Procesamiento y Optimización de Consultas

Mariano Beiró, Lucas Román

Dpto. de Computación - Facultad de Ingeniería (UBA)

Topics

Esquema de procesamiento

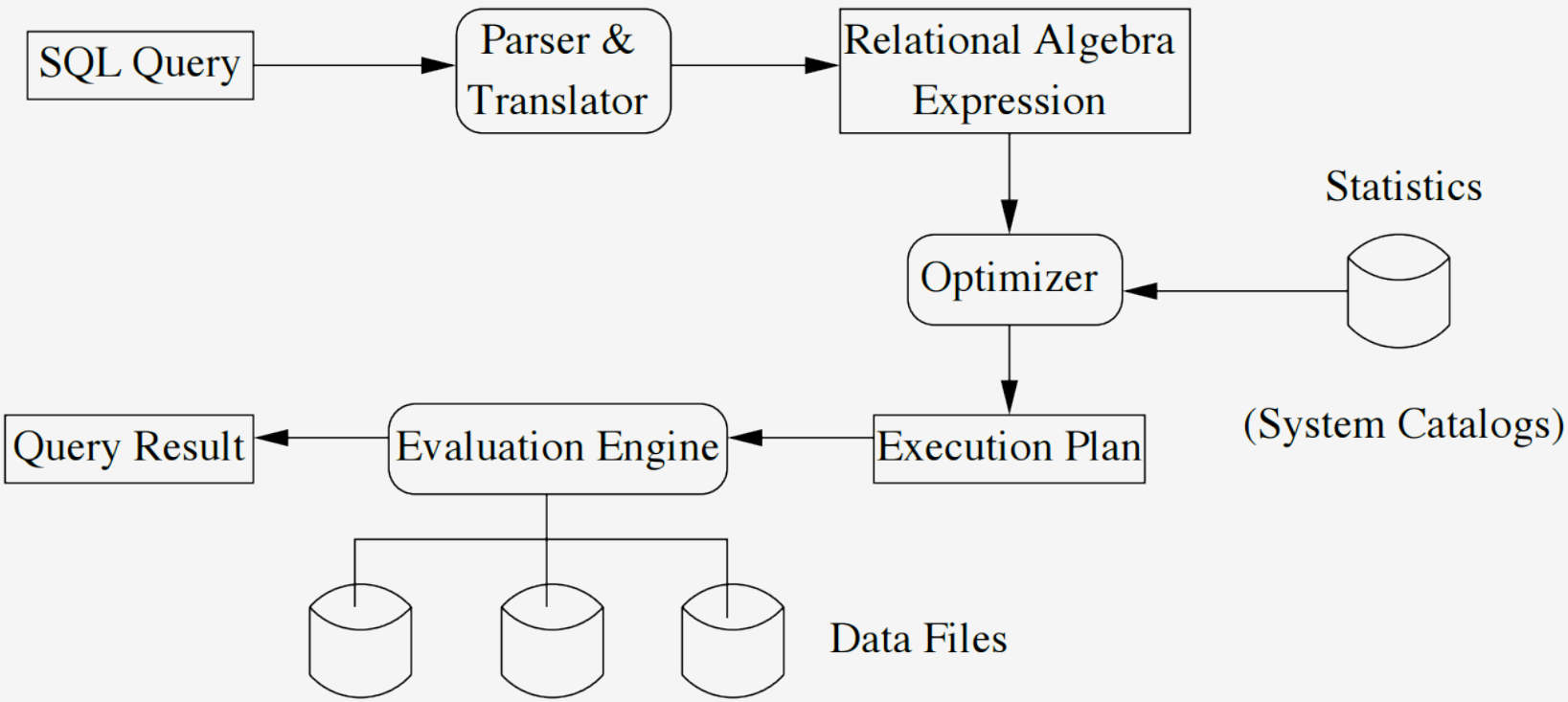


Imagen extraída de http://web.cs.ucdavis.edu/~green/courses/ecs165a-w11/8-query.pdf.

Información de catálogo

- Los SGBD's guardan distinto tipo de información de catálogo que es utilizada para estimar costos y optimizar las consultas. Nosotros utilizaremos la siguiente notación:
  - n(R): Cantidad de tuplas de la relación R.
  - B(R): Cantidad de bloques de almacenamiento que ocupa R.
  - V(A,R): Cantidad de valores distintos que adopta el atributo A en R (variabilidad).
  - F(R): Cantidad de tuplas de R que entran en un bloque (factor de bloque). $F(R) = \frac{n(R)}{B(R)} \Rightarrow B(R) = \lceil \frac{n(R)}{F(R)} \rceil$.
- También se almacena información sobre la cantidad de niveles que tienen los índices construídos, y la cantidad de bloques que ocupan sus hojas.
  - Height(I(A, R)): Altura del índice de búsqueda I por el atributo A de la relación R.
  - Length(I(A, R)): Cantidad de bloques que ocupan las hojas del índice I.

Información de catálogo

Mantenimiento

- Actualizar la información de catálogo en cada operación de ABM puede ser muy costoso.
- Los SGBD's suelen hacerlo con cierta periodicidad, o cuando están ociosos, o cuando el usuario lo indica explícitamente.

Plan de consulta y plan de ejecución

- La optimización de una consulta se inicia con una expresión en álgebra relacional.
- La expresión se optimiza a través de una heurística y utilizando reglas de equivalencia, obteniendo un plan de consulta.
- Luego, cada plan de consulta *lógico* se *materializa* para obtener un plan de ejecución en el que se indica el procedimiento *físico*: estructuras de datos a utilizar, índices, algoritmos a utilizar, etc.
- Para comparar distintos planes de ejecución, necesitamos estimar su costo. Algunos de los factores que inciden en la performance son:
  - El costo de acceso a disco (lectura o escritura)
  - El costo de procesamiento
  - El costo de uso de memoria
  - El costo de uso de red
- **Sólo estudiaremos los costos de acceso a disco**, por ser los de mayor envergadura en bases de datos relacionales centralizadas.

Indices

- Los índices son estructuras de búsqueda almacenadas y actualizadas por el SGBD, que agilizan la búsqueda de registros a partir del valor de un atributo o conjunto de atributos.
- Pueden implementarse con distintas estructuras de datos:
  - Árboles (binarios, B, B+, B*, ...)
  - Tablas de hash
- En los SGBD's los índices se clasifican en distintos tipos:
  - Cuando el índice se construye sobre el campo de ordenamiento clave de un archivo ordenado de registros, se denomina índice primario.
  - Cuando se construye sobre el campo de ordenamiento del archivo físico pero este no es clave, el índice se denomina índice de clustering.
  - Los índices que se construyen sobre campos que no son los campos de ordenamiento del archivo se denominan índices secundarios.
- Observación: Un archivo sólo puede tener un único índice primario o de clustering.

Indices

Definición en SQL

- SQL no dispone de una sentencia estándar para la definición de índices, aunque la mayoría de los SGBDs tiene una sentencia del tipo **CREATE INDEX** con la siguiente sintaxis:

```
CREATE [UNIQUE] INDEX nombreIndice
ON tabla (A1, ..., An);
```

- PostgreSQL
  - `ON tabla [ USING BTREE | GiST | HASH ]` $(A_1, \ldots)$
  - Para claves múltiples (más de un atributo) sólo se admite `BTREE`.
  - El comando `CLUSTER tabla [ USING nombreIndice ]` la reorganiza físicamente.
- DB2
  - `ON tabla (A1, A2, ..., An) [CLUSTER]`
  - La opción `CLUSTER` indica que el índice será de ordenamiento.

Indices

Definición en SQL

- SQLServer
  - `CREATE INDEX nombre ON tabla (A1, A2, ..., An)` (Índice secundario)
  - `CREATE CLUSTERED INDEX nombre ON tabla (A1, A2, ..., An)` (Índice de ordenamiento -primario o de clustering-)
  - `CREATE UNIQUE INDEX nombre ON tabla (A1, A2, ..., An)` (Índice de no-ordenamiento sobre atributo clave)
- MySQL
  - `CREATE [ UNIQUE | FULLTEXT | SPATIAL ] INDEX nombre [ USING BTREE | HASH ]`
  - `FULLTEXT` permite indexar tablas con datos de tipo `TEXT` para búsquedas con `MATCH ... AGAINST`.
  - `SPATIAL` permite indexar tipos de dato espaciales (`POINT` y `GEOMETRY`)

Selección

- Partimos de una selección básica del tipo $\sigma_{cond}(R)$, en donde *cond* es una condición atómica del tipo:
  - $A_i \odot c$, con $c \in dom(A_i)$

  , en donde $\odot$ es un operador de comparación.
- Existen distintas estrategias de búsqueda, según los recursos con los que contamos.
- Analizaremos distintas situaciones para la comparación por $=$.

Selección

File scan vs. Index scan

- Los métodos de file scan recorren el/los archivo/s en busca de los registros que cumplen con la condición.
- Búsqueda lineal: Consiste en explorar cada registro, analizando si se verifica la condición. Cuando no queda otro camino...

$$cost(S_1) = B(R)$$

- Los métodos de index scan utilizan un índice de búsqueda.
- Búsqueda con índice primario: Cuando A_i es un atributo clave del que se tiene un índice primario.
  - Sólo una tupla puede satisfacer la condición.
  - Si utilizamos un árbol de búsqueda:

$$cost(S_{3a}) = Height(I(A_i, R)) + 1$$

  - Si utilizamos una clave de hash:

$$cost(S_{3b}) = 1$$

Selección

File scan vs. Index scan

- **Búsqueda con índice de clustering:** Cuando A_i no es clave pero se tiene un índice de ordenamiento (clustering) por él.
  - Las tuplas se encuentran contiguas en los bloques, los cuales estarán disjuntos.

$$cost(S_5) = Height(I(A_i, R)) + \left\lceil \frac{n(R)}{V(A_i, R) \cdot F(R)} \right\rceil$$

- **Búsqueda con índice secundario:** Cuando A_i no tiene un índice de clustering, pero existe un índice secundario asociado a él.

$$cost(S_6) = Height(I(A_i, R)) + \left\lceil \frac{n(R)}{V(A_i, R)} \right\rceil$$

- Los cálculos que hemos visto pueden extenderse para otros tipos de comparación ($<, \leq, >, \geq, \neq$).

Selección

Selecciones complejas

- Si la selección involucra la conjunción de varias condiciones simples, pueden adoptarse distintas estrategias:
  - Si uno de los atributos tiene un índice asociado, se aplica primero esta condición, y luego se selecciona del resultado a aquellas tuplas que cumplen con las demas condiciones.
  - Si hay un índice compuesto que involucra a atributos de más de una condición, se utiliza este índice y luego se seleccionan las tuplas que cumplen los demás criterios.
  - Si hay índices simples para varios atributos, se utilizan los índices por separado y luego se intersecan los resultados.
- Si la selección involucra una disyunción de condiciones simples, debemos aplicar las mismas por separado y luego unir los resultados.
  - Si uno de los atributos no dispone de índice, hay que usar fuerza bruta.

Proyección

- Dividiremos el análisis de la proyección $\pi_X(R)$ en dos casos:
  1. X es superclave.
     - No es necesario eliminar duplicados. $cost(\pi_X(R)) = B(R)$.
  2. X no es superclave.
     - Debemos eliminar duplicados. Llamaremos $\hat{\pi}_X(R)$ a la proyección de *multisets* (i.e., sin eliminar duplicados). Podemos:
       1. Ordenar la tabla: Si $B(\hat{\pi}_X(R)) \leq M$ podemos ordenar en memoria. De lo contrario, el costo usando sort externo será:

$$cost(\pi_X(R)) = cost(Ord_M(R)) = 2 \cdot B(R) \cdot \lceil log_{M-1}(B(R)) \rceil - B(R)$$

Representa cant. de etapas del sort

       2. Utilizar una estructura de hash: Si $B(\hat{\pi}_X(R)) \leq M$ también podemos hacer el hashing en memoria, con costo $B(R)$. Utilizando hashing externo el costo es de $B(R) + 2 \cdot B(\hat{\pi}_X(R))$.
- Observación: Si la consulta SQL no incluye `DISTINCT`, entonces el resultado es un multiset, y el costo es siempre $B(R)$.

Unión e intersección

- Primero ordenamos las tablas R y S.
  - Si alguna no entra en memoria, usamos *sort externo*.
- Asumimos que no se devuelven repetidos (comportamiento *default* en SQL).
  - Se puede modificar sencillamente en caso de querer repetidos.
- Procesaremos ambas tablas ordenadas haciendo un *merge* que avanza por las filas r_i y s_j ordenadas de cada tabla.
- El costo es:

$$cost(R\ [\cup|\cap|-]\ S) = cost(Ord_M(R)) + cost(Ord_M(S)) + 2 \cdot B(R) + 2 \cdot B(S)$$

- Para la unión ($\cup$), debemos devolver todas las filas:
  1. Si $r_i = s_j$ devolver una de ellas y avanzar sobre ambas tablas hasta que cambien de valor.
  2. Sino, devolver la menor y avanzar sobre su tabla hasta que cambie de valor.
  3. Cuando alguna tabla se termine, devolver todo lo que queda de la otra, sin duplicados.

Unión e intersección

- Para la intersección ($\cap$), devolver las tuplas que están en ambas relaciones:
  1. Si $r_i \neq s_j$ avanzar sobre la tabla de la menor de ellas un lugar, sin devolver nada.
  2. Si $r_i = s_j$ devolver una de ellas y avanzar sobre ambas tablas hasta que cambien de valor.
  3. Cuando alguna tabla se termine, finalizar.
- Para la diferencia ($-$), devolver las tuplas que están en R pero no es S:
  1. Si $r_i > s_j$ avanzar sobre la tabla S hasta que cambie de valor.
  2. Si $r_i < s_j$ devolver r_i y avanzar sobre la tabla R hasta que cambie de valor.
  3. Si $r_i = s_j$ avanzar sobre ambas tablas hasta que cambien de valor.
  4. Cuando R se termine, finalizar. Cuando S se termine, devolver todo lo que queda de R, sin duplicados.

Junta

- La operación de junta es una de las más frecuentes y demandantes.
- Existen distintos métodos para calcularla:
  - Método de loops anidados por bloque
  - Método de único loop
  - Método sort-merge
  - Método de junta hash (variante GRACE)
- Observación: A continuación presentaremos los métodos, y sólo indicaremos el costo de lectura de datos y cálculo del resultado. Para calcular el costo de almacenamiento (que no siempre se realiza para las operaciones intermedias) es necesario estimar la cardinalidad del resultado.

Junta

Método de loops anidados por bloque

- Dadas dos relaciones R y S, el método de loops anidados por bloque consiste en tomar cada par de bloques de ambas relaciones, y comparar todas sus tuplas entre sí.
- Si por cada bloque de R se leen todos los bloques de S, el costo de procesar dicho bloque es $1 + B(S)$, y el total es de $B(R) \cdot (1 + B(S))$. Utilizando las tuplas de S como pivotes, el costo total sería de $B(S) \cdot (1 + B(R))$.
- El costo del método es entonces:

$$cost(R * S) = min(B(R) + B(R) \cdot B(S), B(S) + B(R) \cdot B(S))$$

- Esta estimación es un peor caso, suponiendo que sólo podemos tener un bloque de cada tabla simultáneamente en memoria ($M_i = 2$). Si pudiéramos cargar una de las tablas completa en memoria y sobrara un bloque, tendríamos el mejor caso:

$$cost(R * S) = B(R) + B(S)$$

Junta

Método de único loop

- Si el atributo de junta tiene un índice asociado en R, por ejemplo, podemos recorrer las tuplas de S y para cada una de ellas buscar en el índice la/s tupla/s de R en que el atributo coincide.
- Si el índice es primario, el costo será:

$$cost(R * S) = B(S) + n(S) \cdot (Height(I(A, R)) + 1)$$

- Si el índice es de clustering, puede haber más de una coincidencia:

$$cost(R * S) = B(S) + n(S) \cdot \left(Height(I(A, R)) + \left\lceil \frac{n(R)}{V(A, R) \cdot F(R)} \right\rceil \right)$$

- Si el índice es secundario:

$$cost(R * S) = B(S) + n(S) \cdot \left(Height(I(A, R)) + \left\lceil \frac{n(R)}{V(A, R)} \right\rceil \right)$$

Junta

Ejemplo

Ejemplo

Estimar el costo de la junta
$Clientes(nroCliente, nombre) * Ordenes(nroCliente, nroOrden)$,
utilizando un índice de clustering por nroCliente en la tabla Ordenes.

Clientes	Ordenes
n(Clientes) = 5000	n(Ordenes)=10000
F(Clientes) = 20	F(Ordenes)=25
	Height(I(nroCLiente, Ordenes)) = 4
V(nroCliente, Clientes) = 5000	V(nroCliente, Ordenes)=2500

Respuesta

$cost(Clientes * Ordenes) = \frac{5000}{20} + 5000 \cdot (4 + 1) = 25250$

Junta

Método de sort-merge

- Consiste en ordenar los archivos de cada tabla por el/los atributo/s de junta.
- Si entran en memoria, el ordenamiento puede hacerse con *quicksort*, y el costo de acceso a disco es sólo $B(R) + B(S)$.
- Si los archivos no caben en memoria debe utilizarse un algoritmo de *s*ort externo. El costo de ordenar R y volverlo a guardar en disco ordenado es de aproximadamente $2 \cdot B(R) \cdot \lceil log_{M-1}(B(R)) \rceil$ $.\rightarrow$ ¡El log estima la cantidad de etapas del sort externo!
- Una vez ordenados, se hace un *merge* de ambos archivos que sólo selecciona aquellos pares de tuplas en que coinciden los atributos de junta. El *merge* recorre una única vez cada archivo, con un costo de $B(R) + B(S)$.
- El costo total es entonces:

$$cost(R*S) = B(R)+B(S)+2\cdot B(R)\cdot\lceil log_{M-1}(B(R))\rceil+2\cdot B(S)\cdot\lceil log_{M-1}(B(S))\rceil$$

Junta

Método de junta hash (variante GRACE)

- La idea de este método es particionar las tablas R y S en m grupos utilizando una función de hash $h(X)$ aplicada sobre los atributos de junta X.
- Atención: Que dos tuplas $r \in R$ y $s \in S$ cumplan que $h(r.X) = h(s.X)$ no implica que $r.X = s.X$!
- Costo del particionado: $2 \cdot (B(R) + B(S))$
  - Porque es necesario leer todos los bloques y reescribir sus datos en otro orden.
- Luego, cada par de grupos R_i y S_i se combina verificando si se cumple la condición de junta con un enfoque de fuerza bruta.
  - Observación: No es necesario combinar R_i y S_j para $i \neq j$
  - ¿Por qué? $r.X = s.X \rightarrow h(r.X) = h(s.X)$

Junta

Método de junta hash (variante GRACE)

- Hipótesis: m fue escogido de manera que para cada par de grupos (R_i, S_i) al menos uno entre en memoria, y sobre un bloque de memoria para hacer desfilar al otro grupo.
- Costo de la combinación de R_i y S_i: $B(R_i) + B(S_i)$. Esto se deduce de:
  - Observación 1: $F(R_i) = F(R)$ y $F(S_i) = F(S)$
  - Observación 2: $\sum_{i=1}^{m} n(R_i) = n(R)$ y $\sum_{i=1}^{m} n(S_i) = n(S)$
- El costo total es:

$$cost(R * S) = 3 \cdot (B(R) + B(S))$$

Pipelining

- En muchos casos, el resultado de un operador puede ser procesado por el operador siguiente en forma parcial (es decir sin necesidad de que el operador anterior haya terminado de generar todas las tuplas).
- Esta estrategia se denomina pipelining, y los SGBD suelen utilizarla en los planes de ejecución siempre que sea posible.
- Al calcular el costo de dos operadores anidados $O_2(O_1(R))$ debemos considerar que en caso de utilizar pipelining no será necesario tener todos los bloques de la salida de O_1 para comenzar a calcular O_2. En particular, no tendremos que materializar toda la salida de O_1 por falta de espacio en memoria.

Ejemplo

Ejemplo: Obra social

La base de datos de una obra social cuenta con las siguientes tablas:

- Afiliaciones(nro_socio, cod_plan)
- Planes(cod_plan, servicio)

Existe también un índice de clustering por "cod_plan" para Planes. Se requiere hacer la junta natural de ambas tablas.

Afiliaciones	Planes
n(Afiliaciones) = 3000000	n(Planes)=10000
B(Afiliaciones) = 300000	B(Planes)=500
	Height(I(cod_plan, Planes)) = 2
	V(cod_plan, Planes)=40

Indique cuál de los siguientes métodos de junta es más conveniente:

- Junta hash GRACE
- Único loop con índice

Ejemplo

Solución

Junta hash GRACE

$cost(Afiliaciones * Planes) = 3 \cdot (B(Afiliaciones) + B(Planes)) \approx \mathbf{900000}$

Único loop con índice

$cost(Afiliaciones * Planes) = B(Afiliaciones) + n(Afiliaciones) \cdot \left(Height(I(cod\_plan, Planes)) + \frac{n(Planes)}{V(plan, Planes) \cdot F(Planes)} \right)$

$cost(Afiliaciones * Planes) = 300000 + 3000000 \cdot \left(2 + \frac{10000}{40 \cdot 20}\right) \approx$ **45000000**

$\rightarrow$ En este caso es más eficiente la junta hash GRACE.

Estimación de la cardinalidad

Concepto

- Como parte de la estimación del costo de una consulta, es necesario a veces estimar el tamaño de las relaciones intermedias (la cardinalidad) antes de calcularlas.
- Se espera que una estimación de cardinalidad cumpla con los siguientes requisitos:
  - Sea precisa.
  - Sea fácil de calcular.
  - No dependa de la forma en que esa relación intermedia se calculó.
- Veremos reglas de estimación de la cardinalidad a través de ejemplos para los siguientes operadores:
  - Proyección
  - Selección
  - Junta

Estimación de la cardinalidad

Proyección

- Ejemplo: $Persona(DNI, nombre, f\_nacimiento, género)$
  - 40 millones de tuplas
  - El DNI es un entero de 4 bytes
  - El nombre es un string variable de tamaño promedio 15 bytes
  - La fecha de nacimiento es un timestamp de 4 bytes
  - El género es un caracter

  Supongamos que los bloques son de 1024 bytes con un header de 24 bytes.
- La estimación de la cantidad de bloques que ocupa la relación es:

$$B(Persona) = \frac{40 \cdot 10^6 \cdot (4 + 15 + 4 + 1)}{10^3} = 960000$$

- Ahora queremos estimar $B(\pi_{DNI}(Persona))$. La cantidad de tuplas no se modifica, por lo tanto:

$$B(\pi_{DNI}(Persona)) = \frac{40 \cdot 10^6 \cdot 4}{10^3} = 160000$$

Estimación de la cardinalidad

Selección

- La selección reduce el número de tuplas en el resultado, aunque mantiene el tamaño de cada tupla.
- Para estimar el tamaño de una selección de la forma $\sigma_{A_i=c}(R)$, utilizaremos la variabilidad de A_i en R ($V(A_i, R)$), que es la cantidad de valores distintos que puede tomar el atributo A_i en dicha relación.
- Realizaremos la siguiente estimación:

$$n(\sigma_{A_i=c}(R)) = \frac{n(R)}{V(A_i, R)}$$

- La fracción $\frac{1}{V(A_i,R)}$ se denomina selectividad de A_i en R.

Estimación de la cardinalidad

Selección

- Ejemplo: $Persona(DNI, nombre, f\_nacimiento, gnero)$.
- Para estimar $n(\sigma_{genero='F'}(Persona))$, consideremos que hay dos géneros posibles. Luego:

$$n(\sigma_{genero='F'}(Persona)) = \frac{n(Persona)}{V(genero, Persona)} = \frac{40 \cdot 10^6}{2} = 20 \cdot 10^6$$

$$B(\sigma_{genero='F'}(Persona)) = \frac{n(\sigma_{genero='F'}(Persona)) \cdot (4 + 15 + 4 + 1)}{10^3} = 480000$$

- Dificultades:
  - No nos permite estimar selecciones con otros operadores ($\leq, \geq, \neq$).
  - La estimación asume que el valor c se toma al azar. Si no es así, entonces es sesgada.
- Un método más avanzado consiste en utilizar un histograma para la distribución de A_i.

Estimación de la cardinalidad

Selección - Estimación con histograma

- El histograma nos resume la distribución de los valores que toma un atributo en una instancia de relación dada.
- Es útil cuando un atributo toma valores discretos.
- Ejemplo: Película(id, nombre, género)
  - n(Película) = 728
  - V(género, Película) = 9

	drama	comedia	suspenso	otros
Película.género	150	140	128	310

- El histograma nos dice que $n(\sigma_{genero='comedia'}(Peliculas)) = 140$
- ¿Podemos estimar mejor $n(\sigma_{genero='terror'}(Peliculas))$ utilizando el histograma?

$$n(\sigma_{genero='terror'}(Peliculas)) = \frac{n(Pelicula) - (418)}{V(genero, Pelicula) - 3} = \frac{310}{6} = 52$$

Estimación de la cardinalidad

Junta

- Consideremos la junta de $R(A, B)$ y $S(B, C)$.
- En principio, $0 \leq n(R * S) \leq n(R) \cdot n(S)$, dependiendo de como estén distribuídos los valores de B en una y otra relación.
- Dadas las variabilidades $V(B, R)$ y $V(B, S)$, asumiremos que los valores de B en la relación con menor variabilidad están incluídos dentro de los valores de B en la otra relación.
  - En el caso en que el atributo de junta es clave primaria en una relación y clave foránea en la otra, esta hipótesis se cumple.
- Supongamos que $V(R, B) \geq V(S, B)$ y tomemos una tupla en $t_R \in R$ y una tupla en $t_S \in S$. Sabemos que $t_S.B$ está incluído dentro de los valores que toma B en R. Luego, $P(t_S.B = t_R.B) = \frac{1}{V(R,B)}$.
- De manera análoga, si $V(R, B) \leq V(S, B)$ entonces que $t_R.B$ está incluído dentro de los valores que toma B en S. Luego, $P(t_R.B = t_S.B) = \frac{1}{V(S,B)}$.

Estimación de la cardinalidad

Junta

- En general, $P(t_R.B = t_S.B) = \frac{1}{max(V(R,B),V(S,B))}$, que es la selectividad de la junta (js). Luego:

$$n(R * S) = js \cdot n(R) \cdot n(S) = \frac{n(R) \cdot n(S)}{max(V(R,B), V(S,B))}$$

Ejemplo

Estimar la cardinalidad de $R(A,B) * S(B,C) * T(C,D)$, siendo:

R(A,B)	S(B,C)	T(C,D)
n(R) = 1000	n(S)=2000	n(T)=5000
V(R,B) = 20	V(S,B) = 50	
	V(S,C)=100	V(T,C)=500

Respuesta

$n(R * S * T) = 400000$

Estimación de la cardinalidad

Junta

- Para estimar el factor de bloque del resultado, asumiremos que si una tupla de R ocupa $\frac{1}{F(R)}$ bloques y una tupla de S ocupa $\frac{1}{F(S)}$ bloques, entonces una tupla del resultado ocupa menos de $\frac{1}{F(R)} + \frac{1}{F(S)}$, y por lo tanto el factor de bloque es al menos:

$$F(R * S) = \left(\frac{1}{F(R)} + \frac{1}{F(S)} \right)^{-1}$$

- La fórmula subestima el factor de bloque, porque no tiene en cuenta que los atributos de junta se repiten en ambas tablas.
- La cantidad de bloques será (sobreestimación):

$$B(R * S) = \frac{js \cdot n(R) \cdot n(S)}{F(R * S)} = js \cdot B(R) \cdot B(S) \cdot (F(R) + F(S))$$

Estimación de la cardinalidad

Junta - Estimación con histograma

- Ejemplo:
  - $R(A, B)$, con $V(B, R) = 18$
  - $S(B, C)$, con $V(B, S) = 15$
- Supongamos que disponemos de un histograma que nos muestra los k valores más frecuentes de B en cada una de las relaciones. En este caso, $k = 5$.

	4	12	14	20	22	30	otros
R.B	200		320	120	150	65	550
S.B	150	100		180	210	85	410

- Para cada valor x_i del que conocemos $f_R(x_i)$ [1] y $f_S(x_i)$, sabemos que la cantidad de tuplas en el resultado será: $f_R(x_i) \cdot f_S(x_i)$.

	4	12	14	20	22	30	otros
R.B	200		320	120	150	65	550
S.B	150	100		180	210	85	410
$R * S$	30000			21600	31500	5525	

[1] $f_R(x_i) = n(\sigma_{B=x_i}(R))$.

Estimación de la cardinalidad

Junta - Estimación con histograma

- Para aquellos x_i de los que sólo conocemos $f_R(x_i)$ ó $f_S(x_i)$, estimaremos el faltante a partir de la columna "otros" y de la variabilidad.
- Por ejemplo, si conocemos sólo $f_R(x_i)$, entonces:

$$f_S(x_i) = \frac{f_S(otros)}{V(B,S) - k}$$

	4	12	14	20	22	30	otros
R.B	200	43	320	120	150	65	550 507
S.B	150	100	41	180	210	85	410 369
$R * S$	30000	4300	13120	21600	31500	5525	

- Actualizamos también las frecuencias de "otros", y el valor de k, que se convierte en $k' = 6$.

Estimación de la cardinalidad

Junta - Estimación con histograma

- Finalmente estimamos las tuplas correspondientes a "otros" en el resultado utilizando la estimación simple (equiprobable):

$$f_{R*S}(otros) = \frac{f_R(otros) \cdot f_S(otros)}{max(V(R,B) - k', V(S,B) - k')}$$

	4	12	14	20	22	30	otros
R.B	200	43	320	120	150	65	550 / 507
S.B	150	100	41	180	210	85	410 / 369
$R * S$	30000	4300	13120	21600	31500	5525	15590

- La estimación final es:

$$n(R * S) = \sum_i f_{R*S}(x_i) = 121635$$

- La simple estimación (sin histograma) nos hubiera dado como resultado $n(R * S) = 88594$ (verificar).

Reglas de equivalencia

- Selección
  - $\sigma_{c_1 \wedge c_2 \wedge \ldots \wedge c_n}(R) = \sigma_{c_1}(\sigma_{c_2}(\ldots(\sigma_{c_n}(R))\ldots))$ (Cascada)
  - $\sigma_{c_1 \vee c_2 \vee \ldots \vee c_n}(R) = \sigma_{c_1}(R) \cup \sigma_{c_2}(R) \cup \ldots \cup \sigma_{c_n}(R)$
  - $\sigma_{c_1}(\sigma_{c_2}(R)) = \sigma_{c_2}(\sigma_{c_1}(R))$ (Conmutatividad)
- Proyección
  - $\pi_{X_1}(\pi_{X_2}(\ldots(\pi_{X_n}(R))\ldots)) = \pi_{X_1}(R)$ (Cascada)
  - $\pi_X(\sigma_{cond}(R)) = \sigma_{cond}(\pi_X(R))$ (Conmutatividad con σ)
- Producto cartesiano y junta
  - $R \times S = S \times R$ (Conmutatividad)
  - $R * S = S * R$
  - $(R \times S) \times T = R \times (S \times T)$ (Asociatividad)
  - $(R * S) * T = R * (S * T)$

Reglas de equivalencia

- Operaciones de conjuntos
  - $R \cup S = S \cup R$ (Conmutatividad)
  - $R \cap S = S \cap R$
  - $(R \cup S) \cup T = R \cup (S \cup T)$ (Asociatividad)
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- Otras mixtas
  - Dado $\sigma_c(R * S)$, si c puede escribirse como $c_R \wedge c_S$, con c_R y c_S involucrando sólo atributos de R y de S respectivamente, entonces:

$$\sigma_c(R * S) = \sigma_{c_R}(R) * \sigma_{c_S}(S)$$

    (Distribución de la selección en la junta)
  - Dado $\pi_X(R * S)$, si todos los atributos de junta están incluídos en X, entonces llamando X_R y X_S a los atributos de R y S que están en X respectivamente:

$$\pi_X(R * S) = \pi_{X_R}(R) * \pi_{X_S}(S)$$

    (Distribución de la proyección en la junta)

Heurísticas de optimización

Reglas generales

- La aplicación de las reglas de equivalencia a una expresión algebraica para obtener otra de menor costo se conoce como optimización algebraica.
- Las siguientes son algunas reglas generales utilizadas para optimizar algebraicamente una consulta:
  1. Realizar las selecciones lo más temprano posible.
  2. Remplazar productos cartesianos por juntas siempre que sea posible.
  3. Proyectar para descartar los atributos no utilizados lo antes posible.
     - Entre la selección y la proyección, priorizar la selección.
  4. En caso de que hayan varias juntas, realizar aquella más restrictiva primero.
     - Optar por árboles *left-deep* ó *right-deep* para acotar las posibilidades.

Heurísticas de optimización

Ejemplo: 2010 World Cup Dataset

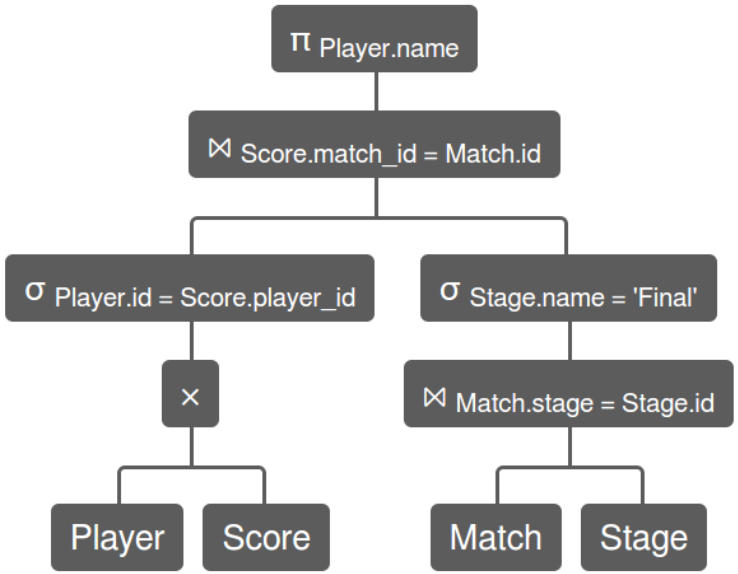
- Esquema de base de datos relacional:
  - Continent(id, name)
    (1, 'Africa')
  - NationalTeam(id, name, group, short_name, continent)
    (1, 'South Africa', 'A', 'RSA', 0)
  - Match(id, home, away, match_datetime_gmt, stage)
    (1, 1, 2, '2010-06-11 14:00:00', 1)
  - Player(id, name, birth_date, height, playing_position, local_club, national_team, national_team_tshirt)
    (53, 'Edinson Cavani', '1987-02-14', 188, 'FW', 'Palermo [ITA]', 3, 7)
  - Score(id, match_id, team_id, player_id, minute, score_type)
    (1, 1, 1, 8, '55', 1)
  - Stage(id, name)
    (3, 'Quarter-finals')
  - Asumiremos que "name" es siempre clave candidata.

Heurísticas de optimización

Ejemplo: 2010 World Cup Dataset

Ejemplo: 2010 World Cup Dataset

Para calcular el listado de jugadores que convirtieron algún gol en la final del mundial, un motor de bases de datos construye el siguiente plan de consulta:

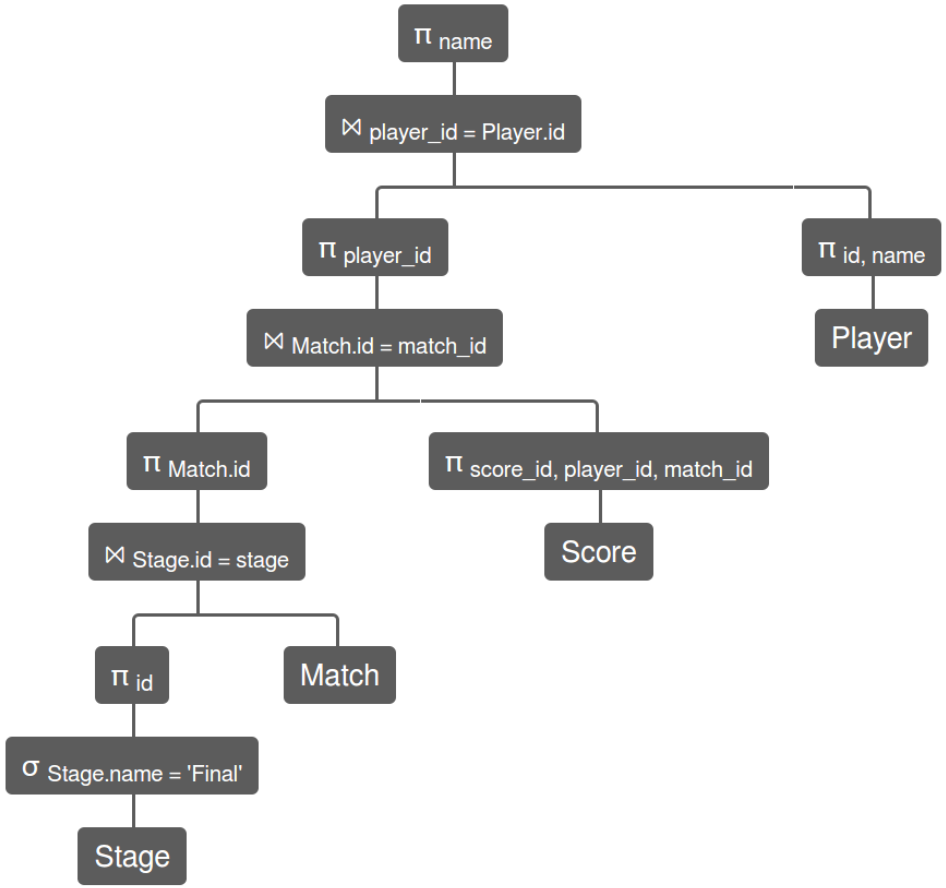


Aplique las heurísticas estudiadas para optimizar el plan.

Heurísticas de optimización

Ejemplo: 2010 World Cup Dataset

Solución



Bibliografía

[ELM16] Fundamentals of Database Systems, 7th Edition.
R. Elmasri, S. Navathe, 2016.
Capítulo 17, Capítulo 18

[SILB19] Database System Concepts, 7th Edition.
A. Silberschatz, H. Korth, S. Sudarshan, 2019.
Capítulo 15, Capítulo 16

[GM09] Database Systems, The Complete Book, 2nd Edition.
H. García-Molina, J. Ullman, J. Widom, 2009.
Capítulo 15, 16

[CONN15] Database Systems, a Practical Approach to Design, Implementation and Management, 6th Edition.
T. Connolly, C. Begg, 2015.
Capítulo 23