

Resumen BD Parte I

Capítulo 1

• Problemas surgidos antes de las BD:

- * Redundancia
- * Dificultad de acceso (a la info)
- * Dependencia de la estructura de almacenamiento
- Def: SGBD: nivel de software q' se encuentra entre los archivos q' componen la BD y los programas de aplicación

• Funciones del SGBD

- * Consulta + Actualización de los Datos
- * Mantenimiento de Esquemas → Def: desc. de la estructura de la info almacenada en DB.
- * Manejo de Transacciones. → Def: programa de aplicación q' accede y actualiza una parte también generalmente pequeña de la BD.

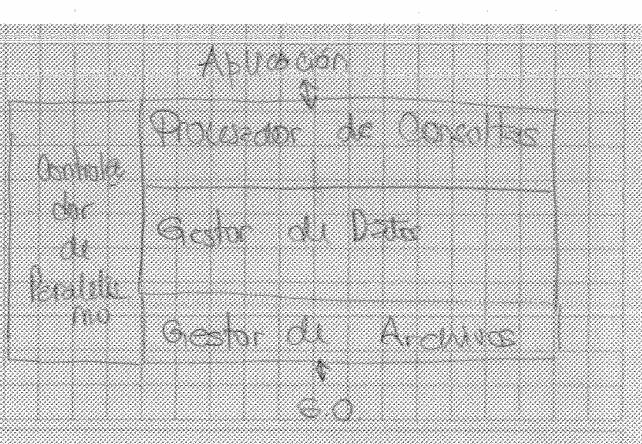
• Funciones de abd (Adm. de BD)

- * Def. el esquema
- * Def. estructuras de almacenamiento
- * Modificación de estructuras Físicas + Lógicas
- * Autorización de Acceso
- * Especificación de Restricciones

• Arquitectura de un SGBD

Módulos q' forman parte

- * Gestor de Archivos
- * Gestor de Datos
- * Procesador de Consultas
- * Controlador del Paralelismo
- * Sistema de Recuperación



DEPENDENCIA FUNCIONAL

- * Def: restricción si el conjunto de tuplas q' pueden aparecer en una relación
 Sea R un esquema de relación y X, Y dos subconjuntos de R \Rightarrow
 $X \rightarrow Y$ y pueden existir 2 tuplas $t_1, t_2 \Rightarrow$
 $t_1[X] = t_2[X] \quad \wedge \quad t_1[Y] \neq t_2[Y]$

* x es superclase de R \equiv $x \rightarrow R$

Axiomas de Armstrong

- Reflexividad: Si $Y \subseteq X$, inferir $X \rightarrow Y$
 - Aumento: P/ cualquier conjunto W , de $X \rightarrow Y$ inferir $XW \rightarrow YW$
 - Transitividad: De $X \rightarrow Y$ e $Y \rightarrow Z$, inferir $X \rightarrow Z$

* Dado un conjunto de df's F, sea lo ~~extensivo~~, def, denominado F^t el conjunto de todos los df's q' median inferirse de F usando los axiomas de ~~df's Armstrong~~

$$F^+ = \{x \rightarrow y \mid F\} \models x \rightarrow z y\}$$

* lascusura de un conjunto de atributos: conjunto de att A-E.R.tg' $X \rightarrow A$
puede ser derivado desde F utilizando los axiomas.

$$X_F^+ = \{ A \in R \mid F|_A = X \rightarrow A \}$$

* Sea F un conjunto de df's y $X \rightarrow Y$ una df, luego $F \vdash X \rightarrow Y$ si $\forall x$

\Leftarrow $F \circ G$ son équivalences si $F^{-1} = G^+$

* Cumplimiento Minimal: conjuntos fm. equivalentes a F, pero con la cond. q' sea mínima, q' no haya at. redundantes en las dependencias ni tengan dependencias redundantes.

Fin de cuestionario minimal si Fase 4

- 1) todo la divisi^{on} de fm tiene un! att
 - 2) si se ^{un} a ^{un} a ^{un} es redundante, no contante att redundante.
 - 3) fm no contiene ^{un} redundante.

Formas Normales

corresponden esquemas de relación desmonstrados de det. tipos de anomalías.

PRIMERA FORMA NORMAL.

los componentes de c/u tupla son valores atómicos, y no conjuntos o tup.

SEGUNDA FORMA NORMAL.

TERCERA FORMA NORMAL (preserva las df) es más recomendable que FNBC

Un esquema de relación R está en 3FN si para todo df no trivial $X \rightarrow Y$ o X es superclase de R o Y es un subconjunto de alguna clase de R.

CUARTA FORMA NORMAL.

Un esquema de relación R ~~debe~~ df a dmr está en 4FN si $\forall dm_{R'} X \rightarrow Y$ no trivial X es superclase de R (un esquema que satisface 4FN también cumple 3FN y FNBC)

QUINTA FORMA NORMAL.

Un esquema de relación R y un conjunto de df a df's D. Se dice que R está en 5FN si $\forall df$ se cumple:

* la df es trivial

* la df puede inferirse a partir de las df que poseen como li. alguna clase de R

FORMA NORMAL DE BOYCE-CODAS (no pierde info pero no preserva las df)

Un esquema de relación R está en FNBC si $\forall df X \rightarrow Y$ o $Y \subseteq X$ o X es superclase de R. Es decir que si df no trivial X debe ser superclase de R.

Estructura Física de Datos

* Manejo de Buffers: la estrategia óptima para el manejo de buffers es la opuesta a las LRU.

Si el gestor de buffers cuenta con suf. info de alto nivel sobre q' vta intentando hacer el sistema, puede seleccionar estrategias de mayor efecto.

* Representación de Relaciones

- máquinas

- ~~clases~~ clícas: en un archivo con reg. de long. fija.

- máquinas

- ~~excepcion~~ grandes: en un archivo se pueden combinar vs relaciones ~~entre~~

- cuando si agrupan A/ el valor de algunos de sus atributos: esquema de "agrupa
mismo" (clustered)

* Diccionario de Datos

Además hay q' almacenarlo. Pero puede considerarse como otra BD
y manejarse con los mismos métodos.

* Indexación

a) Método Secuencial Indexado: archivo almacenado sec. en orden de
número primaria. Además se construye un índice en un archivo auxil.
Este es costoso de mantener cuando hay altas, bajas y modificaciones.
Cuando se llena un bloque se sueltan en el índice al principio
el overflow y punteros q' apuntan desde un bloque del archivo a
ese listo encadenado de bloques de overflow.

b) Índice Denso: archivo que no requiere secuencialidad, solo sigue
secuencialidad del índice. Tiempo t entrada en el índice x/c reg.

c) Árbol B: índice de múltiple nivel, garantiza q' no se desperdicia
espacio

* Hashing: permite calcular inmediatamente el bloque de disco donde se
halló el reg. con esa clave.

Lo q' se hace es una f. q' simula una asignación aleatoria con dist.
uniforme de claves a bloques, de modo q' ningún bloq. se cargue
excesivamente de reg.

* Hashing extensible

Acceso x campo vs clave: se pueden crear índices o emplear f. de hashing
vs claves, con lo q' existan múltiples reg. con un mismo
campo, en el q' caso no se cumple esto. Tmb han sido técnicas excluyentes

Capítulo 4 - Procesamiento de Consultas

* Costo de Procesamiento

- Comunicaciones: tránsferencia de datos entre donde están almacenados y donde se realiza el cálculo.

Factores

- Acceso a memoria secundaria: transferir bloques de datos de memoria principal a memoria secundaria y moverse.
- Tiempo del Procesador: costo de un bucle de unidad central de procesamiento.

* Etapas en el procesamiento de consultas

1º Análisis Sintáctico: traducir la consulta a una forma interna que sea fácilmente manipulable.

2º Optimización de Consultas

a) Transformar la forma interna de la consulta en otra equivalente más económica de ejecutar.

b) Plan de Acceso: tomando el resultado de a) y teniendo en cuenta las estructuras físicas (índices, etc.) querer una estrategia detallada p/ la ejecución de la consulta.

* Transformaciones

• Selección: aplicar los op. selección tan tanto como sea posible. Reducción de tuplas.

• Junta natural: usar un ordenamiento óptimo de los op. de junta. La junta es asociativa y conmutativa.

• Proyección: reduce el tamaño de cada tupla y \Rightarrow el resultado total de los resultados intermedios.

* Estimaciones del Costo de Procesamiento

- Depende de
- Cardinalidad de la relación R: n_R
 - Long (en bytes) de la tupla de R: l_R
 - P/c de A y de la relación R, la cardinalidad de T(A|R), cant de valores de A q' aparecen en R: $V(A|R)$

- Producto Cartesiano: $n_R \times n_S$ n de tupla ocupa $l_R + l_S$ bytes.

- Selección: cardinalidad de $S_{A=k}(R)$: $n_R / V(A|R)$ (A aparece con dist. uniforme)

- Junta R x S: Se puede estimar como $(a) n_R n_S / V(A|R)$ siendo posible también usar $n_R n_S N(A|R)$ (b)

Si (a) es muy < de (b) \Rightarrow es plausible suponer q' una de las relaciones contiene sólo algunos de los valores de A q' aparecen en la otra y en este caso la menor de los estimadores es la más exacta. Los resultados, ambos teóricos, est. altos q' tienen pocos val de A en común y se dice más info p/ el otro.

* Indices

Hay 2 factores a considerar:

- Presencia de Índices: si tenemos q' resultados: select A₁, ..., A_n from R where P₁ and P₂ and P₃... y uno de los P_i es de la forma A=k y 3 un índice si el att A es probablemente ventajoso obtener todos los tuplos q' satisfagan esta cond usando el índice.
- Tipo de Índice - "Índice de Agrupamiento" (clustering index): indica si un att tq las tuplas q' tienen el mismo valor en ese att tienden a aparecer físicamente contig.

Consideremos todos los estrategias q' consisten en una de las 2 posibilidades siguientes:

- Elegir un P_i obtener todos los tuplos q' satisfagan P_i y verificar cuales de ellos satisfagan los demas predicados.
- Hacer un barrido a la relación completa verificando que los tuplos satisfagan todos los predicados.

Calcularemos el costo de q' posibilidades tomando en cuenta la presencia del índice y si se trata o no de índices de agrupamiento y finalmente elegiremos como plan de ejecución aquellos q' resulte de menor costo.

* Métodos de Cálculo de Juntas

- Iteración Ingenio: no hace nada, solo examina los pares de tuplos post.
- Iteración x Bloques: realiza la comparación con los bloques en memoria devanta la mayor cant de bloques posibles de una relación y va comparando con los bloques de la otra.
- Método de Sort-Merge: si ninguna de las 2 relaciones entra en memoria se los puede ordenar (sort) de acuerdo al atributo de juntar y luego calcular la junta mediante un barrido secuencial en ambas relaciones (merge). Costo: cant de bloques ocupados x q' relación + costo del sort B. log B
- Indices: Este método resultaría más costoso q' la iteración bloques a menos q'
 - los tuplos de uno de los relaciones no se encuentre almacenado
 - solo una fraccion de los tuplos participan en la junta.

2) Hash

- Método Simple Junta Hash: Aplico una f_c de hash a la relación R y supuestamente la mas directa sobre los att en cuestion, se reporta q' la dist q' genera esta f_c es uniforme. En la tabla q' genera en memo se pueden almacenar los tuplos o los id's (identificadores de tupla). Esta f_c transforma el att en una direc dentro de la tabla.

B₁ + B₂

Obligatoriamente hay q' considerar algún método para la resolución de colisiones.

Esta primera fase se denomina "Fase Constructiva".

Si existe con una segunda fase "Exploratoria" que aplica lo q' de hasta a la otra relación S, de este manera es posible ubicar los tuplos de R que se corresp. con ~~se~~ q' corresp. allí de S.

El costo del método es la suma de lo cost. de bloques de ambas relaciones.

ii) Método de "Greco" (cuando las relaciones no entran en memoria ppal).

✓ Existe 2 posibles funciones

(B_k+B_s)x2 a) Función de particionamiento q' divide los 2 relaciones en 2 subconjuntos distintos. Esta función aplicada a cualquier tuplo produce un resultado entre 0 y M-1.

b) Se juntan los q' corresp. particiones usando el "método simple de junta hash".

* Junta de Múltiples Relaciones. - PIPELINE

Una posibilidad es juntarlas como fuentes binarias y resolverlos con los métodos descritos anteriormente.

Algo q' también vale hacerse a veces es emplear el método del "pipeline". Es decir q' se ejecutan los fuentes y se mantienen los resultados intermedios q' irán empacados individualmente a cada una de los fuentes. Esto puede beneficiar a algunos tuplos y no a otros.

* Método de Índice de Junta con Mapa de Bits.

i) Índice de Tupla: es una relación binaria q' identifica cada una de los tuplos resultado de la junta & medio de ~~que~~ punteros a las respectivas tuplas de los relaciones q' forman parte de la junta. Algo q' los punteros pueden ser adresas direc. de memoria o cualquier identificador q' ubique de manera única a q' de los tuplos. Una vez confeccionado el índice, se realizan los siguientes pasos:

- Se leen los punteros de una tupla q' ~~se~~ a los tuplos de los relaciones involucrados.
- Se concatenan los campos de los 2 tuplos y se genere lo tupla resultado q' será almacenado.

El costo de este método es lo lectivo de cada una de los tuplos involucrados q' se lea el puntero q' acceder a los mismos.

ii) Índice de mapa de Bits: supongamos q' tengo una relación que posee una columna A cuya |dom(A)| = M q' que la relación posee N tuplos. Si denominara matriz de mapeo del bits a los M vectores binarios de long. N. Queda q' q' de los bits del vector represente una fila de la tabla, q' poseiere 1 en si q' valle 0 y en 0 en caso contrario.

Una ventaja de este método es q' los vectores son fácilmente comprimibles y ademas se redimensionan los entradas y salidas al dispositivo de almacenamiento.

Falta explicar: permite juntar los fuentes de una selección con múltiples fuentes.

Cap 8

Control del Paralelismo y Recuperación

* Transacción

Pasos de:

Durante de la
Transacción

Abort / Commit

Def: sucesión de acciones. No puede hacerse paralelamente
Non gral consiste en 2 operaciones:
a) op. de BD (ej: leer, guardar)
b) op. de transacción (ej: Abort o Commit)

trx termina
"OK"

Se intela implícitamente al ejecutar una sentencia q' requiere el trx para
trans

* Propiedades A.C.I.D

prop. provista x el mecanismo de control del paralelismo de un SGBR

Atomidad: "una trx se ejecuta completamente o no se ejecuta"

el resultado de una trx es un estado consistente

Consistencia: "una trx transforma la BD de un modo consistente a otro modo consistente".

"Estado consistente": aquel q' satisface todos los cond. de integridad

Aislamiento: q' trx se ejecuta aislado, sin interferencia con otra trx.

"Conjunto de Trx Aislado": el efecto de su ejecución en paralelo es exactamente el mismo q' si los trx se ejecutaran uno a uno.

Durabilidad: garantiza q' las actualizaciones a la BD realizadas x trx q' ejecutan commit serán durables y públicas. Es decir los cambios serán almacenados en memoria perdurable y serán visibles x otra trx.

* Ejecución de Transacciones

* Transacción Correcta: la consistencia de la BD se satisface antes y luego de haber ejecutado la trx.

* Ejecución Entrelazada Correcta de un conjunto de trx: Si es equivalente a alguna ejecución serial

* Transacciones Equivalentes: 2 ~~trx~~ ejecuciones de trx E₁, E₂ son equivalentes si:

1) A partir del mismo estado inicial, E₁ y E₂ producen el mismo estado final de la BD.

2) Sea T_i la trx q' ejecuta un cierto par de lectura del ítem en E₁ y E₂, el valor leído debe ser el mismo en ambas ejecuciones.

* SERIALIZABLE: una ejecución de trx es serializable si es equivalente a una ejecución serial.

"TODA EJECUCIÓN SERIALIZABLE PRESERVA EL ESTADO DE LA BD"

"EL OBJ DEL MÓDULO DE SIST q' CONTROLA LA EJECUCIÓN DE LAS TRX EN PARALELO, PLANIFICADOR (SCHEDULER), ES ASEGURAR q' TODAS LAS TRX SEAN SERIALIZABLES"

x Candados y Exclusión Mutua

- **LOCKING (cierra):** El efecto de un pedido de cierre s/ un ítem x una trx es lo obtención de un candado (lock) s/ un ítem x en este ítem.
- Notación: $LL(x)$: pediras de lectura s/ un ítem (x)
 $LE(x)$: " " escritura " " " "
 $LI(x)$: liberación de x

- Para garantizar la exclusión mutua:

1. Una transacción no puede leer ni actualizar un ítem x si/tauto no haya ejecutado la operación $LL(x)$ v $LE(x)$

2. Una trx q' intenta obtener un candado s/ un ítem x q' ha sido cerrado x un trx T debe esperar h/ q' T ejecute la op. $LI(x)$.

Esta tiene un problema y es q' si los 2 trx son de lecturas, une de ellos quedará bloqueada sin la necesidad de exclusión mutuamente \Rightarrow

Para garantizar la serialidad, se modifica lo reglo 2:

2. Una trx q' desea obtener un candado en modo M s/ un ítem x q' ha sido cerrado x otra trx T con un modo q' conflictiva con M debe esperar h/ q' T libere el ítem x.

* Corazón de Ejecuciones Serializables - ¿una ejecución E1 es serializable?

- **Protocolo de cierra:** es un conjunto de reglas s/ el uso de candados b/g Locking Protocol. Si todo trx A compromete a seguir estas reglas, el protocolo garantiza q' la ejecución es serializable.

- **Grafo de precedencia:** contiene un modo x q' trx T_i . Si T_i ejecuta una op. de cierra s/ un ítem x en modo M y T_j ejecuta posteriormente una op. de cierra s/ el mismo ítem x en un modo q' conflictiva a M \Rightarrow G contiene un arco que va de T_i a T_j .

- **Teorema:** Una ejecución es serializable si su grafo de precedencia es aciclico

* Cierre de Dos Fases (degrade: el desempeño de ejecución, es decir, el paralelismo)

- **Protocolo Dos Fases:** Ninguna trx puede hacer una op. de cierra una vez q' ha liberado algún ítem q' había cerrado.

Si como q' inicio una trx s/ fase uno en lo cual no obtiene los candados q' pide en la fase 2, liberando alguno q' ha liberado en la fase 1, se pide q' cierra esta op. de cierra (q' q' LE) s/ ningún ítem.

- **Teorema:** Supongamos q' todas las trx T_1, T_2, \dots, T_n obedecen el protocolo de cierra de 2 fases. Toda ejecución de este conjunto de Trx es serializable.

Este teorema es más sencillo y necesario q' cada ejecución de un conjunto de trx sea serializable.

Que sea cond. nec. no significa q' si un conjunto de trx no es de dos forma necesariamente q' uno ejecución no serializable de dichos trx.

Es posible demostrar q' dado cualquier conjunto de Trx T_1 , Si al menos 1 de ellos no es de 2 fases q' una transacción adicional T_{n+1} , tq' es posible ejecutar $T_1 \dots T_{n+1}$ en forma no serializable.

• Bloqueos x error - Problemas del Protocolo de 2 fases

* Los protocolos

* Los errores de ejecución como ctrl de paralelismo traen 2 problemas:

- 1) Bloqueo (dead lock): sit. en lo cual 2 o más trx quedan esperando indefinidamente x la liberación de un conjunto de condado. Para resolver esto hay 2 métodos:

"Junz sit. de Bloqueo si el grifo es cíclico"
a) "Detección": el syst. examina periódicamente los trx en ejecución. Si comprobaba la existencia de dos o más trx. bloqueados, el sistema aborta uno de ellos, VICTIMA, la q' debe recomenzar.

Hay 2 o más trx bloqueados si no han salido del estados de espera luego de hacer transacciones mut superior al um.

b) "Prevención"

- 2) Postergación Indefinida (livelock o indefinite postponement): ocurre cuando una trx está esperando q' se libere un condado y q' una vez liberado intenta obtener otra trx de forma q' en indefinidamente. La solución, un fácil es incorporar un sistema de colas p/c de los trx.

x Niveles de Aislamiento

Basta la dif. de los protocolos q' permanentemente no desminuye el rendimiento de los trx, se han def. niveles de aislamiento;

- 1) Lectura No Committed (Read Uncommitted)
- 2) Lectura Committed (Read Committed)
- 3) Lectura Repetible (Repeatable Read)

A) Serializable.

Fenómenos q' pueden producir un comportamiento anómalo:

- a) Lectura Sucia (Dirty Read): Si una trx lee un valor modificado x otra trx q' aún no ha ejecutado el commit y antes de hacerlo abierta la primera trx ha empleado un valor sucio.

- b) Lectura No Repetible (Unrepeatable Read): cuando una trx T_1 lee un dato y cuando intenta leerlo de nuevo el dato ~~ha sido~~ no ha sido modificado x otra trx q' ha realizado commit. El dato leído nunca podra ~~reflejarse~~ ser eliminado.

- c) Fantasma (Phantom): cuando una trx T_1 lee un conjunto de tuplas

como resultado de una búsqueda, si luego una trx T2 crea nuevos tuplos o modifica alguno q' tmb satisfagan la búsqueda realizada \Rightarrow si T1 efectúa la búsqueda nuevamente el resultado obtenido será #.

- 1) exhibe los 3 fenómenos descriptos y consecuencia de ejecución al mismo tiempo q' se ejecutan concurrentemente.
Se usa cuando los tablos son estáticos o cuando la vel. es más importante q' la exactitud + creto de exactitud
- 2) exhibe los fenómenos de lectura no repetible y fantasma.
- 3) exhibe el fenómeno de fantasma
- 4) no exhibe ningún fenómeno, si garantizo la serialidad de la ejecución.

* Recuperación Transaccional

- Recuperación: actividad de asegurar q' los fallos de soft. y hard. no corrompan los datos persistentes. El responsable de esto es el GR.
- Gestor de Recuperación - GR - Recovery Manager, responsable de garantizar la atomicidad y durabilidad a una trx. Se ocupa de los abortos, de los commit. y de reiniciar.
- Reiniciar (Restart): restaura una BD ante una falla del sistema, a un estado consistente inmediatamente anterior a la falla.

* Tipos de fallas:

- 1) de transacción: cuando una trx aborta
- 2) de sistema: cuando se afecta de manera impredecible la memoria volátil
- 3) de Medio: cuando se afecta la memoria no volátil.

• ¿Cómo hace el GR para cumplir con su misión cuando ocurre un fallo de sistema?

Utiliza un archivo auxiliar "bitácora" o log, en el q' registra los cambios q' producen las trx a la BD.

* Gestión de Logs o Bitácoras

- Log = archivo secuencial q' contiene todos los datos relativos a los cambios y cambios efectuados a cualquier tabl. de la BD.
- Gestor de Logs (GL) Log manager: componente del mot. q' se encarga de los logs de generación y lectura del log.
- Reg. del log: contiene:
 - a) Encabezado
 - b) Cuerpo (tamaño variable)
- Encabezado tiene: 1) Número de rec. del reg del log (LSN): q' es el nº del primer byte del archivo y este formado por el nº de archivo + direcc. relativa en bytes del reg. en el archivo.

- Rehacer (Redo): cuando hay actualizaciones brechas a un commit q' otros no fueron realizados en la memoria no volatile \Rightarrow los deben ser rehaciendo en la op. reiniciar.
- Idempotencia de Reiniciar: puede ocurrir una falla durante un reinicio \Rightarrow es nec. q' el reiniciar sea idempotente: cualquier secuencia de op. de una ejecución reiniciar incompleta debe tener el mismo efecto q' una ejecución completa de reiniciar.

* Algoritmos de Recuperación

- Algoritmo Deshacer/Rehacer - Algo de Recuperación con Actualización Inmediata en la BD:

Incorpora al log los registros listos: trx activas, trx q' lucieron commit y trx q' abortaron.
Tipos de reg's del log q' son utilizados:

- 1) Actualización: documenta un op. de grabación de unatrx.
- 2) Commit: expresa q' la trx llegó al commit.
- 3) Abort: expresa q' la trx llegó al abort
- 4) Checkpoint: documenta la realización de un checkpoint.

La op. de "Reiniciar" proesa el log en 2 pasadas. Una p/ "Deshacer" y proesa los reg's del log desde el final al el 1º check point q' encuentre o al el comienzo del archivo. En la 2º pasada "Rehacer" y proesa los reg's desde el 1º check point encontrado o desde el comienzo del archivo al el final del archivo.

"Deshacer" procede arrroen q' reg. de log:

- si es un reg. de commit, agrega el id en una lista de commit
- si es un reg. "abort" agrega en una lista de abort
- si es un reg. de actualización:
 - 1) si la trx está en la lista de commit, ignora el reg
 - 2) si no está en la lista de commit ni abort estuvo activa en la falla \Rightarrow p/ considerarla abortada la agrega en la lista de abort
 - 3) si la trx está en la lista de abort graba la imagen prima al item de dato en memoria cache si la trx. no tiene más reg's de actualización p/ deshacer, es eliminado de la lista de abort

"Rehacer" realiza p/ x q' reg. de actualización de una trx en la lista de commit, actualiza el item de dato en memoria cache.

* Checkpoints o Puntos de Control

P/ agilizar el proceso de "reiniciar" y la generación de los checkpoints nacen los "checkpoints binarios" o fuzzy checkpoints y se realizan durante los procesos de modificación de los objetos.

Para realizar un checkpoint binario, el GR:

- 1) No acepta ninguna op. de modif, abort o commit
- 2) Construye una lista con los pag. en memoria cache q' fueron modif. (pag. sucia)

3) Construye una lista de todas las trx activas con los LSN q' operaron al último log grabado del log p/c tx.

4) Acepta sucesivamente los op. de inicio, commit y abort

5) Mientras ejecuta los op. del paso 4) realiza flush de los log.s sucios, esto es hace el gestor de cache en lo mas q' tiempo.

Al grabar un log de check point al log, incluye la lista de trx activas

El siguiente check point solo puede hacerse luego del paso 6)

* Ricuperación del medio

Siempre se daña el disco duro, se suele recurrir a backups y al log de la BD p/ recuperarla, pero esto puede tardar mucho tiempo → Se usan, muchos veces, discos respaldados, q' se tiene una copia de respaldo actualizada al instante

También se usa la Tecnología RAID q' uso es discos redondos

• Copia de Archivo

Para hacer una copia de la BD es nec. copiarlo íntegramente. Pero esto puede tardar mucho tiempo → se hacen copias diferentes con los datos q' fueron modificados solamente.

Antes de hacer
un plazo de

Protocolos al respata el GR:

- un flush de una pg. & sincroniza el log con el buffer (LSN de la pg. q se quita)

 - Grabar Antes en el log (WAL - Write Ahead log) → Ver libro
 - Forzar el log en el conductor (FLC - Force Log at Conductor) - Establecer q los reg. del log deben ser grabados al mismo momento que se volatil como parte del Commit de la tx.
 - Dedicar q busca el check point mas reciente y borrar hasta el final del log p/ reaclararse así mismo. Luego ~~se~~ comienza la fase de reconstrucción p/ lo cual aplica un algo.
 - Deshacer (UNDO) es necesario hacerlo cuando se hace cambios en la ejecución de una tx q no llegó al commit. Se puede recomendar un estado anterior al un item de la BD a partir del ultimo cambio de ese item y reg. del log.

Resumen pl el final de temas puntuales

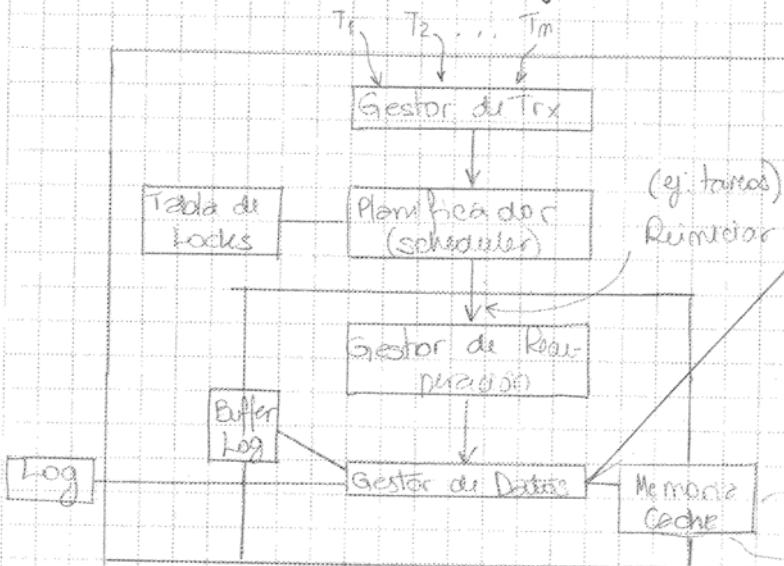
- Que no están en el libro

HOJA:

FECHA: Feb'08

* Protocolos de concurrencia (2 fases y Arbol)

- Def: "Control de concurrencia": es un ctrl q' hace el mgt. pl reinfcar q' 2 trx no molesten.
- Gestor de Trx que se encarga de distribuir y atender los trx.



Cuando uno trx hace un lock el gestor de trx va a log p/dibujar lo trx q'de hacer el lock q'de el start.

DISCO

Scheduler trabaja en 2 fases y q' las trx cumplen con el protocolo FASE 1: el scheduler le coloca a q' se lock y registra esto en la tabla. FASE 2: si ya tiene el lock q'de manda a ejecutar la sp. FASE 1: hace unlock cuando q' otra trx hace un commit o un update pag. de la sp.

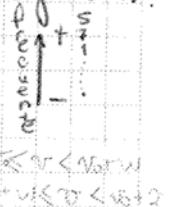
- Problema al Protocolo de 2 fases, cuando trabajo con índices xq' si yo accedo al índice y hago los locks de los nodos del comienzo (lockeo tmb) la raíz \Rightarrow cuando quiero acceder con otro tiempo q' esperar q' se desbloquee la raíz \Rightarrow para seleccionar el tiempo q' desbloquear la concurrencia \Rightarrow utiliza el protocolo de arbol.

* Protocolos de Arbol

- 1) Una trx puede aplicar un lock s/cualquier nodo, a partir de allí, cualquier otro nodo q' quiere lockear, tmb tiene q' lock su padre
- 2) En cualquier momento se puede hacer un unlock
- 3) A ningún nodo puedo bloqearlo 2 veces.

* Histogramas. (Estimación del tamaño del resultado de una junta).

- 1) Armar un histograma buscando valores más frecuentes



Ej: Histograma

Select Evento, Dia, Julio, Dia, Evento, Euro, Julio
Where Evento.Tiempo = Julio, Tiempo

Rango	Euro	Julio
0-9	0	40
10-19	0	60
20-29	0	80
30-39	0	50
40-49	5	10
50-59	20	5
60-69	50	0
70-79	100	0
80-89	60	0
90-99	10	0
#tuples	245	245

$$40 \cdot 49 \cdot 5.10 = 5 \\ 10 \\ 50 \cdot 58 \cdot 20 \cdot 5 = 10 \\ 10$$

→ ya que es el peor de los casos
que Euro=50, pero solo hay
un 1/10 de la cuenta en el
1/10 del rango es 40-49
5.10. 0.1

$$\text{Del otro lado de Euro: } \frac{\text{mín. m.s}}{\text{N(A,B)}} = \frac{245^2}{10 \cdot 10} = 600$$

Son 10 intervalos f.

Tipos

* Distribuidos Uniformes



* Distribuidos no uniformes



* Igualmente espaciados



Los locks nos brindan q' otras trx usen lo q' lockea. Se puede hacer lock
de una tabla, de una fila.
En gen' los sist. hacen lock de pag. p/ mejorar la performance.

- Tipos de locks:
 - Exclusivos: solo se le da a una trx a la vez
 - Compartidos: puede haber vs. trx simultáneamente con este tipo de lock. Solo se usa p/ leer.

- Lock de update: una trx quiere hacer un update del lock "I" →
"LU" tenta un lock compartido y ahora quiere escribir
A partir que lo obtiene al lock de update el sist. no permite
ningún otra peticion de lock compartido. Una vez q' solo el pase
al item puede estar zorro.

Matriz de Compatibilidad:

		Lock Pedido		
		S	X	V
Lock	S	Y	N	Y
	X	N	N	N
V	N	N	N	N

- Lock de incremento: Ademas de las op. de grabar y leer podemos tener una op de incremento: inc; (X) en donde se incrementa el item de la BD.

T1	T2
SL ₁ (A), L ₁ (A)	SL ₂ (A), L ₂ (A)
IL ₁ (B), inc(B)	IL ₂ (B), inc(B)
U ₁ (A), V ₁ (B)	U ₂ (A), V ₂ (B)

		Lock Pedido		
		S	X	I
Lock		S	Y	N
Previo	X	N	N	N
I	N	N	N	Y

- Lock de upgrade: LS: lock compartido
LX: lock exclusivo.

* Triggers

- Def: tipo de ~~store~~ procedure q' se ejecuta cuando un usuario trae al efectuar un cierto tipo de modif. s/ una tarea específica.
- Def. store procedure: conj. de instrucciones Transact-SQL y de control q' se almacenan en la BD y q' pueden ser ejecutados al ser invocados x su nombre.
- Uso
 - Reforzar la integridad referencial
 - Proporcionar métodos de bondad y actualización es costado.
 - Mantener actualizados datos redundantes en caso de existir
 - Implementar un sist. de log de aquellos acciones q' se deseó sig. (solo las actualizaciones ya q' los triggers no se llaman x select)
- Tipos
 - los q' se activan al tratar algún tipo de actualización q' lo toque en general sin tener distinción de campo
 - los q' se activan al efectuar la modif. q' un/los campos en particular

* Vistas

(se pueden crear vías de otras) la creación ya q' las vistas son tablas las ~~no~~ vistas q' no se puede / o no tiene sentido ordenarlas

- Def: es una tabla virtual, es decir, que al crearla no se está creando una copia de los datos, tan solo se crea la def. de una vista partiendo de los datos. Pueden estar def. si más de una tabla base.

* Uso y Beneficios:

- son usados como mecanismo de seguridad mediante el permiso de acceso a las vistas y no a las tablas propiamente dichas.
- simplifican el manejo de los datos
- proveen independencia lógica de los datos.
- brindan al usuario una percepción simplificada y personalizada de la BD.

* Conflictos en las actualizaciones:

Solo se permite la actualización de las vistas def. s/ una sola tabla base y q' ademas:

- 1) q' fila de la vista se corresponde con una fila + y distinguible en la BD
- 2) q' columna de la vista se corresp. con una columna + y distinguible de la tabla base.

- Al definir una vista no se puede especificar la cláusula ORDER BY

CREATE VIEW nombre AS
SELECT ...

- * "with check options" se coloca al final del create view p/ q' chequé al hacer un insert o modif. si cumple lo cond. de la vista

- 1) los valores q' tome la PK de la relación deben estar bien definidos, es decir q' no puedes haber un obj. (o tupla) del conj. desconocido.
NULL = desconocido

~~Regla 2)~~ Los valores q' tome una FK deben ser totalmente desconocidos o en su defecto deben tomar un valor existente en la relación donde es PK. También está prohibido q' sea parcialmente desconocido.

~~x~~ Restricciones de Dominio

- Hay restricciones de columnas, de filas, de tuplos, de tablas y de la BD.
- "Una PK es una restricción de tabla" y "Una FK es una restricción de la BD"
(ya q' involucra más de 1 tabla)
- Inconvenientes con los NULL

Ej 1: `SELECT * FROM cliente WHERE Loc = "Mer Del Plata";`

- = los clientes que viven en Mar del Plata (si no permite valores nulos en Loc)
- = los clientes q' sabemos q' viven en Mar del Plata, pero posiblemente me falta alguno (si permite valores nulos)

Ej 2: `SELECT * FROM cliente WHERE Loc = "MDP" UNION SELECT * FROM cliente WHERE Loc = "MDP"`

- = todos los clientes (si no permite NULL)
- = los clientes q' sabemos donde viven (ya no me da todo el universo) (si permite los valores NULL)

Ej 3: Salario

$$C_1 \text{ } 10$$

$$C_2 \text{ } 20$$

$$C_3 \text{ } \text{NULL}$$

$$\begin{aligned} \text{sum(salario)} &= 30 \rightarrow \text{implica q' c3 tiene salario} \\ \text{avg(salario)} &= 15 \rightarrow \text{u u u u u } | 15 \\ \text{count(salario)} &= 2 \\ \text{count(*)} &= 3 \end{aligned} \quad \text{Incompatibilidad}$$

R:

A B C

A1 B1 C1

A2 B2 C2

A3 B3 C3

S:

W A

w1 a1

w2 a2

w3 NULL

`SELECT * FROM r WHERE r.A NOT IN
(SELECT S.A FROM s);`

= P1, P2, NULL

No sabemos si a3 está o no en el IN ya q' hay un valor q' no conocemos \Rightarrow la respuesta nos da vacío.

- AL APARECER EL NULL PASAMOS A TENER UNA LÓGICA DE 3 VALORES $\begin{smallmatrix} & & \\ & F & T \\ V & T & F \end{smallmatrix}$
- Hay productos q' permiten ignorar el NULL p/ no pasar a lógica de 3 valores \rightarrow NO SABEMOS
- NULL empeoran la performance ya q' no aparecen en el índice (no tienen entradas) \Rightarrow me obligan a recorrer todo la tabla.

- Restricciones



- con valores NULL
- con valores x defecto
- con checks p/ acotar el dominio.

SQL

- SQL representa todo valor nulo con la misma representación interna independiente de la DB, así se relacion q' se trate.

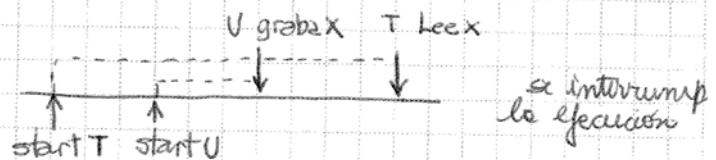
* Protocolo de TimeStamping.

Sí marca a la base de datos con 3 cosas:

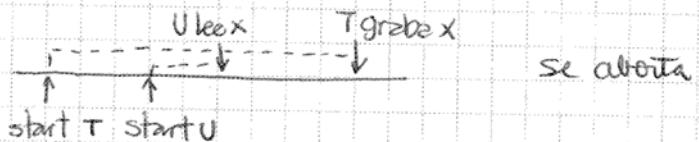
- LTS(x): time stamping de lectura de x (si 2 trx quieren leer x \Rightarrow se lip \wedge a la q tiene mayor TS)
- GTS(x): time stamping de grabación
- C(x): bitmap

Problemas:

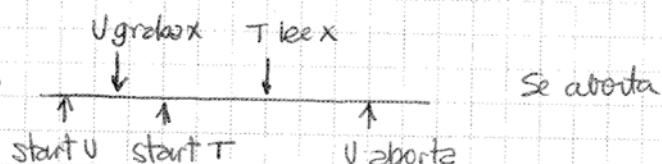
1) Lectura tardía.



2) Grabación tardía



3) Problemas con datos sucesos



Reglas de TimeStamping.

1) Pedido de Lectura

a) $TS(T) \geq GTS(X) \Rightarrow$ lectura físicamente realizabile.

i) Si $C(X) = 1 \Rightarrow$ se otorga el permiso

Si $TS(T) > LTS(X) \Rightarrow LTS(X) = TS(T)$

2) Si $C(X) = 0$ dilatar T si q' $C(X) = 1$ o Tx aborta

b) Si $TS(T) < GTS(X) \Rightarrow$ No es físicamente realizarible (Rollback de T)

2) Pedido de Grabación

a) Si $TS(T) \geq LTS(X) \wedge TS(T) \geq GTS(X) \Rightarrow$ efectua la grabación

1) grabar el valor de x

2) $GTS(X) = TS(T)$

3) $C(X) = 0$

b) Si $TS(T) > LTS(X)$ pero $TS(T) < GTS(X) \Rightarrow$ no es físicamente realizable.

c) Si $TS(T) \subset LTS(X) \Rightarrow$ no es físicamente realizabile (Rollback dir).

* DER - Dependencias Funcionales

No es posible a partir de un DER sacar las DF dado que no son lógicas, sino q' son deformadas x el nombre $\xrightarrow{?}$ reg., a pesar de q' ambas tienen q ver con el negocio.

* Creador del modelo Entidad-Relacional: Peter Chen 1976

• Los autores del " Relacional": E.F. Codd - 1970

• Fundamentos teóricos del modelo entidad - Interrelación

* u u " " Relacional, El cálculo Relacional

* u u " " SQL, Porte de su proj. de investigación al IBM

-* Fundamentos matemáticos del cálculo relacional:

$$\{x_1, \dots, x_n / \rho(x_1, \dots, x_n)\}$$

* u u " " álgebra relacional,

$$(S, \Pi, \cdot)$$

→ Es el uso de la lógica matemática d^o orden como herramienta descriptiva de los datos. Cualquier consulta escrita en Algebra Relacional puede ser expresada en el cálculo.

El cálculo y el álgebra son lenguajes equivalentes.

* CREATE TABLE

MESA	
FECHA	

* Blustering - Índices.

* Agrupamiento - Clustering: tipo de esquema q' agrupa tuplos s/ el valor de algunos de sus attr. De esta manera lo inf se puede acceder leyendo lo mismo cont. de bloques posible en forma secuencial.

La decisión de cómo y cuándo usar agrupamiento no es trivial y depende de las fases relativas de los tipos de consultas.

* Tipos del Índice → el archivo es ~~indexar~~ ^(a) ^{indexar} visto almacenado en forma sec.

(b) " " " " " " " " " " " "

(a) Archivo del archivo secuencialmente indexado tenemos un auxiliar llamado "índice" q' consiste en una clore y una direc. de bloq. Si el índice es suficientemente pequeño pl. residir en memoria → el costo q' tendremos es recorrer el índice al encontrar el reg. buscado. De lo contrario el índice implica ^{archivos} q' ocupo a disco. Se prefiere usar índices de índices. Una org. secuencial es costosa de mantener en presencia de altos, bajos y bajas fluctuaciones.

(b) "Índice Directo": se requiere secuencialidad del archivo q' consta de se tiene una lista de pares k_i : b_i q' el reg. con clore k_i se encierra en el bloq. b_i . Almacenando el índice en forma sec. podemos acceder cualquier reg. a un costo q' es un bloq. más q' el costo de acceder al índice. Pero este costo es superior al anterior q' debemos q' recorrer todo el índice al encontrar el clore k_i .

* Archivos B

Se utilizan pl. mantener índices al nivel múltiple, dado q' no desperdiamos demasiado espacio ya q' el resto tiene espacio p/ n puentes y la mitad será usada. Además el tiempo de búsqueda es logarítmico en la cont. de nodos.

* Hashing:

cuando la cont. de accesos debe llevarse al mínimo es deseable en bloques en esquemas así si ondula → lo fc. de hashing permite localizar inmediatamente el bloque de disco donde se halla el reg. b_i q' se desea es q' es fc. simple una org. aleatoria den dist. una forma de clores a bloques, de modo q' ningún bloq. se cargue excesivamente de reg. Resol. de colisiones

Estimación

Hay 2 estimaciones

a) cost. de tuplas

b) costo de una operación (junta, selección, proy, etc)

$$2) |G_{A=z_0}(R)| = \text{cost. de filas q' d att A toma el valor } z_0 \text{ en la tabla R} = \frac{m_R}{V(A, R)}$$

$$|R \times S| = \text{cost. de filas de la junta} = \frac{m_R \cdot m_S}{\min\{V(A, R), V(A, S)\}}$$

si el att es común en A

mdx{V(A, R), V(A, S)}

si todos A, ..., B son comunes

$$= \frac{m_R \cdot m_S}{\min\{V(A, R), V(A, S)\} \times \dots \times \min\{V(B, R), V(B, S)\}}$$

$$|R \times S| = m_R \cdot m_S$$

Además se pueden utilizar los histogramas para calcular estos valores

b) COSTO DE UNA JUNTA R x S

1) Iteración simple. x q file de R lee todos los filos de S

$$\Rightarrow \text{Costo} = B_R + m_R \cdot B_S$$

2) Iteración x bloque lee un bloq de R y lo compone con todos los bloques de S, luego el 2do bloq de R y los compone con todos los de S y así de

$$\Rightarrow \text{Costo} = B_R + B_S \cdot B_S$$

Si tengo mucha memoria q me entra 1 tabla en memoria \Rightarrow

$$\text{Costo} = B_R + B_S$$

Si en memoria me entran N bloques y uso N-1 p/R y 1 p/S \Rightarrow

$$\text{Costo} = B_R + \frac{B_R \cdot B_S}{N-1}$$

3) Sort-Merge. si ninguna de las 2 relaciones entra en memoria, se puede ordenar de acuerdo al att de junta y luego calcular la junta x medio de un merge secuencial de ambas relaciones ($B_R + B_S$)

$$\text{Costo} = B_R + B_S + B_S \cdot \log_2 B_S$$

\rightarrow costo de procedimiento

4) Índices de no agrupamiento; S, R tienen un att A. Si un índice S/A de cada entrada al índice (x q' valor A) puede dirigirnos a \neq filas de S

Para calcular el costo, considera el peor de los casos q' q' filo vlo en un bloq de \Rightarrow

$$\text{Costo} = \frac{m_R \cdot m_S}{V(A, S)} + B_R$$

5) Índice de Agrupamiento: Idem anterior pero ahora las filas est.
ordenadas físicamente \Rightarrow

$$\text{Costo} = B_R + m_R \frac{B_S}{V(A_S)}$$

OBS: conviene usar índices cuando no mez. revolver todo R.

6) Método simple de Hash cuando 1 clave entra en memoria

$$\text{Costo} : (B_R + B_S)$$

7) Grae:

$$\text{Costo} = 3(B_R + B_S)$$

Cant. de bytes del output: $m_R \cdot m_S (l_R + l_S)$

donde: l_R : long. en bytes de cl tupla de R.

m_R : tuplas q tienen la clave k

FB_R : factor de bloques de r. cant. de negs. q' entran en 1 bloq.

B_R : cant. de bloques nec p/ almacenar R $\frac{m_R}{FB_R}$

Cant. de bloques del output $\leq B_R \cdot m_S + B_S \cdot m_R$