SELECCIÓ DE VISTES MATERIALITZADES

Una de les utilitats de les vistes materialitzades és fer de magatzem de resultats parcials a partir dels quals es puguin calcular diversos resultats finals de manera més ràpida que fent el càlcul a partir de les dades bàsiques, les que tenim a les taules. Pensem especialment en les consultes multidimensionals, que solen constar de moltes JOINs i fer GROUP BY per agregar mesures. Les diverses consultes d'aquest estil que podem definir a partir d'una determinada BD guarden una relació entre elles: donades dues consultes, pot ser que una es pugui obtenir a partir de l'altra. Imagineu, per exemple, que tenim aquesta taula i aquestes consultes:

```
Persona(id, nom, tel, edat, ciutat, ...)
Q1: SELECT edat, ciutat, COUNT(*) FROM Persona
    GROUP BY edat, ciutat
Q2: SELECT edat, COUNT(*)FROM Persona
    GROUP BY edat
```

Fixeu-vos que podem obtenir Q2 a partir de Q1, però no a l'inrevés (penseu un possible contingut de la taula i observeu el resultat de les consultes donat aquell contingut; veieu com obtenir el resultat de Q2 a partir del de Q1?). Noteu, a més, que el resultat de Q1 ocupa força menys que la taula. En aquestes circumstàncies, ens podem plantejar crear una vista materialitzada MV1 corresponent a Q1 de manera que si cal executar Q1 no caldrà llegir la taula (que ocupa més i es triga més a llegir) sinó MV1 (que ocupa menys i es llegeix més ràpid). Però és que quan calgui executar Q2 també podrem optar per anar més de pressa llegint MV1 en comptes de la taula. I l'estalvi no ve només dels blocs que cal llegir sinó també de la menor complexitat del que cal fer per processar les consultes a partir de MV1 respecte de fer-ho a partir de la taula. Alguns SGBD són capaços de detectar, quan se'ls demana d'executar una consulta definida sobre taules (com ara Q2), si els surt més a compte executar una consulta equivalent definida a partir d'una vista materialitzada (com ara MV1). D'això se'n diu reescriptura de consultes (query rewriting): la consulta Q1 s'ha reescrit i se n'ha obtingut una altra que dóna el mateix resultat però que accedeix a MV1.

Filant una mica més prim, a la transparència 4 hi teniu les condicions que cal que es donin perquè una consulta definida sobre taules base es pugui reescriure per tal d'aprofitar una vista materialitzada concreta:

- El predicat de la consulta (condicions del WHERE, incloent les de JOIN) implica el de la vista. Això garanteix que les tuples que hem d'obtenir (les de la consulta) són un subconjunt de les que tenim (les de la vista). En el senzill exemple que hem posat abans, tant un predicat com l'altre eren *true*, o sigui que es comlia la condició.
- El nivell d'agregació de la consulta és major o igual que el de la vista. És a dir, podem agrupar més a la consulta però no a l'inrevés. Això és el que passa a

l'exemple d'abans: Q2 ajunta tuples de Q1 (les que corresponen a la mateixa edat però a ciutats diferents), Q2 té un nivell d'agregació més alt.

- Les agregacions del SELECT de la consulta es poden calcular a partir de les del SELECT de la vista. Això també passa a l'exemple: el nombre de persones d'una edat es pot calcular a partir del nombre de persones de cada ciutat que tenen aquella edat fent una suma.

A la transparència 5 teniu un altre exemple, una mica més complex, de reescriptura de consultes. Es correspon a les taules i vista que teniu al document addicional "Exemple Query Rewriting". Donades les taules i la vista materialitzada *EuroSales*, la consulta de l'apartat "Want" de la transparència 5 es reescriu donant lloc a la consulta de l'apartat "Get". Fixeu-vos que són equivalents; per exemple, analitzeu quin és el resultat de cada consulta amb el contingut d'exemple que hi ha al document "Exemple Query Rewriting". Proveu ara de decidir si els casos de les transparéncies 6 i 7 permeten o no reescriure la consulta . Hauríeu de veure que no, en el primer cas perquè el predicat de la consulta no implica el de la vista i en el segon perquè no és possible calcular el màxim d'una col.lecció de nombres a partir de la suma d'aquesta col.lecció.

Tot això ens obre una possibilitat d'optimització de les consultes en un entorn decisional: podem considerar totes les consultes que es poden fer i analitzar un graf dirigit de consultes definit segons la relació "es pot reescriure en". Aleshores, en triem unes quantes que guardarem com a vistes materialitzades per tal que el SGBD pugui reescriure les consultes que ens podem esperar que seran útils per als usuaris fent servir les vistes materialitzades que hem triat. El problema que se'ns presenta és com triar el millor conjunt de vistes, atès que hi ha moltes possibilitats. Un mètode que podem fer servir simplifica el problema a costa de renunciar a trobar la millor combinació possible. La simplificació la fem en dos aspectes:

- No considerem el graf complet, sinó que, a partir de les consultes que volem optimitzar, generem un conjunt de vistes candidates a ser materialitzades que no inclou totes les vistes possibles. La transparència 12 explica com generem el conjunt de vistes candidates.
- Apliquem, a partir de les candidates, un algoritme greedy que no troba la millor solució però sí una solució que ens garanteix un resultat prou bo. La transparència 13 explica l'algoritme i les següents són un exemple d'aplicació.

La transparència 14 mostra les dades de l'exemple. Tenim una taula que té 100000 tuples i ocupa 10000 blocs. El temps de llegir un bloc és 1 i el temps de procés de CPU l'ignorem (C = 0). L'atribut *pobl* té 200 valors diferents, l'atribut *edat* en té 100 i l'atribut *cand* en té 10. Volem triar vistes materialitzades per optimitzar quatre consultes i tenim 10140 blocs de disc per a la taula i les vistes. Hem de decidir, fent servir el mètode explicat, quines vistes materialitzarem.

Les diferents versions de la transparència 15 reflecteixen quines són les vistes candidates a ser materialitzades, quin és el graf que surt de la relació de reescriptura entre elles i quants blocs ocupa cadascuna:

- Les vistes candidates inicials són les pròpies consultes a optimitzar (transparència 15, versió 1)
- Ajuntant els GROUP BY de la C1 i la C4 obtenim una nova candidata, C5 (versió 2)
- A la versió 3 hi veiem el graf de possibles reescriptures. Noteu que la relació de reescriptura és transitiva. Podríem, per exemple, deixar de posar l'aresta que uneix C5 i C4 perquè ja hi ha un camí de C5 a C4 passant per C3.
- A la versió 4 es compta quantes files té cada vista. Per fer aquest càlcul, mirem quants grups es poden formar amb el GROUP BY de cada vista. Això es fa multiplicant entre ells el nombre de valors diferents que prenen els atributs del GROUP BY. Això sí, si aquest producte supera les files que té la taula, el nombre de grups que es formaran serà el nombre de tuples de la taula.
- A la versió 5 calculem quants blocs ocupa cada vista. Ho fem amb dues proporcions: files i columnes de la vista respecte les de la taula. En el cas de les columnes, tant a la taula com a la vista hi sumem un atribut més que representa l'espai de la informació de control de cada tupla. Aquestes proporcions multiplicades pels blocs de la taula ens donen els blocs de la vista. A la versió 6 de la transparència 15 hi ha el resultat per a totes les vistes candidates.

A la transparència 16 calculem el cost mitjà de les consultes que volem optimitzar en funció de quina vista candidata materialitzem. Omplim una taula de costos on hi ha una fila per cada vista candidata i una columna per cada consulta a optimitzar, i una columna addicional per al cost mitjà. La primera columna, corresponent a la consulta Q1 l'omplim així:

- Si materialitzem C1, podem resoldre Q1 llegint la taula (10000 blocs, cost 10000) o bé llegint C1 (cost 1). Anotem el cost menor, 1, a la primera posició de la columna.
- Si materialitzem C2, podem resoldre Q1 llegint la taula o llegint C2 (perquè el graf ens indica que C2 permet resoldre C1). Anotem 100, els blocs de C2, com a cost a la segona posició de la columna.
- Si materialitzem C3, només podem resoldre Q1 a partir de la taula. Per tant el cost és 10000.

D'aquesta manera omplim les columnes de Q1, Q2, Q4 i Q4. La darrera columna s'obté fent la suma ponderada de les altres columnes, tenint en compte la freqüència de cada consulta que ens venia donada amb les dades del problema. Ara el que fem és triar com a vista a materialitzar la que té un cost mitjà més petit, C5 en aquest exemple.

A la transparència 17 tornem a generar la taula de costos, ara sense la vista C5 que ja hem decidit materialitzar a la primera volta de l'algoritme *greedy*. Però en realitata no

calia fer-ho perquè en l'espai que queda disponible només hi cap la candidata C1. Per tant, el procés acaba donant com a resultat que s'han de materialitzar les vistes C5 i C1.