
Magatzem de columnes

Característiques

PID_00253060

M. Elena Rodríguez González

Jordi Conesa i Caralt



Universitat
Oberta
de Catalunya

Almacenes de columnas: características

- Características de los almacenes de filas
- Características de los almacenes de columnas

EIMT.UOC.EDU

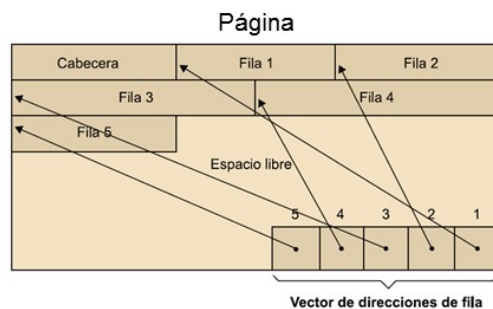
Benvinguts a la segona presentació d'aquest material didàctic dedicat als magatzems de columnes.

En aquesta presentació veurem quines són les característiques principals dels magatzems de columnes i les compararem amb les dels magatzems de files. L'objectiu és proporcionar els elements necessaris per a poder triar en quines circumstàncies és millor optar per un tipus o un altre de producte.

En les explicacions hem intentat ser genèrics i no restringir-nos a un producte específic. Quan això no ens ha estat possible, esmentem el producte en qüestió a què es refereixen les explicacions. En general, aquest producte és Vertica (i en gran manera també C-Store, el prototip del qual deriva). Aquesta elecció es basa en la implantació d'aquest producte i en el volum, detall i qualitat de la informació disponible en publicacions científiques de reconegut prestigi en l'àmbit de les BD. Malgrat tot, i com a recomanació general, no oblideu que abans de decidir quin producte s'utilitzarà, sempre és necessari consultar publicacions especialitzades i la documentació del mateix fabricant.

Almacenamiento por filas

| SALE | | | | | | |
|------|---------|-------------|------------|-----------|----------|-------------|
| | SALE_ID | CUSTOMER_ID | PRODUCT_ID | SALE_DATE | QUANTITY | TOTAL_PRICE |
| 1 | 10 | 2 | 1 | 20160112 | 3 | 200 |
| 2 | 20 | 1 | 3 | 20160107 | 5 | 1000 |
| 3 | 30 | 10 | 2 | 20160229 | 1 | 30 |
| 4 | 40 | 8 | 5 | 20160308 | 10 | 850 |
| 5 | 50 | 2 | 30 | 20160217 | 9 | 1500 |



Modelo de almacenamiento n-ario o NSM

EIMT.UOC.EDU

Els magatzems de files organitzen i guarden les dades en la memòria permanent per files, tal com es mostra a la part dreta de la figura, on podem veure que les files de la nostra taula d'exemple *SALE* es guarden seqüencialment una darrere l'altra en pàgines d'un disc. Aquestes pàgines, al seu torn, pertanyen a un fitxer. Aquesta estratègia d'emmagatzematge es denomina model d'emmagatzematge n-ari. També es coneix amb l'acrònim NSM que deriva del seu nom en anglès (*n-ary storage model*). Aquesta estructura d'emmagatzematge és la que, per defecte, segueixen els proveïdors de les BD relacionals més tradicionals (PostgreSQL, Oracle, SQL Server, DB2, Informix, MySQL, etc.).

En general, no s'atorga massa espai per a cada fitxer, sinó que l'espai es va adquirint de forma automàtica (sense intervenció humana) d'acord amb les necessitats, és a dir, a mesura que es van inserint files en la taula. La unitat d'adquisició d'espai és el que es denomina extensió, que no és més que un conjunt de pàgines consecutives en el disc. Per tant, i a manera de resum, les files es guarden en pàgines que, al seu torn, s'agrupen en extensions que pertanyen a un fitxer. Per defecte, les pàgines i les extensions són de longitud fixa, encara que en general l'SGBD permet definir aquesta longitud, en el moment d'instal·lar l'SGBD.

La pàgina també és la unitat mínima d'accés i transport del sistema d'E/S de l'SGBD. En altres paraules, per a resoldre les peticions formulades pels usuaris és necessari transferir les pàgines que contenen les dades d'interès des de la memòria externa (disc) a la memòria principal de l'ordinador que és on realment es resolen les

peticions. En el cas d'operacions de canvi (execució d'operacions d'INSERT, DELETE i UPDATE), les pàgines modificades hauran de ser bolcades al disc en algun moment.

En el cas més simple, les dades d'una taula es guarden en un sol fitxer i el fitxer només conté dades d'una taula. Aquest és el supòsit que es realitza en el nostre exemple. Com podem veure a la part inferior esquerra de la transparència, la pàgina té una estructura interna que consta dels elements següents:

1. Una capçalera, amb informació de control de la pàgina. Un exemple d'informació de control podria ser l'espai lliure que queda a la pàgina o el percentatge d'espai lliure mínim que es desitja mantenir a la pàgina (per exemple, per a absorbir futures insercions de files o increments en la longitud de les files emmagatzemades).
2. Un cert nombre de files, que pot variar al llarg del temps. Les files no es corresponen exactament amb les files del nivell lògic, per la qual cosa en ocasions reben el nom de registres. Aquestes files contenen, a més de la fila del nivell lògic, informació de control. En general, hi ha informació de control sobre la fila en global i sobre cadascuna de les columnes que incorpora la fila. Un exemple d'informació de control sobre la fila en global és la seva longitud total en *bytes* o la taula a què pertany la fila. Exemples d'informació de control sobre cada columna podrien ser la seva longitud en *bytes* (rellevant en columnes que tenen associat un tipus de dades de longitud variable, com seria el cas de les columnes de tipus `VARCHAR`) o un indicador que informi si la columna pren o no un valor nul (en el supòsit que la columna admeti valors nuls). En conseqüència, cada columna emmagatzema la columna de nivell lògic més la informació de control, per la qual cosa freqüentment reben el nom de camps.
3. Espai lliure.
4. Un vector de direccions (o adreces) de la fila (VDF), que té tants elements com files hi ha a la pàgina. Cada element del VDF conté un apuntador a l'inici de la fila dins de la pàgina. L'element que ocupa la posició de la direcció més alta en el VDF apunta a la fila que ocupa la direcció més baixa a la pàgina, tal com podeu veure a la figura. La fila següent (a continuació de la primera) està apuntada per l'element del VDF a l'esquerra del primer element, i així successivament. El VDF agilita l'accés a les files contingudes a la pàgina, sobretot en el cas de files de longitud variable.

Així doncs, i a manera de resum, les files es van col·locant d'esquerra a dreta en l'ordre creixent de les adreces dins de la pàgina, mentre que els elements del VDF es van col·locant de dreta a esquerra des de la posició més alta i seguint en l'ordre decreixent de les adreces. D'aquesta manera, l'espai lliure queda en la meitat de la pàgina.

Adicionalment, cada fila té un identificador assignat per l'SGBD (en anglès, *rowid* o *record identifier*), que no varia mentre la fila existeix en la BD. Els *rowid*, almenys, queden emmagatzemats en l'índex en forma d'arbre B+ que l'SGBD crea per defecte sobre la clau primària de la taula. L'identificador consta d'una sèrie de *bytes* i conté dues parts diferenciades. En primer lloc, conté informació que permet saber a quina pàgina està emmagatzemada la fila. I en segon lloc, conté informació que permet saber a quina posició del VDF es troba l'apuntador que ens permet accedir a la fila en qüestió dins de la pàgina. A manera d'exemple, assumint *rowids* de 4 *bytes*, i que els 3 *bytes* de més pes identifiquen la pàgina, i el *byte* de menys pes la posició del VDF, i que la pàgina que es mostra a la transparència és la pàgina número 17, el *rowid* de la fila 3 (expressat en binari) seria el que es mostra a la figura 1.

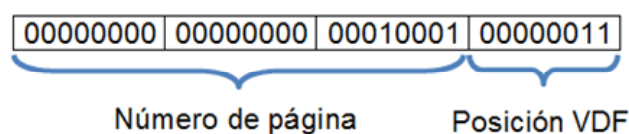


Figura 1. Exemple identificador de fila (*rowid*)

L'esquema d'emmagatzematge que acabem d'explicar cobreix el cas més senzill: les files de la taula s'emmagatzemen en un únic fitxer i aquest fitxer conté dades d'una única taula. Com ja hem estudiat en l'assignatura, la realitat pot ser més complexa, de tal manera que de vegades les dades de diferents taules s'agrupen en un únic fitxer i en unes altres les dades d'una mateixa taula es parteixen en diversos fitxers. Per tant, l'esquema d'emmagatzematge que hem explicat pot sofrir adaptacions que es tradueixen, en general, en l'addició de nova informació de control, sia a les pàgines i a les files, o en altres estructures de l'SGBD (com seria el cas del catàleg de la BD).

L'agrupació de taules en un sol fitxer pot succeir, sia quan les taules són molt petites (i no es desitja consumir massa recursos del sistema), sia quan s'accedeix freqüentment a les taules de forma conjunta (en aquest cas, la idea és tenir la combinació o *join* preconstruïda en el dispositiu d'emmagatzematge extern).

La divisió de les dades en diferents fitxers es pot fer aplicant la fragmentació horitzontal o la fragmentació vertical. La fragmentació horitzontal és especialment útil en el cas de taules amb un volum important de files i consisteix en subdividir una taula en grups de files disjunctes a través d'aplicar algun tipus de condició (per exemple, en funció del valor que pren una columna o grup de columnes de la taula, o de forma aleatòria i uniforme a través de l'aplicació d'una funció de dispersió, –o *hash*, en anglès– sobre una columna de la taula). Cada grup de files resultant d'aplicar la condició s'emmagatzema en un fitxer separat.

Per la seva banda, la fragmentació vertical divideix la taula en grups de columnes i cada grup de columnes s'emmagatzema en un fitxer separat. El benefici principal que s'espera obtenir de la fragmentació vertical és incrementar la ràtio d'informació útil

llegida, és a dir, intentar llegir les columnes estrictament necessàries per a resoldre les consultes formulades pels usuaris o programes. Quan s'aplica la fragmentació vertical, cada columna de la taula original només pot estar en un únic grup, a excepció de la columna o columnes que constitueixen la clau primària de la taula, que hauran d'estar replicades a cada fragment. Encara que persegueixen propòsits similars, no hem de confondre la fragmentació vertical amb l'esquema d'emmagatzematge dels magatzems de columnes (en què entrarem en més detall més endavant en aquesta presentació), tal com mostra l'exemple presentat a la figura 2.

En l'exemple, la nostra taula *SALE* ha estat partida en dos fragments (o si preferiu subtaules) denominats *SALE_FRAG1* i *SALE_FRAG2*: el primer (a la part dreta de la figura) conté les columnes *SALE_ID*, *CUSTOMER_ID*, *SALE_DATE* i *TOTAL_PRICE*, mentre que el segon (a la part esquerra de la figura) conté les columnes *SALE_ID*, *PRODUCT_ID* i *QUANTITY*. Cada fragment s'emmagatzema en pàgines separades (part central de la figura) i les pàgines pertanyen a fitxers diferents (en concret a dos fitxers, un per fragment), encara que això no s'apreciï a la figura. Com podem observar, l'emmagatzematge en el disc de cadascun dels fragments segueix sent per files, encara que es tracti de files més petites pel que fa a la taula original *SALE*. De fet, les files de cada fragment són subconjunts de les files de la taula original.

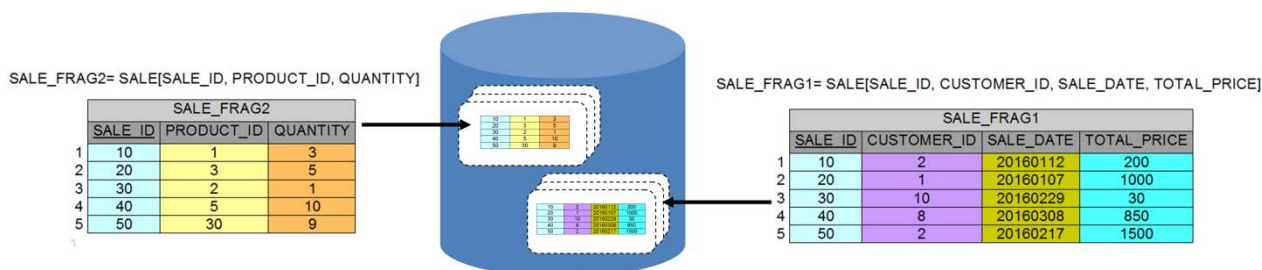


Figura 2. Exemple de fragmentació vertical en un magatzem de files

Clàssicament s'exigeix que l'esquema de fragmentació compleixi les propietats de completesa, disjunció i reconstrucció. La completesa garanteix que no es perdin dades com a conseqüència del procés de fragmentació. Per la seva banda, la disjunció indica que cada element de les dades (fila o columna) només pot estar en un fragment, a excepció de taules que hagin estat fragmentades verticalment. En aquest cas, com ja hem comentat, la clau primària de la taula haurà d'estar a tots els fragments. Finalment, la reconstrucció significa que la taula original s'ha de poder regenerar a partir dels seus fragments.

En essència, les propietats prèvies garanteixen, en primer lloc, que el procés de fragmentació preservi la semàntica de la taula (i per extensió de la BD) i les dades emmagatzemades en aquesta (completesa i reconstrucció). En segon lloc, garanteix que s'evitin redundàncies, és a dir, repeticions de les dades que serien evitables (disjunció). En relació amb aquest últim punt, recordem que les BD clàssiques (les operacionals, la manera de treballar preferida dels magatzems de files) promulguen la normalització de les dades.

Cal destacar que la fragmentació vertical és d'ús poc freqüent en els magatzems de files, perquè per tal que sigui efectiva és necessari saber l'afinitat entre les columnes (és a dir, és necessari saber a quines columnes s'accedirà de forma conjunta en les operacions de consulta i amb quina freqüència) i perquè, a més, exigeix la repetició de la clau primària a tots els fragments (això fa que es requereixi espai d'emmagatzematge extra), així com l'execució d'operacions de combinació (que acostumen a ser costoses) per a donar resposta a les peticions formulades sobre la taula que ha estat fragmentada (recordem que els usuaris/programes treballen sobre la taula de nivell lògic, és a dir, per a aquests és transparent el fet que la taula hagi estat fragmentada). A més, el nombre d'operacions de combinació a executar pot ser elevat si no s'ha dissenyat un bon esquema de fragmentació vertical. També complica l'execució de les operacions de canvi (`INSERT`, `DELETE` i `UPDATE`). Com a alternativa a la fragmentació vertical, els magatzems de files utilitzen índexs (aquests índexs es defineixen sobre les columnes que s'accedeixen més freqüentment) i vistes materialitzades. Tots dos elements també s'han d'utilitzar amb mesura, donat el cost que tenen associat davant els canvis en les dades (execució d'operacions d'`INSERT`, `DELETE` i `UPDATE`).

L'aplicació d'esquemes de fragmentació té interès tant en el cas de les BD centralitzades (aquest és el supòsit de la nostra assignatura) com en el cas de les BD distribuïdes. En les BD distribuïdes, els fragments s'usen com a unitat de distribució de les dades (és a dir, cada fragment s'emmagatzemarà, almenys, en un node de la BD distribuïda).

Adicionalment, les tècniques de fragmentació es poden combinar amb l'ús de la replicació. La replicació és una tècnica que s'utilitza en les BD distribuïdes. La replicació de les dades consisteix en mantenir diferents còpies (o rèpliques) idèntiques d'unes mateixes dades que s'emmagatzemen en diferents dispositius d'emmagatzematge extern.

A la figura 3, podeu veure un exemple d'ús combinat de replicació i fragmentació en un magatzem de files. Específicament, podeu veure com un dels fragments de l'esquema de la fragmentació vertical mostrat a la figura 2 ha estat replicat (el fragment en qüestió és `SALE_FRAG1`). Si us fixeu, es tracta de dues còpies (o rèpliques) idèntiques del fragment, és a dir, contenen les mateixes dades i aquestes estan guardades en el mateix ordre a cadascun dels dispositius d'emmagatzematge extern (disc).

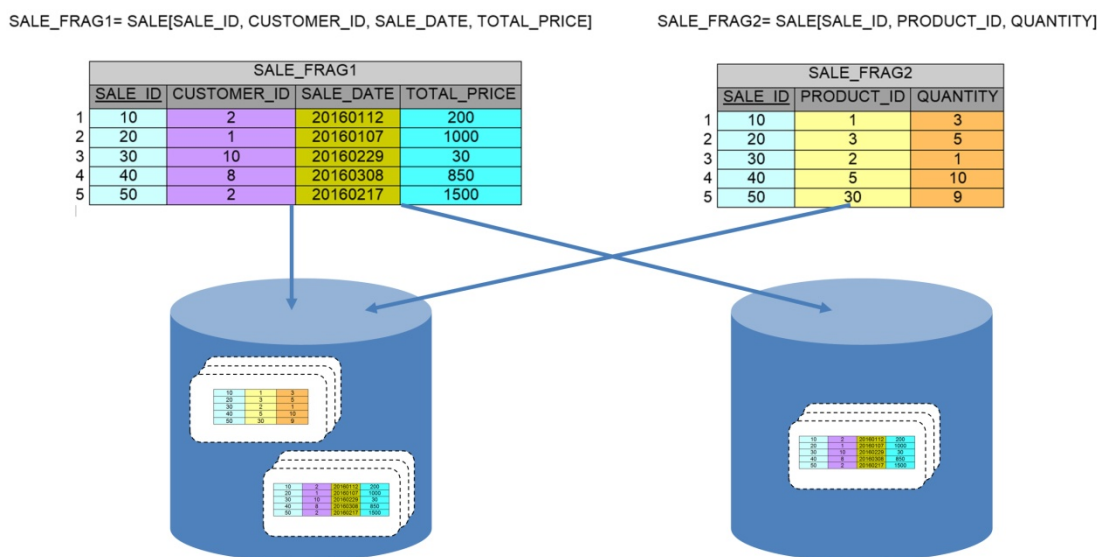


Figura 3. Exemple d'ús combinat de fragmentació vertical i replicació en un magatzem de files

La fragmentació i replicació de les dades en les BD distribuïdes tenen com a objectiu aconseguir principalment els beneficis següents: augmentar la disponibilitat de les dades, repartir la càrrega de treball entre els diferents nodes que formen la BD distribuïda, disminuir els costos de transmissió de les dades a través de la xarxa apropant les dades allà on es necessiten i millorar l'eficiència de les operacions de consulta (*SELECT*). Aquest últim avantatge està relacionat principalment amb l'existència de rèpliques. En contrapartida i com a desavantatges principals, cal destacar que el rendiment empitjora quan les operacions necessiten accedir a múltiples fragments emmagatzemats en diferents nodes i que la replicació introdueix complicacions en la resolució d'operacions de canvi (*INSERT*, *DELETE*, *UPDATE*) sobre la BD, atès que compromet la consistència (les rèpliques d'unes mateixes dades han de contenir els mateixos valors).

No és l'objectiu d'aquest material didàctic dedicat als magatzems de columnes entrar en més detall en les BD distribuïdes, atès que es tracten en altres assignatures. Malgrat això, aquells de vosaltres que estigueu interessats a obtenir informació addicional, podeu consultar la referència setena que s'inclou al final d'aquesta presentació.

Características de los almacenes de filas

- Las tablas contienen un número razonable de filas y columnas.
- Implementan de forma eficiente operaciones de cambio (`INSERT`, `DELETE` y `UPDATE`) sobre la BD.
- Implementan de forma eficiente consultas (`SELECT`) que verifican alguna de las siguientes condiciones:
 - Acceden a filas individuales completas o a un conjunto reducido de filas.
 - Las operaciones de consulta (`SELECT`) son simples: pocas tablas y no incluyen agrupaciones ni cálculos de agregados complejos sobre los datos.

Los almacenes de filas se orientan principalmente a entornos OLTP.

EIMT.UOC.EDU

L'estructura d'emmagatzematge que segueixen els magatzems de files els fa especialment adequats per a ser usats com a BD operacionals, és a dir, s'orienten a gestionar el dia a dia de les organitzacions (entorns OLTP).

En primer lloc, en general, les taules de la BD tenen un nombre raonable de files i de columnes. Pel que fa al nombre de files, si bé no existeix un límit per al nombre de files que pot tenir una taula (més enllà de les imposades per l'equip en què resideix l'SGBD), les taules d'una BD operacional acostumen a estar en ordres no superiors a desenes de milers de files, encara que potser el factor més rellevant sigui el nombre de files accedides per a cada operació executada sobre la BD, que s'estima està en ordres d'entre desenes i centenars de files. En relació amb el nombre de columnes, hem de pensar que les BD operacionals promouen l'ús de taules normalitzades. Això, en essència, significa que cada taula representa un únic concepte del món real, amb la qual cosa s'eviten redundàncies (repeticions que serien evitables) en les dades, fet que afavoreix una resolució eficient de les operacions de canvi sobre la BD (sobretot en el cas de les operacions de `DELETE` i `UPDATE`). Quantes més columnes tingui una taula, hi ha més probabilitats que la taula representi més d'un concepte del món real (és a dir, més probable és que la taula no estigui normalitzada). És per això que ens hauríem de plantejar si seria més convenient subdividir la taula en diverses subtaules, de manera que cadascuna d'aquestes subtaules representi un únic concepte. Però d'altra banda, treballar amb taules no normalitzades també té els seus avantatges en termes de rendiment de la BD. En concret, les consultes (operacions de `SELECT`) es podran

resoldre més eficientment atès que es podran evitar operacions de combinació (*join*). En resum, no és senzill determinar què significa que una taula tingui un nombre raonable de columnes, ja que és necessari considerar diferents factors. Com a recomanació general, i en el cas extrem, la mesura de cada fila d'una taula no ha d'excedir la grandària de la pàgina en què aquesta s'emmagatzema, ja que en aquest cas seria necessari realitzar més d'una operació d'E/S per a recuperar la fila. Per a aquesta aproximació sobre la mesura de la fila, estem considerant columnes que tenen associats tipus de dades tradicionals. Tal com hem estudiat en l'assignatura, les columnes amb tipus de dades no tradicionals (per exemple, les columnes amb tipus d'objectes grans, els LOB) s'emmagatzemen en pàgines separades.

En segon lloc, l'esquema d'emmagatzematge per files afavoreix la resolució eficient de les operacions de canvi (*INSERT*, *DELETE* i *UPDATE*) sobre la BD.

A manera d'exemple, per a inserir una nova fila a la nostra taula d'exemple (la sentència *INSERT* d'SQL es mostra a la part esquerra de la figura 4), l'SGBD haurà de seguir una sèrie de passos. Com a primer pas, haurà de localitzar la pàgina on s'ha d'afegir la fila. Aquesta pàgina es denomina pàgina candidata i hi ha diverses estratègies, com ja hem estudiat en l'assignatura, per a determinar quina serà aquesta pàgina (en el cas més simple, podria ser la primera pàgina amb espai lliure suficient per a absorbir la fila). Una vegada l'SGBD sap quina és la pàgina, utilitza l'espai lliure disponible a la pàgina per a col·locar la fila completa i deixar-la apuntada per un nou element del VDF.

| | |
|--|--|
| <pre>INSERT INTO SALE (SALE_ID, CUSTOMER_ID, PRODUCT_ID, SALE_DATE, QUANTITY, TOTAL_PRICE) VALUES (60, 2, 2, '20160319', 5, 150)</pre> | <pre>SELECT SALE.SALE_ID, SALE.CUSTOMER_ID, SALE.PRODUCT_ID, SALE.SALE_DATE, SALE.QUANTITY, SALE.TOTAL_PRICE FROM SALE WHERE SALE.SALE_ID=30</pre> |
|--|--|

Figura 4. Exemples de sentències SQL

Fixem-nos que una vegada localitzada la pàgina on s'emmagatzema la fila, l'operació d'inserció es pot resoldre amb dues operacions d'E/S (una operació de lectura per a portar la pàgina del disc a la memòria principal i una operació d'escriptura per a


bolcar la pàgina modificada al disc). La inserció també ocasionarà (si més no) que s'emmagatzemi l'entrada corresponent a l'índex en forma d'arbre B+ que l'SGBD ha creat sobre la clau primària de la taula (això, com ja sabem, pot desencadenar noves operacions d'E/S). També és possible que es desencadenin operacions extres d'E/S, si s'han definit altres índexs sobre la taula.

En tercer lloc, els magatzems de files també permeten resoldre eficientment operacions de consulta (`SELECT`) que accedeixen a files completes (una fila o un conjunt de files completes) o quan les consultes són relativament senzilles (afecten poques taules i no inclouen agrupacions o càlculs d'agregats complexos). A manera d'exemple, sobre la nostra taula *SALE*, una consulta com la que es mostra a la part dreta de la figura 4, que recupera totes les dades de la venda identificada amb el valor 30 es resoldria, una vegada localitzada la pàgina que emmagatzema la fila, amb una operació d'E/S. Per a localitzar la pàgina que emmagatzema la fila, l'SGBD usarà l'índex en forma d'arbre B+ creat sobre la clau primària de la taula (això pot desencadenar noves operacions d'E/S, com ja sabem).

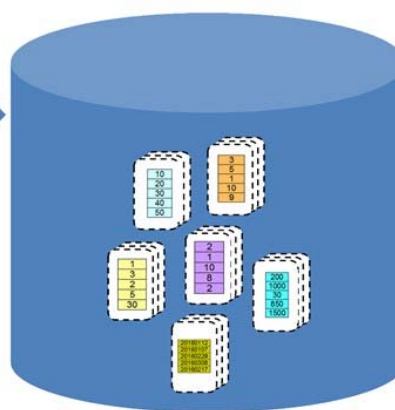
Per finalitzar, simplement recordar que un magatzem de files pot utilitzar diferents mecanismes per a millorar el rendiment en l'execució de les operacions de consulta: fragmentació, índexs sobre les columnes més freqüentment utilitzades en les consultes i vistes materialitzades. Aquests mecanismes permeten tractar amb un ampli ventall de consultes, més enllà de les que hem comentat que es resolen eficientment.

Almacenamiento por columnas

| SALE | | | | | |
|------|---------|-------------|------------|-----------|-------------|
| | SALE_ID | CUSTOMER_ID | PRODUCT_ID | SALE_DATE | TOTAL_PRICE |
| 1 | 10 | 2 | 1 | 20160112 | 3 |
| 2 | 20 | 1 | 3 | 20160107 | 5 |
| 3 | 30 | 10 | 2 | 20160229 | 1 |
| 4 | 40 | 8 | 5 | 20160308 | 10 |
| 5 | 50 | 2 | 30 | 20160217 | 9 |



| | | | | | | | | | | | |
|---|----|---|----|---|----|---|----------|---|----|---|------|
| 1 | 10 | 1 | 2 | 1 | 1 | 1 | 20160112 | 1 | 3 | 1 | 200 |
| 2 | 20 | 2 | 1 | 2 | 3 | 2 | 20160107 | 2 | 5 | 2 | 1000 |
| 3 | 30 | 3 | 10 | 3 | 2 | 3 | 20160229 | 3 | 1 | 3 | 30 |
| 4 | 40 | 4 | 8 | 4 | 5 | 4 | 20160308 | 4 | 10 | 4 | 850 |
| 5 | 50 | 5 | 2 | 5 | 30 | 5 | 20160217 | 5 | 9 | 5 | 1500 |



- Cada columna se almacena en un fichero separado.
- Tiene sus orígenes en el modelo de almacenamiento por descomposición o DSM.

EIMT.UOC.EDU

Les característiques distintives més significatives en l'estratègia d'emmagatzematge que segueixen els magatzems de columnes en relació amb els magatzems de files són les següents:

1. Les dades de cada columna s'emmagatzemen en un fitxer separat.
2. Les dades de les columnes s'emmagatzemen de forma ordenada.
3. Cada fitxer únicament emmagatzema dades d'una columna.
4. Es minimitza la necessitat de guardar informació de control a cada pàgina de dades.
5. No es guarden, de forma explícita, dades que permetin reconstruir les files (identificadors de files –*rowids* en anglès– o la clau primària de la taula).

Aprofundirem una mica més sobre cadascuna d'aquestes característiques a les transparències que veurem a continuació. Abans d'això, no obstant, cal destacar que alguna d'aquestes característiques pot no ser aplicable per a un magatzem de columnes concret o pot ser que s'apliqui de forma diferent a les explicacions realitzades en aquesta presentació. Per tant, sempre serà necessari revisar la documentació que cada fabricant proporciona en relació amb el seu producte.

El magatzems de columnes organitzen i guarden les dades en la memòria permanent per columnes, tal com es mostra a la part dreta de la transparència, on podem veure que cada columna de la nostra taula d'exemple *SALE* es guarda de forma separada. En concret, podem observar que els valors de cada columna es guarden seqüencialment un darrere l'altre en pàgines del disc. Aquestes pàgines, al seu torn, pertanyen a una extensió i s'associen a un fitxer. Per defecte, les pàgines i les extensions són de longitud fixa, encara que es tracta de paràmetres que l'SGDB permet configurar en el moment d'instal·lar l'SGDB. Òbviament, la pàgina segueix sent la unitat mínima d'accés i transport del sistema d'E/S de l'SGDB.

L'estratègia d'emmagatzematge que segueixen els magatzems de columnes té els seus orígens en el denominat model d'emmagatzematge per descomposició, també conegut amb l'acrònim DSM (*decomposition storage model*, en anglès) proposat a principis dels 80. Malgrat que aquest model va tenir una repercussió limitada en el cas de la indústria, l'interès es va veure renovat en la segona meitat de la dècada dels 90 a causa del desenvolupament dels primers magatzems de columnes. El model DSM es basa a portar a l'extrem l'aplicació de les tècniques de la fragmentació vertical que hem discutit anteriorment. En essència, consisteix en partir, per defecte, cada taula de nivell lògic en tants fragments com columnes tingui la taula. La columna que constitueix la clau primària (aquest model assumeix l'ús de claus subrogades, és a dir, parteix del principi que la clau primària estarà formada per una sola columna) queda exclosa de conformar un fragment separat, ja que figurarà a cadascun dels fragments creats amb l'objectiu de poder reconstruir les files de la taula.

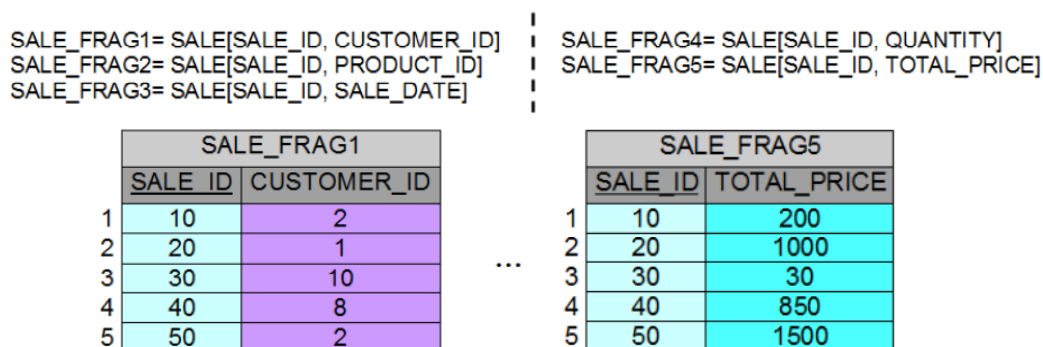
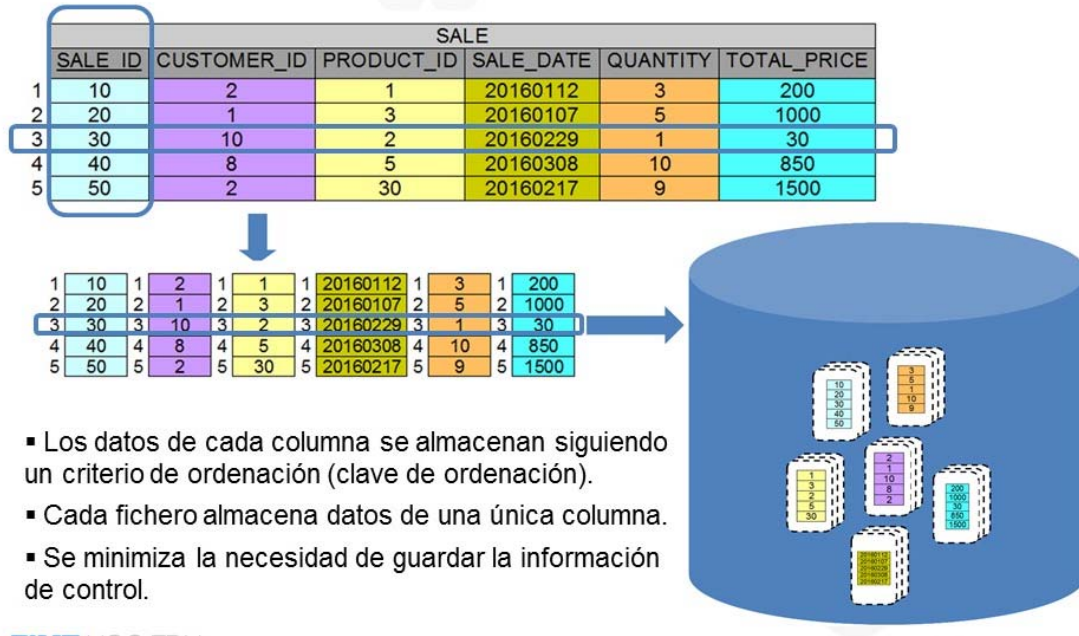


Figura 5. Model DSM per a la taula *SALE*

La figura 5 mostra, de forma esquemàtica, com el procés descrit es concreta en 5 fragments per al cas de la nostra taula d'exemple *SALE*. A més, a la figura, es mostra el contingut de dos d'aquests fragments a manera d'exemple (en concret, els fragments 1 i 5). Cadascun dels fragments creats segueix exactament el mateix esquema d'emmagatzematge que hem explicat anteriorment per als magatzems de files. Aquells de vosaltres que tingueu interès, podeu consultar la segona referència que s'inclou al final de la presentació per a trobar informació addicional sobre aquest model d'emmagatzematge.

Ordenación de datos



EIMT.UOC.EDU

Una altra diferència entre els magatzems de files i els magatzems de columnes és que, en aquests últims, els valors associats a les columnes s'emmagatzemen de forma ordenada en el disc. La situació més freqüent és que l'ordenació es realitzi en el moment de la inserció dels valors, en funció del valor o valors que prenen una columna o grup de columnes. La columna o columnes que es trien per a realitzar l'ordenació reben el nom de clau d'ordenació (en anglès, *sort key*). Aquest és el cas, per exemple, de Vertica (i de C-Store, el prototip en què es basa), d'Actian Vector (prèviament VectorWise) o d'Amazon Redshift. En altres casos, els menys freqüents, l'ordenació de les dades en les estructures d'emmagatzematge es realitza de forma incremental i parcial, en funció de les consultes formulades, i no del temps d'inserció (a manera d'exemple, aquesta és una característica distintiva de MonetDB pel que fa a altres fabricants). Així mateix, cal destacar que en el cas dels magatzems de files, tal com hem estudiat en l'assignatura, també hi ha la possibilitat d'emmagatzemar les dades ordenades a través de la definició d'un índex agrupat però, en general, no és una opció per defecte.

En el cas d'aquest material didàctic dedicat als magatzems de columnes, únicament cobrirem el primer supòsit, és a dir, assumirem que el magatzem de columnes guarda els valors de les columnes de forma ordenada en funció del valor de la clau d'ordenació que s'hagi triat. A més que és l'estratègia més comuna, la dificultat associada a les tècniques que permeten l'ordenació incremental excedeix els objectius d'aquest material didàctic. Malgrat això, aquells de vosaltres que desitgeu ampliar els

coneixements podeu consultar la segona i la cinquena referència que s'inclouen al final d'aquesta presentació.

En el nostre exemple, la columna utilitzada per a realitzar l'ordenació és *SALE_ID*, tal com es pot veure a la part superior de la transparència, on es mostra la visió lògica de la taula *SALE*. Si us fixeu, podeu veure que la fila 1 de nivell lògic de la taula *SALE* és la que té un valor més petit per a la columna *SALE_ID* (en concret el valor 10), la fila 2 conté el valor següent més petit (20) i així successivament fins a la fila 5 que és la que conté el valor màxim de la columna *SALE_ID* (concretament el valor 50).

Els valors de cada fila, una vegada segregats en columnes, ocupen exactament la mateixa posició a cadascuna de les estructures d'emmagatzematge creades. A manera d'exemple, si ens fixem en la fila 3 de nivell lògic (vegeu la part superior de la transparència), els valors associats a cadascuna de les columnes que componen la fila, una vegada segregats (observeu la part inferior esquerra de la transparència), ocupen sempre la posició 3 de cada estructura d'emmagatzematge (en el nostre exemple, es tracta de la primera pàgina de cada fitxer, tal com es pot observar a la part inferior dreta de la transparència). La fila 3 de nivell lògic representa una venda (identificada amb el valor 30 en la columna *SALE_ID*) d'1 unitat (columna *QUANTITY*) del producte 2 (columna *PRODUCT_ID*) efectuada el 29 de Febrer de 2016 (columna *SALE_DATE*) per un import total de 30 (columna *TOTAL_PRICE*) al client 10 (columna *CUSTOMER_ID*).

A diferència dels magatzems de files, els magatzems de columnes, en principi, guarden les dades de cada columna en un fitxer separat i aquest fitxer només guarda les dades d'una columna. Com en el cas dels magatzems de files, la realitat pot ser més complexa. D'una banda, s'ha explorat la possibilitat d'agrupar columnes diferents (sense intercalar o compondre els valors d'aquestes columnes en un format de fila) en un mateix fitxer. Aquest tipus de magatzems de columnes es denominen magatzems multicolumna. Es tracta de projectes de recerca i, per tant, no és una opció disponible en productes comercials (almenys al moment d'escriure aquest material didàctic). D'altra banda, en general, hi ha la possibilitat d'aplicar tècniques de fragmentació horitzontal en un magatzem de columnes (amb independència que aquest constitueixi una BD centralitzada o distribuïda). La fragmentació horitzontal, en el context dels magatzems de columnes, acostuma a rebre el nom de segmentació. Addicionalment, es pot combinar amb l'ús de la replicació (ho veurem més endavant en aquesta presentació).

La fragmentació horitzontal, com en el cas d'un magatzem de files, parteix les files d'una taula en grups disjunts de files que s'obtenen com a resultat d'aplicar algun tipus de condició (per exemple, en funció del valor que pren una columna o grup de columnes de la taula, o de forma aleatòria i uniforme a través de l'aplicació d'una funció de dispersió –o *hash*, en anglès– sobre una columna de la taula). De nou, i com en el cas dels magatzems de files, es tracta d'una estratègia que és especialment útil en

el cas de taules amb un volum important de files. Així mateix, és necessari que l'esquema de fragmentació horitzontal compleixi les propietats de completesa, disjunció i reconstrucció. Cada grup (o fragment) de files resultant d'aplicar la condició, al seu torn, s'emmagatzema per columnes. Això significa que els diferents grups de valors de cada columna corresponents a cada fragment estaran emmagatzemats en fitxers diferents i en funció de la clau d'ordenació triada. Quan la fragmentació es realitza a través d'aplicar una condició sobre el valor (o valors) d'una columna (o columnes), és habitual que es tracti de la columna (o columnes) que constitueix la clau d'ordenació (és a dir, de la columna o columnes que dicta com les dades seran emmagatzemades en el dispositiu d'emmagatzematge extern).

La figura 6 mostra un exemple del que acabem d'explicar. En primera instància, la taula *SALE* es fragmenta horitzontalment en dues, en funció del valor de la columna *SALE_ID* (fixeu-vos que aquesta columna també és la clau d'ordenació triada per a emmagatzemar les dades). El primer fragment (*SALE_HF1*) conté les files amb *SALE_ID* menor o igual a 30, mentre que el segon (*SALE_HF2*) conté les files amb *SALE_ID* major a 30. Finalment, cada fragment de la taula s'emmagatzema per columnes, en fitxers diferents que es troben en diferents discos (assumim una BD distribuïda per a emfatitzar les explicacions).



Figura 6. Exemple de fragmentació horitzontal en un magatzem de columnes

Una altra diferència entre els magatzems de files i els de columnes té a veure amb el fet que els magatzems de columnes permeten minimitzar la informació de control que és necessari guardar a les pàgines de dades, és a dir, a les pàgines que guarden els valors de les diferents columnes, i en conseqüència, dels fitxers que allotgen aquestes pàgines.

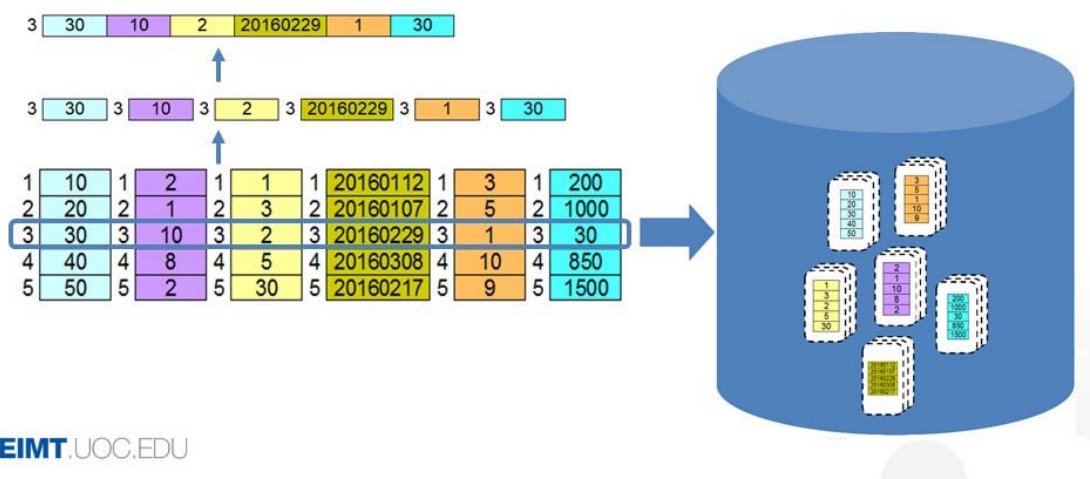
Això és a causa que l'estructura d'emmagatzematge que segueixen els magatzems de columnes fomenta la localitat de les dades, dades que a més tenen un nivell d'entropia baix. Entrarem en aquestes qüestions de manera més formal una mica més endavant dins d'aquesta mateixa presentació però, en essència, el que signifiquen és que les dades de cada columna no es barregen amb dades d'altres columnes que tenen una naturalesa diferent (representen un concepte del món real diferent i potser tindran associat un tipus de dades diferent). En altres paraules, les dades emmagatzemades en un mateix fitxer, en el cas d'un magatzem de columnes, són més homogènies i comparteixen més característiques entre si, que en el cas dels fitxers que guarden les dades d'un magatzem de files.

Si ens fixem en la nostra taula d'exemple *SALE* i, assumint que tots els tipus de dades associades a les columnes són de longitud fixa (hipòtesi raonable veient els valors que s'emmagatzemen), ens podem imaginar cada pàgina de dades com un vector (o *array*) de valors que ocupen el mateix nombre de *bytes* (aquest nombre estarà prefixat per l'SGBD). Aquest fet elimina, per exemple, la necessitat de guardar les dades de control sobre la longitud de cada valor guardat. En un magatzem de files, malgrat que tampoc no sempre sigui necessari guardar aquesta informació per al valor de cada columna, el fet de tenir simplement una columna de longitud variable implica que la fila serà de longitud variable. Al seu torn, això causa la necessitat de guardar la longitud total de la fila en *bytes* (aquest és un exemple d'informació de control que es guarda en un magatzem de files, com ja hem explicat prèviament), i el vector d'adreces de files (VDF) que ens diu a quina posició, dins de la pàgina, comença cada fila.

De nou, la realitat és més complexa. No estem dient que no sigui necessari emmagatzemar informació de control, sinó que és necessari guardar-ne menys pel que fa a un magatzem de files. És més, aquesta informació de control freqüentment es pot guardar en altres estructures de dades auxiliars gestionades per l'SGBD i, fins i tot, en el catàleg de la BD. Un fet que pot causar la necessitat de guardar la informació de control a les pàgines de dades d'un magatzem de columnes és l'aplicació de tècniques de compressió de dades (introduïrem aquestes tècniques una mica més endavant en aquesta mateixa presentació).

Identificadores de fila implícits

- No es necesario guardar de forma explícita en las páginas información que permita reconstruir las filas.
- Para reconstruir la fila *i*-ésima es suficiente acceder a la posición *i*-ésima de cada columna y concatenar los valores contenidos.



EIMT.UOC.EDU

En algunes de les estratègies que permeten aproximar-se (amb menor o major fortuna) a l'esquema d'emmagatzematge d'un magatzem de columnes, com seria el cas de la fragmentació vertical o el model d'emmagatzematge DSM discutits amb anterioritat, hem vist que sempre era necessari emmagatzemar la clau primària i que aquesta era utilitzada per a executar les operacions de combinació (en anglès, *join*) que permeten reconstruir les files que conformen l'extensió de la taula. En el cas d'un magatzem de columnes, podem apreciar que no es guarda, de forma explícita, cap tipus d'informació destinada a aquest efecte –ni la clau primària ni un identificador de la fila (o *rowid*, en anglès)– conjuntament amb les dades de cada columna.

Això és així perquè els magatzems de columnes treballen amb el que es denomina identificadors de files implícits (en anglès, *implicit rowid*). En altres paraules, l'SGBD és capaç de calcular quin és l'identificador de cada fila de la taula. Aquest procediment és més senzill quan els valors associats a les columnes s'emmagatzemen de forma ordenada en el disc, en funció del valor o valors que prenen una columna o grup de columnes (la denominada clau d'ordenació) i serà l'únic cas que cobrirem en aquest material didàctic.

Quan els valors associats a les columnes s'emmagatzemen de forma ordenada en el disc, en funció d'una clau d'ordenació, es garanteix que el valor trobat per a cada columna en la posició *i*-èsima de cada estructura d'emmagatzematge (és a dir, de cada fitxer) pertany a la fila *i*-èsima de la taula. Per tant, la posició que ocupa cada valor a cadascuna de les estructures d'emmagatzematge és l'identificador implícit de

la fila. En conseqüència, per a reconstruir la fila i -èsima, simplement és necessari accedir a la posició i -èsima de cadascuna de les estructures d'emmagatzematge creades i concatenar els valors recuperats.

En realitat, el procés que acabem de descriure és més complex, però sempre hi ha un procediment que l'SGBD és capaç de derivar. La complexitat pot variar depenent de tot un seguit de factors com serien, entre d'altres, si la columna és de longitud fixa o variable, o de si s'han usat o no tècniques de compressió de dades.

Per a possibilitar el procés de reconstrucció de les files, l'SGBD també disposa d'estructures de dades auxiliars que permeten saber, per a cada columna d'una taula, el total de pàgines que hi ha en el disc que emmagatzemen les dades d'aquesta columna. A més, per a cada pàgina, l'SGBD coneix quin és el número de la pàgina (és a dir, el seu identificador), la primera fila emmagatzemada a la pàgina (en altres paraules, la primera posició o identificador implícit de la fila que hi ha a la pàgina d'acord amb el criteri d'ordenació aplicat), així com el valor mínim i màxim que hi ha emmagatzemats en aquesta pàgina per a aquesta columna.

Aquestes estructures auxiliars, a més de permetre la reconstrucció de les files de la taula, també poden ajudar a resoldre els accessos per valor sobre la BD. Cadascuna d'aquestes estructures auxiliars construïda sobre cada columna constitueix un índex no dens (en anglès, *sparse index*). Un índex no dens no emmagatzema de forma exhaustiva informació relativa a tots els valors que existeixen per a la columna indexada, sinó que emmagatzema un resum d'aquesta informació, la que es considera suficient per al propòsit (o propòsits) de l'índex. Cal destacar que, depenent del producte en consideració, el contingut i estructuració de l'índex pot variar lleugerament. Nosaltres hem descrit el que implementa Vertica.

La figura 7 mostra un exemple d'un índex no dens com el descrit per al cas de la columna `SALE_ID` de la nostra taula d'exemple `SALE`. L'exemple assumeix l'existència de 3 pàgines que emmagatzemen dades per a la columna `SALE_ID` (a totes les transparències i figures d'aquest document només hem mostrat la primera d'aquestes pàgines). A la primera pàgina (identificada amb el número 15), el primer valor que s'emmagatzema és el de la fila 1 (d'acord amb l'ordenació realitzada) i els valors mínim i màxim emmagatzemats a la pàgina per a la columna són, respectivament, el 10 i el 50. Per a la segona pàgina (identificada amb el número 16), el primer valor que s'emmagatzema es correspon amb la fila 6 i els valors mínim i màxim són el 60 i el 100. Finalment, la tercera pàgina és la identificada amb el número 17, la primera fila que s'emmagatzema es correspon amb la fila 12 i els valors mínim i màxim són, respectivament, el 110 i el 150.

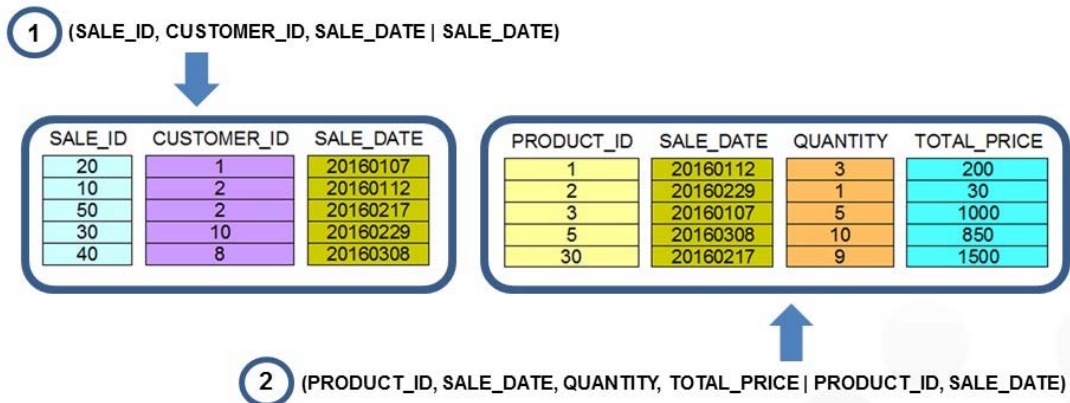
| | SALE.SALE_ID | | | |
|---|------------------|-------------------------------|---------------------------|---------------------------|
| | Número de página | Primera posición en la página | Valor mínimo en la página | Valor máximo en la página |
| 1 | 15 | 1 | 10 | 50 |
| 2 | 16 | 6 | 60 | 100 |
| 3 | 17 | 12 | 110 | 150 |
| 4 | ... | ... | ... | ... |

Figura 7. Exemple d'índex no dens sobre *SALE_ID*

Fixeu-vos que aquestes estructures d'índex són completament diferents als índexs disponibles per als magatzems de files, com seria el cas dels arbres B+ o dels índexs de dispersió que hem estudiat en l'assignatura. Aquests tipus d'índexs constitueixen exemples d'índexs densos (*dense indexes*, en anglès). En un índex dens, per a cada valor v_i present en la columna sobre la qual es construeix l'índex, es guarda una entrada. Aquestes entrades inclouen el valor v_i i l'identificador o identificadors (*rowids*) de les files que contenen un valor igual a v_i per a la columna indexada. La construcció i el manteniment d'un índex no dens són més senzills que en el cas d'un índex dens. De forma similar, si ho comparem, tenen una grandària (en *bytes*) molt inferior a la d'un índex dens.

Proyecciones

- Las columnas pueden ser almacenadas diversas veces por grupos (redundancia de los datos), usando criterios de ordenación diferentes.



EIMT.UOC.EDU

Els magatzems de columnes que mantenen les dades ordenades físicament en el disc en funció del valor que pren una columna o grup de columnes (la clau d'ordenació), en general, permeten emmagatzemar les dades d'una mateixa taula (en concret, grups de columnes de la taula) diverses vegades, d'acord amb diferents criteris d'ordenació (o sigui, en funció de diferents claus d'ordenació). En definitiva, aquesta estratègia introdueix una certa redundància (o repetició) en l'emmagatzematge físic de les dades, que s'orienta a millorar el rendiment de les operacions de consulta (`SELECT`) que es formulen sobre la BD. En concret, Vertica/C-Store i Amazon Redshift incorporen aquesta característica. Per a les explicacions que venen a continuació ens centrarem en les possibilitats que ofereix Vertica (i el prototip en què es basa, C-Store).

Cada grup de columnes que s'ordena segons una clau d'ordenació (és a dir, en funció del valor que pren una columna o columnes presents en el grup), rep el nom de projecció. En aquesta transparència es mostren dues projeccions sobre la nostra taula d'exemple `SALE`. La primera de les projeccions (a l'esquerra de la transparència) inclou les columnes `SALE_ID`, `CUSTOMER_ID` i `SALE_DATE` de la taula `SALE` i la clau d'ordenació que s'ha triat és la columna `SALE_DATE`. Per la seva banda, la segona projecció (mostrada a la part dreta de la transparència) inclou les columnes `PRODUCT_ID`, `SALE_DATE`, `QUANTITY` i `TOTAL_PRICE`, i la clau d'ordenació triada està formada per dues columnes, en concret, les columnes `PRODUCT_ID` i `SALE_DATE` (les dades s'ordenen primer atenent l'identificador del producte i, per a cada identificador de cada producte diferent, per la data en què s'efectua la venda).

La notació que utilitza Vertica/C-Store per a representar cada projecció és la que teniu posada al costat de cada exemple. Entre parèntesi, en primer lloc, s'inclouen separades per comes les columnes que intervenen en la projecció. En segon lloc, i després del caràcter «|» s'inclouen la columna o columnes que constitueixen la clau d'ordenació per a la projecció (si hi ha més d'una columna, estaran separades per comes). Les columnes presents a cadascuna d'aquestes projeccions, al seu torn, s'emmagatzemen en fitxers separats en el disc. A la figura 8, s'il·lustra aquesta situació per al cas de la primera projecció d'exemple.

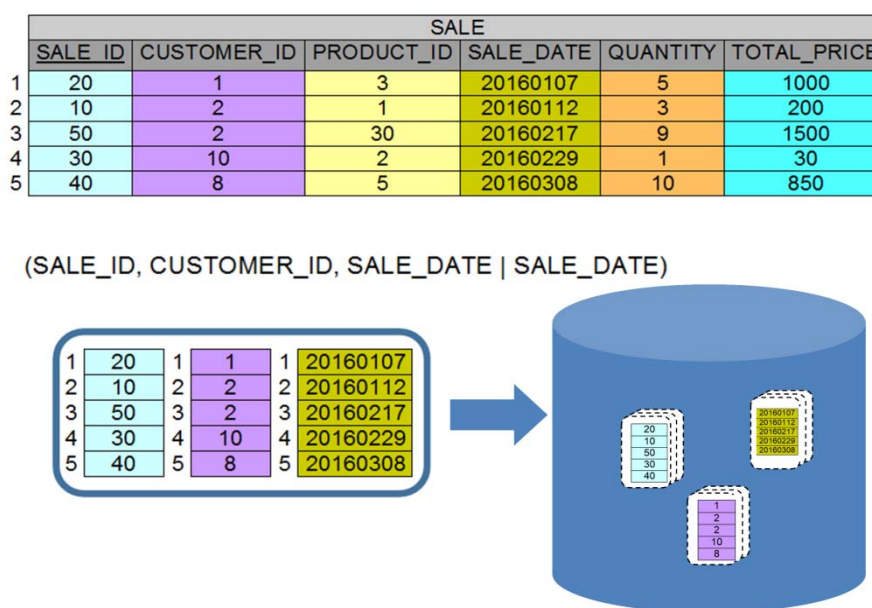


Figura 8. Exemple de projecció sobre SALE

Concretament, a la figura es mostra la nostra taula d'exemple SALE a nivell lògic ordenada en funció de la columna SALE_DATE (la clau d'ordenació de la projecció). A continuació podeu observar les 3 columnes que conformen aquesta projecció (SALE_ID, CUSTOMER_ID i SALE_DATE). Finalment, podeu veure com aquestes columnes s'emmagatzemen en estructures d'emmagatzematge separades en el dispositiu d'emmagatzematge extern.

Amb Vertica, per a cada taula de nivell lògic, sempre hi ha una projecció (amb la seva corresponent clau d'ordenació) que inclou totes les columnes de la taula. Aquesta projecció es defineix quan es crea la taula (és a dir, quan s'executa la sentència CREATE TABLE) i rep, en terminologia de Vertica, la denominació de superprojecció. L'existència d'una superprojecció per taula garanteix que es preservarà la semàntica de la BD, és a dir, es garanteix que no es perdran columnes (ni les seves dades associades) en el procés de creació de les projeccions. Sobre la nostra taula d'exemple SALE, la superprojecció (a més d'incloure totes les columnes) té com a clau d'ordenació la columna SALE_ID, és a dir, és l'exemple que hem usat en aquesta presentació fins a aquest punt, i que es mostra a la figura 9. A partir d'aquí, Vertica permet definir tantes projeccions (a través de l'execució de la sentència CREATE

PROJECTION) sobre grups de columnes com es desitgi (nosaltres n'hem definit dues a manera d'exemple), amb les seves corresponents claus d'ordenació per a cada projecció. Res no impedeix que una mateixa columna figuri en més d'una projecció, amb els seus valors reordenats en funció de la clau d'ordenació triada per a la projecció. Sobre els exemples de projeccions que mostrem en aquesta transparència, aquest és el cas de la columna *SALE_DATE* que apareix en ambdues projeccions.

A la figures 8 i 9 podem veure que els valors de cada fila de la taula de nivell lògic, una vegada segregats en columnes i dins de cada projecció, ocupen exactament la mateixa posició a cadascuna de les estructures d'emmagatzematge creades. Cadascuna d'aquestes posicions, dins de cada projecció, constitueix l'identificador implícit de la porció de la fila emmagatzemada en la projecció (cas de la figura 8) o de la fila completa si es tracta de la superprojecció (figura 9). Com ja hem explicat amb anterioritat, l'SGBD és capaç de derivar de forma automàtica aquests identificadors implícits.

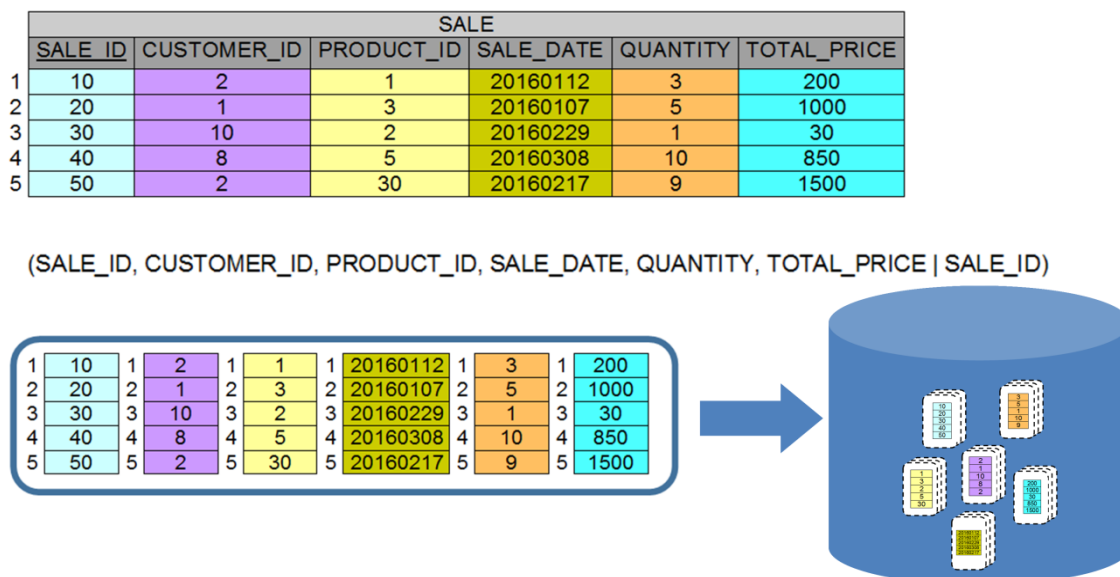


Figura 9. Exemple de superprojecció sobre *SALE*

Cal observar que l'identificador implícit de cada fila dins de cada projecció és diferent, atès que aquest s'assigna en funció de l'ordenació realitzada. En altres paraules, cada projecció constitueix una permutació de les files (o de trossos de files) de la taula de nivell lògic.

| SALE | |
|--|---|
| (SALE_ID, CUSTOMER_ID, PRODUCT_ID, SALE_DATE, QUANTITY, TOTAL_PRICE SALE_ID) | (SALE_ID, CUSTOMER_ID, SALE_DATE SALE_DATE) |
| 1 | 2 |
| 2 | 1 |
| 3 | 4 |
| 4 | 5 |
| 5 | 3 |

Figura 10. Taula de correspondències entre projeccions de la taula *SALE*

La taula de la figura 10 mostra la correspondència que hi ha entre els identificadors de files assignats a cadascuna de les projeccions de les figures 8 i 9. És a dir, la taula ens diu, per a cada identificador de la fila de la superprojecció (que està ordenada en funció de la columna *SALE_ID*) i que es mostra en la columna de l'esquerra, quin identificador li correspon en la projecció que hem creat sobre el grup de columnes *SALE_ID*, *CUSTOMER_ID* i *SALE_DATE* de la taula *SALE* amb clau d'ordenació *SALE_DATE* (segona columna de la taula). Cal destacar que la taula mostrada a la figura 10 s'inclou a efectes docents, és a dir, per a aclarir les explicacions realitzades. Vertica, tal com veurem més endavant, únicament utilitza una projecció per cada taula implicada en les operacions de consulta que es formulin sobre la BD. Això vol dir que Vertica mai no reconstruirà una fila de nivell lògic a partir de les dades que existeixin d'aquesta fila en més d'una projecció.

Els exemples de projeccions que hem vist fins ara només inclouen columnes d'una taula, la taula sobre la qual es defineix la projecció. Aquesta taula, en terminologia de Vertica, rep el nom de taula àncora (*anchor table*, en anglès). Una projecció també pot incloure columnes d'altres taules, sempre que la taula àncora inclogui una clau forana cap a aquestes altres taules. Aquest tipus de projeccions reben el nom de projeccions precombinades (*prejoin projections*, en anglès) i tenen com a objectiu, com el seu propi nom indica, facilitar l'execució d'operacions de combinació (*join*) entre les taules. Les projeccions precombinades són d'especial interès en el cas de models multidimensionals.

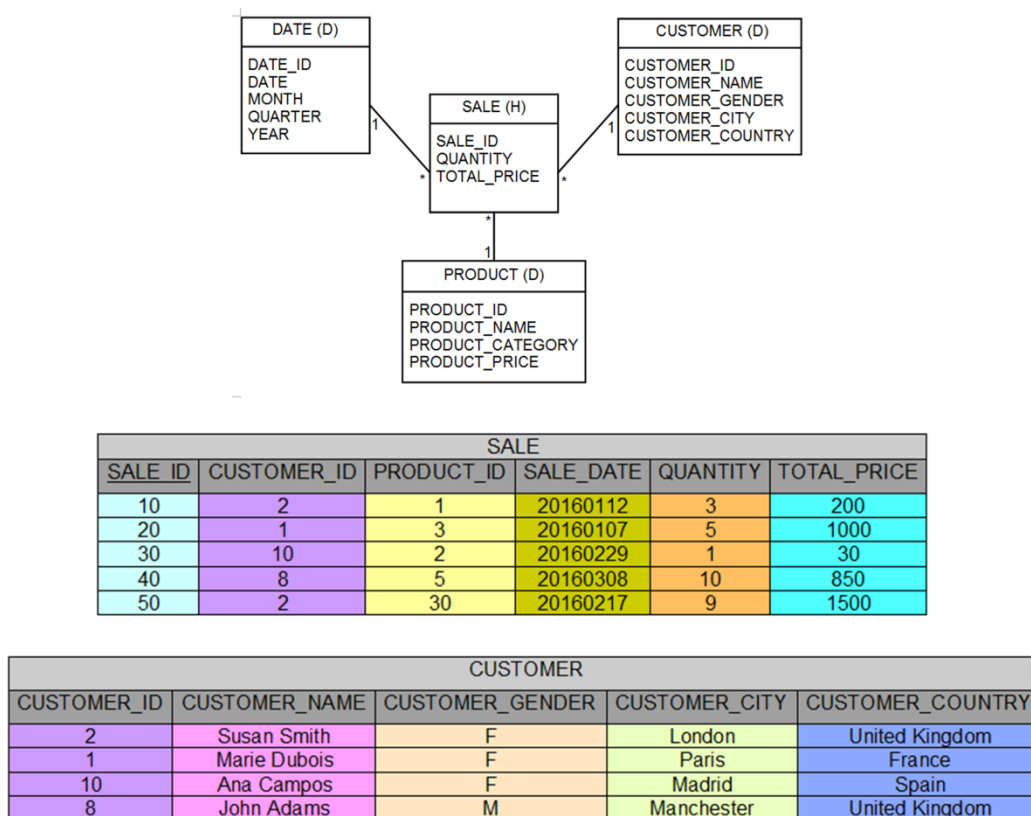


Figura 11. Model multidimensional i taules d'exemple

Si recordeu, la nostra taula d'exemple *SALE* representa una taula de fets d'un model multidimensional en estrella. El model conceptual (en forma de diagrama de classes UML) d'aquest model multidimensional es mostra a la part superior de la figura 11. En aquesta mateixa figura, s'inclou el model lògic de dues de les taules associades al model multidimensional, en concret, la nostra taula *SALE* (la taula de fets) i la taula *CUSTOMER* (una de les taules que representa una de les possibles dimensions d'anàlisi). En el model lògic, l'associació binària amb connectivitat un a molts que hi ha en el model conceptual entre les taules *CUSTOMER* i *SALE* queda representada per una clau forana que es defineix a la taula *SALE* (en concret, es tracta de la columna *CUSTOMER_ID*). Malgrat que no es mostrin a la figura 11, per a la resta de taules que representarien les altres dimensions d'anàlisi (les taules *PRODUCT* i *DATE*), se segueix exactament la mateixa estratègia (en altres paraules, les columnes *PRODUCT_ID* i *SALE_DATE* de la taula *SALE* són claus foranes cap a les taules *PRODUCT* i *DATE*, respectivament).

Doncs bé, podem crear projeccions sobre la taula *SALE* que incloguin columnes de les taules que representen les diferents dimensions d'anàlisi. Específicament, a la figura 12, podem veure una projecció que incorpora la columna *CUSTOMER_COUNTRY* de la taula *CUSTOMER*. Aquesta mateixa columna constitueix la clau d'ordenació de la projecció. Pel que fa a la notació, podeu observar que les columnes de la segona taula que s'han afegit a la projecció porten anteposat el nom de la taula (*CUSTOMER.CUSTOMER_COUNTRY*). En l'extensió associada a la projecció, podeu veure que s'ha realitzat la combinació entre ambdues taules, s'han seleccionat les dades de les columnes indicades en la projecció i les dades estan ordenades en funció de la columna *CUSTOMER.CUSTOMER_COUNTRY* que és la clau d'ordenació. Òbviament, cadascuna de les columnes presents en la projecció seran emmagatzemades en un fitxer separat. A la figura també es mostra una consulta d'exemple que es resoldrà eficientment usant aquesta projecció.

(SALE_ID, CUSTOMER_ID, PRODUCT_ID, CUSTOMER.CUSTOMER_COUNTRY | CUSTOMER.CUSTOMER_COUNTRY)

| SALE_ID | CUSTOMER_ID | PRODUCT_ID | CUSTOMER.CUSTOMER_COUNTRY |
|---------|-------------|------------|---------------------------|
| 20 | 1 | 3 | France |
| 30 | 10 | 2 | Spain |
| 10 | 2 | 1 | United Kingdom |
| 40 | 8 | 5 | United Kingdom |
| 50 | 2 | 30 | United Kingdom |

```

SELECT
  CUSTOMER.CUSTOMER_COUNTRY
  COUNT(*) AS TOTAL_SALE
FROM SALE INNER JOIN CUSTOMER ON
  SALE.CUSTOMER_ID = CUSTOMER.CUSTOMER_ID
GROUP BY
  CUSTOMER.CUSTOMER_COUNTRY

```

Figura 12. Exemple de projecció precombinada

Com ja hem comentat, el benefici principal que s'espera obtenir de la creació de projeccions és millorar el rendiment dels accessos pel valor de les dades en el cas d'operacions de consulta (execució de sentències *SELECT*). En conseqüència, la definició de projeccions constitueix una decisió de disseny de la BD. Per a crear una projecció, és necessari decidir quin grup de columnes s'inclourà en la projecció (i de quina taula o taules procedeixen) i quina serà la clau d'ordenació. I aquestes decisions es prenen tenint en compte les consultes que es realitzaran. Addicionalment, Vertica

proporciona eines que guien l'administrador de la BD en la creació de projeccions mitjançant l'examen de mostres de les dades i sobre la base de l'històric de les consultes que s'han formulat, facilitant així l'afinament (en anglès, *tuning*) de la BD.

Durant el procés d'optimització de cada consulta, Vertica triarà les projeccions més adequades per a resoldre la consulta. Com a màxim, per a cada taula (o grup de taules, si s'ha definit una projecció precombinada) implicada en la consulta, triarà una projecció. En el cas que no existeixi, utilitzarà la superprojecció. De forma similar al que hem fet a la figura 12, a la figura 13 es presenten dues consultes que es resoldran de forma eficient per a cadascuna de les dues projeccions que es mostren en aquesta transparència.

En resum, la definició de projeccions persegueix un objectiu similar al de la definició d'índexs i vistes materialitzades en els magatzems de files. Pel que fa als índexs, l'ordenació física de les dades és, en si mateixa, un mecanisme d'indexació i és similar, conceptualment, a l'ús d'índexs en forma d'arbre B+ agrupats que coneixem dels magatzems de files. Cal destacar que Vertica no permet als usuaris crear, de forma explícita, índexs de tipus B+ (agrupats o no agrupats), de dispersió o de mapes de bits (en anglès, *bitmaps*) sobre les taules. El fet que els usuaris no puguin (en general) definir índexs com els descrits, no significa que els magatzems de columnes no usin índexs, el que significa és que es creen principalment de forma implícita i sota el control de l'SGBD, sense la intervenció humana. De fet i tal com hem explicat anteriorment, Vertica utilitza (si més no) índexs no densos. En relació amb les vistes materialitzades, cal destacar que tampoc no són suportades amb Vertica (ni en altres magatzems de columnes), per tant, aquestes queden subsumides per la definició de projeccions precombinades.

Si bé l'ús de projeccions ajuda a resoldre de forma eficient les operacions de consulta, també tenen associats dos desavantatges principals, que es relacionen amb el fet de la redundància que introdueixen. En primer lloc, es requereix un espai extra per a l'emmagatzematge de les dades. L'ús de tècniques de compressió de dades (que seran tractades amb posterioritat) permet alleujar aquest desavantatge. En segon lloc, es complica l'execució d'operacions de canvi sobre la BD (operacions d'INSERT, DELETE i UPDATE), no només a causa de la redundància, sinó també de l'ordenació física de les dades que imposen les projeccions. Més endavant, en aquesta mateixa presentació, veurem com els magatzems de columnes gestionen l'execució de les operacions de canvi. Per tot això i malgrat que no existeixen limitacions en el nombre de projeccions a definir, la documentació de Vertica comenta que, sobre la base de la seva experiència en projectes reals, l'habitual és tenir entre zero i tres projeccions per taula que incloguin un nombre reduït de columnes, a més de la superprojecció. En resum, el que ens està advertint la documentació de Vertica, és que s'ha de fer un ús mesurat de les projeccions si no volem que els seus beneficis quedin anul·lats pels costos de manteniment d'aquestes estructures. En essència, és exactament la mateixa

recomanació que realitzen els venedors de magatzems de files en relació amb la definició d'índexs i vistes materialitzades.

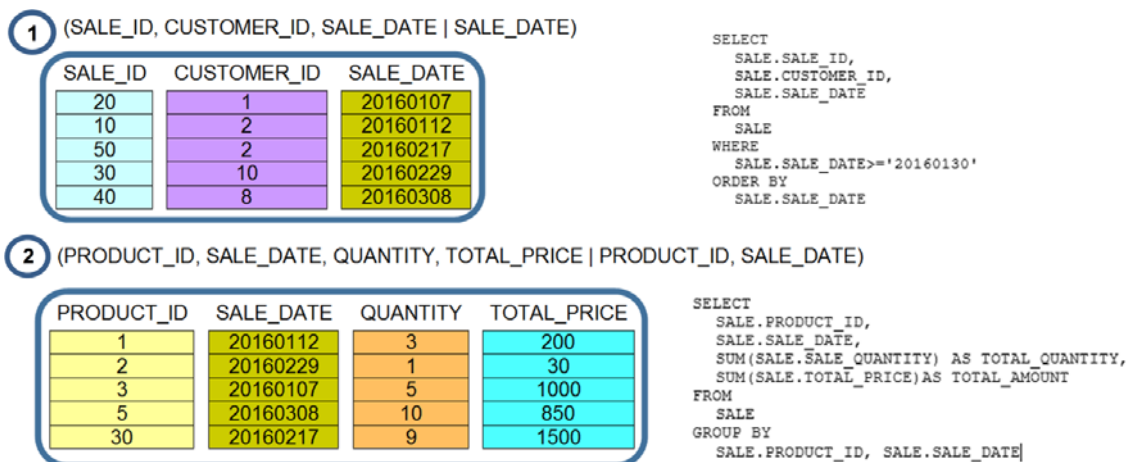


Figura 13. Exemples de consulta per projecció

Per finalitzar, i tal com ja hem explicat, les projeccions introdueixen redundàncies (o repeticions) en l'emmagatzematge físic de les dades. Per tant, guarden una certa relació amb el mecanisme de replicació de les dades, encara que es tracta de conceptes diferents.

Com ja sabem, la replicació de les dades consisteix en mantenir diferents còpies (o rèpliques) idèntiques d'unes mateixes dades i el seu ús es concentra en les BD distribuïdes, en combinació amb la utilització de tècniques de fragmentació. Mentre que en un magatzem de files podem aplicar tant tècniques de fragmentació horitzontal com vertical, en el cas dels magatzems de columnes únicament s'usa (tal com ja hem explicat amb anterioritat) la fragmentació horitzontal (que es coneix, en general, amb la denominació de segmentació).

La figura 14 conté un exemple d'ús combinat de replicació i fragmentació horitzontal de dades, adaptat al cas particular de Vertica. La fragmentació horitzontal s'aplica a nivell de projecció. Per la seva banda, la replicació s'aplica sia a nivell de projecció (en el cas de no haver-se aplicat la fragmentació horitzontal a la projecció) o a nivell de cadascun dels fragments resultants d'aplicar la fragmentació horitzontal (a aquests fragments els denominarem subprojeccions). En concret, la figura 14 conté un exemple on es pot veure, en primera instància, com la nostra taula SALE (o sigui, la superprojecció) ha estat fragmentada en dues subprojeccions. En segona instància, la figura també mostra com una de les subprojeccions resultants (la de l'esquerra), al seu torn, ha estat replicada. Com en el cas dels magatzems de files, es tracta de dues còpies (o rèpliques) idèntiques del fragment (o sigui de la subprojecció): contenen les mateixes dades i aquestes estan guardades amb el mateix ordre a cadascun dels dispositius d'emmagatzematge extern (disc).

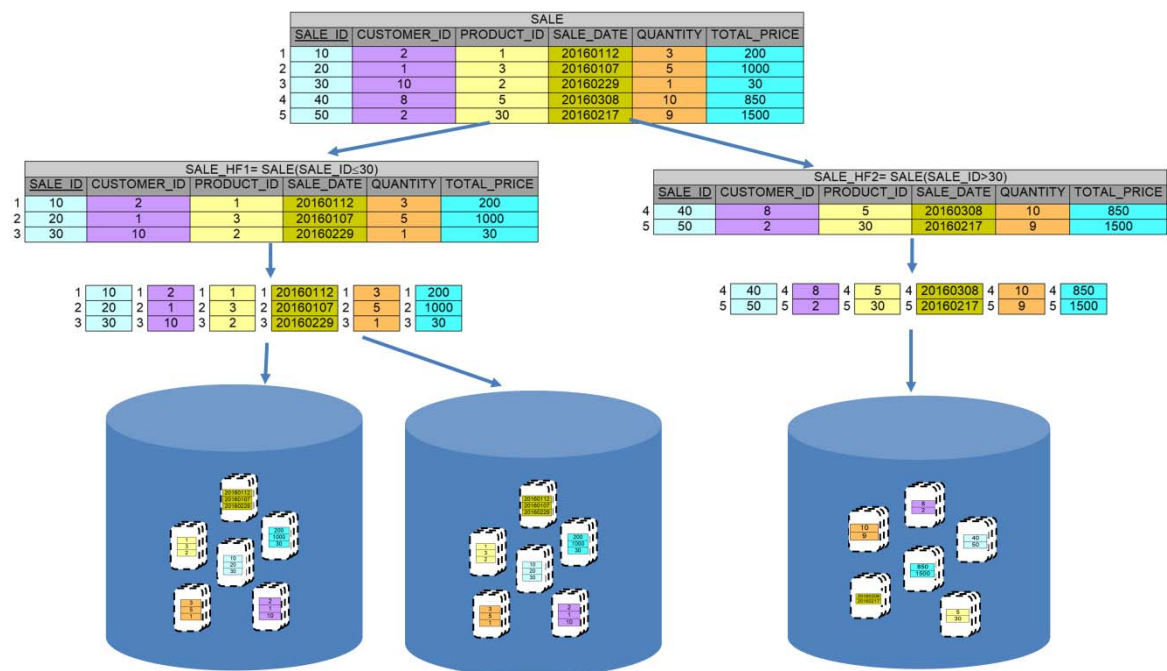
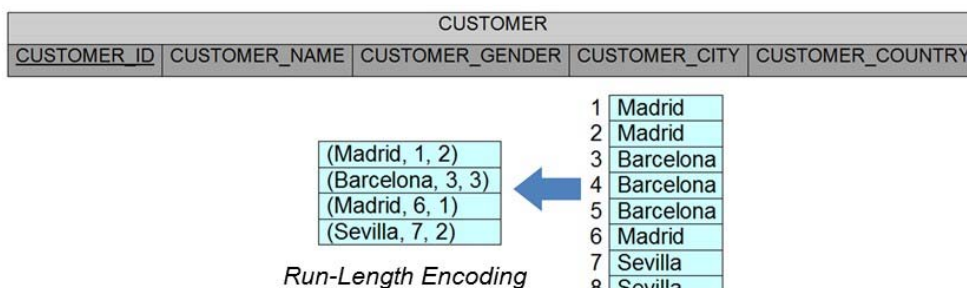


Figura 14. Exemple d'ús combinat de la fragmentació horitzontal i la replicació en un magatzem de columnes

Compresión de datos

- El almacenamiento por columnas promueve la localidad de datos con un nivel de entropía bajo \Rightarrow facilita la compresión de los datos.
- La compresión de datos se basa en buscar repeticiones en una serie de datos que se representan de forma resumida.



EIMT.UOC.EDU

L'estructura d'emmagatzematge que segueixen els magatzems de columnes té dues característiques principals.

En primer lloc, un augment de la localitat de les dades. Això és així perquè els valors d'una mateixa columna estan propers en el dispositiu d'emmagatzematge extern, atès que es guarden en un mateix fitxer, sense barrejar-se amb dades de naturalesa diferent, com serien els valors d'altres columnes de la taula. Això implica que la lectura de les dades associades a una mateixa columna (per exemple, la columna *CUSTOMER_CITY* de la taula *CUSTOMER* mostrada a la transparència) es pugui resoldre de forma eficient, atès que les operacions d'E/S de lectura de pàgines que es desencadenen recuperen únicament les dades potencialment útils. En el cas d'un magatzem de files, les pàgines de dades d'una taula contenen files completes, per la qual cosa accedir als valors d'una columna concreta implica un procés d'extracció, descartant els valors que no són d'interès de cada fila (sobre la nostra taula d'exemple, els valors de les columnes *CUSTOMER_ID*, *CUSTOMER_NAME*, *CUSTOMER_GENDER* i *CUSTOMER_COUNTRY*). A més, i en relació amb un magatzem de columnes, s'incorre en un cost superior en termes d'operacions d'E/S per a recuperar les dades d'interès.

En segon lloc, s'incrementa la capacitat de compressió de les dades. Intuïtivament les dades guardades en un magatzem de columnes són més compressibles que les dades emmagatzemades en un magatzem de files, atès que tenen un nivell d'entropia baix. Això és així a causa que són dades similars (és a dir, dades que representen un mateix concepte del món real, com seria el cas de les ciutats dels clients, representat a través

de la columna *CUSTOMER_CITY* a la taula *CUSTOMER*) es guarden en un mateix fitxer, sense barrejar-se amb altres dades de naturalesa diferent (com seria el cas de la resta de columnes de la taula). En altres paraules, les ciutats dels clients s'assemblaran més entre si, que el que es podrien semblar a altres conceptes que poguessin envoltar-les —per exemple, el gènere o el país dels clients (columnes *CUSTOMER_GENDER* i *CUSTOMER_COUNTRY*, respectivament)—, que seria el que passaria si s'hagués utilitzat un magatzem de files.

La compressió de dades es basa fonamentalment a buscar repeticions en una sèrie de dades (el conjunt de valors de cadascuna de les columnes d'una taula pot ser vist com una sèrie), que es representen de forma resumida.

Hi ha diferents algorismes de compressió, que seran objecte d'una presentació separada dins d'aquest material didàctic dedicat als magatzems de columnes. L'elecció de quin algorisme utilitzar depèn de diversos factors com serien: el nombre de valors diferents que pren la columna, el nombre de valors repetits que existeixen per a cada valor, la freqüència d'aparició de la columna en predicats de selecció o, fins i tot, el tipus de dades associat. Quan el magatzem de columnes guarda les dades seguint algun criteri d'ordenació, l'ordenació utilitzada pot augmentar la ràtio de compressió. En altres paraules, productes com ara Vertica/C-Store, Actian Vector/VectorWise o Amazon Redshift fan un ús més extensiu de les tècniques de compressió de dades que aquells productes que realitzen l'ordenació de les dades de forma incremental i parcial, en funció de les consultes plantejades (cas de MonetDB). Addicionalment, productes com ara Vertica/C-Store, Actian Vector/VectorWise o Amazon Redshift acostumen a incorporar utilitats que ajuden a triar els algorismes de compressió, a través de l'anàlisi de mostres de dades de les diferents taules.

A la transparència, podem veure un petit exemple que mostra com seqüències de valors repetits de forma consecutiva de ciutats de clients (columna *CUSTOMER_CITY* de la taula *CUSTOMER*) queden representats mitjançant tripletes. Aquestes tripletes contenen el valor en consideració, la primera posició que presenta el valor (recordem que aquesta posició representa la primera fila de nivell lògic que conté el valor) i el nombre de vegades que es repeteix el valor en qüestió en files consecutives. Podem veure que el valor Madrid està a la fila 1 i conté dues repeticions seguides, que el valor Barcelona segueix a la fila 3 i conté tres repeticions consecutives, que el valor Madrid torna a aparèixer a la fila 6 en una ocasió i que finalment el valor Sevilla apareix a la fila 7 amb dues repeticions consecutives.

L'algorisme que hem usat en el nostre exemple rep el nom de *run-length encoding* (abreujat com RLE). Aquest algorisme resulta especialment útil per a columnes que contenen un nombre de valors diferents relativament baix que presenten un gran nombre d'ocurrències. Fixeu-vos que la ràtio de compressió hauria estat superior en el cas que les dades haguessin estat ordenades per la columna *CUSTOMER_CITY*, tal com es mostra a la figura 15.

La compressió millora el rendiment (a més de la reducció d'espai en el disc), ja que permet reduir el nombre d'operacions d'E/S necessàries per a transferir les dades del disc a la memòria. De forma similar, la transferència de dades de la memòria a la CPU (el lloc on finalment seran executades les operacions sobre les dades) a través de la jerarquia de les memòries *cache* també serà més eficient. Finalment, un altre element que millora el rendiment és la capacitat de treballar sobre dades comprimides. Com veurem més endavant, els magatzems de columnes poden implementar operacions capaces d'operar directament sobre dades comprimides durant el procés de resolució de consultes.

El desavantatge principal de la compressió de les dades és el temps necessari que s'ha d'invertir en el procés de compressió i descompressió de dades (especialment problemàtic en les operacions de canvi). Tot això reforça l'interès de poder operar directament sobre dades comprimides.

| CUSTOMER | | | | |
|-------------|---------------|-----------------|---------------|------------------|
| CUSTOMER_ID | CUSTOMER_NAME | CUSTOMER_GENDER | CUSTOMER_CITY | CUSTOMER_COUNTRY |
| 1 | Barcelona | | | |
| 2 | Barcelona | | | |
| 3 | Barcelona | | | |
| 4 | Madrid | | | |
| 5 | Madrid | | | |
| 6 | Madrid | | | |
| 7 | Sevilla | | | |
| 8 | Sevilla | | | |

| |
|-------------------|
| (Barcelona, 1, 3) |
| (Madrid, 4, 3) |
| (Sevilla, 7, 2) |




Figura 15. RLE sobre *CUSTOMER_CITY* (dades ordenades)

Otras características de los almacenes de columnas

- Las tablas contienen un número elevado de filas (por ejemplo, datos históricos) y columnas.
- Las operaciones de cambio (`INSERT`, `DELETE` y `UPDATE`) son problemáticas.
- Implementan de forma eficiente consultas (`SELECT`) que verifican alguna de las siguientes condiciones:
 - Las búsquedas se basan en el valor que toman algunas columnas.
 - Se realizan agrupaciones y agregaciones sobre un volumen importante de datos.
 - Realizan cálculos sobre un número reducido de columnas.
- Las operaciones de consulta (`SELECT`) se benefician de la compresión de datos y de las diferentes ordenaciones de datos que se hayan definido.

Los almacenes de columnas se orientan principalmente a entornos OLAP.

EIMT.UOC.EDU

L'estructura d'emmagatzematge que segueixen els magatzems de columnes els fa especialment adequats per a ser usats com a BD en entorns de naturalesa analítica, és a dir, en entorns que són intensius en lectures, com seria el cas d'entorns OLAP. Aquests entorns, com ja sabem, es caracteritzen per llegir i processar grans quantitats de dades, amb l'objectiu d'obtenir nou coneixement i utilitzar-lo per a impulsar la presa de decisions.

Aquest tipus de BD, en comparació amb les BD operacionals (que s'orienten a entorns OLTP), solen tenir una càrrega de treball més imprevisible i donen suport a un nombre més reduït d'usuaris (aquests usuaris són, principalment, analistes de negoci i directius d'empresa). Addicionalment, es tracta de BD que contenen taules que acostumen a emmagatzemar un nombre elevat de files, com seria el cas, per exemple, de dades històriques. Això significa que el nombre de files de les taules és molt superior al d'una BD operacional i, en conseqüència, el nombre de files processades en les operacions de consulta (sentències `SELECT`) també és superior, podent estar en ordres de magnitud que van de desenes de milers a centenars de milers de files i, fins i tot, milions.

De forma similar, les taules també poden contenir un nombre elevat de columnes, atès que es tracta de taules que representen models multidimensionals en què se solen aplicar estratègies de desnormalització. Els problemes que hem comentat anteriorment en relació amb el nombre de columnes en el cas dels magatzems de files (BD operacionals), deixen de ser un condicionant en el cas dels magatzems de

columnes a causa de l'estratègia d'emmagatzematge que segueixen aquests últims (cada columna s'emmagatzema per separat).

Les dades que es processen en els entorns OLAP s'afegeixen a la BD a través de processos de càrrega de dades. Inicialment, en el moment de la creació de la BD, es produeix un procés de càrrega massiva de dades. Amb posterioritat, i amb una certa periodicitat, es poden executar processos incrementals nous de càrrega que tenen com a objectiu afegir les dades més recents. En general, els processos de càrrega incrementals són treballs per lots que s'executen en segon pla, o quan no hi ha usuaris treballant amb la BD (a la nit o durant el cap de setmana, per exemple). Els magatzems de columnes donen suport a aquest tipus de processos de càrrega de dades. No obstant això, cada vegada és més important la necessitat d'informació en temps real en els processos de presa de decisions de les organitzacions. Això implica que les BD que donen suport a aquests processos de decisió, com seria el cas dels magatzems de columnes, han de ser capaces d'acceptar un degoteig constant de canvis en les dades (execució d'operacions individuals d'INSERT, DELETE i UPDATE).

Com ja hem esmentat en aquest material didàctic, les operacions de canvi en els magatzems de columnes són problemàtiques, a causa de l'estratègia d'emmagatzematge seguida. A més, els problemes es veuen agreujats per l'ús de tècniques de compressió de dades i la definició de múltiples projeccions (en el cas que el magatzem de columnes incorpori aquesta possibilitat).

Il·lustrem aquesta problemàtica de forma simplificada, suposant que no s'usen tècniques de compressió de dades i que només hi ha una projecció per taula (en concret, la superprojecció). Per a això, recuperem la sentència d'inserció sobre la nostra taula d'exemple *SALE* que ja hem utilitzat en el cas dels magatzems de files i que tornem a mostrar a la dreta de la figura 16. La figura també mostra (a l'esquerra) la superprojecció amb clau d'ordenació *SALE_ID* que hem definit sobre la taula *SALE*. Finalment, i encara que no sigui realista atenent les explicacions prèvies, assumirem que la taula *SALE* únicament conté les files que es mostren a la figura.

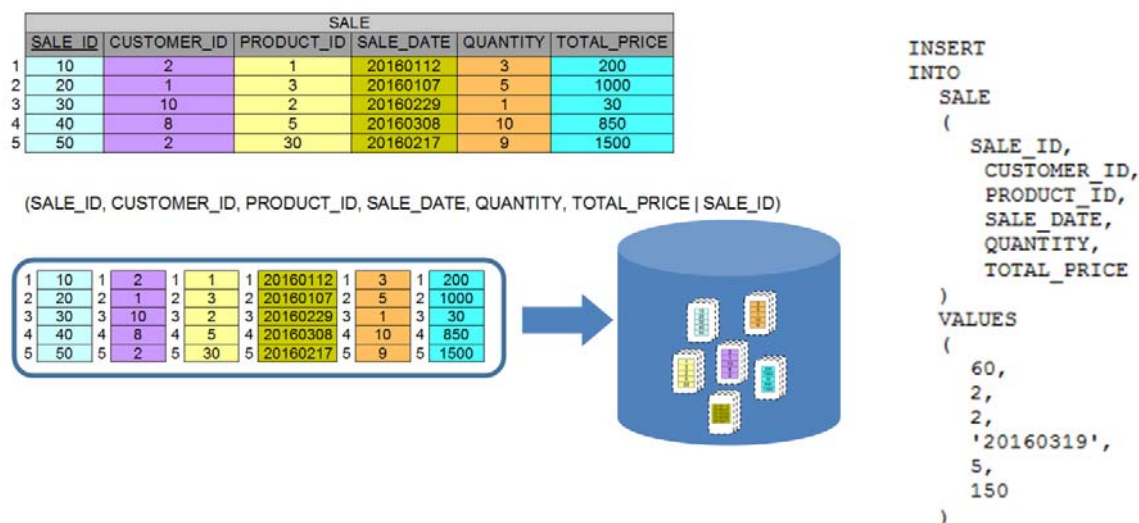


Figura 16. Inserció d'una fila en un magatzem de columnes

En aquestes circumstàncies, si es decideix inserir la nova fila a la taula *SALE* seguint una estratègia com la que empen els magatzems de files, en primer lloc, és necessari que l'SGBD determini la posició (és a dir, el *rowid* o identificador implícit) de la nova fila. La posició es determina sobre la base de la clau d'ordenació utilitzada en la superprojecció. Atès que es tracta de la venda *SALE_ID* amb valor 60, la posició de la nova fila serà la 6.

En segon lloc, l'SGBD haurà de segregar els valors associats a les diferents columnes de la fila. Atès que la nostra taula *SALE* té 6 columnes, la fila s'haurà descompost en 6 valors. El valor de cada columna s'emmagatzema en un fitxer separat. Per tant, en tercer lloc i per a cada fitxer, l'SGBD haurà de localitzar la pàgina que ha d'emmagatzemar el valor de cada columna. En altres paraules, per a cadascuna de les estructures d'emmagatzematge, és necessari que l'SGBD localitzi la pàgina que ha d'emmagatzemar el valor associat a la posició 6. Sobre el nostre exemple, es tractarà de l'última pàgina que conté valors (o una nova pàgina, si l'última està plena) a cadascun dels fitxers. Per a determinar la posició i localitzar les pàgines de cada estructura d'emmagatzematge, l'SGBD utilitza els índexs no densos que hem discutit anteriorment (podeu consultar la figura 7). Aquests índexs, tal com hem explicat amb anterioritat, es defineixen sobre cadascuna de les columnes de la taula (recordeu que si s'han definit diferents projeccions, hi haurà un índex no dens per a cada columna present en la projecció).

En resum, per a inserir la nova fila, l'SGBD haurà de llegir 6 pàgines de disc, o sigui, haurà d'efectuar 6 operacions d'E/S. Una vegada afegits els valors de cada columna a cadascuna de les pàgines llegides, serà necessari bolcar els canvis al disc. Això desencadena 6 noves operacions d'E/S en el disc (en aquest cas es tractaran d'operacions d'escriptura). Consegüentment, la inserció de la fila d'exemple a la taula *SALE*, en el cas d'un magatzem de columnes, requereix 12 operacions d'E/S. Estem

ometent les operacions d'E/S (de lectura i escriptura) addicionals que es puguin desencadenar en les estructures auxiliars (els índexs no densos) com a conseqüència de l'execució de l'operació d'inserció.

Quan vam examinar aquest mateix exemple per al cas d'un magatzem de files, vam veure que l'operació d'inserció, una vegada localitzada la pàgina que ha de contenir la fila i deixant al marge les operacions d'E/S necessàries per a l'actualització de les estructures auxiliars (índexs), es podia resoldre amb dues operacions d'E/S.

Per tant, podem veure que el cost de la resolució de les operacions d'inserció a un magatzem de columnes és molt superior al d'un magatzem de files. Les explicacions que acabem de realitzar són extrapolables a la resta d'operacions de canvi (DELETE i UPDATE) i també a consultes que recuperen files individuals completes o conjunts petits de files completes. Aquest seria el cas, per exemple, d'una sentència SELECT que recuperi la fila corresponent a una venda concreta (per exemple, la fila amb SALE_ID igual a 30). Si recordeu, vam discutir aquest exemple (mostrat a la part dreta de la figura 4) quan vam analitzar les característiques dels magatzems de files.

Les explicacions que acabem de donar i que comparen el procés d'inserció a un magatzem de files i a un de columnes, es mostren de forma gràfica (i en dos passos) a les figures 17 i 18. A cadascuna de les figures, es mostra el magatzem de files a l'esquerra i a la dreta hi ha el magatzem de columnes.

A la figura 17 (primer pas) es mostren les operacions d'E/S de lectura en cada cas: una operació d'E/S en el cas d'un magatzem de files i sis en el cas d'un magatzem de columnes.

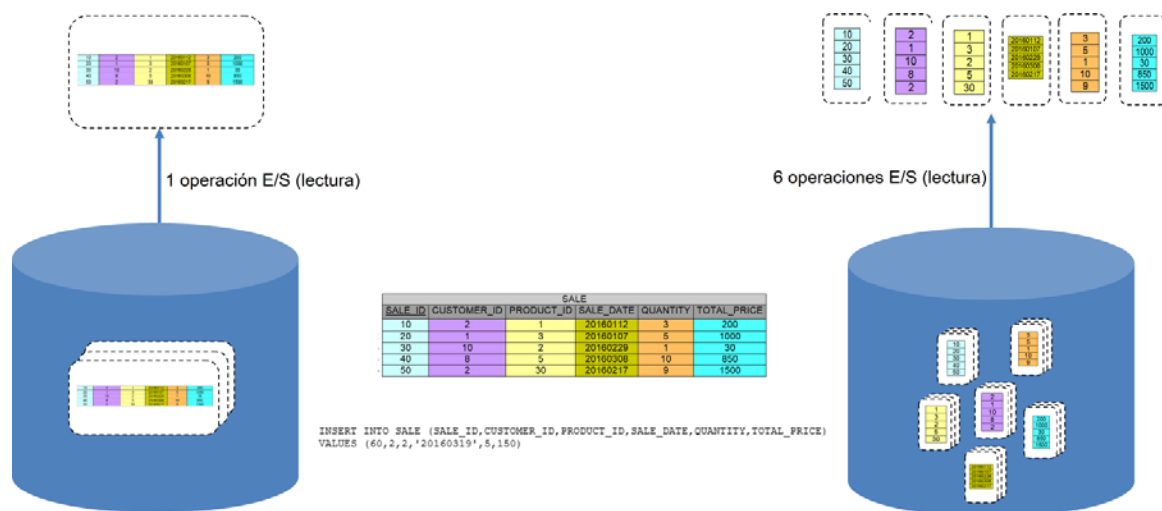


Figura 17. Comparació inserció magatzems de files contra magatzems de columnes (pas I)

La figura 18 mostra, una vegada resolta l'operació d'inserció en la memòria principal i com a segon pas, el procés de bolcat (o d'escriptura) de les pàgines modificades al dispositiu d'emmagatzematge extern: una operació d'E/S en el cas del magatzem de

files i sis operacions d'E/S en el cas del magatzem de columnes. Observeu també que hem afegit la nova fila a la taula *SALE* (la nova extensió de la taula tant es pot veure en el model lògic com en el físic, és a dir, en les estructures d'emmagatzematge de cada tipus de BD). Per a no complicar innecessàriament les explicacions, podeu veure que la fila a inserir cap a les pàgines inicialment contemplades en cada cas (en altres paraules, aquestes pàgines no estan plenes i poden absorbir la inserció de valors nous).

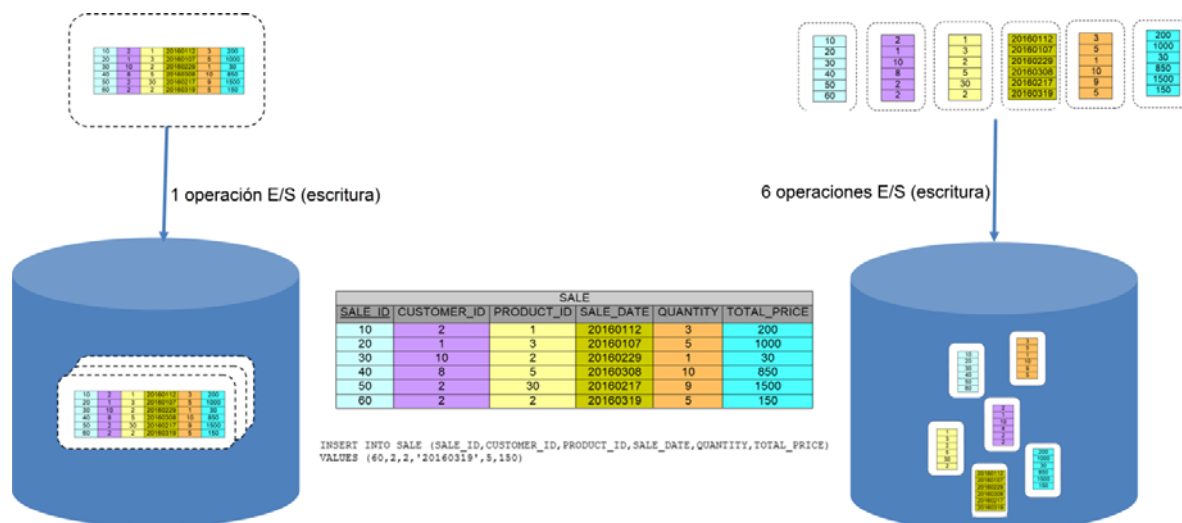


Figura 18. Comparació inserció magatzems de files contra magatzems de columnes (pas II)

Cal destacar que el cost en què s'incorre en l'execució d'operacions de canvi seguint l'enfocament clàssic dels magatzems de files (i que acabem d'exemplificar per al cas d'operacions d'inserció) és inacceptable.

En conseqüència, els magatzems de columnes adopten un enfocament, o millor dit, una arquitectura absolutament diferent per a tractar les operacions de canvi. La terminologia i el funcionament específic pot variar en funció del magatzem de columnes que estiguem considerant. A continuació explicarem, a grans trets, aquesta arquitectura, la qual també es mostra de forma esquemàtica a la figura 19. La terminologia que usarem és la de Vertica/C-Store. Aquells de vosaltres que vulgueu aprofundir en aquestes qüestions podeu consultar algunes de les referències que teniu al final d'aquesta presentació (en concret, la segona i la sisena referència).

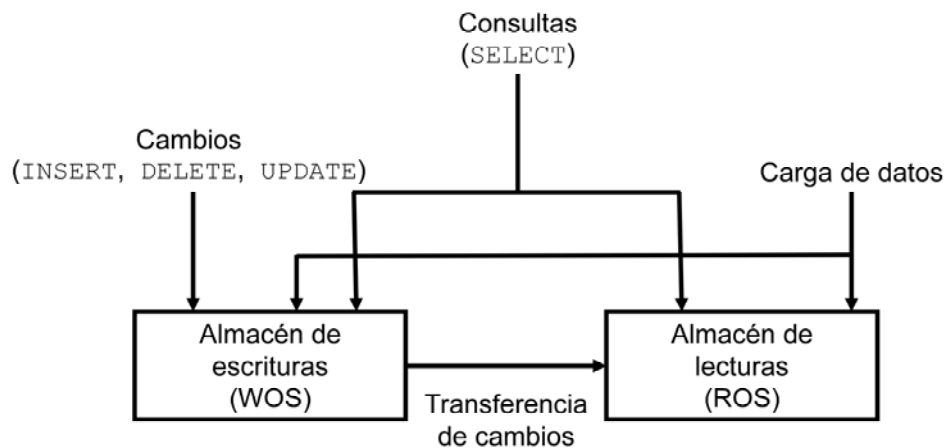


Figura 19. Arquitectura d'un magatzem de columnes

Tal com es pot observar a la figura 19, l'arquitectura consta dels elements següents:

- Un magatzem d'escriptures (en anglès, *write optimized store* o WOS). Aquest magatzem està dissenyat per a resoldre de forma eficient les operacions de canvi (`INSERT`, `DELETE`, `UPDATE`) sobre la BD. Així mateix pot tractar amb processos de càrrega de dades incrementals. El WSO resideix en la memòria principal de l'ordinador i, en general, les dades que guarda no estan comprimides. L'estructura d'emmagatzematge seguida acostuma a ser per columnes. En el cas de Vertica, a manera d'exemple, conté la mateixa estructura de projeccions dissenyada per a la BD (és a dir, segueix el disseny del magatzem de columnes, que és justament el magatzem de lectures o ROS que es mostra a la dreta de la figura 19). Les operacions de canvi funcionen de la manera següent:
 - Operacions d'inserció (`INSERT`): s'apliquen com hem explicat anteriorment, però sense incórrer en el cost de les operacions d'E/S, atès que les insercions es realitzen en estructures de dades que resideixen en la memòria principal de l'ordinador. L'estalvi d'operacions d'E/S també és aplicable a la resta d'operacions de canvi (esborrar i modificar).
 - Operacions d'eliminació (`DELETE`): es pren nota de l'eliminació d'una fila o d'un conjunt de files, usant algun tipus d'estructura auxiliar. A manera d'exemple, en el cas de Vertica, per a cada projecció, s'emmagatzema una columna especial denominada columna d'eliminació (*delete column*, en anglès). Per a cada fila (o tros de fila) esborrada en cada projecció i en la posició que representa l'identificador implícit de la fila (la seva *rowid*), s'emmagatzema l'instant del temps en què la fila deixa d'existir en la BD.
 - Operacions de modificació (`UPDATE`): queden implementades a través d'una seqüència d'eliminacions seguida d'insercions. En altres paraules, l'execució d'una operació de modificació implica l'eliminació de la fila o

files que es modificaran, seguida per la inserció de la fila o files que contenen els valors nous. Per a realitzar les eliminacions i insercions necessaris per a resoldre l'operació de modificació se segueixen les estratègies que hem descrit anteriorment.

- Un magatzem de lectures (*read optimized store*, en anglès, abreujat com ROS). Aquest magatzem està dissenyat per a resoldre de forma eficient les operacions de lectura (`SELECT`) que es formulin sobre la BD. En definitiva, es tracta del magatzem de columnes que hem estat tractant de llarg a llarg en aquesta presentació. D'una banda, admet l'execució de processos de càrrega (en concret, el procés inicial de càrrega massiva de dades). D'altra banda, permet l'execució de canvis que es transfereixen des del WOS. L'SGBD (específicament l'optimitzador de consultes), per a donar resposta a les operacions de lectura (`SELECT`) que es formulin sobre la BD, atacarà ambdues estructures d'emmagatzematge (el WOS i el ROS). L'objectiu és proporcionar informació completa i precisa per a facilitar els processos de presa de decisions dels usuaris.
- Un procés que transfereix de forma asíncrona i en segon pla els canvis del WOS al ROS. Aquest procés, en el cas de Vertica, rep el nom de *tuple mover*. La freqüència amb la qual es transfereixen els canvis és un paràmetre que es pot configurar. Atès que el WOS resideix en la memòria principal de l'ordinador (i en conseqüència, en comparació amb el disc, té una capacitat d'emmagatzematge limitada) és bastant habitual que aquest procés s'executi sempre que la quantitat de dades guardades en el WOS excedeixi un cert llindar. El procés que transfereix els canvis bàsicament és un procés que fusiona el contingut del WOS amb el ROS. En altres paraules, aplica el conjunt d'insercions i purga les dades que s'han eliminat en el ROS. Durant aquest procés, òbviament, s'incorrerà en un cost en termes d'operacions d'E/S (tant de lectura com d'escriptura) com les que hem vist en el nostre exemple d'inserció d'una fila (vegeu les figures 17 i 18). La diferència fonamental és que s'intentarà amortitzar al màxim aquest cost, aplicant un conjunt de canvis sobre una mateixa pàgina o conjunt de pàgines. En altres paraules, si sobre el nostre exemple d'inserció teníem un cost de dotze operacions d'E/S per a inserir una única fila, la idea ara és que amb aquestes dotze operacions d'E/S puguem inserir, esborrar i eliminar un conjunt de files. A mesura que aquest procés de fusió es va executant, es poden eliminar les dades contingudes en el WOS.

Una vegada vist com els magatzems de columnes resolen les operacions de canvi, podem seguir amb les operacions de consulta (`SELECT`).

Com segurament heu intuït al llarg d'aquesta presentació, els magatzems de columnes s'orienten a facilitar la resolució eficient de les consultes que són típiques

en entorns OLAP. Aquestes consultes verifiquen alguna (o diverses) de les condicions següents: les cerques es basen en el valor que prenen algunes columnes, es realitzen agrupacions i agregacions sobre un volum important de dades i els càlculs s'efectuen sobre un nombre reduït de columnes.

Al llarg d'aquesta presentació hem donat alguns exemples d'aquest tipus de consultes i també s'entrarà en aquestes qüestions en una altra presentació separada d'aquest material didàctic dedicat als magatzems de columnes. En aquest punt simplement comentar que, en general, per a la resolució d'aquest tipus de consultes, l'SGBD majoritàriament es basa en la realització de recorreguts seqüencials sobre les estructures d'emmagatzematge que s'hagin creat.

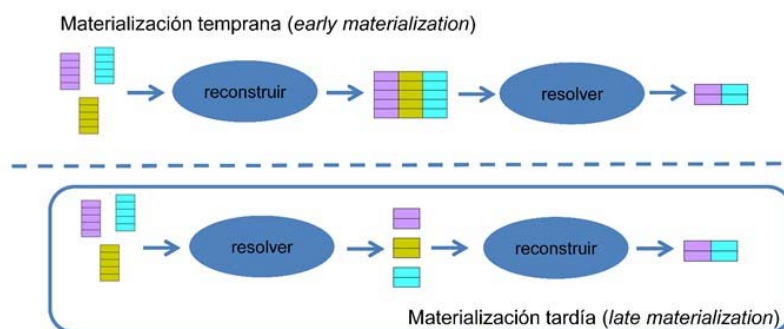
També, i com ja hem comentat en diverses ocasions, l'ús de tècniques de compressió de dades i la definició de diferents ordenacions de dades (projeccions) pot ajudar a millorar el rendiment de les operacions de consulta. En aquest sentit, i arribats a aquest punt, volem comentar que la clau d'ordenació que hem triat per a la superprojecció de la nostra taula d'exemple *SALE* (aquesta clau d'ordenació és la columna *SALE_ID*) no és la més apropiada per a resoldre de forma eficient el tipus de consultes que clàssicament es plantegen en entorns OLAP. L'elecció s'ha realitzat per motius docents, és a dir, per a facilitar la comprensió dels diferents conceptes introduïts al llarg d'aquesta presentació.

Optimización de consultas

▪ Mecanismos específicos para optimizar la ejecución de las consultas:

- Materialización tardía
- Las operaciones actúan sobre múltiples valores a la vez (*block iteration*).
- Las operaciones trabajan sobre datos comprimidos.

} Procesamiento vectorial de consultas



EIMT.UOC.EDU

Com hem vist, els magatzems de columnes resolen eficientment les operacions de consulta (*SELECT*), com seria el cas d'aquelles que es plantegen en entorns OLAP, atès que només han de llegir aquelles columnes que són rellevants per a la resolució de cada consulta. Aquesta visió pot ser simplista, ja que pot conduir a la suposició (errònia) que és possible simular un magatzem de columnes en un magatzem de files a través de la utilització de diferents tècniques i obtenir un rendiment equivalent.

En concret, les tècniques que es poden aplicar en un magatzem de files per a realitzar aquesta simulació són la fragmentació vertical de les taules que conformen la BD, la definició d'índexs sobre les columnes més freqüentment usades en les consultes (en el cas extrem, sobre totes les columnes de cada taula) perquè aquestes puguin ser accedides de forma independent o definint vistes materialitzades. Si bé aquestes tècniques poden ser útils per a millorar el rendiment de certs tipus de consultes, tal com hem argumentat anteriorment, tenen un cost associat en la resolució de les operacions de canvi (*INSERT, DELETE, UPDATE*) sobre la BD i en el manteniment de les estructures auxiliars definides (els índexs, per exemple). En la primera referència que trobareu al final d'aquesta presentació, teniu un estudi per si us interessa aprofundir en aquesta qüestió.

En conseqüència, hi ha diferències fonamentals entre tots dos tipus de sistemes, més enllà de les estructures d'emmagatzematge utilitzades, i que afecten el rendiment. Aquestes diferències tenen a veure amb els mecanismes especialitzats que els magatzems de columnes implementen i que són utilitzats durant el procés de

resolució de les consultes. Aquests mecanismes són diferents als usats pels magatzems de files. Aquests mecanismes són principalment tres:

1. Materialització tardana (en anglès, *late materialization*)
2. Operacions que actuen alhora sobre blocs de valors (*block iteration*, en anglès). La conjunció d'aquest mecanisme amb el previ, també es coneix com a processament vectorial de consultes (en anglès, *vectorized query processing*).
3. La possibilitat que les operacions treballin sobre dades comprimides.

Dels mecanismes previs, els que tenen un impacte major en el rendiment són la materialització tardana i la capacitat d'operar sobre les dades comprimides (podeu consultar la primera referència que s'inclou al final d'aquesta presentació per a obtenir informació més detallada sobre aquesta qüestió).

En un magatzem de columnes, les dades de cada taula s'emmagatzemen en diverses ubicacions en el disc (com ja sabem, cada columna es guarda en un fitxer diferent i cada fitxer només guarda dades d'una columna). No obstant això, els usuaris i els programes que accedeixen a la BD, en última instància, esperen que el resultat d'una consulta sigui un conjunt de files.

Per tant, en el pla d'accés de la majoria de consultes, les dades de diverses columnes s'hauran de combinar en files d'informació. Aquest procés de reconstrucció de files, que també es denomina materialització de files (o simplement materialització), és una operació molt comuna en un magatzem de columnes. Conceptualment, aquesta operació de reconstrucció és molt similar a una operació de combinació.

La qüestió fonamental és a quin moment, durant el procés de resolució de la consulta, es realitza la reconstrucció de les files. En principi, hi ha dos enfocaments (o possibilitats): materialització primerenca (en anglès, *early materialization*) i materialització tardana. Cadascun d'aquests enfocaments es mostra de forma gràfica a la part inferior de la transparència.

La materialització primerenca (vegeu la primera de les figures de la part inferior de la transparència), després de llegir del disc les dades associades a les columnes rellevants per a la resolució de la consulta, realitza la reconstrucció de les files. En definitiva, la materialització primerenca reconstrueix la porció de la taula (o taules) que intervenen en la consulta, guardant-les en la memòria. Aquest procés de reconstrucció s'efectua abans de començar a aplicar el pla d'accés de la consulta. A partir d'aquí, i atès que les dades ja estan en un format de fila, es poden aplicar els algorismes implementats pels magatzems de files per a la resolució de les operacions indicades en la consulta (operacions de selecció, agregacions, combinacions etc.). Si bé això permet estalviar costos de desenvolupament, desaprofita gran part del potencial dels magatzems de columnes a efectes de rendiment. En concret, no es poden aplicar la resta de

mecanismes que hem esmentat (*block iteration* i la possibilitat d'operar directament sobre les dades comprimides).

Per aquests motius els magatzems de columnes, avui, acostumen a aplicar la materialització tardana, la qual es mostra en la segona de les figures de la part inferior de la transparència. En aquest cas, la idea és que les operacions implicades en el pla de la consulta operin directament sobre columnes i no sobre files. En conseqüència, el que es desitja és, en primer lloc, postergar la reconstrucció de les files i, en segon lloc, reconstruir únicament les files necessàries. Per tant, la idea és que la reconstrucció de les files es realitzi tan tard com sigui possible en el pla de la consulta. En l'escenari ideal, la materialització no es realitza fins al final, just abans que el resultat de la consulta sigui retornat als usuaris o als programes que accedeixen a la BD.

A manera d'exemple, i per a veure la diferència entre tots dos enfocaments, suposem que la consulta actua sobre una taula que emmagatzema una gran quantitat de files i que sobre la consulta (si més no) s'apliquen operacions de selecció (predicats) sobre una o diverses columnes. Com més restrictius siguin els predicats de les operacions de selecció (és a dir, quantes menys files verifiquin les condicions expressades en la consulta), més inútil és reconstruir totes les files a l'inici del pla de la consulta, atès que la majoria d'aquestes files seran descartades immediatament.

El segon dels mecanismes usats pels magatzems de columnes per a resoldre eficientment les consultes és la possibilitat que les operacions que s'executen en el pla de la consulta actuïn sobre blocs de valors.

Això vol dir que les operacions actuen sobre columnes i que les operacions reben els valors de la columna a què necessiten accedir en bloc. El resultat d'execució de l'operació és un bloc nou de valors que es passa a la següent operació a executar en el pla d'accés de la consulta. En el cas més simple, assumint que no s'usa la compressió de dades i que les columnes tenen un tipus de dades de longitud fixa, ens podem imaginar aquests blocs de valors com a vectors (*arrays*). Les operacions, durant la seva execució, iteren sobre aquests vectors.

Aquest funcionament difereix significativament dels magatzems de files en què, per a executar una operació sobre una sèrie de files, és necessari iterar sobre cada fila. En cada iteració, sobre la fila en curs, és necessari extreure els valors que són rellevants per a la resolució de l'operació, descartant la resta. En ocasions, el cost d'extreure els valors necessaris de la fila pot excedir el cost d'execució de l'operació que sobre aquests valors es desitja realitzar. Fixeu-vos que l'extracció dels valors de la fila equival a accedir als valors de les columnes que són rellevants en la resolució de l'operació.

El tercer mecanisme implementat en els magatzems de columnes és la possibilitat que les operacions siguin capaces d'operar directament sobre columnes sobre les quals s'hagi usat alguna estratègia de compressió de dades. En definitiva, es tracta

d'evitar incórrer en el cost que suposa la descompressió de les dades. Idealment, la descompressió s'hauria de postergar fins al moment de presentar el resultat d'execució de la consulta als usuaris o programes que accedeixen a la BD.

A més dels mecanismes presentats, és important destacar que poden haver-n'hi altres. Si el magatzem de columnes permet guardar grups de columnes amb diferents ordenacions en el disc (és a dir, si el producte permet definir projeccions), l'elecció de quina és la projecció més convenient per a la resolució de la consulta plantejada ajudarà a millorar encara més el rendiment. Tal com hem vist, la possibilitat de guardar grups de columnes amb diferents ordenacions (projeccions) està disponible, per exemple, a Vertica/C-Store i Amazon Redshift.

Algun d'aquests mecanismes seran tractats amb major profunditat dins d'aquest material didàctic dedicat als magatzems de columnes. Per tant, en aquest punt, és suficient entendre les idees generals subjacents de cadascuna d'aquestes tècniques.

Almacenes de columnas: características

- Características de los almacenes de filas
- Características de los almacenes de columnas

EIMT.UOC.EDU

I fins aquí hem arribat amb aquesta presentació, en què hem analitzat les característiques principals dels magatzems de columnes, comparant-les amb els magatzems de files.

Esperem que us hagi agradat la presentació i que aquesta us hagi servit de molta ajuda.

Ara us presentarem un conjunt d'enllaços i referències d'interès sobre aquest tema.

Referencias

- D.J. Abadi, S. Madden, N. Hachem (2008). Column-stores vs. Row-stores: How different are they really? *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 967–980.
- D.J. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden (2012). The Design and Implementation of Modern Column-Oriented Database Systems. *Foundations and Trends in Databases*, 5(3), pages 197-280.
- J. Conesa, M.E. Rodríguez (2013). Diseño físico de bases de datos. Fundación UOC. PID_00203528.
- Hewlett Packard Enterprise (2016). Vertica Concepts, HPE Vertica Analytic Database, Software version 7.2.x.
- S. Idreos, M.L. Kersten, S. Manegold (2007). Database Cracking. *Proceedings of the 3rd International Conference on Innovative Data Systems Research (CIDR)*, pages 68-78.
- A. Lamb, M. Fuller, R. Varadarajan, N. Tran, B. Vandiver, L. Doshi, C. Bear (2012). The Vertica Analytic Database: C-Store 7 Years Later. *Proceedings of the VLDB Endowment*, 5(12), pages 1790-1801.
- M.E. Rodríguez, J. Conesa (2014). *Bases de datos distribuidas: Diseño*. (<https://vimeo.com/channels/basesdedatos/121665573>)
- R. Segret (2004). Componentes de almacenaje de una base de datos. Fundación UOC. P03/75053/02085.
- M. Stonebraker, D.J. Abadi, A. Batkin, X. Chen, M. Cherniak, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, S. Zdonil (2005). C-Store: A Column-oriented DBMS. *Proceedings of the 31st Very Large Data Bases (VLDB) Conference*, pages 553-564.

EIMT.UOC.EDU

Que tingueu un bon dia!