Introducción a las Bases de Datos y Bases de Datos

PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS

TEORÍA 10

2017

Tecnicaturas TUPAR y TUDAI

PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS

- 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.

PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS

En un sistema no relacional:

- Consultas expresadas en lenguaje procedural de bajo nivel
- El usuario selecciona la estrategia de ejecución realizando una optimización "manual". Decide las operaciones necesarias y su orden de ejecución. Si se equivoca, el sistema no puede mejorar automáticamente la situación.

En un sistema relacional:

- Consultas expresadas con SQL: indica QUÉ datos pero no CÓMO recuperarlos.
- El SGBD puede seleccionar una **estrategia de ejecución** y tiene mayor **control sobre** el **rendimiento** del sistema.
- La **optimización "automática"** es un desafío (para lograr un tiempo de ejecución de consultas aceptable) y una oportunidad (el alto nivel semántico de una expresión relacional permite su optimización antes de la ejecución)

PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS

Procesamiento de Consultas: Actividades involucradas en la recuperación de datos de la BD

Optimización de Consultas: Elección de una estrategia de ejecución eficaz para procesar cada consulta sobre la base de datos

- La **optimización** [automática] es...
  - Un **reto**: obligatorio para lograr un **tiempo de ejecución de consultas aceptable**
  - Una **oportunidad**: el alto nivel semántico de una expresión relacional permite su **optimización antes de la ejecución**

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.

– Existen 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

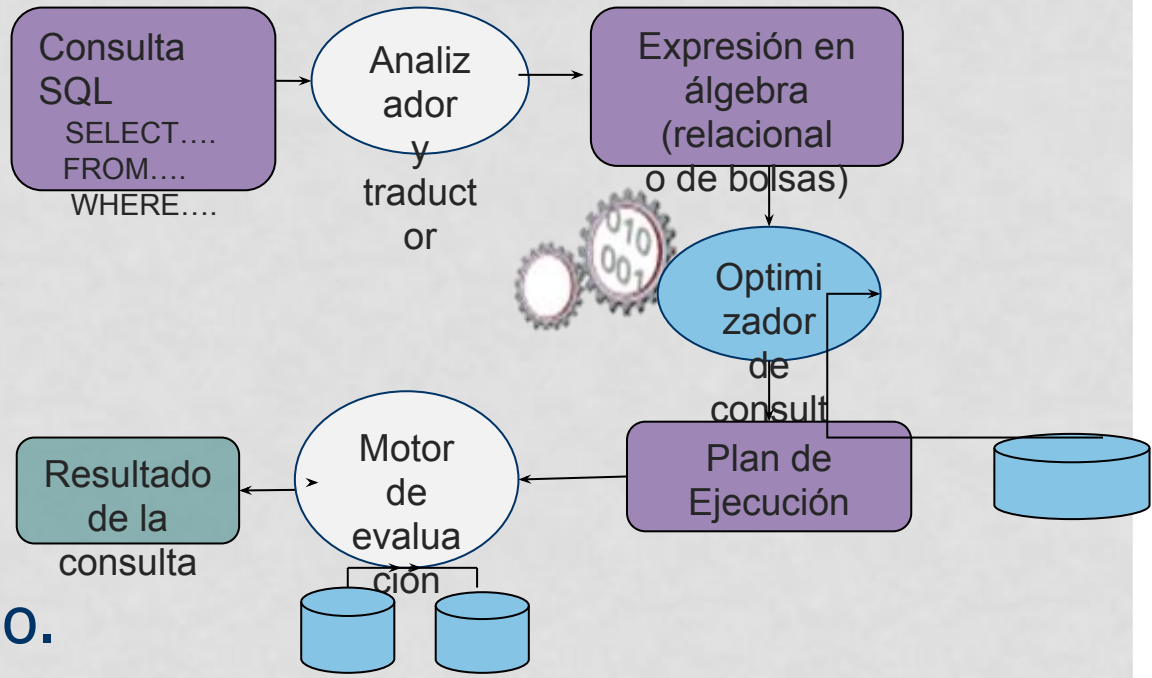
Ventajas de la **optimización automática**

- 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: 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.

PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS

PASOS EN EL PROCESAMIENTO DE CONSULTAS (PyOC)

1. **Análisis léxico, sintáctico y validación**
2. **Optimización**
3. Generación de código
4. Ejecución



Describiremos el 1er.

Veremos en detalle el 2do.

PYOC: ANÁLISIS LÉXICO, SINTÁCTICO Y VALIDACIÓN

- **Análisis léxico:** **Identificar** los **componentes** (léxicos) en el texto de la consulta (SQL)
- **Análisis sintáctico:** **Revisar** la **sintaxis** de la consulta (corrección gramática)
- **Validación semántica:** **Verificar** la validez de los **nombres** de las tablas, vistas, columnas, etc. y si tienen sentido
- **Traducción** de la consulta a una representación interna eliminando peculiaridades del lenguaje de alto nivel (SQL), la mejor elección: Álgebra Relacional

PYOC: ANÁLISIS LÉXICO, SINTÁCTICO Y VALIDACIÓN

π Proyeccción
σ Selección
⋈Ensamble
X Producto Cartesiano

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';
```

$$\Pi_{nombre}(\sigma_{nombre\_tarea='PROMOCION'}(VOLUNTARIO \bowtie TAREA))$$

resultado
↑
proyectar(V.nombre)
↑
seleccionar(T.nombre_tarea='PROMOCION')
↑
ensamblar
↑ ↑
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 técnicas principales son las siguientes:
  - **Optimización heurística:** **Ordenar** las operaciones de la consulta para **incrementar** la **eficiencia** de su **ejecución**
  - **Estimación de costos:**
    - Estimar sistemáticamente el **costo de cada estrategia** de ejecución y
    - **Elegir** el plan (estrategia) con el **menor costo** estimado

PYOC: OPTIMIZACIÓN

- Optimización Heurística
  - **Aplicación 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**

- Varias expresiones del álgebra relacional pueden corresponder a la misma consulta

- Lenguajes de consulta, como SQL permiten expresar una misma consulta de muchas formas diferentes, pero el rendimiento NO debiera depender de cómo sea expresada la consulta, no siempre ocurre.

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

PYOC: OPTIMIZACIÓN

- El Optimizador tiene un conjunto de **técnicas** para realizar cada operación

  Ejemplo: técnicas para implementar una **σ**

  - Búsqueda lineal
  - Búsqueda binaria
  - Vía un índice primario o clave de dispersión
  - Vía un índice de agrupamiento
  - Vía un índice secundario

- ¿Cómo elige el Optimizador las técnicas adecuadas en cada caso?

PYOC: OPTIMIZACIÓN

Estadística
(diccionario de datos)

Interdependencia entre las operaciones de bajo nivel

OPTIMIZADOR

Elección de varias **técnicas candidatas** para 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

El Optimizador genera **varios planes de ejecución**

- **Plan de Ejecución** = combinación de **técnicas** candidatas
  - una técnica por cada operación elemental de la consulta
- Cada técnica tendrá asociada una **estimación del costo**

Costo(técnica$_i$) = cant.accesos a bloque de disco necesarios

- Transferencias de bloques memoria ↔ disco
- **La estimación exacta de costos es difícil**, dado que para estimar los accesos a bloque es necesario estimar el tamaño de las tablas (base o generadas como resultados intermedios)
  - ➢ depende de los **valores actuales** de los datos

PYOC: OPTIMIZACIÓN

- 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 suele utilizar **técnicas heurísticas** para mantener el conjunto de planes de consulta generados en un nro. razonable.

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

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

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

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

REGLAS DE EQUIVALENCIA

- **Expresiones de cómputo escalar**
  - El Optimizador debe conocer **reglas de transformación de expresiones aritméticas**, pues 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**

Ejemplo de transformación (1)

Nombres de los voluntarios que mayores de 18 años y que realizan la tarea de PROMOCION
Relaciones: V Voluntario, T Tarea
Condiciones C1: T.nombre_tarea='PROMOCION'
C2: date_part('year',age(V.fecha_nacimiento)) > 18

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

$$\Pi_{nombre}(\ ((\sigma_{C1}(V)) \bowtie (\sigma_{C2}(T))\ )$$

Acercar la selección tan tempranamente como sea posible reduce el tamaño de la relación (tabla) a ensamblar.

Ejemplo de transformación (2)

Nombres de las tareas que realizan los voluntarios mayores de 18 años que pertenecen a la institución FUNDACION ALERTA BOSQUES

Relaciones: V Voluntario, T Tarea, Institución I

Condiciones C1: date_part('year',age(V.fecha_nacimiento)) > 18

C2: I.nombre_institucion = 'FUNDACION ALERTA BOSQUES'

$$\Pi_{nombre\_tarea}(\sigma_{C1\ AND\ C2}(V \bowtie T \bowtie I))$$

$$\Pi_{nombre\_tarea}((\sigma_{C1\ AND\ C2}(V \bowtie I)) \bowtie T) \qquad (1)$$

$$\Pi_{nombre\_tarea}(((\sigma_{C1}(V)) \bowtie (\sigma_{C2}(I))) \bowtie T) \qquad (2)$$

(1) Asociar los ensambles apropiadamente

(2) Ejecutar las selecciones lo más temprano posible

⋈ Implementación de Ensambles

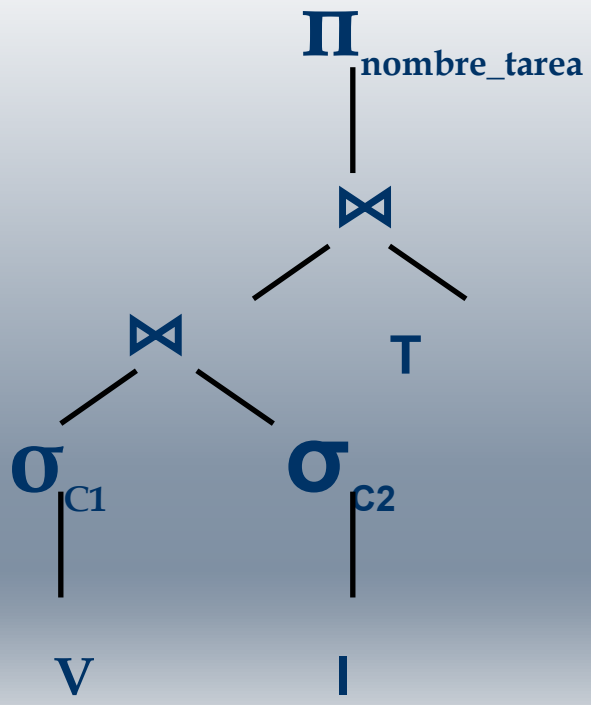
- Las **técnicas** para realizar ensambles pueden ser:
  1. **Fuerza Bruta**
  2. **Búsqueda por Índice**
  3. **Búsqueda** Hash
  4. **Mezcla**
  5. **Hash**
  6. Combinaciones de las anteriores

Plan de Evaluación

Ensamblar por hash

Ensamblar por bucle

Ambos búsqueda lineal



Índices

Realmente importantes para disminuir el tiempo de procesamiento de una consulta.

```
SELECT nombre, apellido
FORM voluntario V
WHERE nombre = 'Alexander'
```

- Barrido secuencial de la tabla voluntario puede demorar mucho tiempo dependiendo de su tamaño, entre otras características.
- Una alternativa de mejora es crear un índice.

Creación de índices

Sintaxis

```
CREATE INDEX IN_NombreIndice
ON tabla
[ USING método ]
( { columna | (expresión) } ) [ ASC | DESC ])
```

Método puede ser árbol B, hash u otro específico para el tipo de dato.

Expresión puede ser una función o una expresión de más de una columna. Por ej. extract(year from fecha)

Creación de índices

Ejemplo

```
CREATE INDEX inx_voluntario
ON voluntario ( nombre )
```

Ayuda para la consulta

```
SELECT nombre, apellido
FORM voluntario V
WHERE nombre = ‘Alexander´
```

Pero NO contribuye en

```
SELECT nombre, apellido
FORM voluntario V
WHERE telefono like '+54%'
```

Creación de índices

Ejemplo

```
CREATE INDEX inx_voluntario_2
ON voluntario ( horas_aportadas)
```

Ayuda para consultas por rango, en particular los soportados en árboles B+

```
SELECT nombre, apellido
FORM voluntario V
WHERE horas_aportadas between 2700 AND 3100
```

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

σ : Selección

Uso de índices

Ejemplo

```
SELECT nombre, apellido
FORM voluntario V
WHERE telefono like '+54%'
```

Puede usarse un índice sobre teléfono; sólo útil para las consultas que comiencen con un conj. de caracteres, NO para por ej. '%45*'

Las alternativas puede ser:

- Encontrar el camino más restrictivo y luego aplicar verificaciones para los atributos que no se usaron con el índice, o bien
- Obtener los conjuntos de referencias a registros para cada atributo, usando cada índice disponible, hallar la intersección de las referencias y verificar la condición sobre los atributos remanentes.

π : Proyección

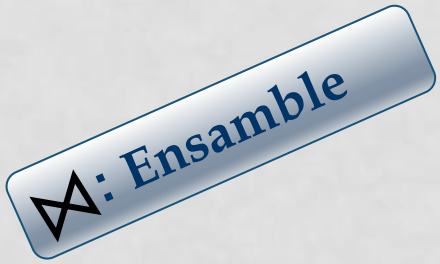
Uso de índices

- **Dos pasos:**
  1. remover atributos no deseados.
  2. remover duplicados del resultado.

- **Refinamientos para el 2do paso:**
  - Si hubiese algún índice para todos los atributos deseados, se puede hacer un barrido del índice solamente.
  - Si el índice contiene un subconjunto de los atributos deseados, se pueden remover los duplicados *localmente*.

Ejemplo

```
SELECT DISTINCT id_tarea
FORM voluntario V
```



Implementación de op. relacionales

Para estimar el tamaño de los ensambles se usa la estimación del producto cartesiano. Existen varias formas de calcular el costo asociado al ensamble de relaciones.

Por ciclos anidados: (costoso)

No necesita índices y se puede utilizar sin importar la condición del ensamble. Examina cada pareja de filas en las dos relaciones:

for each tupla *tr* in R **do begin**
  for each tupla *ts* in S **do begin**
    if (*tr,ts*) satisface la condición **C** del ensamble $R \bowtie_C S$
then
      añadir *tr.ts* al resultado;
  end
end

⋈: Ensamble

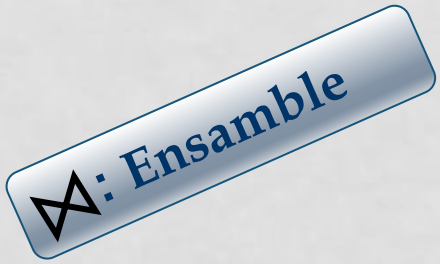
Implementación de op. relacionales

POR CICLOS ANIDADOS

- Sustituye las búsquedas en archivos por búsquedas en un índice.

- Para cada tupla *tr* de la relación 'externa' R se usa el índice para buscar tuplas en S que cumplan la condición de ensamble con la tupla *tr*.

- Se deben crear los índices **apropiados**.



Implementación de op. relacionales

- ✓ Requiere una única pasada sobre los datos de cada tabla
  1) Construir tabla de hashing H para S sobre el o los atributos S.C sobre los cuales se ensambla, utilizando h(S.C)

  Cada entrada de H contiene:

  Valor de S.C y (opcional) valores de otras columnas de S

  Puntero a la fila correspondiente en el archivo.

  2) Examinar R y aplicar la misma función h(R.C)

  Si una fila de R resulta ser sinónimo en H con filas de S, entonces, si S.C = R.C, se generan las filas ensambladas adecuadas.
- ✓ Ventaja sobre la técnica de mezcla:
  - ✓ No se requiere que R y S estén ordenadas
  - ✓ No se requiere ordenarlas dinámicamente

Columnas a Indexar

- Columnas de clave primaria y extranjeras.
- Columnas sobre las que se apliquen frecuentemente funciones de agregación.
- Columnas frecuentemente usadas para ORDER BY, GROUP BY, y DISTINCT

Plan de ejecución de la consulta

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

- Query execution plan:

  En Postgresql sentecia **EXPLAIN**
  EXPLAIN SELECT ….