# Base de Datos II

## Unidad 3 – Optimización

## Objetivos

- Optimización: Definición.
- Estimación de estadísticas.
- Buenas prácticas.
- Vistas Materializadas.
- Implementación de operadores relacionales:
  - Fuerza bruta.
  - Búsqueda con índice.
  - Búsqueda con dispersión.
  - Mezcla
  - Dispersión.

# Unidad 3 : Optimización

La optimización de consultas, es el proceso de selección del plan de evaluación de las consultas más eficiente de entre muchas estrategias disponibles para el procesamiento de una consulta dada, especialmente si la consulta es compleja.

*Un aspecto de la optimización de consultas tiene lugar en el nivel del algebra relacional, donde el DBMS intenta hallar una expresión que sea equivalente a la expresión dada, pero de ejecución mas eficiente.*

*Otro aspecto es la elección de una estrategia detallada para el procesamiento de la consulta, como puede ser la selección de un algoritmo que se utilizará para ejecutar una operación, la selección de índices concretos, que se van a emplear, etc.*

## Optimización

- La diferencia de costos (en términos de tiempo de evaluación) entre una estrategia buena y una mala suele ser sustancial. Por lo tanto, vale la pena que el sistema pase una importante cantidad de tiempo en la selección de una buena estrategia.
- Ejemplo: *Obtener el nombre de los proveedores que proporcionan la parte P2.*

## Optimizando la consulta

- Supongamos que la BD contiene 100 proveedores y 10000 envíos, de los cuales solo 50 son de la parte P2. Supongamos que las tablas proveedor y envío están representadas en disco como dos archivos guardados por separado.
- Si el sistema evalúa la expresión “directamente” (sin optimizar), la secuencia sería:
  - Juntar proveedor con envío: este proceso implica la lectura de 10000 envíos, la lectura de cada uno de los 100 proveedores 10000 veces (una vez por cada envío) la construcción de 10000 tuplas juntadas y la escritura de 10000 tuplas de nuevo en disco.
  - Restringir el resultado del anterior paso, solo a las tuplas para la parte P2: involucra la lectura de las 10000 tuplas juntadas hacia memoria, y produce un resultado de 50 tuplas que quedan en memoria principal.
  - Proyectar el resultado sobre proveedor: esto produce el resultado final deseado.

## Optimizando la consulta

- El siguiente procedimiento es equivalente (produce el mismo resultado) pero es mas eficiente:
  - Restringir los envíos solamente a las tuplas para la parte P2: esto involucra la lectura de 10000 tuplas, pero produce un resultado de 50 tuplas, que pueden mantenerse en memoria principal.
  - Juntar el resultado del paso anterior, con proveedor: esta paso involucra la lectura de 100 proveedores (solo una vez, y no por cada envío de P2) y produce nuevamente un resultado de 50 tuplas.
  - Proyectar el resultado del paso anterior sobre proveedor.

## Comparando números de la consulta

- El 1º de estos procedimientos implica la E/S de 1.030.000 tuplas.
- La 2º involucra 10.100.
- Si tomamos la cantidad de E/S de las tuplas como medida de desempeño, el 2º procedimiento es un poco mas de 100 veces mejor que el 1º.
- Un cambio muy simple en el algoritmo de ejecución ha producido una reducción drástica en el desempeño. La mejora sería todavía mas drástica si los envíos estuvieran además indexados en la parte, ya que la cantidad de tuplas de envíos leídas sería 50 y no 10000, y el nuevo procedimiento sería 7000 veces mejor que el original.

## Optimizador – Información del catálogo

- $n_r$ , el número de tuplas de la relación r.
- $b_r$ , el número de bloques que contiene tuplas de la relación r.
- $t_r$ , el tamaño de cada tupla de la relacion r en bytes.
- $f_r$ , el factor de bloqueo de la relación r, es decir, el número de tuplas de la relación r que caben en un bloque.
- V(A,r), el número de valores distintos que aparecen en la relación r para el atributo A. Si A es una clave de la relación r, entonces V(A,r) es $n_r$.
- La ultima estadística también se puede calcular para conjuntos de atributos, en vez de solo para atributos aislados.
- Las estadísticas sobre los índices, como las alturas de los arboles B+ y el número de páginas hojas de los índices, también se conservan en el catalogo.
- Los optimizadores verdaderos suelen conservar información estadística adicional para mejorar la precisión de sus estimaciones de costos de los planes de evaluación.

## Actualización de estadísticas

- Si se desean conservar estadísticas precisas, cada vez que se modifica una relación también hay que actualizar las estadísticas.
- Esto supone una “sobrecarga adicional”.
- La mayoría de los DBMS no actualizan las estadísticas con cada modificación, sino en los momentos de menor carga.
- En consecuencia, puede que las estadísticas utilizadas para escoger una estrategia no sean lo suficientemente exactas.

## Estimación del tamaño de la selección

- La estimación del tamaño del resultado de una operación de selección depende del predicado de la selección.
- Si hacemos una selección por igualdad, en la que el atributo *A* tome el valor *a (A = a)*, si se supone una distribución uniforme de los valores, se puede estimar que el resultado de la selección sería: $n_r$ / *V(A,r)* tuplas, suponiendo que el valor *a* aparece en el atributo *A* de algún registro de *r*. Pese al hecho de que la suposición de distribución uniforme no suele ser correcta, resulta una aproximación razonable de la realidad.

## Estimación del tamaño de la selección

- Considere una selección donde *A* <= *v*, si el valor real *v* utilizado en la comparación esta disponible en el momento de calculo de costos, puede hacerse una estimación mas precisa. Los valores mínimo y máximo del atributo *A* están almacenados en el catálogo. Suponiendo nuevamente una distribución uniforme, se puede estimar el nº de registros que cumpliran con la condición *A<=v* como *0* si *v<min(A,r)*, y como *n* si *v>=min(A,r)* y como

***$n_r$ * ( v – min(A,r)) / (max(A,r) – min(A,r))*** ***en cualquier otro caso***

## Estimación del tamaño de la selección

- En algunos casos, cuando la consulta forma parte de un procedimiento almacenado, puede que el valor *v* no este disponible cuando se optimice la consulta. En estos casos se supondrá que aproximadamente la mitad de los registros cumplen la condición de comparación. Esta estimación puede resultar muy imprecisa, pero es lo mejor que se puede hacer sin información.

## Estimación del tamaño de las reuniones

- El producto cartesiano r * s contiene $n_s$ * $n_r$ tuplas. Cada tupla de r *s ocupa $t_r$ + $t_s$ bytes, de donde se puede calcular el tamaño del producto cartesiano.
  - Si la intersección entre R y S es vacía, es decir las relaciones no tienen atributos en común, entonces se puede usar la técnica de estimación anterior para los productos cartesianos.
  - Si la intersección entre R y S es una clave de R, entonces se sabe que cada tupla de *s* se combinará como máximo con una tupla de *r.* Por lo tanto, el nº de tuplas no es mayor que el numero de tuplas de S.
  - El costo mas difícil es que la intersección entre R y S no sea una clave de R ni de S.
  - …. Se realizan estimaciones para todas las consultas: proyección, agregación, operaciones de conjunto, etc.

## Procesamiento de consultas

1. Convertir la consulta a su forma interna.
2. Convertir a la forma canónica.
3. Seleccionar procedimientos candidatos de bajo nivel.
4. Generar planes de consulta y seleccionar el de menor costo.

## 1- Convertir la consulta a su forma interna

- Implica transformar la consulta original en su representación interna que sea mas adecuada para manejarla en la máquina. El procesamiento de vistas también se realiza en esta etapa.
- Por lo general, la forma interna seleccionada es algún tipo de árbol de sintaxis abstracto o árbol de consulta.
- Podemos pensar que esta representación interna representa alguno de los formalismos del algebra relacional o el cálculo relacional.

## 2 – Conversión a la forma canónica

- En esta etapa, el optimizador hace optimizaciones que son garantizadas como buenas, sin tener en cuenta las rutas de acceso físicas ni los valores actuales de los datos.
- En este paso, el optimizador convierte la representación interna en alguna forma canónica equivalente, con el objeto de encontrar una representación que sea en cierta forma, mas eficiente que la anterior.

(A join B) where < restricción sobre A>

(A where < restricción sobre A>) join B

## 3 – Selección de procedimientos candidatos de bajo nivel

- El optimizador debe elegir como ejecutar dicha consulta transformada. Aquí entran en juego consideraciones tales como la existencia de índices u otras rutas de acceso físicas, la distribución de los valores, el agrupamiento físico de los datos almacenados, etc.
- La estrategia básica consiste en considerar la expresión como la especificación de una serie de operaciones de "bajo nivel" con cierta independencia entre si.
- Luego para cada operación de bajo nivel, el implementador tendrá una serie de procedimientos de implementación predefinidos.
- Cada procedimiento tendrá una formula de costo asociada, que indique el costo de ejecutar ese procedimiento.

## 4 – Generación de planes de consulta y selección del mas barato

- La ultima etapa del proceso de optimización involucra la construcción de un conjunto de planes de consulta candidatos, seguidos de una selección del mejor de estos planes.
- Cada plan se construye por medio de combinación de una serie de procedimientos predefinidos candidatos.
- La elección del plan de menor costo, requiere de un método para asignar un costo a cualquier plan dado. Este costo es básicamente la suma de los costos de los procedimientos individuales que conforman ese plan.
- La estimación precisa de los costos puede ser un problema difícil.

## Divide y vencerás

- Una estrategia es la descomposición de la consulta, y es una estrategia muy usada en ambientes distribuidos y de procesamiento en paralelo.
- La idea básica de la descomposición de la consulta es dividir una consulta que involucra muchas variables en una secuencia de consultas pequeñas que generalmente involucren cada una , a una o dos de estas variables, usando separación y sustitución de tuplas para lograr esta descomposición.

## Buenas prácticas

- Buenas prácticas Generales: recomendaciones de diseño y configuración de carácter general, aplicable a cualquier motor de base de datos relacional.
- Buenas prácticas Particulares: recomendaciones únicamente aplicables a un determinado motor de base de datos particular (ej. MySQL, SQL Server, Oracle, etc.).

## Buenas prácticas Generales: Diseño de la BD

- Normalizar las tablas, hasta la tercera forma normal, para asegurar que no hay duplicidad de datos y se aprovecha al máximo el almacenamiento en las tablas. Si hay que desnormalizar alguna tabla se debe pensar en la ocupación y en el rendimiento antes de proceder.
- Los primeros campos de cada tabla deben ser aquellos campos requeridos y dentro de los requeridos primero se definen los de longitud fija y después los de longitud variable.
- Ajusta al máximo el tamaño de los campos para no desperdiciar espacio.
- Es muy habitual dejar un campo de texto para observaciones en las tablas. Si este campo se va a utilizar con poca frecuencia o si se ha definido con gran tamaño, es mejor crear una nueva tabla que contenga la clave primaria de la primera y el campo para observaciones.

## Buenas prácticas Generales: Gestión y elección de índices

- Los índices son campos elegidos arbitrariamente por el constructor de la base de datos que permiten la búsqueda a partir de dicho campo a una velocidad notablemente superior. Esto se ve contrarrestada por el hecho de ocupar mucha más memoria (el doble más o menos) y de requerir para su inserción y actualización un tiempo de proceso superior.
- No podemos indexar todos los campos de una tabla extensa ya que doblamos el tamaño de la base de datos. Tampoco sirve de mucho el indexar todos los campos en una tabla pequeña ya que las selecciones pueden efectuarse rápidamente de todos modos.
- Un caso en el que los índices pueden resultar muy útiles es cuando realizamos peticiones simultáneas sobre varias tablas.
- Los índices pueden resultar contraproducentes si los introducimos sobre campos triviales a partir de los cuales no se realiza ningún tipo de petición ya que, además del problema de memoria ya mencionado, estamos ralentizando otras tareas de la base de datos como son la edición, inserción y borrado.
- Entonces ¿Qué indexamos? ….

## Buenas prácticas Generales: Donde escribir las sentencias

- Siempre es preferible utilizar consultas almacenadas dentro del motor de base de datos y no dentro de la aplicación, una consulta almacenada en un procedimiento almacenado, se ejecuta mucho mas rápido y directamente sobre el motor que cualquier consulta externa.
- Muchas de las aplicaciones sobre todo de reporting permiten unir y realizar los joins y consultas directamente en la herramienta produciendo una baja de performance realmente considerable. Lo correcto y más eficiente seria generar la consulta en un sp con todos los joins y guardar el resultado en una tabla temporal, de donde posteriormente el reporte solo deberá mostrar los datos sin necesidad de trabajarlos primero. Dependiendo los joins que intervengan y los registros involucrados se pude mejorar drásticamente la performance.

## Buenas prácticas Generales: Formulación de las consultas

- No utilizar sin necesidad SELECT * por que el motor debe leer primero la estructura de la tabla antes de ejecutar la sentencia.
- Seleccionar solo aquellos campos que se necesiten, cada campo extra genera tiempo extra.
- Utilizar Inner Join , left join , right join, para unir las tablas en lugar del where, esto permite que a medida que se declaran las tablas se unan mientras que si utilizamos el where el motor genera primero el producto cartesiano de todos los registros de las tablas para luego filtrar las correctas, un trabajo definitivamente lento.

## Buenas prácticas Particulares: MySQL

- Especificar el alias de la tabla delante de cada campo definido en el select, esto le ahorra tiempo al motor de tener que buscar a que tabla pertenece el campo especificado.
- El orden de ubicación las tablas en el from deberían ir en lo preferible de menor a mayor según el número de registros , de esta manera reducimos la cantidad de revisiones de registros que realiza el motor al unir las tablas a medida que se agregan.

## Buenas prácticas Particulares: SQL Server

- Si desea guardar objetos grandes (BLOB) en la base de datos como por ejemplo: Imágenes, documentos, etc. No lo haga en la misma tabla de la entidad y cree una nueva tabla con el ID y su campo pesado.
- Cada tabla soporta un solo índice Clustered ya que es el ordenamiento físico de la misma. Se debe prestar atención a la creación de este tipo de índices ya que puede tener impacto negativo en la performance. Se recomienda que todas las tablas tengan Clustered index ya que las que no lo tienen son menos performantes.
- Se recomiendan utilizar el estándar (YYYYMMDD HH:mm:ss) para la definición de las fechas.
- Evitar subconsultas en campos del SELECT.
- Utilizar BETWEEN en lugar de IN u OR.
- Evitar operaciones aritméticas del lado de las columnas del WHERE y ON del JOIN.

# Optimizadores en DBMS: MySQL

- MySQL Query Esta herramienta gráfica es provista por por MySQL para crear, ejecutar, y optimizar consultas dirigidas a bases de datos MySQL.
- myisamchk : Una utilidad para describir, testear, optimizar y reparar tablas. La forma es: myisamchk -r *nombre_tabla*
- Puede optimizar una tabla de la misma forma usando el comando SQL OPTIMIZE TABLE . Realiza una reparación de la tabla y un análisis de las claves, y también ordena el árbol de índices para obtener un mejor rendimiento en la búsqueda de claves.
- SELECT BENCHMARK(*loop_count,expression):* devuelve el tiempo en segundos que consume la expresion, la cantidad de veces requeridas. Todas las funciones MySQL pueden ser optimizadas. BENCHMARK() es una herramienta excelente para buscar donde estan los problemas de una query.

# Optimizadores en DBMS: MySQL

- EXPLAIN *tbl_name :* es un sinonimo *DESCRIBE tbl_name or SHOW COLUMNS*
- EXPLAIN SELECT *select_options:* MySQL explica cómo procesaría el SELECT, proporcionando información acerca de cómo las tablas se unen y en qué orden.
- ANALYZE TABLE tt: permite analizar una tabla, e informa errores sobre esta.

# Optimizadores en DBMS: SQL Server

- SQL Server Management Studio provee una herramienta gráfica para analizar planes de consulta, utilización de índices y costos.
- SQL Server Index Tuning Advisor es una herramienta automática para detectar problema en la configuración de índices (índices que falta o que sobran).
- SQL Trace: es una herramienta para analizar las sentencias ejecutadas en forma on-line, ver sus costos, sus planes de ejecución, cadenas de bloqueos, deadlock, etc.

# Pasos en MySQL para optimizar una consulta WHERE

1. Elimina los paréntesis innecesarios:

   ((a AND b) AND c OR (((a AND b) AND (c AND d))))

   ->(a AND b AND c) OR (a AND b AND c AND d)

2. Simplificación constante:

   (a<b AND b=c) AND a=5

   -> b>5 AND b=c AND a=5

3. Eliminación continua de condiciones (es necesario por la simplificación constante):

   (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)

   -> B=5 OR B=6

4. Las expresiones constantes utilizados por los índices se evalúan sólo una vez.

## Pasos en MySQL para optimizar una consulta WHERE

5. COUNT (*) sobre una sola tabla sin un WHERE se recupera directamente de la tabla de información para tablas MyISAM y HEAP. Esto también se hace para cualquier expresión no NULL cuando se utiliza con una sola tabla.
6. Detección temprana de expresiones constantes inválidas : MySQL detecta rápidamente que algunos comandos SELECT no son posibles y no devuelve ningúna fila.
7. HAVING se fusiona con WHERE si no se utiliza GROUP BY o funciones de grupo (COUNT (), MIN (), y así sucesivamente).
8. Para cada tabla en un join, se construye un WHERE más simple para obtener una rápida evaluación del WHERE de la tabla y también para omitir registros tan pronto como sea posible.
9. Etc …..

## Vistas materializadas

Cuando se define una vista, normalmente la BD solo almacena la consulta que define la vista.

Una vista materializada es una vista cuyo contenido se calcula y se almacena. Constituyen datos redundantes, pero resulta mucho mas económico en muchos casos leer el contenido de la vista materializada que calcular el contenido de una vista ejecutando la consulta que la define.

Las vistas materializadas resultan importantes para la mejora del rendimiento de algunas aplicaciones.

## Vistas Materializadas

- Mantenimiento de las vistas
  - Un problema de las vistas materializadas es su mantenimiento.
  - Pueden mantenerse mediante código escrito (triggers o procedimientos almacenados).
- ¿Qué vistas materializar? Esta decisión debe tomarse teniendo en cuenta la carga de trabajo del sistema

## Implementación de los operadores relacionales

A continuación, explicaremos algunos métodos directos para implementar algunos operadores relacionales, especialmente el join. Estos métodos pertenecen al 3º paso en el procesamiento de consultas, que llamamos “procedimientos de implementación de bajo nivel”

## Fuerza bruta

- Es lo que podria llamarse el caso “llano” donde se inspeccionan todas las combinaciones posibles de tuplas (cada tupla de R es examinada en conjunción con cada tupla de S).
- Observemos los costos asociados al enfoque de fuerza bruta (mirando solamente el costo de E/S, aunque en realidad sean importantes otros costos): el enfoque requiere claramente un total de *m + ( m * n)* operaciones de lectura de tuplas, pero la cantidad de escrituras será distinta según la cardinalidad de la relación:
  - En el caso de que un join de muchos a uno (en particular de clave externa con una clave candidata coincidente), es claro que la cardinalidad del resultado es igual a la cardinalidad ( m o n) de R o de S, dependiendo del cual representa el lado de la clave externa del join.
  - En el caso de muchos a muchos, si suponemos una distribución uniforme de los valores, la cantidad total de tuplas del join sería (m * n) / dCRoS , donde dCRoS es la cantidad de valores distintos del atributo de junta C en la relacion R o S (el menor de los dos).
- Este es el procedimiento del peor caso, ya que asume que la relacion S no esta indexada ni dispersada sobre el atributo de junta C.

## Búsqueda con índice

Consideremos el caso en el que existe un índice X, sobre el atributo S.C de la relación interna.

La ventaja de esta técnica sobre la fuerza bruta es que para una tupla dada de la relación externa R, se puede ir “directamente” a las tuplas coincidentes de la relación interna S. La cantidad de lecturas de tuplas de las relaciones R y S es simplemente la cardinalidad del resultado juntado.

## Búsqueda con dispersión

Es similar a la búsqueda con índice, con excepción de que la ruta de acceso rápida a la relación interna S, sobre el atributo de junta S.C, se hace con dispersión en lugar de un índice.

Se accede a los datos a través de una estructura hash.

## Mezcla

La técnica de mezcla supone que las dos relaciones R y S están físicamente guardadas en secuencia por los valores del atributo de junta C. Si este es el caso, las dos relaciones pueden ser revisadas en secuencia física, las dos revisiones pueden ser sincronizadas y la unión completa puede ser realizada en una sola pasada sobre los datos.

- El agrupamiento físico de los datos relacionados lógicamente es uno de los factores críticos de desempeño.
- En ausencia de tal agrupamiento, a menudo es buena idea ordenar una o ambas relaciones en tiempo de ejecución.

## Dispersión

Al igual que la técnica de mezcla, la técnica de dispersión requiere de una sola pasada sobre cada una de las dos relaciones.

La primera pasada construye una tabla de dispersión para la relación S sobre los atributos de junta S.C; las entradas de esa tabla contienen el valor del atributo de junta y un apuntador hacia la tupla correspondiente en el disco.

La segunda pasada revisa la relación R y aplica la misma función de dispersión al atributo de junta R.C.

La gran ventaja de esta técnica sobra la de mezcla es que las relaciones R y S no necesitan estar guardadas en ningún orden específico y no es necesario ordenamiento.

## Objetivos

- ✓ Optimización: Definición.
- ✓ Estimación de estadísticas.
- ✓ Buenas prácticas.
- ✓ Vistas Materializadas.
- ✓ Implementación de operadores relacionales:
  - ✓ Fuerza bruta.
  - ✓ Búsqueda con índice.
  - ✓ Búsqueda con dispersión.
  - ✓ Mezcla
  - ✓ Dispersión.

**¿Dudas, consultas?**

Unidad 3