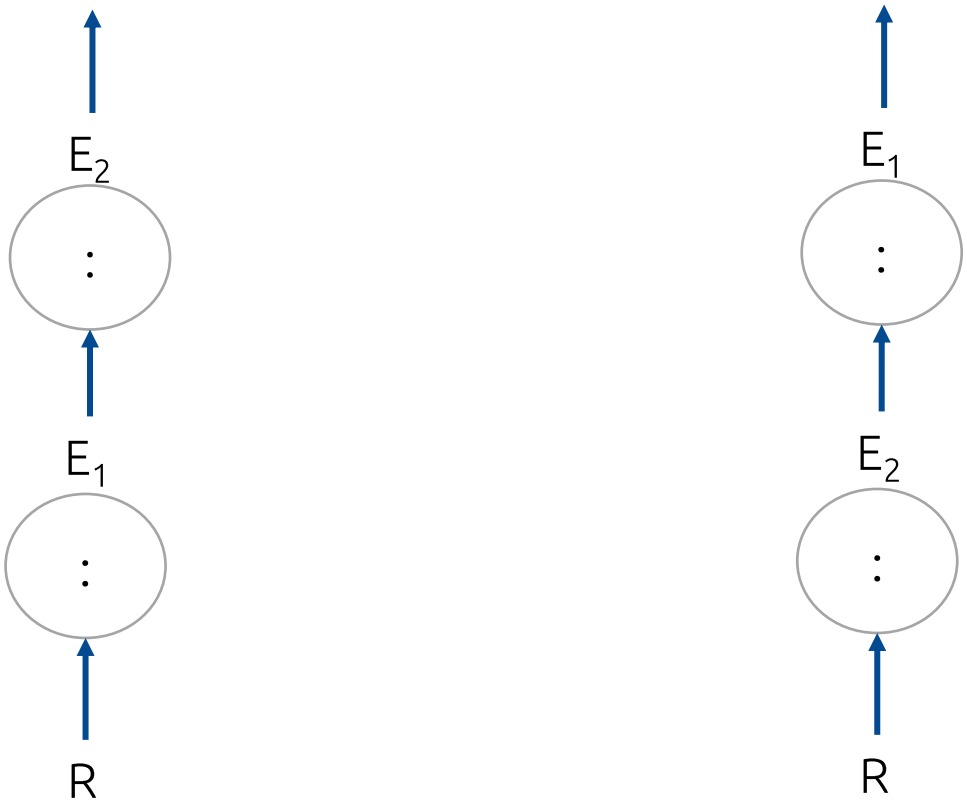
- Para una selección sobre una relación R, en donde la condición de selección E se puede expresar como una conjunción E_1 AND E_2 AND ... AND E_n es equivalente a una aplicación reiterada de selecciones basadas en estas subcondiciones, es decir la siguiente equivalencia se cumple:
  - R: (E_1 AND E_2 AND ... AND E_n) = (((R : E_1) : E_2) ....) : E_n.

r_1: Explosión en cadena de los operadores



r_2: Conmutatividad de la selección

- La siguiente equivalencia se verifica:
  - ((R : E_1) : E_2) ((R : E_2) : E_1).



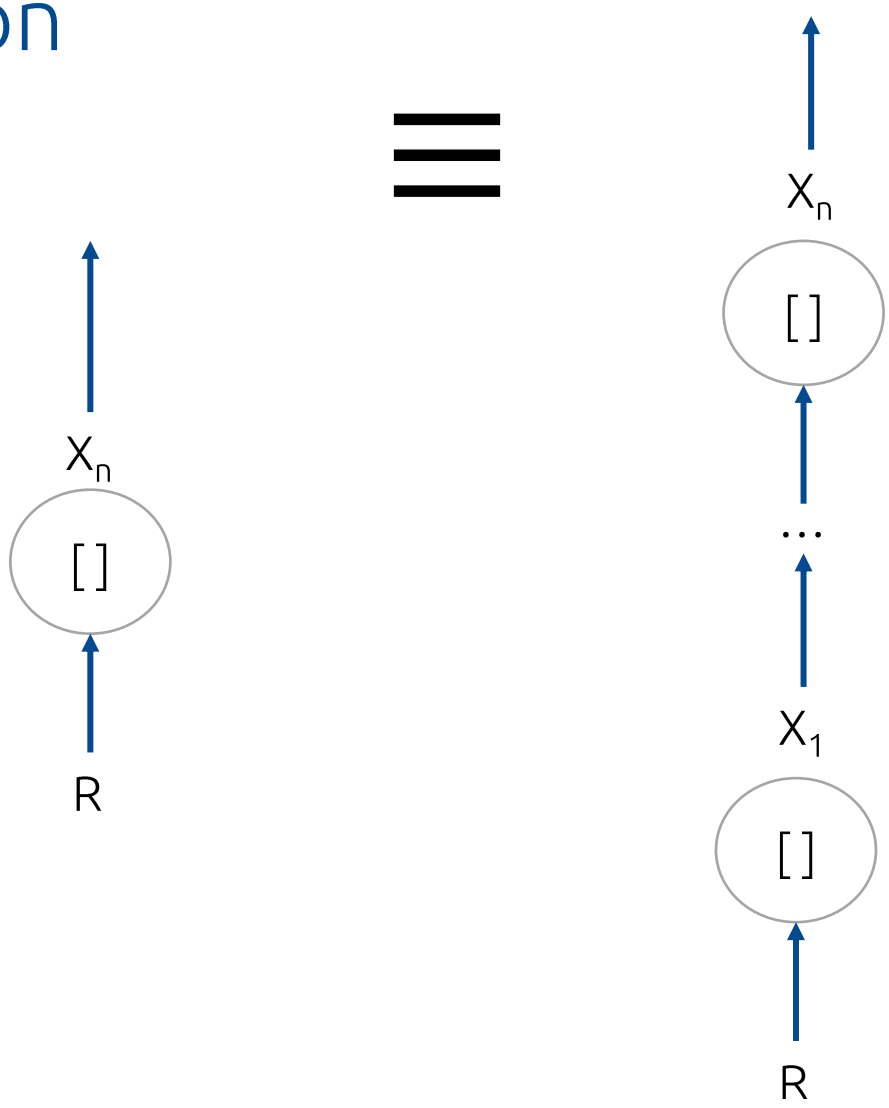
Es importante en el caso de que E_i sea sobre un campo con índice.

r_3: Explosión en cadena de la proyección

- Efectuar una serie de proyecciones sobre una misma relación es equivalente a realizar la proyección únicamente sobre el último conjunto de atributos proyectados, es decir, la siguiente equivalencia se cumple:
  - $(((R[X_1])[X_2]) \ldots . [X_n]) = R[X_n]$.

- En este caso, para que las proyecciones reiteradas se cumplan se debe tener el siguiente orden de inclusiones:
  - X1 X2 ..... Xn.

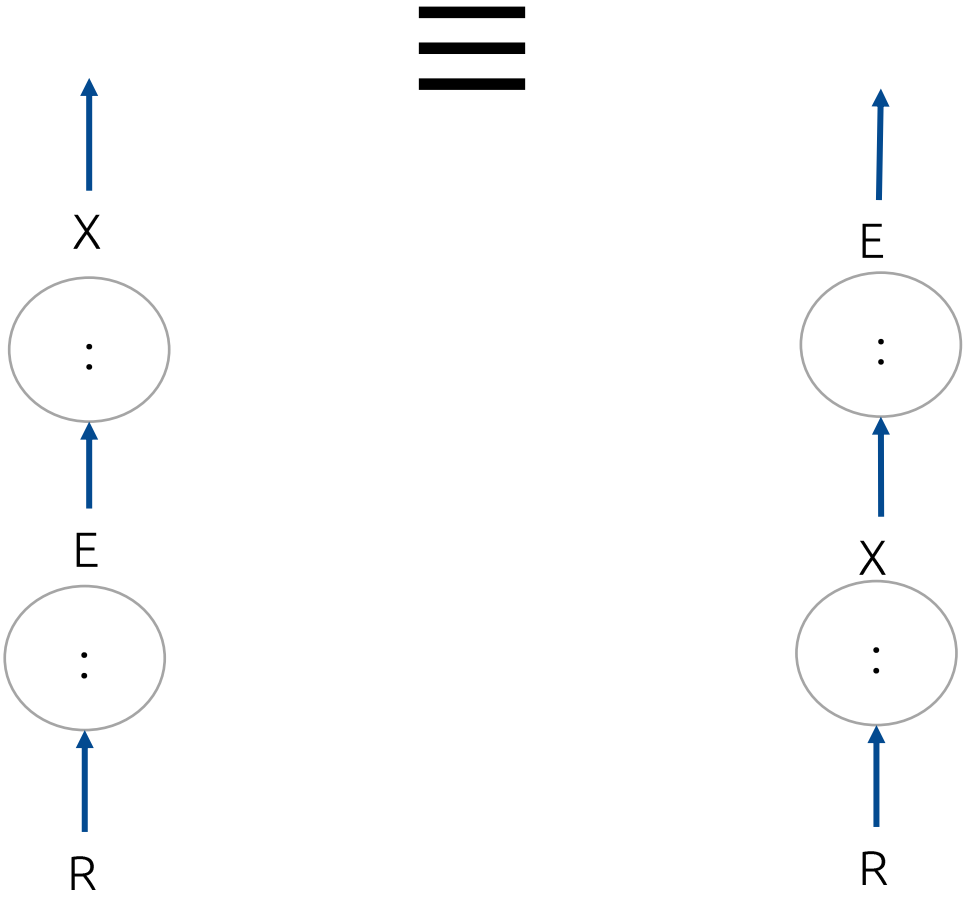
r_3: Explosión en cadena de la proyección



r_4: Transposición de la proyección con la selección

- Si los atributos ubicados en la condición de selección de E se encuentran en el conjunto de atributos X sobre el cual se quiere hacer la proyección:
  - R[X] : E (R : E) [X].

r_4: Transposición de la proyección con la selección



r_5: Transposición de la selección con el producto cartesiano o con el Join

- Cuando se desean invertir los operadores de selección y producto cartesiano en una expresión en donde se encuentran involucrados dos relaciones, según los atributos que aparecen en la condición de selección, se pueden caracterizar tres casos.

r_5: Transposición de la selección con el producto cartesiano o con el Join

- En primer lugar, si la condición E solo contiene atributos de una de las relaciones (suponer que sean los atributos de S) entonces la equivalencia se expresa como:
  - (R X S) : E = R X (S : E).

- Por otra parte, si la condición E se puede representar como E_1 AND E_2, en donde E_1 es una subcondición que solo contiene atributos de R y E_2 contiene atributos de S, entonces la equivalencia es la siguiente:
  - (R X S) : E_1 AND E_2 = (R : E_1) X (S : E_2).

r_5: Transposición de la selección con el producto cartesiano o con el Join

- Finalmente, si la condición E se puede representar como E_1 AND E_2, en donde E_1 es una subcondición que solo contiene atributos de R y E_2 contiene atributos de R y de S, entonces la equivalencia es la siguiente:
  - $(R \times S) : E_1 \text{ AND } E_2 = ((R : E_1) \times S) : E_2$.

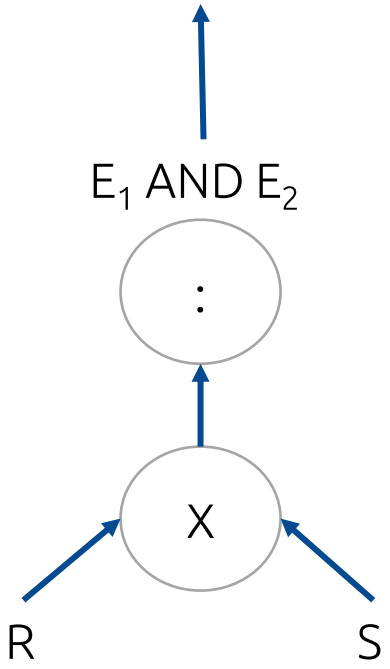
- Estas equivalencias también se cumplen si en vez del producto cartesiano se tiene el operador θ Join debido a la equivalencia siguiente:
  - $R< A \theta B>S = (R \times S) : (A \theta B)$.

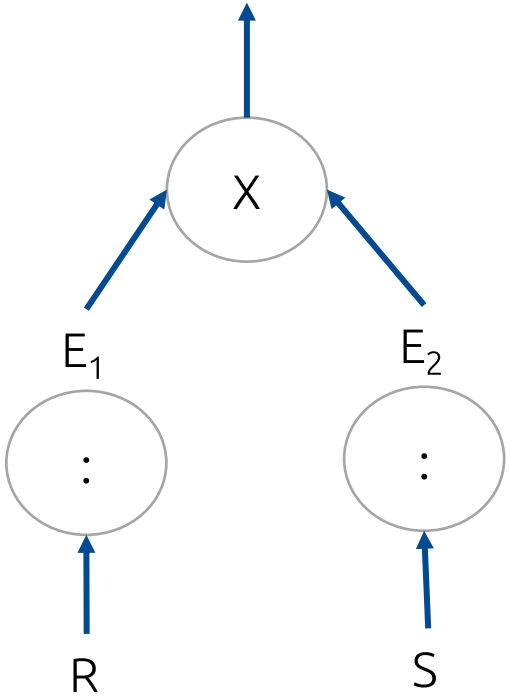
r_5: Transposición de la selección con el producto cartesiano o con el Join

- Entonces, según el caso, se tienen las siguientes equivalencias del operador Join:
  - $(R\langle A\ \theta\ B\rangle S) : E = R\langle A\ \theta\ B\rangle (S : E)$.
  - $(R\langle A\ \theta\ B\rangle S) : E_1$ AND $E_2 = (R : E_1)\ \langle A\ \theta\ B\rangle\ (S : E_2)$.
  - $(R\langle A\ \theta\ B\rangle S) : E_1$ AND $E_2 = ((R : E_1)\langle A\ \theta\ B\rangle S) : E_2$.

r_5: Transposición de la selección con el producto cartesiano o con el Join

- Se puede apreciar la representación el caso de la transposición de la selección y el producto cartesiano, en donde la condición $E = E_1$ AND E_2, E_1 solo contiene atributos de R y E_2 solo atributos de S.



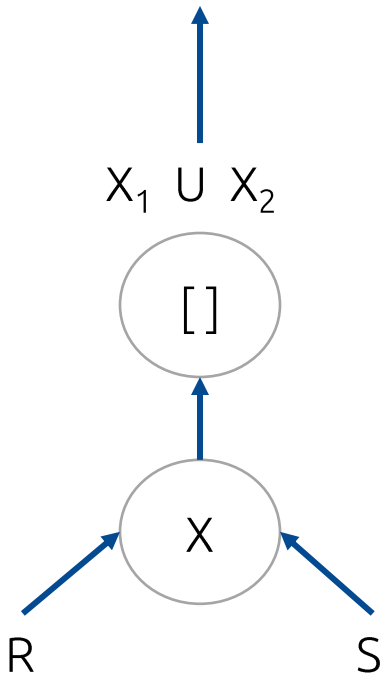


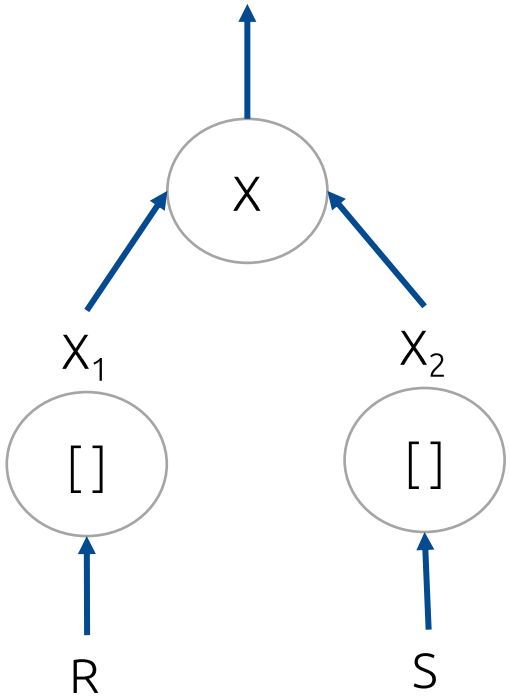
r_6: Transposición de la proyección con el producto cartesiano o con el Join

- En este caso, si R y S son dos relaciones y $X = X_1, X_2$ es un subconjunto de atributos, en donde X_1 son atributos de R y X_2 son atributos de S, entonces se verifica la siguiente equivalencia:
  - $(R \times S)[X] = R[X_1] \times S[X_2]$.

- Esta misma regla se cumple cuando se sustituye el producto cartesiano por el θ Join, es decir, se tiene la siguiente equivalencia:
  - $(R \langle A \theta B \rangle S)[X] = R[X_1] \langle A \theta B \rangle S[X_2]$.

r_6: Transposición de la proyección con el producto cartesiano o con el Join



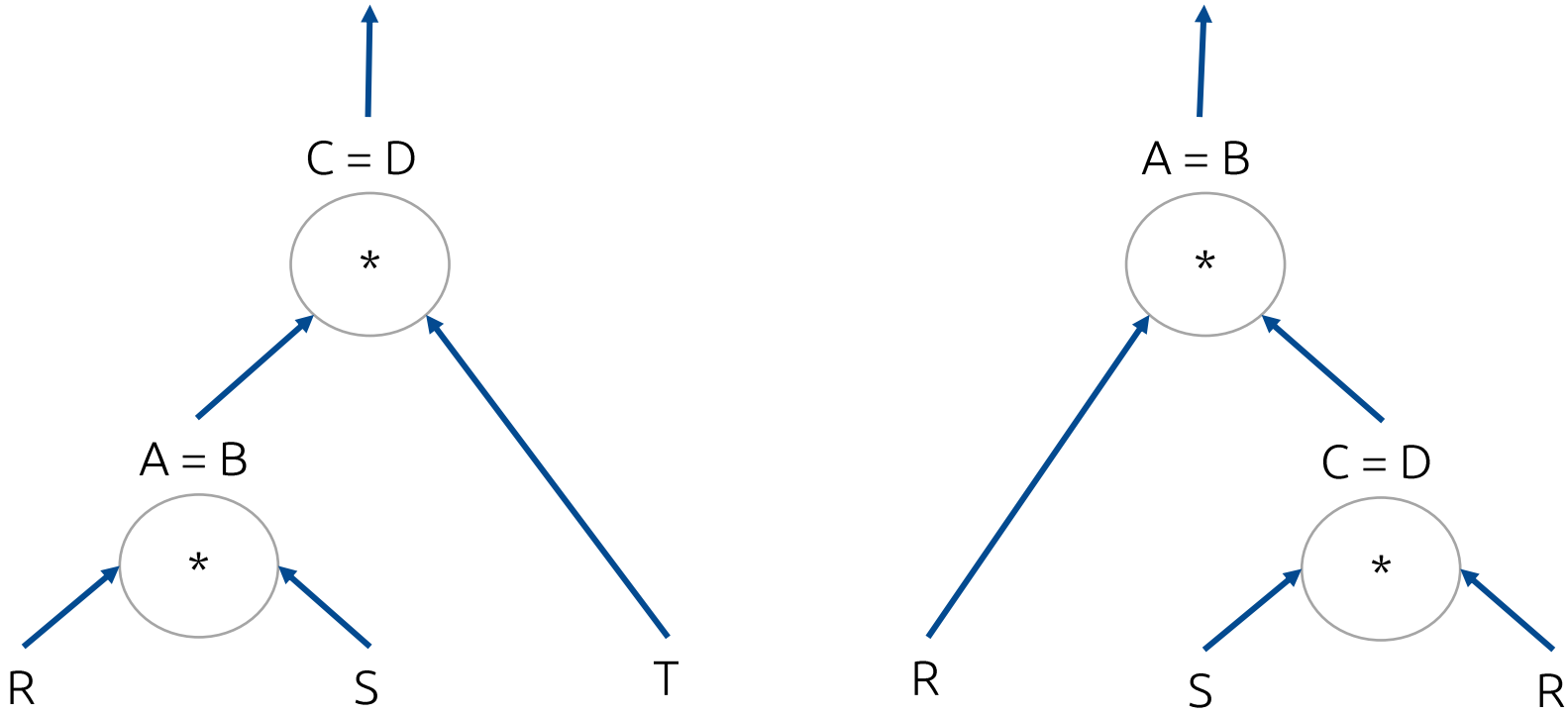


r_7: Conmutatividad de la unión y la intersección

- Se tienen las siguientes reglas:
  - $R \cup S \equiv S \cup R$.
  - $R \cap S \equiv S \cap R$.

r_8: Asociatividad del Join

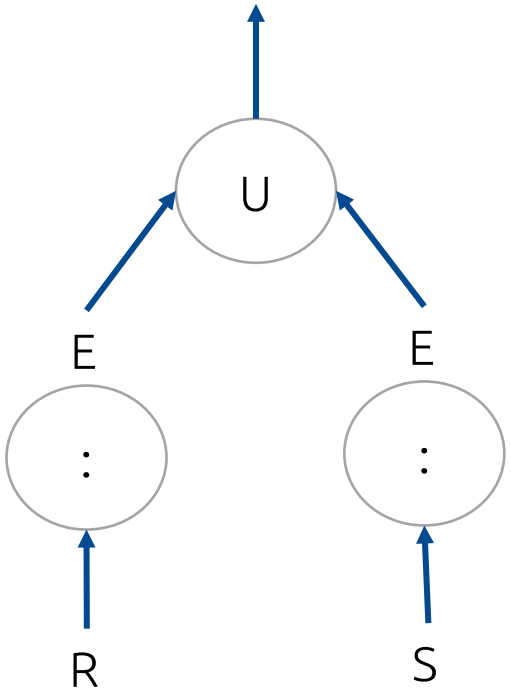
- La siguiente equivalencia se cumple:
  - (R <A θ B>S) <C θ D>T = R<A θ B> (S<C θ D>T).

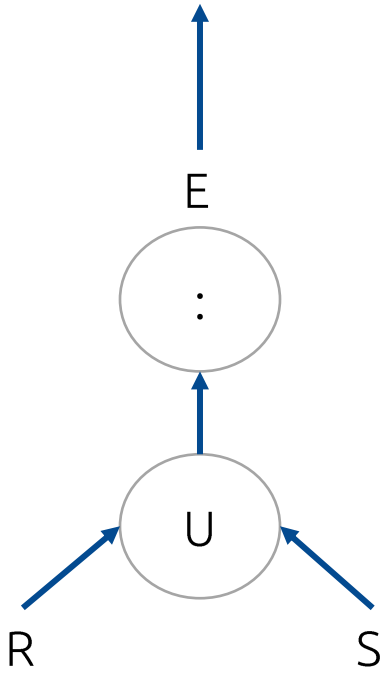


r_9: Transposición de la selección con la unión, intersección o con la diferencia

- Es lo mismo hacer una selección sobre una unión de relaciones, que hacer primero la selección sobre cada una de las relaciones y posteriormente efectuar la unión de estas relaciones. Esta regla también se cumple para los operadores intersección y diferencia, es decir, se cumple lo siguiente:
  - $(R \cup S) : E = (R : E) \cup (S : E)$.
  - $(R \cap S) : E = (R : E) \cap (S : E)$.
  - $(R - S) : E = (R : E) - (S : E)$.

r_9: Transposición de la selección con la unión, intersección o con la diferencia





r_{10}: Transposición de la proyección con la unión

- $(R \cup S)[Z] \equiv R[Z] \cup S[Z]$.
- Esta regla también es aplicable a la intersección y a la diferencia, es decir, las siguientes equivalencias se verifican:
  - $(R \cap S)[Z] \equiv R[Z] \cap S[Z]$.
  - $(R - S)[Z] \equiv R[Z] - S[Z]$.

Algoritmo de optimización de expresiones algebraicas

- Paso 1:
  - Separar las selecciones que contienen varios predicados en forma conjuntiva, usando la regla r_1 cuando pertenecen a diferentes tablas.
- Paso 2:
  - Descender las selecciones lo más bajo posible en el árbol, con ayuda de las reglas r_4, r_5 y r_9.
- Paso 3:
  - Agrupar las selecciones que se realizan sobre una misma relación usando la regla r_1.

Algoritmo de optimización de expresiones algebraicas

- Paso 4:
  - Descender las proyecciones lo más bajo posible en el árbol, con ayuda de las reglas r_6 y r_{10}.
- Paso 5:
  - Agrupar las proyecciones sucesivas dejando los atributos restantes y eliminar eventuales proyecciones inútiles que hubiesen podido aparecer (proyección sobre todos los atributos de una relación usando la regla r_3).

Gracias - ¿Preguntas?