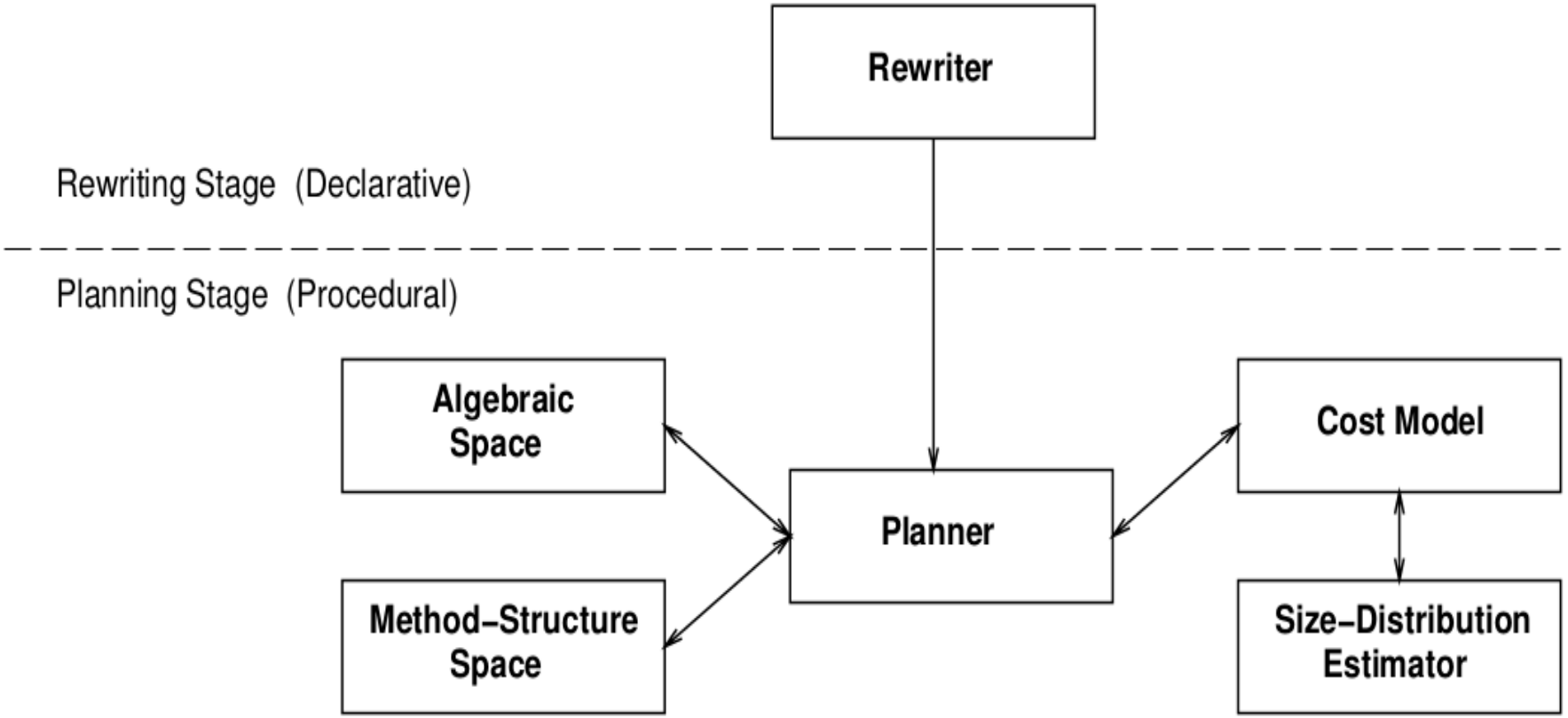
Universidad del Valle

Optimización de Consultas

Introducción

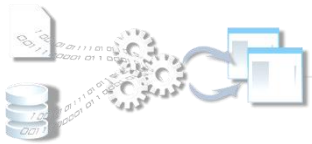
Transformar consulta de alto nivel en otra equivalente más eficiente

Optimizador: Arquitectura



Introducción

- Tópico central en BD
- Aspectos a considerar
  - Técnicas almacenamiento e indexación
  - Tamaños páginas
  - Protocolos paginación
  - Propiedades datos almacenados
  - Propiedades estadísticas
  - Implementación específica operadores



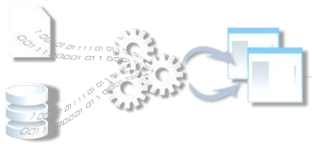
Optimización

Niveles aplicación

- Físico: Supuesto BD memoria secundaria
- Lógico: equivalencias algebráicas, reescritura de consultas

Optimización: Niveles

- **Físico**
  - Técnicas acceso, propiedades estadísticas datos, gestión de buffers
  - Gestión memoria secundaria (disco)
  - Copias porciones de datos buffers
  - Almacenamiento intermedio
  - Intercambio páginas disco, memoria,
  - Indices (B-trees, Inidices Hash)
  - **Acciones**
  - Generar flujos (streams) tuplas
  - Manipulación streams
  - Combinación



Optimización: Niveles

- **Lógico**
  - Implementación Operadores Algebráicos
  - Arboles de Consulta
  - Re-escritura consultas
  - Selección Planes

Plan Evaluación Consultas

Especificación de secuencia de operadores para calcular una consulta

- Consulta
- Estado de BD almacenada
- Colección de índices

Implementación operadores: Selección

- *scan* directo relación argumento (tiempo lineal)
- Indices B-Tree o tablas de Hash: reducir tiempo de búsqueda
  - Salida Simple (2 o 3 pg)
  - Salida grande
- relación entrada clusterizada (tuplas con valor de atributo dado en la misma página o páginas contiguas)

Implementación operadores: Proyección

Más compleja, dos operaciones:

1. tuple rewriting
2. duplicate elimination

- Llevar tuplas a memoria principal
- Reescritura con valores requeridos
- Puede producir duplicados
- Remover duplicados: ordenar listas de tuplas y remover duplicados (n log(n))
- Lenguajes permiten duplicados en resultado intermedio y final
- Comando explícito para eliminarlos

Implementación de operadores: Equijoin

- Más costoso, incluye 2 relaciones
- Implementación ingenua $\bowtie_f$, (orden n_1x n_2, tamaños de relaciones de entrada I_1, $I_{2)}$

J := ϕ

for each u **in** I_1

for each v **in** I_2

if u and v are joinable **then J := J** $\cup$ **{**$u \bowtie_f v$**}**

Implementación de operadores: Equijoin

- Sort-merge mejora desempeño
- Entradas ordenadas con base en atributos del join
- Scan simultáneo de ambas relaciones
- Orden $\max(n_1 \log(n_1) + n_2 \log(n_2)$, tamaño de salida$)$
- Otra estrategia: Reemplazar inner loop por medio de la recuperación de índices para tuplas en I_2 que se emparejan con los de I_1.
- Asumiendo que un número pequeño de tuplas de I_2, emparejan con tupla dadas en I_1, el join se calcula en tiempo proporcional al tamaño de I_1.

Árboles y re-escritura de consultas

- Planes de evaluación alternativos
- usan reglas transformación local (re-escritura)
- Tipos de transformaciones en optimización
  - Desde lenguajes de alto nivel (álgebra) a lenguaje físico
  - Dentro del mismo lenguaje, implementación equivalentes

Optimizacion:

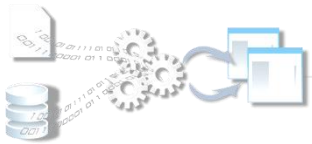
- Desde el punto de vista lógico
- Reglas Algebraicas

Reglas algebráicas: Ejemplo

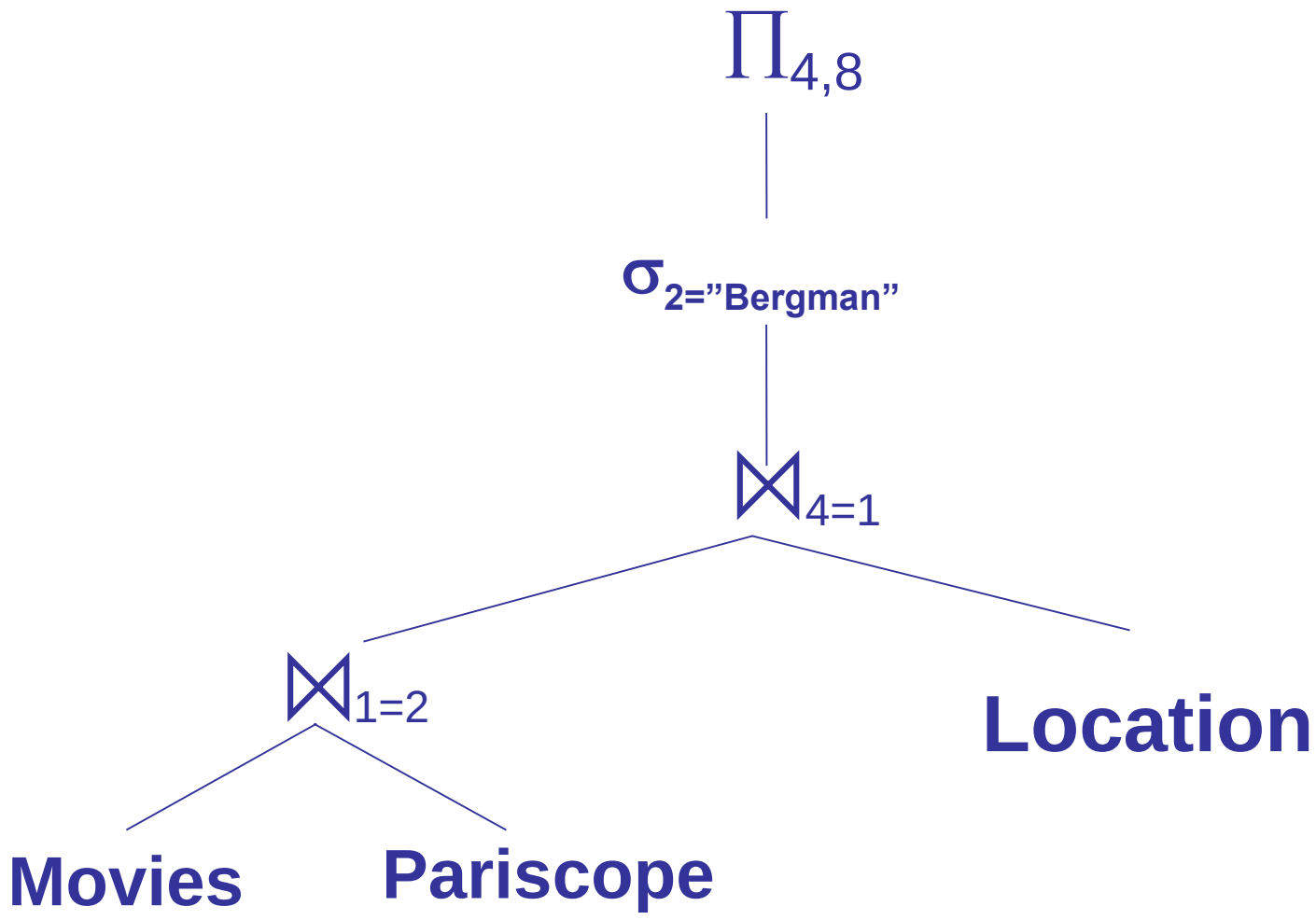
$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, \text{“Bergman”}, x_{ac}),$
$Pariscope(x_{th}, x_{ti}, x_s),$
$Location(x_{th}, x_{ad}, x_p).$

Álgebra SPC generalizada (selección conjuntiva positiva y equi-join)

$q_1 = \Pi_{4,8}\ \sigma_{2=\text{“Bergman”}}\ ((Movies \bowtie_{1=2} Pariscope) \bowtie_{4=1} Location)$



Árbol de Consulta

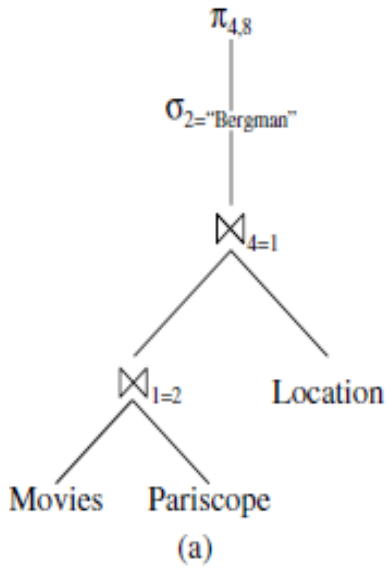


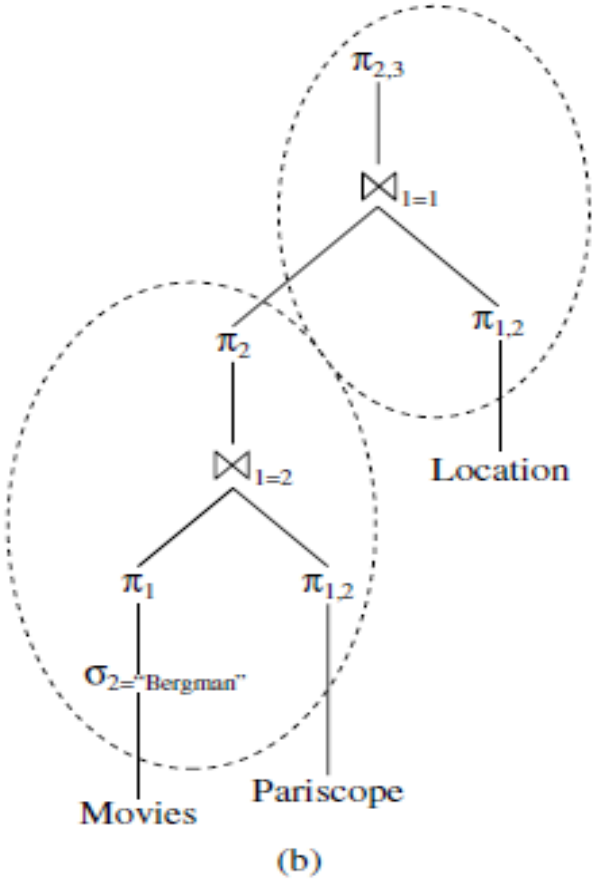
Árbol de Consulta

$q_1 = \Pi_{4,8}\ \sigma_{2=\text{“Bergman”}}\ ((\text{Movies} \bowtie_{1=2} \text{Pariscope}) \bowtie_{4=1} \text{Location})$

Vs

$q2 = \Pi_{4,8}\ ((\sigma_{2=\text{“Bergman”}}(\text{Movies})) \bowtie_{1=2} \text{Pariscope}) \bowtie_{4=1} \text{Location})$





q1

q2

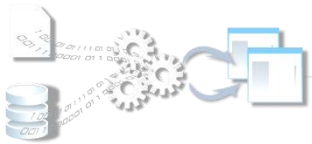
Cómo estimar costo

Movies: 10000 tuplas (≈ 5 tuplas por película)

Pariscope: cerca de 200 tuplas

Location: cerca de 100 tuplas

Supuesto: relación con 50 tuplas/página sin índices

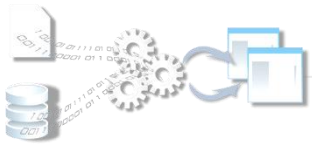


Costo consulta: ingenuo

Resultados intermedios por cada nodo interno en el query tree de q_1

- Join Movies y Pariscope ≈ 200 x 5 = 1000 tuplas

  (≈ 40 páginas; casi el doble de atributos de salida)
- Join con Location: 1000 tuplas (18 por página), total 55 páginas

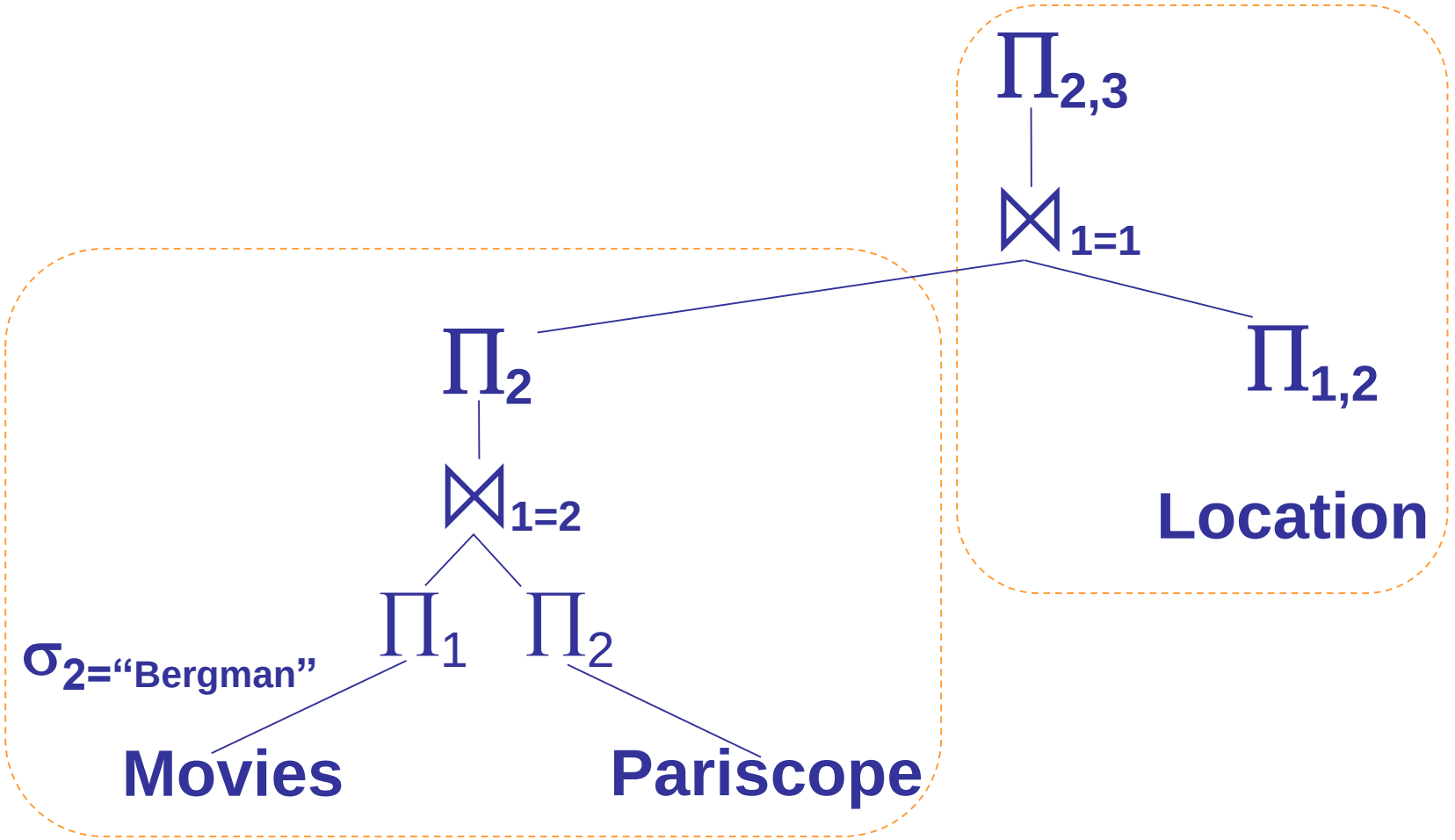


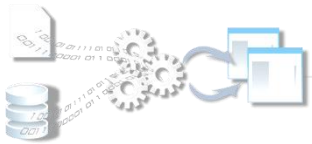
Costos de evaluación

Asumiendo 4 películas de “Bergman” en 1 o 2 teatros se tendría un resultado final de 6 tuplas:

- ≈ 206 páginas para leer la entrada (10300/50)
- 95 para resultados intermedios
- páginas requeridas para ejecutar el join

Query Tree





Costo consulta

- q_2 producida de q_1 empujando selecciones y proyecciones tan abajo del árbol como sea posible
- Reduce el tamaño de resultados intermedios
- Asume que cerca de 20 películas de “Bergman” hay en Movies

Costos de consulta

- Selección sobre Movies: ≈100 tuplas

  (5 tuplas/película y 20 películas de “Bergman”)
- Proyección ajusta a una página
- Unión con Pariscope produce ≈ 6 tuplas
- Solo una página se necesita para resultados intermedios

Reglas algebráicas de consulta

Para cada consulta SPC (generalizada) ***q*** *hay una consulta SPC generalizada* ***q'*** *en forma normal tal que* ***q*** ≡ ***q'***

Reglas algebráicas de consulta

$$\sigma_F(\sigma_{F'}(q)) \leftrightarrow \sigma_{F \wedge F'}(q)$$

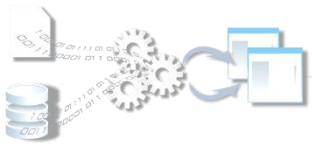
$$\Pi_j(\Pi_k(q)) \leftrightarrow \Pi_j(q)$$

$$\sigma_F(\Pi_j(q)) \leftrightarrow \Pi_j(\sigma_{F'}(q))$$

$$q_1 \bowtie q_2 \leftrightarrow q_2 \bowtie q_1$$

$$\sigma_F(q_1 \bowtie_G q_2) \rightarrow \sigma_F(q_1) \bowtie_G q_2$$

$$\sigma_F(q_1 \bowtie_G q_2) \rightarrow q_1 \bowtie_G \sigma_{F'}(q_2)$$



Reglas algebráicas de consulta

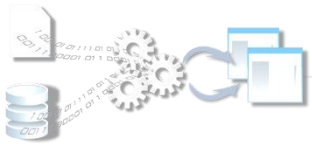
$$\sigma_F(q_1 \bowtie_G q_2) \rightarrow q_1 \bowtie_{G'} q_2$$

$$\Pi_i(q_1 \bowtie_G q_2) \rightarrow \Pi_j(q_1) \bowtie_{G'} q_2$$

$$\Pi_i(q_1 \bowtie_G q_2) \rightarrow q_1 \bowtie_{G'} \Pi_k(q_2)$$

Generación y selección plan de evaluación

- Planes alternativos
- Estimar costo de ejecución
- Seleccionar el más barato
- Estimación costo
  - Tamaño de relaciones
  - Propiedad de índices
  - Factor de selectividad de join y selección

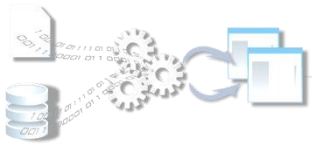


Generación y selección plan de evaluación

SELECT-FROM-WHERE

SPC $\prod_i(\sigma_F(R_1 x \ldots x R_n))$

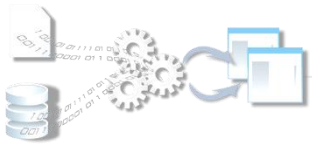
- Join relaciones en FROM
- Usar condición de join del WHERE
- Proyectar sobre columnas en SELECT



Generación Planes

$$\sigma_F(q_1 \bowtie_G q_2) \rightarrow \sigma_F(q_1) \bowtie_G q_2$$

- Empujar selecciones(deseable), no siempre es buena
- Muchos factores influencian
- Uso técnica analítica imposible encontrar plan óptimo

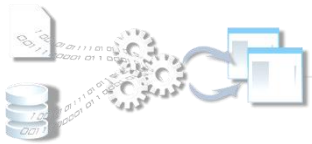


Generación Planes

$$\sigma_F(q_1 \bowtie_G q_2) \rightarrow \sigma_F(q_1) \bowtie_G q_2$$

En la práctica

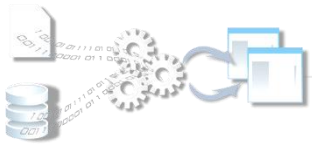
- Generación de un número de planes de evaluación alternativos
- Estimación costo de ejecución plan
- Escogencia plan de menor costo



Optimización

Problema de búsqueda difícil

- Espacio de búsqueda (Planes)
- Técnica de estimación de costos (recursos ejecución)
- Algoritmo enumeración



Optimización de consultas

CommercialOptimizers: Current relational DBMS optimizers are very complex pieces of software with many closely guarded details, and they typically represent 40 to 50 man-years of development effort!