Bases de Datos I

Procesamiento y Optimización de consultas

Facultad de Ciencias Exactas
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

INGENIERÍA DE SISTEMAS

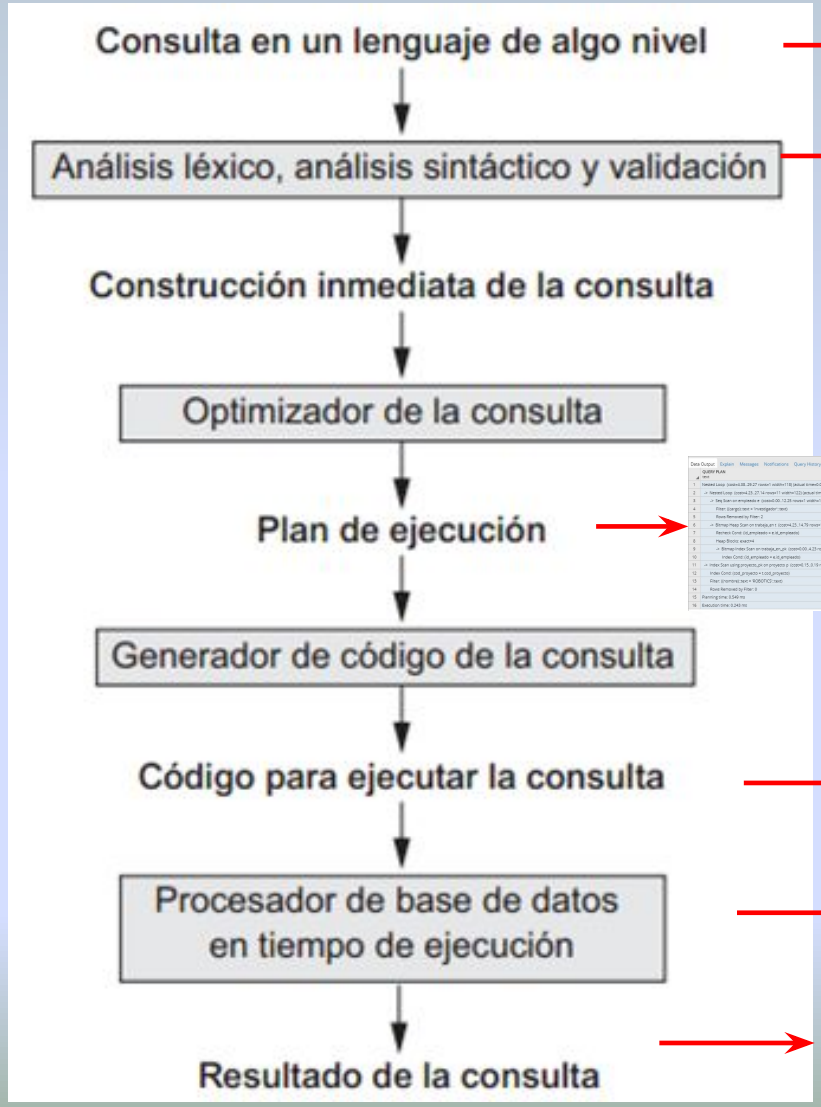
Cursada 2019

Introducción

Implica:

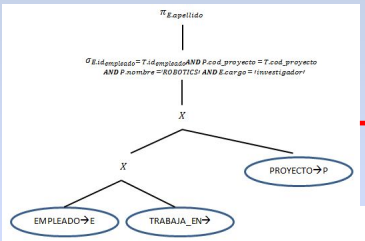
- Comprender las **tareas de procesamiento y optimización de consultas** realizadas por un SGBD.
- Conocer las **reglas heurísticas y de transformación** de expresiones del álgebra relacional y cómo aplicarlas para mejorar la eficiencia de una consulta.
- Conocer las diferentes **estrategias de implementación** de **operaciones relacionales**, en particular la de ensamble, y cómo evaluar el costo estimado de cada estrategia.
- Identificar la **información estadística** de la base utilizada para estimar el costo de ejecución de las operaciones del álgebra relacional.

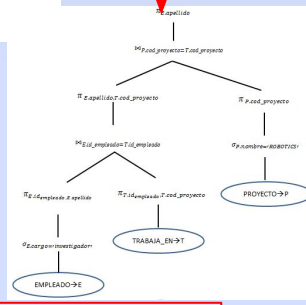
Introducción



SELECT…

Compilador





Modo compilado o interpretado

Tiempo de ejecución

Si hay error de ejecución presenta el mensaje correspondiente

Elmasri, R.; Navathe, S.B.: **Fundamentos de Sistemas de Bases de Datos**. 5ª Edición. Addison-Wesley. **(Cap. 15)**

Procesamiento y Optimización de Consultas

Objetivos del procesamiento de consultas

- Transformar una consulta SQL en una **estrategia de ejecución eficaz**, expresada en un lenguaje de bajo nivel.
- Ejecutar dicha estrategia para recuperar los datos requeridos.

Optimización de Consultas: Elección de una estrategia de ejecución eficaz para procesar. Puede haber muchas transformaciones equivalentes para una misma consulta.

Objetivo de la optimización de consultas

- Elegir la **estrategia de ejecución** que **minimiza el uso de los recursos**

– En general, **no se garantiza que la estrategia elegida** por el SGBD **sea** la **óptima**, pero seguro que será una **estrategia razonablemente eficiente**

Procesamiento y Optimización de Consultas

Ventajas de la **optimización automática**

- El usuario no se preocupa de **cómo** formular la consulta
- El Módulo **Optimizador trabaja "mejor"** (a veces!!!) que el DBA o programador, pues:
  - Dispone de **información estadística** en el diccionario de datos del SGBD
    - Estimar mejor la eficiencia de cada posible estrategia... y así (con mayor probabilidad) elegirá la más eficiente.
  - Si **cambian las estadísticas** (por ej. si se reorganiza el esquema de BD)
    - **Re-optimización** (quizá ahora convenga elegir **otra** estrategia)
      - SGBD Relacional o post-relacional: El Optimizador re-procesa la consulta original
      - SGBD No Relacional: hay que modificar el software!
  - Por lo general el Optimizador por ser un módulo del SGBD puede **considerar más estrategias** que un DBA manualmente.

PyOC: Análisis léxico, sintáctico y validación

Nombres de los voluntarios que realizan la tarea de PROMOCION

```
SELECT nombre
FROM voluntario V, tarea T
WHERE V.id_tarea = T.id_tarea
    AND T.nombre_tarea='PROMOCION';
```

$$\pi_{nombre}(\sigma_{V.id\_tarea = T.id\_tarea \text{ AND } nombre\_tarea='PROMOCION'}(VOLUNTARIO \times TAREA))$$

PyOC: Análisis léxico, sintáctico y validación

$\pi_{nombre}(\sigma_{V.id\_tarea = T.id\_tarea \text{ AND } nombre\_tarea='PROMOCION'}(VOLUNTARIO \times TAREA))$

resultado
↑
proyectar(V.nombre)
↑
seleccionar (V.id_tarea = T.id_tarea) AND
(T.nombre_tarea='PROMOCION')
↑
prod.cartesiano
↑ ↑
VOLUNTARIO(V) TAREA(T)

Árbol de Consulta

(o árbol sintáctico abstracto)

es la **representación de una expresión algebraica**

PyOC: Optimización

- El Optimizador de Consultas suele combinar varias técnicas, las principales son las siguientes:
  - **Optimización heurística:** **Ordena** las operaciones de la consulta para **incrementar** la **eficiencia** de su **ejecución**
  - **Estimación de costos:**
    - Estima el **costo de cada estrategia** de ejecución y
    - **Elige un** plan (estrategia) con el **menor costo** estimado

PyOC: Optimización

Optimización Heurística

Implica aplicar de **reglas de transformación y heurísticas** para modificar la representación interna de una consulta (Álgebra Relacional o **Árbol de consulta**) a fin de mejorar su rendimiento

- El Optimizador **selecciona el plan de ejecución más económico**
- En general, existen **muchos** (¡demasiados!) **posibles planes de ejecución** para una consulta

La tarea de **obtener el plan más económico**, **tendría un costo elevado…**

Entonces se suelen utilizar **técnicas heurísticas** para mantener el conjunto de planes de consulta generados en una cantidad razonable.

PyOC: Optimización

- El **Analizador Sintáctico** genera **árbol de consulta inicial**
  - sin optimización ⇨ ejecución ineficiente

- El **Optimizador** de Consultas **transforma el árbol** de consulta inicial en **árbol de consulta final** equivalente y eficiente

  Aplicación de **reglas de transformación** guiadas por **reglas heurísticas**

  Conversión de la consulta en su **forma canónica equivalente**

PyOC: Optimización

- Obtenida la forma canónica de la consulta, el Optimizador decide **cómo evaluarla**

Estimación sistemática **de costos**:
Estimación y comparación de los **costos de ejecutar** una consulta con **diferentes estrategias**, y elegir la estrategia con menor costo estimado

- El punto de partida es considerar la **consulta** como una serie de **operaciones elementales interdependientes**

Operaciones del Álgebra Relacional: π, σ, $\bowtie$, $\cup$, $\cap$

El Optimizador tiene un conjunto de técnicas para realizar cada operación

PyOC: Optimización

Información estadística

- Para cada **tabla**
  - **Cardinalidad** (cant.filas), **Factor de bloqueo** (cant. de filas por bloque), **bloques ocupados**,
  - **Método de acceso primario y otras** estructuras de acceso (hash, índices,etc.),
  - **Columnas indexadas**, de **dispersión**, de **ordenamiento** (físico o no), etc.
- Para cada **columna**
  - Nº de **valores distintos** almacenados,
  - Valores **máximo y mínimo**, etc.

El **éxito** de la estimación del **tamaño** y **costos** de las operaciones incluidas en una consulta, depende de la cantidad y actualidad de la información estadística almacenada en el diccionario de datos del SGBD

PyOC: Optimización

- **Método heurístico** de optimización de consultas utiliza **reglas de transformación** para convertir una expresión de álgebra relacional en otra forma equivalente que sepa que es más eficiente.

- Utiliza las reglas de transformación para **reestructurar** el árbol de álgebra relacional canónico.

- Construcción árbol canónico, se obtiene de aplicar:
  1. Producto cartesiano (X) a relaciones del FROM
  2. Condiciones de selección y ensamble de la cláusula WHERE
  3. Proyección sobre los atributos de la cláusula SELECT.

PyOC: Optimización

Nombres de los voluntarios que mayores de 18 años y que realizan la tarea de PROMOCION

```
SELECT nombre
FROM voluntario V, tarea T
WHERE V.id_tarea = T.id_tarea                                  C1
AND T.nombre_tarea='PROMOCION'                                 C2
AND date_part('year',age(V.fecha_nacimiento)) > 18;            C3
```

Árbol canónico desarrollado en clase

$$\Pi_{nombre}(\sigma_{C1\ AND\ C2\ AND\ C3}(V \times T))$$

$$\Pi_{nombre}(\sigma_{C2\ AND\ C3}(V \bowtie T))$$

$$\Pi_{nombre}(\ ((\sigma_{C3}(V)) \bowtie (\sigma_{C2}(T))\ ) \ldots\ \text{Otros!!}$$

Reglas de Equivalencia

1. Selección Conjuntiva: una secuencia de selecciones sobre una relación A puede transformarse en una sola selección

$$\sigma_{C1}(\sigma_{C2}(A)) \equiv \sigma_{C1 \text{ AND } C2}(A)$$

2. La selección es conmutativa,

$$\sigma_{C1}(\sigma_{C2}(A)) \equiv \sigma_{C2}(\sigma_{C1}(A))$$

3. En una secuencia de proyecciones sobre una relación A pueden ignorarse todas, salvo la última (si cada columna mencionado en la última, también aparece en las demás)

$$\Pi_{L2}(\Pi_{L1}(A)) \equiv \Pi_{L2}(A), \text{ sii } L2 \subseteq L1$$

Reglas de Equivalencia

4. La selección, puede combinarse con productos cartesianos y ensambles generales (Theta join).

$$\sigma_{C1}(A \times B) \equiv (A \bowtie_{C1} B) \qquad \sigma_{C1}(A \bowtie_{C2} B) \equiv A \bowtie_{C1 \text{ AND } C2} B$$

5. Los ensambles naturales y generales son conmutativos.

$$A \bowtie_{C} B \equiv B \bowtie_{C} A$$

6. a) Los ensambles naturales y b) generales son asociativos

a) $(A \bowtie B) \bowtie C \equiv A \bowtie (B \bowtie C)$

b) $(A \bowtie_{C1} B) \bowtie_{C2 \text{ AND } C3} C \equiv A \bowtie_{C1 \text{ AND } C3} (B \bowtie_{C2} C)$

c2 involucra atributos sólo de B y C

Reglas de Equivalencia

7. σ es distributiva respecto de ensambles si la condición de selección ...

- C1 involucra sólo atributos de una de las expresiones a ensamblar (sea A)

$$\sigma_{C1}(A \bowtie_C B) \equiv (\sigma_{C1}(A)) \bowtie_C B$$

- C1 involucra sólo los atributos de A y C2 sólo los de B.

$$\sigma_{C1\ AND\ C2}(A \bowtie_C B) \equiv (\sigma_{C1}(A)) \bowtie_C (\sigma_{C2}(B))$$

se **reduce el número de filas examinadas en la siguiente operación** en secuencia: join (así, esa operación será más rápida y también producirá menos filas)

se **reduce el nro. de columnas que se involucran en la siguiente operación** en secuencia: join (por tanto, esa operación necesitará menos tiempo para su ejecución y producirá menos columnas)

Reglas de Equivalencia

8. π es distributiva para el ensamble general si:

- si π involucra sólo atributos de L1 $\cup$ L2 (L1 y L2 conj. de atributos de A y B, L1 $\cap$ L2 $\neq \emptyset$)

$$\pi_{L1 \cup L2}(A \bowtie_c B) \equiv (\pi_{L1}(A)) \bowtie_c (\pi_{L2}(B))$$

- L1, L3 y L2, L4 conj. de atributos de A y B respectivamente; L3 y L4 involucrados en la condición C, pero no en L1 $\cup$ L2.

$$\pi_{L1 \cup L2}(A \bowtie_C B) \equiv$$

$$(\pi_{L1 \cup L2}((\pi_{L1 \cup L3}(A)) \bowtie_C (\pi_{L2 \cup L4} B))$$

Reglas de Equivalencia

9. Son conmutativas: **UNIÓN e INTERSECCIÓN** y **NO** conmutativas: **DIFERENCIA** y **DIVISIÓN**

$$A \cup B \equiv B \cup A \qquad A \cap B \equiv B \cap A$$

10. Son asociativas: **UNIÓN** e **INTERSECCIÓN** y **NO** asociativas: **DIFERENCIA** y **DIVISIÓN**

$$(A \cup B) \cup C \equiv A \cup (B \cup C)$$
$$(A \cap B) \cap C \equiv A \cap (B \cap C)$$

se **reduce el número de filas examinadas en la siguiente operación** en secuencia: Θ
(así, esa operación será más rápida y también producirá menos filas)

Reglas de Equivalencia

11. π es distributivo respecto de la $\cup$

$$\pi_p(A \cup B) \equiv (\pi_p(A)) \cup (\pi_p(B))$$

12. σ es distributivo respecto de la $\cup$, $\cap$ y $-$; $\Theta \in \{ \cup, \cap, - \}$

$$\sigma_c(A \Theta B) \equiv (\sigma_c(A)) \Theta (\sigma_c(B))$$

Reglas de Equivalencia

- **Expresiones de cómputo escalar**
  - El Optimizador debe conocer **reglas de transformación de expresiones aritméticas** porque aparecen en las consultas
  - Reglas de transformación basadas en propiedades Conmutativa, Asociativa y Distributiva

- **Expresiones condicionales** (booleanas)
  - El Optimizador debe saber aplicar reglas generales a **operadores**
    - **de comparación** (>, <, ...) y
    - **lógicos** (AND, OR, ...)

Reglas Heurísticas

Algunas **buenas heurísticas** que pueden ser aplicadas durante el procesamiento de consultas

1. Ejecutar **operaciones de restricción σ tan pronto como sea posible**
2. Ejecutar **primero las restricciones σ más restrictivas** (las que producen menor nro. de filas)
3. **Combinar** un **producto cartesiano ×** con una σ subsiguiente cuya condición represente una condición de reunión, **convirtiéndolas en un ensamble** ⋈
4. **Ejecutar** las operaciones de **Π tan pronto como sea posible**

σ : Selección

Implementación de op. relacionales

Sin índices: Los algoritmos que 'barren' tablas (archivos) recuperando registros que cumplen con la condición de la selección.

Algoritmos básicos:

- **Búsqueda lineal:** Se examinan todos los bloques y se toman los que cumplan la condición.
- **Búsqueda binaria:** Si el archivo está ordenado en función del atributo del que se está realizando la selección.

Con índices:

- **Igualdad basada en clave:** Se usará el índice primario para recuperar el único registro.
- **Igualdad basada en atributo no clave:** Se usará el índice de agrupamiento para recuperar varios registros.
- **Otros casos de igualdad**: Utilizar índices secundarios para recuperar un registro si es único o varios si no lo es.

σ : Selección

Implementación de op. relacionales

Sin índices: Debe barrer todo el archivo (tabla).

Con índice sobre el atributo de selección: Se utilizará el índice para encontrar el conjunto de entradas que satisfacen la condición, y luego se recuperarán los registro de datos.

Importante: el índice con hash sólo es eficiente para selecciones por igualdad

Estimación del tamaño del resultado:

(Tamaño de V) * factor de reducción

Operación de Selección

Algoritmos de búsqueda para localizan y recuperan registros de archivos con br bloques y su costo en términos de operaciones E/S

- **Búsqueda lineal:** *br*
- **Búsqueda binaria:** $\lceil \log_2 (\mathrm{br}) \rceil$, archivo ordenado según un atributo y condición de la selección.

Con Caminos de acceso (índices)

- **Índice primario:** Altura del árbol + 1
- **Índice secundario:** Altura del árbol + n registros

Con condiciones de comparación σ A <= x (T)

- **índice primario o secundario de comparación:** Si A >= x

Con ensambles

- **Por bucle**
- **Por bucle indexado**
- **Por ordenación-mezcla**

Plan de ejecución de la consulta

Algunos SGBD permiten a los usuarios inspeccionar la estrategia del optimizador para ejecutar determinada consulta.

Algunos SGBD permiten "fijar" el plan de consulta. (hints) Postgresql NO.

Query execution plan:

En Postgresql sentencia **EXPLAIN**

EXPLAIN SELECT ….

Bibliografía

Elmasri, R.; Navathe, S.B.: **Fundamentos de Sistemas de Bases de Datos**. 3ª Edición. Addison-Wesley. **(Cap. 18)**

Elmasri, R.; Navathe, S.B.: **Sistemas de bases de datos. Conceptos fundamentales**. 2ª Ed. Addison-Wesley Iberoamericana. **(Cap. 16)**

Connolly et al.: **Database Systems: A Practical Approach to Design, Implementation and Management**. 2nd Ed. Addison-Wesley **(Cap. 18)**