

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

2.praktiskais darbs

**Daudzdimensiju datu bāzes izveide**

Izstrādāja: Natans Šalamberidze

171RMC203

Pārbaudīja: lektors Jānis Eiduks

2023./2024. māc. Gads

**Darba uzdevums**

1. Jāveic daudzdimensiju datu bāzes struktūras projektēšana: fakti, dimensijas (vismaz 3), hierarhijas. Par pamatu var tikt izmantota 1. praktiskajā darbā izveidotā struktūra.

2. Jādefinē atbilstošās faktu un dimensiju tabulas. Jāizveido to struktūras grafiskais attēlojums (tas ļaus studentiem pašiem un arī pasniedzējiem vieglāk uztvert definēto datu struktūru un vaicājumus).

3. Izmantojot PL/SQL valodu un gadījuma vērtību ģeneratorus DBMS\_RANDOM.Value, DBMS\_RANDOM\_STRING tabulas jāaizpilda ar datiem (var tikt izmantoti 1. praktiskajā darbā iegūtie rezultāti). Faktu tabulas rakstu skaits >100 000.

4. Jāveic pārbaude (SELECT vaicājums), ka kopējā datu struktūra ir darba spējīga.

5. Izveidotajai datu struktūrai jāizdomā un jārealizē (ar vaicājumiem) dināmisks datu analīzes uzdevums ( vismaz 5 vaicājumi loģiskā secībā). Piemēram, gribam noskaidrot kāpēc viens veikals ir ieguvis mazākus ienākumus, kā pārējie. Jāizanalizē veikalu darbība, jāatrod atšķirības, jānovērtē, kas ir tas atšķirīgais, kas novedis pie mazākiem ienākumiem (protams, studentiem būs citi piemēri).

6. No iepriekš izmantotajiem vaicājumiem, jāizvēlas 3 sarežģītākie laikietilpīgākie vaicājumi un jāizpilda vairākos variantos iegūstot to izpildes laikus, izmantojot EXPLAIN PLAN komandu, jeb uzraksītu nelielu izpildes laika notekšanas PL/SQL programmu (statistika jāieraksta datu tabulās (ne datu bāzē, bet aprakstā):

1) pirmais variants,bez indeksu izmantošanas;

2) otrais variants izmantojot parastos B-koka indeksus;

3) trešais variants izmantojot BitMap dažāda tipa indeksus (3 tipi).

7. Jāizveido 3 materializētie skati (izmantojot iepriekšējos vaicājumus). Jāfiksē to izpildes ātrums. Jāpārbauda vai notiek pārrakstīšanas process (rewriting)).

8. Jāveic dimensiju aprakstu definēšana (create dimension).

9. jāizveido materializēto skatu hierarhija, kura varētu izmantot dimensiju aprakstus.

10. Jāizdara un jāuzraksta **rezultatos pamatoti secinājumi** par daudzdimensiju datu bāzes realizēšanā iegūtajiem rezultatiem.

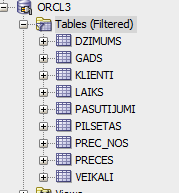
Jādod **loģisks pamatojums secinājumiem**.

11. Jānoformē praktiskais darbs (obligāti WORD dokuments). Visas SQL komandas un PL/SQL programmas darba aprakstā j**āievieto teksta formātā** (ja ir vajadzība, lai ātri varētu veikt pārbaudi). Rezultāti tiek iekļauti darbā attēlu formātā, **"izgriežot"** (lai nav jāievieto viss ekrāna attēlojums) **ekrāna vajadzīgo daļu**.

**1. Jāveic daudzdimensiju datu bāzes struktūras projektēšana: fakti, dimensijas (vismaz 3), hierarhijas. Par pamatu var tikt izmantota 1. praktiskajā darbā izveidotā struktūra.**

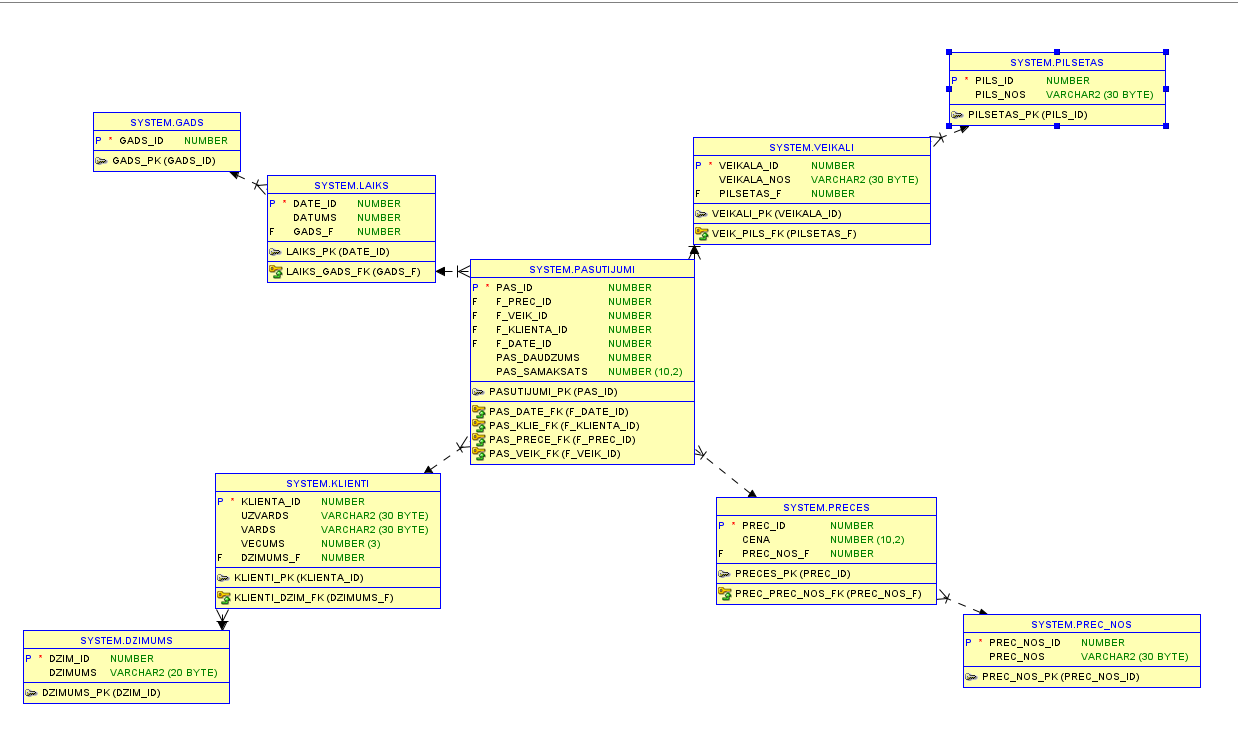
**3. Izmantojot PL/SQL valodu un gadījuma vērtību ģeneratorus DBMS\_RANDOM.Value, DBMS\_RANDOM\_STRING tabulas jāaizpilda ar datiem (var tikt izmantoti 1. praktiskajā darbā iegūtie rezultāti). Faktu tabulas rakstu skaits >100 000.**

Tiks izmantotas 1.praktiska darba tabulas.



Pasutījumi – Faktu tabula.

**2. Jādefinē atbilstošās faktu un dimensiju tabulas. Jāizveido to struktūras grafiskais attēlojums (tas ļaus studentiem pašiem un arī pasniedzējiem vieglāk uztvert definēto datu struktūru un vaicājumus).**

Ir izveidota ER Diagramma ar SQL Developer palīdzību.

**4. Jāveic pārbaude (SELECT vaicājums), ka kopējā datu struktūra ir darba spējīga.**

select \* from pasutijumi p, klienti k, dzimums d, laiks l,

preces pr, prec\_nos nos, veikali v, pilsetas pil

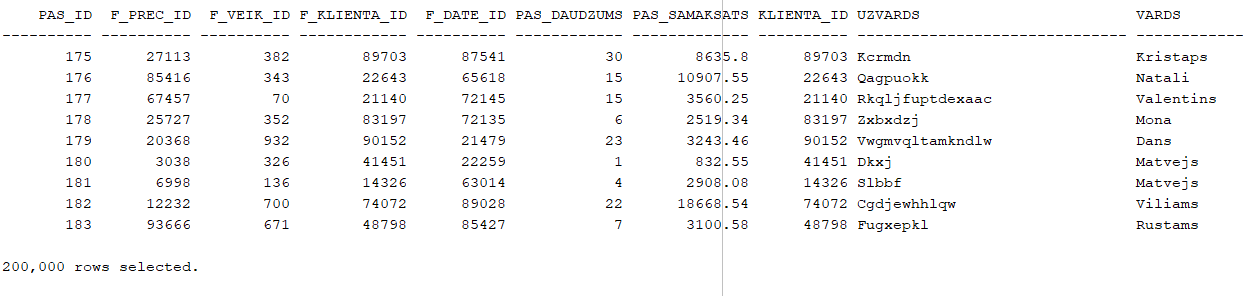
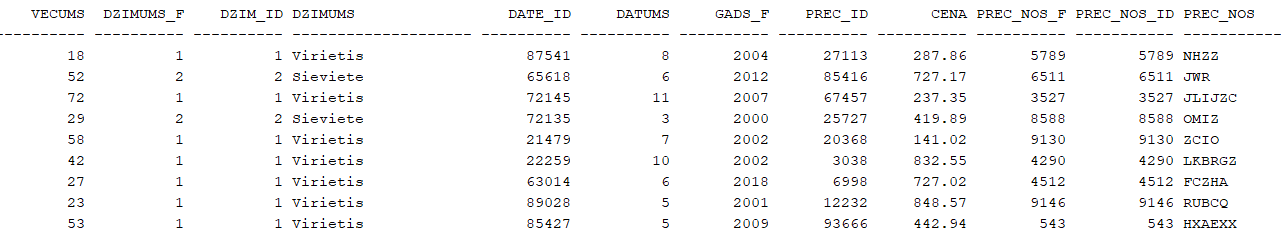
where

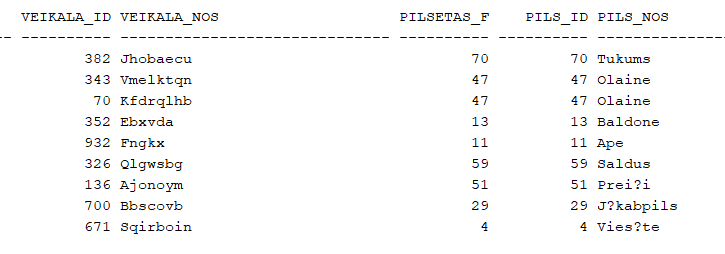
p.F\_klienta\_ID = k.klienta\_ID AND k.dzimums\_f = d.dzim\_ID

AND p.f\_date\_ID = l.date\_id AND p.f\_prec\_id = pr.prec\_id

AND pr.prec\_nos\_f = nos.prec\_nos\_id

AND p.f\_veik\_ID = v.veikala\_id AND v.pilsetas\_f = pil.pils\_id





**5. Izveidotajai datu struktūrai jāizdomā un jārealizē (ar vaicājumiem) dināmisks datu analīzes uzdevums ( vismaz 5 vaicājumi loģiskā secībā).**

**Vaicājums No1**

Ir sasummēti visi ieraksti: Cena, Daudzums, Samaksa visiem veikaliem, visas pilsētas, katrā gadā no 2000-2023.

**select l.gads\_f, sum(pr.cena) as Preces\_cenas\_summa ,**

**sum(p.pas\_daudzums) as Summa, sum(p.pas\_samaksats) as Daudzums**

**from pasutijumi p, laiks l, preces pr**

**where**

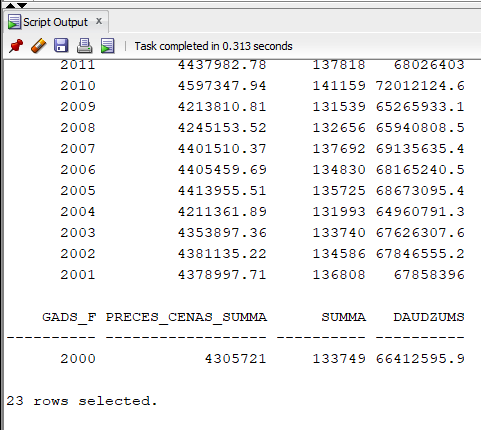
**p.f\_date\_ID = l.date\_id**

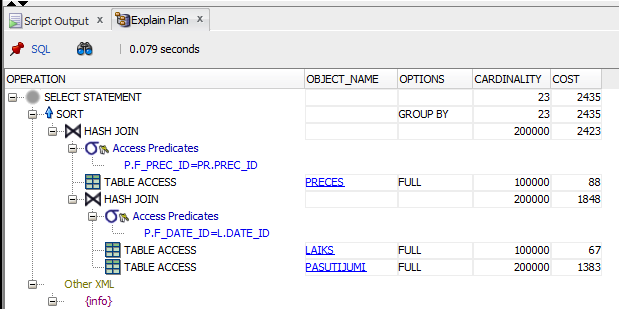
**AND**

**p.f\_prec\_id = pr.prec\_id**

**group by l.gads\_f**

**order by l.gads\_f desc**

****

****

**Vaicājums No2**

Ir iegūta informācija par samaksātu summu pasūtījumiem un pasūtījuma daudzumu abām dzimumiem, par visiem gadiem.

**select d.dzimums, sum(p.pas\_samaksats) as Summa, sum(p.pas\_daudzums) as Daudzums**

**from pasutijumi p, klienti k, dzimums d**

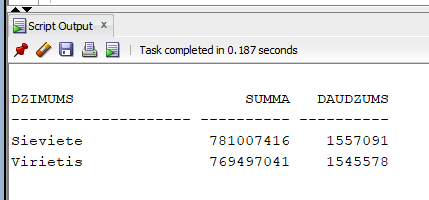
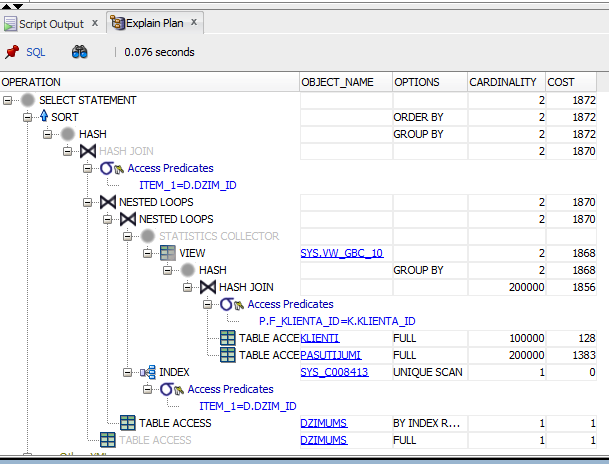
**where**

**p.f\_klienta\_ID = k.klienta\_id**

**and k.dzimums\_f = d.dzim\_id**

**group by d.dzimums**

**order by sum(p.pas\_samaksats) desc**

****

**Vaicājums No3**

Ir iegūta informācija par samaksātu summu pasūtījumiem un pasūtījuma daudzumu, pa veikaliem kur, veikala nosaukums sākas ar -**Vu**

**select veikala\_nos, sum(pas\_samaksats) as Summa**

**from pasutijumi p, veikali v, pilsetas pils**

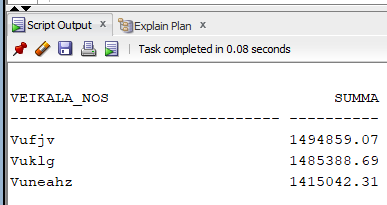
**where**

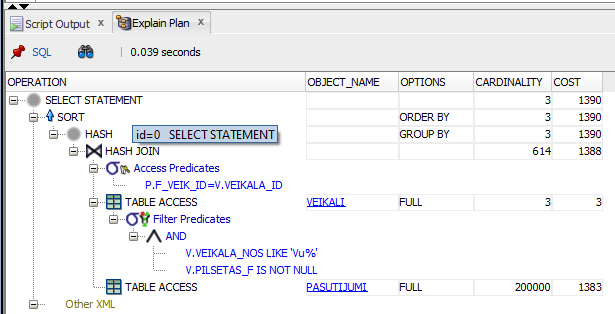
**p.f\_veik\_ID = v.veikala\_id AND v.pilsetas\_f = pils.pils\_id**

**AND v.veikala\_nos LIKE 'Vu%'**

**group by veikala\_nos**

**order by Summa desc**

****

****

**Vaicājums No4**

Ir iegūta informācija par samaksātu summu pasūtījumiem, pa klientiem kur, klienta vārds sākas ar -**Ma**

**select vards, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums**

**from pasutijumi p, klienti k**

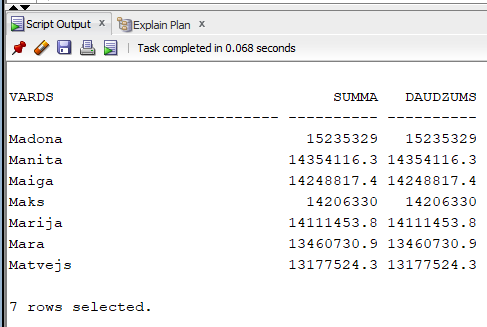
**where**

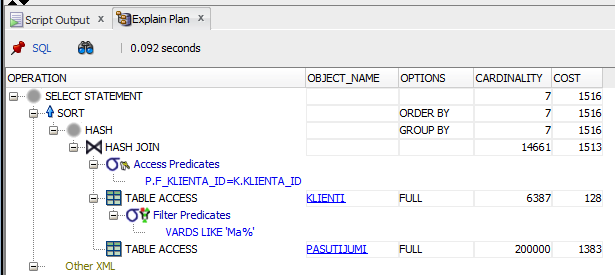
**p.f\_klienta\_ID = k.klienta\_id**

**AND vards LIKE 'Ma%'**

**group by vards**

**order by Summa desc**

****

****

**Vaicājums No5**

Ir iegūta informācija par veiktiem pirkumiem veikalos katrā pilsētā 2005. gadā.

**select pils\_nos, count(v.veikala\_nos) as Veikala\_Nosaukums**

**from pasutijumi p, laiks l, veikali v, pilsetas pils**

**where**

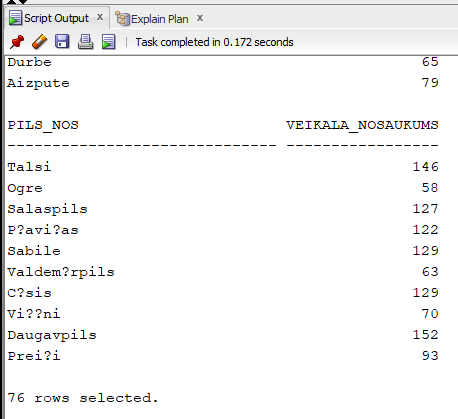
**p.f\_date\_ID = l.date\_id**

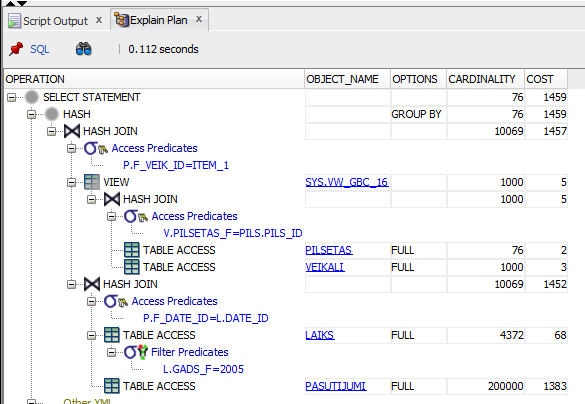
**AND p.f\_veik\_ID = v.veikala\_id**

**AND v.pilsetas\_f = pils.pils\_id**

**AND l.gads\_f = 2005**

**group by pils\_nos**

****

****

**6. No iepriekš izmantotajiem vaicājumiem, jāizvēlas 3 sarežģītākie laikietilpīgākie vaicājumi un jāizpilda vairākos variantos iegūstot to izpildes laikus, izmantojot EXPLAIN PLAN komandu, jeb uzraksītu nelielu izpildes laika notekšanas PL/SQL programmu (statistika jāieraksta datu tabulās (ne datu bāzē, bet aprakstā):**

**1) pirmais variants,bez indeksu izmantošanas;**

**2) otrais variants izmantojot parastos B-koka indeksus;**

**3) trešais variants izmantojot BitMap dažāda tipa indeksus (3 tipi).**

Analizējot COST vērtības visām 5 vaicājumiem, es nolēmu paņemt 1,3 un 4.

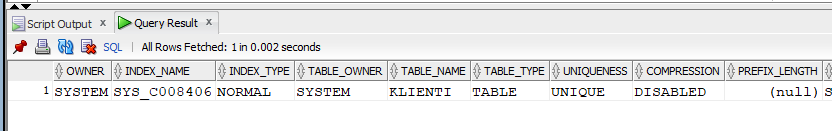
6.1) **Pirmais variants, bez indeksu izmantošanas**

Sākumā es pārbaudīju vai eksistē jebkādi indeksi katrā izveidotajā tabulā ar comandu-

SELECT \* FROM all\_indexes

WHERE

table\_name = ‘KLIENTI’;



Kā ir redzams, nav nekādu indeksu, tikai ir viens SYS indekss, es varu kļūdīties, bet manuprāt tas ir tabulas primāra atslēga kura ir saģenerēta kad mēs veidojam tabulas. Katrā tabulā bija tikai viens šis SYS indekss un nekādu citu. Tāpēc COST vērtības bez indeksu izmantošanas mums ir jau zināmas(2435- Vaicājums No1, 1390- Vaicājums No3, 1516- Vaicājums No4)

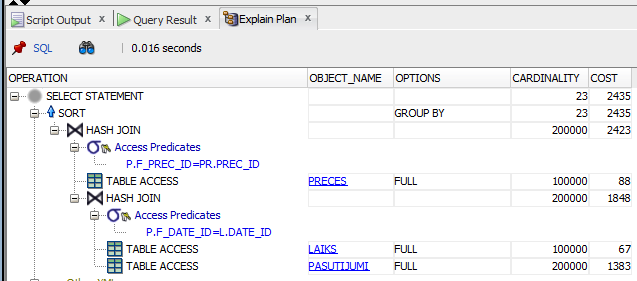
6.2) **Otrais variants izmantojot parastos B-koka indeksus;**

**Vaicājums No1 Vērtība bez indeksa - 2435**

CREATE INDEX idx\_pasutijumi\_date\_id ON PASUTIJUMI(f\_date\_ID);

CREATE INDEX idx\_pasutijumi\_prec\_id ON PASUTIJUMI(f\_prec\_id);

CREATE INDEX idx\_laiks\_gads\_f ON LAIKS(gads\_f);



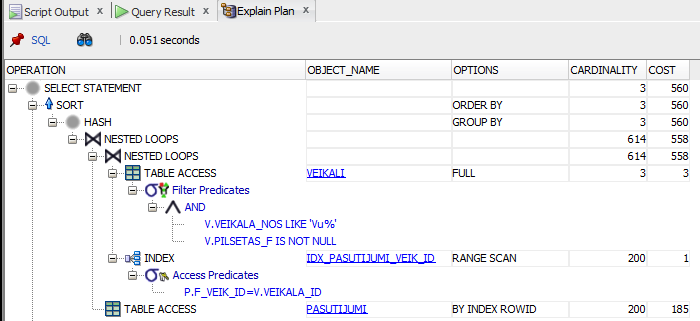
Mēs varam redzēt ka COST vērtība nav mainījusies, tāpēc ka mēs neprasījām informāciju par konkrētu kolonu, bet vaicājums iet cauri visai DB ar komandu GROUP BY. **Un indeksēšana šeit nepalidzēja.**

**Vaicājums No3 Vērtība bez indeksa - 1390**

CREATE INDEX idx\_pasutijumi\_veik\_id ON PASUTIJUMI(f\_veik\_ID);

CREATE INDEX idx\_veikali\_pilsetas\_f ON VEIKALI(pilsetas\_f);

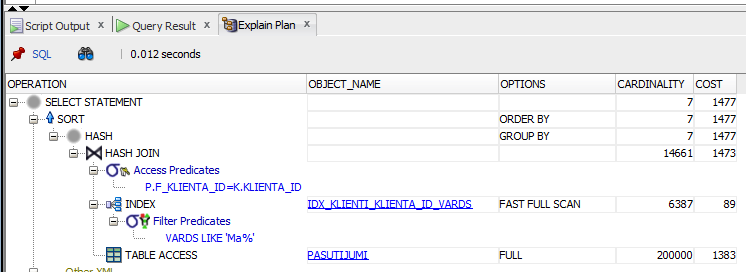
CREATE INDEX idx\_veikali\_veikala\_nos ON VEIKALI(veikala\_nos)



Šeit jau ir redzama atšķirība, ka COST vērtība **samazinājās apmēram uz 59%. Bija-1390, kļuva-560**. Arī ir redzams kolonnā OPTIONS, kā tabula PASUTIJUMI ir skanēta nevis FULL ar CARDINALITY 200000, bet jau BY INDEX ROWID ar CARDINALITY 200. **idx\_pasutijumi\_veik\_id** indeks izdarīja savu darbu.

**Vaicājums No4 Vērtība bez indeksa – 1516**

CREATE INDEX idx\_klienti\_klienta\_id\_vards ON KLIENTI(klienta\_id, vards);



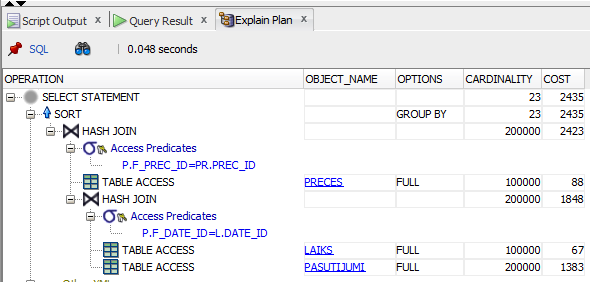
Šeit arī ir redzama atšķirība starp COST vērtībām **bija 1516 un kļuva 1477**, mēs to dabūjam ar indeksa palīdzību kurš iedarbināja FAST FULL SCAN, tieši šajā darbībā mēs samazinām vērtību par 39.

6.3) **trešais variants izmantojot BitMap dažāda tipa indeksus (3 tipi).**

**Vaicājums No1 Vērtība bez indeksa - 2435**

**Tips No1**

create BitMap index Bit\_prec on PRECES(cena);



Redzam ka vērtība nav mainies. BitMap indeksa izveide šeit būs nefiktīva, jo BitMap indeksus parasti izmanto zemas kardinalitātes kolonnām ar ierobežotu atšķirīgu vērtību kopu.

**Vaicājums No3 Vērtība bez indeksa - 1390**

**Tips No2**

**Ir BitMap indeksa izmantošana saistot divas tabulas (faktu tabula un dimensujas tabula)**

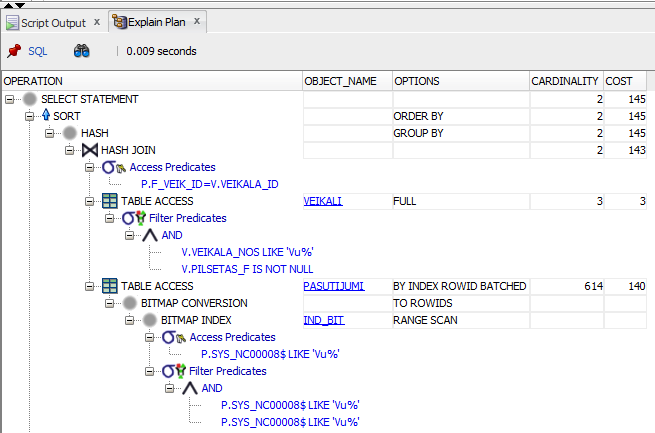
create BitMap index Ind\_Bit on PASUTIJUMI(VEIKALA\_NOS)

from pasutijumi p, veikali v

where

p.f\_veik\_ID = v.veikala\_id

Pēc 3 vaicājuma izpildes mēs redzam ka **IND\_BIT** indeks veiksmīgi pārskrēja faktu tabulu. Un COST vērtība samazinājās**, bija 1390 kļuva 145**. vajag ari pieminēt ka BitMap indeksēšana ar šo vaicājumu nodarbojas labāk nekā B-koka indeksēšana. Protams saņemtie tabulas dati nav mainījušies.



**Vaicājums No4 Vērtība bez indeksa - 1516**

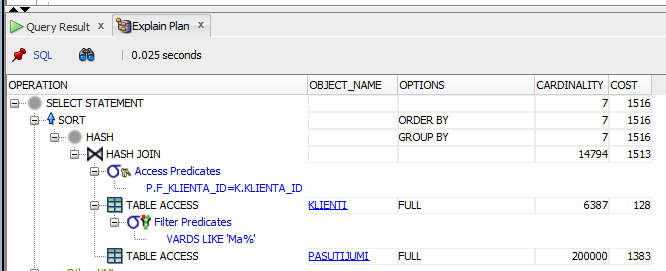
**Tips No3**

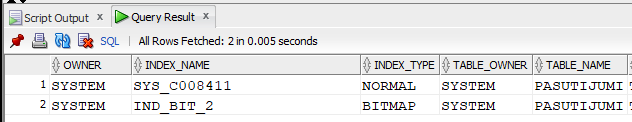
**Vienas dimensiju tabulas viena lauka sasistē ar faktu tabulu.**

create BitMap index Ind\_Bit\_2 on PASUTIJUMI (k.vards)

from pasutijumi p, klienti k

where k.klienta\_id = p.f\_klienta\_id;





Šajā vaicājumā nevar redzēt indeksu kuru es uztaisīju un godīgi sakot es nezinu kāpēc tā sanāk.

7. **Jāizveido 3 materializētie skati (izmantojot iepriekšējos vaicājumus). Jāfiksē to izpildes ātrums. Jāpārbauda vai notiek pārrakstīšanas process (rewriting)).**

**Vaicājums No1 Vērtība bez indeksa - 2435**

create materialized view GADS\_MS

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select l.gads\_f, sum(pr.cena) as Preces\_cenas\_summa ,

sum(p.pas\_daudzums) as Summa, sum(p.pas\_samaksats) as Daudzums

from pasutijumi p, laiks l, preces pr

where

p.f\_date\_ID = l.date\_id

AND

p.f\_prec\_id = pr.prec\_id

group by l.gads\_f

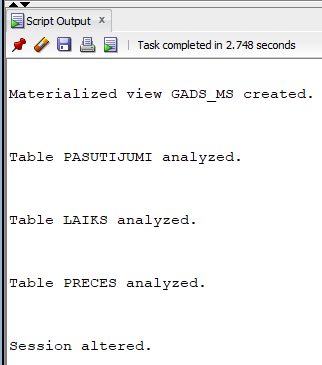
order by l.gads\_f desc;

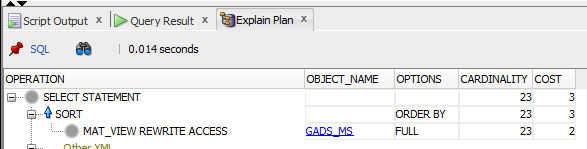
analyze table pasutijumi COMPUTE STATISTICS;

analyze table laiks COMPUTE STATISTICS;

analyze table preces COMPUTE STATISTICS;

alter session set OPTIMIZER\_MODE = ALL\_ROWS;





Ir redzama ļoti lielas atšķirības COST vērtībai, ka arī vaicājuma izpildes laiku. Tas nav pārsteidzoši, jo materializētie skati (MATERIALIZED VIEW), kā arī parastie skati (VIEW) ir domāti lai samazināt vaicājuma veiktspēju.

**Vaicājums No3 Vērtība bez indeksa - 1390**

create materialized view VEIKALS\_MS

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select veikala\_nos, sum(pas\_samaksats) as Summa

from pasutijumi p, veikali v, pilsetas pils

where

p.f\_veik\_ID = v.veikala\_id AND v.pilsetas\_f = pils.pils\_id

AND v.veikala\_nos LIKE 'Vu%'

group by veikala\_nos

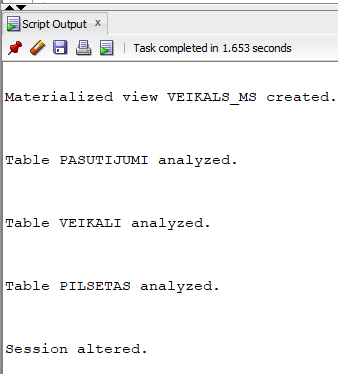
order by Summa desc

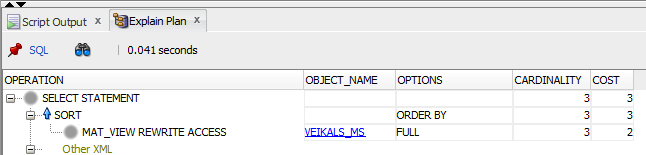
analyze table pasutijumi COMPUTE STATISTICS;

analyze table veikali COMPUTE STATISTICS;

analyze table pilsetas COMPUTE STATISTICS;

alter session set OPTIMIZER\_MODE = ALL\_ROWS;





Kā bija analizēts pirmajā gadījumā. Būtiski samazinājās COST vērtība un vaicājuma izpildes laiks. Protams 3 vaicājumā būs tādi paši rezultāti.

**Vaicājums No4 Vērtība bez indeksa – 1516**

create materialized view VARDS\_MS

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select vards, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums

from pasutijumi p, klienti k

where

p.f\_klienta\_ID = k.klienta\_id

AND vards LIKE 'Ma%'

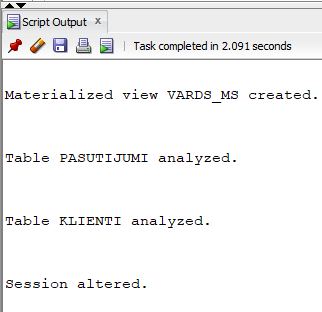
group by vards

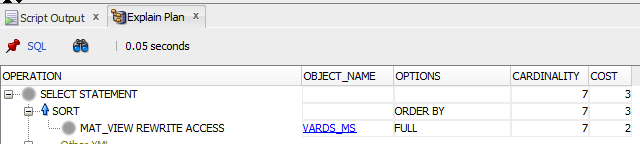
order by Summa desc

analyze table pasutijumi COMPUTE STATISTICS;

analyze table klienti COMPUTE STATISTICS;

alter session set OPTIMIZER\_MODE = ALL\_ROWS;





Kā bija domāts, ir dabūti ļoti labi rezultāti, kā iepriekšējos variantos. Var secināt ka materializētie skati ļoti labi strādā ar daudz dimensijas datubāzēm.

8. **Jāveic dimensiju aprakstu definēšana (create dimension).**

Veidojam dimensijas tabulam VEIKALI, PRECES, LAIKS, KLIENTI.

CREATE dimension DIM\_VEIKALI

level L\_PIL is PILSETAS.PILS\_NOS

level L\_VEIK is VEIKALI.VEIKALA\_NOS

hierarchy HIERARCHY\_veikali(

L\_VEIK child of

L\_PIL

JOIN KEY(VEIKALI.pilsetas\_f ) REFERENCES L\_PIL)

ATTRIBUTE L\_PIL DETERMINES PILSETAS.PILS\_ID

ATTRIBUTE L\_VEIK DETERMINES VEIKALI.VEIKALA\_ID;

CREATE dimension DIM\_PRECES

level L\_PRNOS is PREC\_NOS.prec\_nos

level L\_PRECES is PRECES.CENA

hierarchy HIERARCHY\_preces(

L\_PRECES child of

L\_PRNOS

JOIN KEY(PRECES.PREC\_NOS\_F ) REFERENCES L\_PRNOS)

ATTRIBUTE L\_PRECES DETERMINES PRECES.PREC\_ID

ATTRIBUTE L\_PRNOS DETERMINES PREC\_NOS.PREC\_NOS\_ID;

CREATE dimension DIM\_LAIKS

level L\_GADS is GADS.GADS\_ID

level L\_DATE is LAIKS.DATUMS

hierarchy HIERARCHY\_laiks(

L\_DATE child of

L\_GADS

JOIN KEY(LAIKS.GADS\_F) REFERENCES L\_GADS )

ATTRIBUTE L\_GADS DETERMINES GADS.GADS\_ID

ATTRIBUTE L\_DATE DETERMINES LAIKS.DATE\_ID;

CREATE dimension DIM\_Klienti

level L\_DZIM is DZIMUMS.DZIMUMS

level L\_KLT is KLIENTI.KLIENTA\_ID

hierarchy HIERARCHY\_klienti(

L\_KLT child of

L\_DZIM

JOIN KEY(KLIENTI.DZIMUMS\_F) REFERENCES L\_DZIM)

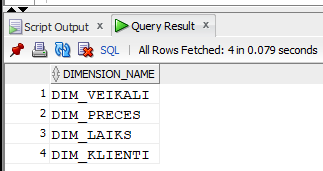
ATTRIBUTE L\_DZIM DETERMINES DZIMUMS.DZIM\_ID

ATTRIBUTE L\_KLT DETERMINES KLIENTI.KLIENTA\_ID;

Lai pārbaudīt izveidotus dimensijas izmantojam komandu:

select DIMENSION\_NAME

from ALL\_DIMENSIONS;



9. **jāizveido materializēto skatu hierarhija, kura varētu izmantot dimensiju aprakstus.**

Iepriekšējā uzdevumā es izveidoju dimensiju tabulai Klienti un tagad ir uztaisījis nelielu materializēta skata hierarhiju. Es paņēmu vaicājumu No4.

select vards, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums

from pasutijumi p, klienti D1

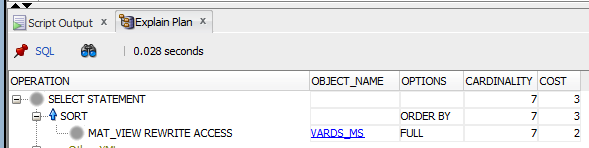
where

p.f\_klienta\_ID = D1.klienta\_id

AND vards LIKE 'Ma%'

group by vards

order by Summa desc



create materialized view MAT\_SKATS\_1

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select D1.vards v, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums

from pasutijumi p, klienti D1

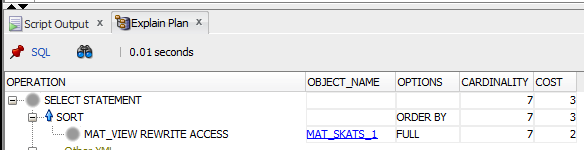
where

p.f\_klienta\_ID = D1.klienta\_id

AND vards LIKE 'Ma%'

group by D1.vards

order by Summa desc;



create materialized view MAT\_SKATS\_1\_1

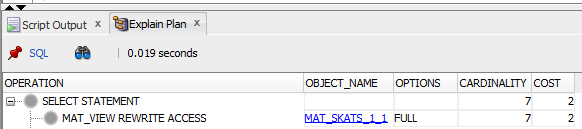
BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select v, SUM(Summa), SUM(Daudzums)

from MAT\_SKATS\_1

group by v



create materialized view MAT\_SKATS\_2

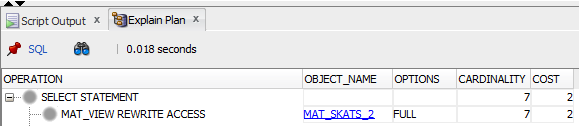
BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select v

from MAT\_SKATS\_1\_1

group by v



**Tabula ar datiem**

|  |  |  |  |
| --- | --- | --- | --- |
| Vaicājums | 1. | 3. | 4. |
| Bez indeksēšanas | 2435 | 1390 | 1516 |
| B-koka indk. | 2435 | 560 | 1477 |
| BitMap indk. | 2435 | 145 | Indeks nav redzams |
| Materializētais skats | 3 | 3 | 3 |

Tabulā ir parādītas visas vaicājuma vērtības (COST) un indeksēšanas tipu, kuras bija izmantotas tajos vaicājumos.

DARBA APRAKSTS

/\*

Select vaicājums lai pārbaudīt tabulas

/\*

select \* from pasutijumi p, klienti k, dzimums d, laiks l,

preces pr, prec\_nos nos, veikali v, pilsetas pil

where

p.F\_klienta\_ID = k.klienta\_ID AND k.dzimums\_f = d.dzim\_ID

AND p.f\_date\_ID = l.date\_id AND p.f\_prec\_id = pr.prec\_id

AND pr.prec\_nos\_f = nos.prec\_nos\_id

AND p.f\_veik\_ID = v.veikala\_id AND v.pilsetas\_f = pil.pils\_id

/\*

1 vaicāajums

/\*

select l.gads\_f, sum(pr.cena) as Preces\_cenas\_summa ,

sum(p.pas\_daudzums) as Summa, sum(p.pas\_samaksats) as Daudzums

from pasutijumi p, laiks l, preces pr

where

p.f\_date\_ID = l.date\_id

AND

p.f\_prec\_id = pr.prec\_id

group by l.gads\_f

order by l.gads\_f desc

/\*

2 vaicāajums

/\*

select d.dzimums, sum(p.pas\_samaksats) as Summa, sum(p.pas\_daudzums) as Daudzums

from pasutijumi p, klienti k, dzimums d

where

p.f\_klienta\_ID = k.klienta\_id

and k.dzimums\_f = d.dzim\_id

group by d.dzimums

order by sum(p.pas\_samaksats) desc

/\*

3 vaicāajums

/\*

select veikala\_nos, sum(pas\_samaksats) as Summa

from pasutijumi p, veikali v, pilsetas pils

where

p.f\_veik\_ID = v.veikala\_id AND v.pilsetas\_f = pils.pils\_id

AND v.veikala\_nos LIKE 'Vu%'

group by veikala\_nos

order by Summa desc

/\*

4 vaicāajums

/\*

select vards, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums

from pasutijumi p, klienti k

where

p.f\_klienta\_ID = k.klienta\_id

AND vards LIKE 'Ma%'

group by vards

order by Summa desc

/\*

5 vaicāajums

/\*

select pils\_nos, count(v.veikala\_nos) as Veikala\_Nosaukums

from pasutijumi p, laiks l, veikali v, pilsetas pils

where

p.f\_date\_ID = l.date\_id

AND p.f\_veik\_ID = v.veikala\_id

AND v.pilsetas\_f = pils.pils\_id

AND l.gads\_f = 2005

group by pils\_nos

/\*

Visi B-koka indeksi kuri bija uzrakstīti

/\*

CREATE INDEX idx\_pasutijumi\_date\_id ON PASUTIJUMI(f\_date\_ID);

CREATE INDEX idx\_pasutijumi\_prec\_id ON PASUTIJUMI(f\_prec\_id);

CREATE INDEX idx\_laiks\_gads\_f ON LAIKS(gads\_f);

CREATE INDEX idx\_pasutijumi\_veik\_id ON PASUTIJUMI(f\_veik\_ID);

CREATE INDEX idx\_veikali\_pilsetas\_f ON VEIKALI(pilsetas\_f);

CREATE INDEX idx\_veikali\_veikala\_nos ON VEIKALI(veikala\_nos)

CREATE INDEX idx\_klienti\_klienta\_id\_vards ON KLIENTI(klienta\_id, vards);

/\*

Visi BitMap indeksi kuri bija uzrakstīti

/\*

create BitMap index Bit\_prec on PRECES(cena);

create BitMap index Ind\_Bit on PASUTIJUMI(VEIKALA\_NOS)

from pasutijumi p, veikali v

where

p.f\_veik\_ID = v.veikala\_id

create BitMap index Ind\_Bit\_2 on PASUTIJUMI (k.vards)

from pasutijumi p, klienti k

where k.klienta\_id = p.f\_klienta\_id;

/\*

Visi materializētie skati

/\*

create materialized view GADS\_MS

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select l.gads\_f, sum(pr.cena) as Preces\_cenas\_summa ,

sum(p.pas\_daudzums) as Summa, sum(p.pas\_samaksats) as Daudzums

from pasutijumi p, laiks l, preces pr

where

p.f\_date\_ID = l.date\_id

AND

p.f\_prec\_id = pr.prec\_id

group by l.gads\_f

order by l.gads\_f desc;

analyze table pasutijumi COMPUTE STATISTICS;

analyze table laiks COMPUTE STATISTICS;

analyze table preces COMPUTE STATISTICS;

alter session set OPTIMIZER\_MODE = ALL\_ROWS;

create materialized view VEIKALS\_MS

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select veikala\_nos, sum(pas\_samaksats) as Summa

from pasutijumi p, veikali v, pilsetas pils

where

p.f\_veik\_ID = v.veikala\_id AND v.pilsetas\_f = pils.pils\_id

AND v.veikala\_nos LIKE 'Vu%'

group by veikala\_nos

order by Summa desc

analyze table pasutijumi COMPUTE STATISTICS;

analyze table veikali COMPUTE STATISTICS;

analyze table pilsetas COMPUTE STATISTICS;

alter session set OPTIMIZER\_MODE = ALL\_ROWS;

create materialized view VARDS\_MS

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select vards, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums

from pasutijumi p, klienti k

where

p.f\_klienta\_ID = k.klienta\_id

AND vards LIKE 'Ma%'

group by vards

order by Summa desc

analyze table pasutijumi COMPUTE STATISTICS;

analyze table klienti COMPUTE STATISTICS;

alter session set OPTIMIZER\_MODE = ALL\_ROWS;

/\*

4 izveidotas dimensijas

/\*

CREATE dimension DIM\_VEIKALI

level L\_PIL is PILSETAS.PILS\_NOS

level L\_VEIK is VEIKALI.VEIKALA\_NOS

hierarchy HIERARCHY\_veikali(

L\_VEIK child of

L\_PIL

JOIN KEY(VEIKALI.pilsetas\_f ) REFERENCES L\_PIL)

ATTRIBUTE L\_PIL DETERMINES PILSETAS.PILS\_ID

ATTRIBUTE L\_VEIK DETERMINES VEIKALI.VEIKALA\_ID;

CREATE dimension DIM\_PRECES

level L\_PRNOS is PREC\_NOS.prec\_nos

level L\_PRECES is PRECES.CENA

hierarchy HIERARCHY\_preces(

L\_PRECES child of

L\_PRNOS

JOIN KEY(PRECES.PREC\_NOS\_F ) REFERENCES L\_PRNOS)

ATTRIBUTE L\_PRECES DETERMINES PRECES.PREC\_ID

ATTRIBUTE L\_PRNOS DETERMINES PREC\_NOS.PREC\_NOS\_ID;

CREATE dimension DIM\_LAIKS

level L\_GADS is GADS.GADS\_ID

level L\_DATE is LAIKS.DATUMS

hierarchy HIERARCHY\_laiks(

L\_DATE child of

L\_GADS

JOIN KEY(LAIKS.GADS\_F) REFERENCES L\_GADS )

ATTRIBUTE L\_GADS DETERMINES GADS.GADS\_ID

ATTRIBUTE L\_DATE DETERMINES LAIKS.DATE\_ID;

CREATE dimension DIM\_Klienti

level L\_DZIM is DZIMUMS.DZIMUMS

level L\_KLT is KLIENTI.KLIENTA\_ID

hierarchy HIERARCHY\_klienti(

L\_KLT child of

L\_DZIM

JOIN KEY(KLIENTI.DZIMUMS\_F) REFERENCES L\_DZIM)

ATTRIBUTE L\_DZIM DETERMINES DZIMUMS.DZIM\_ID

ATTRIBUTE L\_KLT DETERMINES KLIENTI.KLIENTA\_ID;

/\*

Materializēta skata hierarhija ar

/\*

create materialized view MAT\_SKATS\_1

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select D1.vards v, sum(pas\_samaksats) as Summa, sum(pas\_samaksats) as Daudzums

from pasutijumi p, klienti D1

where

p.f\_klienta\_ID = D1.klienta\_id

AND vards LIKE 'Ma%'

group by D1.vards

order by Summa desc;

create materialized view MAT\_SKATS\_1\_1

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select v, SUM(Summa), SUM(Daudzums)

from MAT\_SKATS\_1

group by v

create materialized view MAT\_SKATS\_2

BUILD IMMEDIATE

ENABLE QUERY REWRITE as

select v

from MAT\_SKATS\_1\_1

group by v

SECINĀJUMI

Ir paveikts liels darbs ar daudzdimensiju datu bāzi. Pats galvenais šim praktiskajām darbam ir labi saprot kas un kā, vajadzētu būt izpratnei par visām lietām kas bija šeit aplūkots un par visam uzdevumiem kuri bija uzdotas. Jā par to visu ir zināšanas un izpratne, tikai tad tu varēsi izveidot stabilo, labu datu bāzi, kurā būs lietoti visas izveidotas tabulas un dati tajos tabulās. Tas pats attiecās vaicājumiem. Vajag saprast kāda veida un tipa indeksēšanu lietot katrām vaicājumam, jo ka mēs jau zinām, ne katra indeksēšana palīdz samazināt vaicājuma veiktspēju. Lai labi būtu redzami rezultāti, vajag rakstīt pareizos, pilnvērtīgos vaicājumus.

Un atlasiet tos, kas ir piemēroti darba analīzei. Runājot par rezultātiem, manuprāt tie ir ne tik sliktas. Es daudz ko uzzināju par indeksiem un pārliecinājās, ka darbojas skati, it īpaši materializētie skati. Pateicoties uzskatiem, man izdevās ievērojami samazināt izmaksas. B-koka un BitMap indeksēšana ari nebija veltīga. Visi rezultāti ir atrodami tabulā. Kāpēc man neizdevās sasniegt pozitīvu rezultātu ar pirmo pieprasījumu? Manuprāt tāpēc ka BitMap indeksēšana nav domāti lielai datubāzei ar ierakstiem, manā gadījumā <10000. Veicot šo darbu, es praksē uzzināju, kādus vaicājumus vislabāk izmantot B-koka un BitMap indeksus. Es apmierināts ar savu darbu, jo neskatoties uz kļūdām un nepilnībām es daudz ko jauno uzzināju. Un beigas es gribu atvainoties, ka darbs nav nodots paredzētajā laikā.