

SQL andmekäitluskeele lausete töötlemine ja optimeerimine

Teema 7

Andmebaasid II 2017

© Erki Eessaar

SQL andmekäitluskeele lausete töötlemine ja töötlemise tulemused

- ♦ Dekompositsioon
 - Analüüsi puu
- ♦ Täitmisplaani koostamine ja parandamine
 - Loogiline täitmisplaan – relatsioonalgebra puu
 - Füüsiline täitmisplaan – relatsioonalgebra operatsioonide realiseerivate sisemise taseme operatsioonide kogum
- ♦ Täitmisplaani täitmine
 - Füüsilises täitmisplaanis ette nähtud operatsioonide täitmine ettenähtud järjekorras

Kõiki neid tegevusi teeb andmebaasisüsteem!!!

15.12.2017

Teema 7

2

Andmebaasid II 2017

© Erki Eessaar

SQL andmekäitluskeele lausete töötlemine (2)

SELECT nimetus FROM Aine_seisundi_liik;

Execution Plan

Plan hash value: 970303617

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	156	2 (0)	00:00:01
1	INDEX FAST FULL SCAN	AK_C_AINE_SEISUND_NIMETUS	3	156	2 (0)	00:00:01

NIMETUS

kinnitatud
loodud
mitteaktuaalne

Lause täitmise
tulemus

Täitmisplaan – **imperatiivne**
programm lause täitmiseks –
kuidas andmed leida

SQL SELECT lause –
mida tuleb leida
(**deklaratiivne** lause)

15.12.2017

Teema 7

3

Andmebaasid II 2017

© Erki Eessaar

Faas 1 – dekompositsioon

- ♦ Luuakse analüüsi puu. **Süntaksi** kontroll.
- ♦ Kui lause on tehtud vaate põhjal, siis vaate **lahतिकirjutamine** e vaate aluseks oleva lause ning vaate põhjal tehtud lause **mestimine** ja analüüsi puu täiendamine.
- ♦ **Semantiline** analüüs.

15.12.2017

Teema 7

4

Andmebaasid II 2017

© Erki Eessaar

Päringu näide

```
SELECT A.aine_kood, Ais.nimetus
FROM Aine A INNER JOIN Aine_seisund Ais
ON A.aine_seisund_id=Ais.aine_seisund_id
WHERE A.aine_kood='IDU3381' AND
Ais.nimetus='kinnitatud';
```

15.12.2017

Teema 7

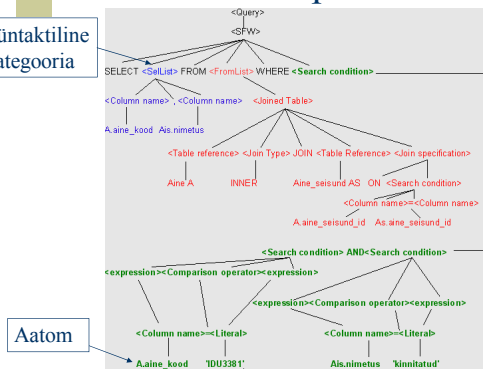
5

Andmebaasid II 2017

© Erki Eessaar

Analüüsi puu näide

Süntaktiline
kategooria



15.12.2017

Teema 7

6

Päringut saaks lihtsustada

- ♦ `SELECT aine_kood FROM Aine WHERE aine_kood='IDU3381' AND aine_seisund_id=2;`
- ♦ Tegelikult tahan teada, kas aine IDU3381 on seisundis kinnitatud või mitte.
- ♦ SQLi olemuslik puudus on, et seal ei saa hästi formuleerida jah/ei vastust andvaid päringuid.

Vaate lahtikirjutamine – näide

- ♦ `CREATE VIEW IDU_Ained AS SELECT aine_kood, nimetus, on_aktuaalne FROM Aine WHERE aine_kood LIKE 'IDU%' WITH CHECK OPTION;`
- ♦ `SELECT aine_kood FROM IDU_Ained WHERE on_aktuaalne=TRUE;`
- ♦ `SELECT aine_kood FROM Aine WHERE aine_kood LIKE 'IDU%' AND on_aktuaalne=TRUE;`



Vaate lahtikirjutamine (2)

- ♦ Keerukamate lausete puhul võib juhtuda, et optimeerija ei suuda vaadet ja selle põhjal tehtud päringut mestida.
- ♦ Vaate alampäring täidetakse eraldi, selle tulemus on sisend ülejäänud päringusse.
 - Täitmisplaanis VIEW operatsioon.
- ♦ Ei pruugi olla optimaalseim lahendus, st lause täitmise kiirus pole parim võimalik.

Vaate lahtikirjutamine (3)

- ♦ Kašnikova, D., 2015. *Vaadete mõju päringute täitmisplaanide koostamisele kahe andmebaasisüsteemi näitel*. Magistritöö. TTÜ Informaatikainstituut. [WWW] <https://digi.lib.ttu.ee/i/?3676>
- ♦ Kokkuvõte:
 - Vaadete kasutamine mõjutab täitmisplaanide valikut ja võib mõnikord viia ebaotstarbekate plaanide valimiseni.
 - Vaadetele siiski palju muid eeliseid, mis seda kompenseerivad. Soovitan vaateid kasutada!

Semantiline analüüs

- ♦ Kas baastabelid/hetktõmmised ja nende veerud on olemas?
- ♦ Kas kasutajal on olemas vajalikud õigused?
- ♦ Kas lauses on antud piisavalt informatsiooni selle täitmiseks?
 - `SELECT aine_seisund_id FROM Aine A INNER JOIN Aine_seisund AS ON A.aine_seisund_id=As.aine_seisund_id;`
- ♦ Kas lauses andmebaasiobjektile rakendatav operatsioon on selle objekti tüübi jaoks sobiv?

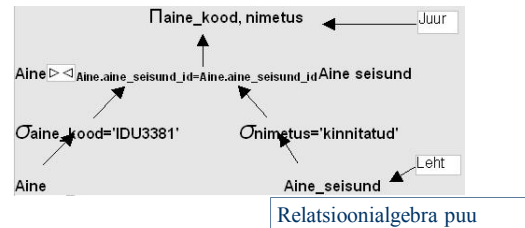
Faas 2 – täitmisplaani koostamine ja parandamine

- ♦ Loogilise täitmisplaani koostamine analüüsi puu põhjal.
- ♦ Loogilise täitmisplaani parandamine.
 - Normaliseerimine – tingimuse konjunktiivsele normaalkujule viimine (valikuline).
 - Tingimuse lihtsustamine (valikuline).
 - Semantiline teisendamine (valikuline).
 - Vastuolude otsimine (valikuline).
 - Optimaalse tegevuste järjekorra valimine.

Faas 2 – täitmisplaani koostamine ja parandamine

- ◆ Füüsilise täitmisplaani koostamine.
 - Täitmisplaanide genereerimine ja igaühe maksumuse arvutamine.
 - Sisemise taseme operatsioonide valimine.
 - Madalaima maksumusega plaani valimine.

Loogilise täitmisplaani näide



Møisted

- ◆ **Konjunktsioon** – loogiline korrutamine – binaarne loogiline operatsioon:
 - Tähistuse näited: $A \& B$; $A \text{ AND } B$
- ◆ **Disjunktsioon** – loogiline liitmine – binaarne loogiline operatsioon:
 - Tähistuse näited: $A \vee B$; $A \text{ OR } B$

Tingimuse normaliseerimine

- ◆ Tingimuse viimine konjunktiivsele normaalkujule.
 - Antud valemiga loogiliselt samaväärne valem
 - Kujutab endast **elementaardisjunktsioonide konjunktsiooni**
- ◆ $A > B \text{ OR } (C = D \text{ AND } E < F) \Rightarrow (A > B \text{ OR } C = D) \text{ AND } (A > B \text{ OR } E < F)$

Tingimuse lihtsustamine

Loogiliselt samaväärne väljend	Näide päringu tingimuse liitumistaseest. Tingimus asendatakse loogiliselt samaväärse, kuid lihtsama tingimusega
$p \wedge p \equiv p$ $p \wedge (\text{false}) \equiv \text{false}$	aine_kood=>IDU3381' AND (aine_kood=>IDU3381 AND ainepunkte=2) false (sellise tingimuse ei saa vastata ükski)rijet
$p \wedge \text{eitus}(p) \equiv \text{false}$	Lause sisaldab vastutulu, kui lause tingimus ei rahulda ühtegi kirjet
$p \wedge (p \vee q) \equiv p$	aine_kood=>IDU3381' AND aine_kood=>IDU3381' \equiv false (ei leita ühtegi kirjet)
$p \vee p \equiv p$	aine_kood=>IDU3381' OR ainepunkte=2 OR ainepunkte=3 \equiv ainepunkte=2 OR ainepunkte=2 \equiv ainepunkte=2
$p \vee (\text{false}) \equiv p$	Oppe ainepunkte arvi ei saa samal ajal olla ühest väiksem ja 3-st suurem. Järjearve tingimuse saab liitustada, sest teine disjunkt on vastutoluine.
$p \vee \text{eitus}(p) \equiv \text{true}$	aine_kood=>IDU3381' OR (aine_kood=>IDU3381' AND ainepunkte=3) \equiv aine_kood=>IDU3381' aine_kood=>IDU3381' OR aine_kood=>IDU3381' true (sellise tingimuse vastavad kõik kirjed)
$p \vee (p \wedge q) \equiv p$	aine_kood=>IDU3381' OR (aine_kood=>IDU3381' AND ainepunkte=3) \equiv

Tingimus
teisendatakse
samaväärseks,
kuid
lihtsamaks
tingimuseks.

NB! Eeldusel,
et veerg
aine_kood on
NOT NULL!

Tingimuse lihtsustamine (2)

- ◆ `ainepunkte>2 AND ainepunkte>3` \equiv `ainepunkte>3`
- ◆ `ainepunkte>2 OR ainepunkte>3` \equiv `ainepunkte>2`
- ◆ `aine_kood LIKE 'IDU%' AND aine_kood LIKE 'I%'` \equiv `aine_kood LIKE 'IDU%'`
- ◆ `aine_kood LIKE 'IDU%' OR aine_kood LIKE 'I%'` \equiv `aine_kood LIKE 'I%'`
- ◆ `NOT (ainepunkte>20)` \equiv `ainepunkte<=20`
- ◆ `ainepunkte>2 AND 2<ainepunkte` \equiv `ainepunkte>2`
- ◆ `ainepunkte+2-2>2` \equiv `ainepunkte>2`

Andmebaasid II 2017 © Erki Eessaar

Tingimuse lihtsustamine (PostgreSQL)

```
scott=# EXPLAIN SELECT * FROM Emp WHERE sal=1000 AND sal=1000;
QUERY PLAN
Seq Scan on emp (cost=0.00..1.18 rows=1 width=32)
Filter: (sal = 1000::numeric)
(2 rows)

scott=# EXPLAIN SELECT * FROM Emp WHERE sal=1000 AND sal<>1000;
QUERY PLAN
Seq Scan on emp (cost=0.00..1.21 rows=1 width=32)
Filter: ((sal <> 1000::numeric) AND (sal = 1000::numeric))
(2 rows)

scott=# set constraint_exclusion=on;
SET
scott=# EXPLAIN SELECT * FROM Emp WHERE sal=1000 AND sal<>1000;
QUERY PLAN
Result (cost=0.00..0.01 rows=1 width=0)
One-Time Filter: false
(2 rows)
```

15.12.2017 Teema 7 19

Andmebaasid II 2017 © Erki Eessaar

Tingimuste lihtsustamisest

- Raja, K., 2017. *Päringu filtri predikaadi automaatne lihtsustamine kahe SQL-andmebaasisüsteemi näitel*. Bakalaureusetöö. TTÜ Tarkvarateaduse instituut. [WWW] <https://digi.lib.ttu.ee/i/?8925>
 - Uuring PostgreSQL 9.5 ja Oracle 12c Release 1 põhjal.
 - Kokku katsetati 8 erinevat päringu tüüpi ning 26 erinevat loogikaavaldise (filtri predikaadi) tüüpi.

15.12.2017 Teema 7 20

Andmebaasid II 2017 © Erki Eessaar

Tingimuste lihtsustamisest (2)

- 154-st katsetatud päringust suutis Oracle lihtsustada 88 ja PostgreSQL 46 lause filtri predikaate.
- PostgreSQLi *constraint_exclusion=ON* pole võluvits.
 - Iga päringu tüübi korral lisandus selle määramisel maksimaalselt üks kuni kaks tingimust, mida PostgreSQL oskas tänu sellele lihtsustada.

15.12.2017 Teema 7 21

Andmebaasid II 2017 © Erki Eessaar

Milliseid tingimusi osati/ei osatud lihtsustada?

A, B, C – loogika-avaldised; a – veerg

<ul style="list-style-type: none"> Kumbki ei osanud <ul style="list-style-type: none"> A AND NOT (A) => FALSE A OR NOT (A) => TRUE a+6-6>6 => a>6 A OR (NOT(A) AND B) => A OR B A AND NOT(B) OR B => A OR B (A OR B) AND (A OR NOT(B)) => A (A OR B) AND (A OR C) => A OR (A AND B) 	<ul style="list-style-type: none"> Mõlemad oskasid <ul style="list-style-type: none"> A AND A => A A OR A => A A OR (A AND B) => A NOT (a>6) => a<=6 NOT (NOT (a)) => a NOT(A OR B) => NOT(A) AND NOT(B) NOT(A AND B) => NOT(A) OR NOT(B) A AND B OR A AND C => A AND (A OR B)
---	--

15.12.2017 Teema 7 22

Andmebaasid II 2017 © Erki Eessaar

Tingimuste lihtsustamisest (3)

- Tänapäeva andmebaasisüsteemide optimeerimismoodulite üks suurimaid probleeme on, et need **hindavad operatsioonide tulemuseks olevat ridade hulka valesti (enamasti alahindavad)**.
- Tulemuseks on *mitteoptimaalsed täitmiskavad* ja lausete aeglane täitmine.

15.12.2017 Teema 7 23

Andmebaasid II 2017 © Erki Eessaar

Tingimuste lihtsustamisest (4)

- Lihtsustamata tingimused võivad ajada andmebaasisüsteemil "**pea sassi**" ja panustavad seega mitteoptimaalsete täitmiskavade koostamisse.
 - SELECT ... FROM K INNER JOIN P USING(k) INNER JOIN T USING (p) WHERE (piirang tabelile K) AND (piirang tabelile P) AND (piirang tabelile T);
 - Andmebaasisüsteem hindab tingimustele vastavate ridade hulka valesti ning valib ebasoodsa ühendamise algortimi ning ühendamise järjekorra.

15.12.2017 Teema 7 24

Semantiline teisendamine

- ♦ Lause *lihtsustamine* vastavalt andmebaasis deklareeritud kitsendustele.
 - Andmebaasisüsteem **saab eeldada** andmete kooskõla kitsendustega.
- ♦ Näide:
 - [Linn]-1-----0..*-[Osakond]
 - [Osakond] -1-----0..*-[Töötaja]
 - Leidke töötajad, kes töötavad linnas asuvas osakonnas => Leidke kõik töötajad.

15.12.2017

Teema 7

25

Semantiline teisendamine (2)

- ♦ Enne:
 - SELECT Töötaja.* FROM (Töötaja INNER JOIN Osakond USING (osakonna_nr)) INNER JOIN Linn USING (asukoht);
- ♦ Pärast:
 - SELECT * FROM Töötaja;
 - Kui poleks välisvõtmete deklaratsioone, siis võiks näiteks tabelis *Töötaja* olla osakonna numbreid, mida pole tabelis *Osakond*. Siis pole need kaks päringut enam loogiliselt samaväärsed.

15.12.2017

Teema 7

26

Semantiline teisendamine (3)

- ♦ Kui süsteem saab täita lihtsama lause, siis suure tõenäosusega täidetakse lause **kiiremini**.
- ♦ Seega kitsenduste deklareerimine aitab lisaks kõigele muule parandada jõudlust.
- ♦ Veel üks näide.
 - CHECK (**palk>0**)
 - Päringu SELECT * FROM Töötaja WHERE **palk<0**; tulemuseks saab süsteem automaatselt määrata FALSE (tulemuse tabel ei sisalda ühtegi rida).
 - Oracle (12.1), seda ei oska. PostgreSQLis tuleb oskus sisse lülitada.

15.12.2017

Teema 7

27

Semantiline teisendamine (4)

```
experiment_views=# EXPLAIN ANALYZE SELECT Count(*) AS cnt FROM health_care_visit
WHERE visit_fee<0;
```

QUERY PLAN

```
Finalize Aggregate (cost=18384.65..18384.66 rows=1 width=8) (actual
time=12232.588..12232.588 rows=1 loops=1)
-> Gather (cost=18384.44..18384.65 rows=2 width=8) (actual time=12232.510..12232.568
rows=3 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Partial Aggregate (cost=17384.44..17384.45 rows=1 width=8) (actual
time=12168.129..12168.129 rows=1 loops=3)
-> Parallel Seq Scan on health_care_visit (cost=0.00..17384.33 rows=41 width=0)
(actual time=12168.047..12168.047 rows=0 loops=3)
Filter: (visit_fee < 0)::numeric)
Rows Removed by Filter: 333333
```

```
Planning time: 77.520 ms
Execution time: 12236.776 ms
(6 rows)
experiment_views=# SET constraint_exclusion = on;
SET
experiment_views=# EXPLAIN ANALYZE SELECT Count(*) AS cnt FROM health_care_visit
WHERE visit_fee<0;
```

QUERY PLAN

```
Aggregate (cost=0.00..0.01 rows=1 width=8) (actual time=0.004..0.004 rows=1 loops=1)
-> Result (cost=0.00..0.00 rows=0 width=0) (actual time=0.001..0.001 rows=0 loops=1)
One-Time Filter: false
Planning time: 0.403 ms
Execution time: 0.041 ms
(5 rows)
```

- PostgreSQL
- Tabelis *health_care_visit* 1000000 rida.
- Jõustatud kitsendus: *visit_fee=0*
- Plaan koostamise aeg võib suureneeda.
- Täitmise aeg väheneb.

15.12.2017

Teema 7

28

Tabeli elimineerimine – sisuliselt semantiline teisendamine

- ♦ Mitmed andmebaasisüsteemid (näiteks Oracle 12c Enterprise Edition, PostgreSQL 10, MariaDB, MS SQL Server) rakendavad täitmisplaani koostamise/optimeerimise käigus **tabeli elimineerimise** teisendust.
- ♦ Kui päringus viidatakse mõnele tabelile T, mida tegelikult pole päringu täitmiseks vaja lugeda, siis andmebaasisüsteem koostab täitmisplaani, mille alusel tabelit T ei loeta.

15.12.2017

Teema 7

29

Tabeli elimineerimine – näide Oracles

- ♦ Tabelid:
 - [Dept]-0..1-----0..*-[Emp]
- ♦ Päring:
 - SELECT empno FROM Emp INNER JOIN Dept USING (deptno);
- ♦ Andmebaasisüsteem saab aru, et päringu täitmiseks piisab vaid tabeli *Emp* andmete lugemisest.

```
CRSUD10KCL > SELECT empno FROM Emp INNER JOIN Dept USING (deptno);
14 rows selected.

Execution Plan
-----
Plan hash value: 3956160932

| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time | | |
|---|---|---|---|---|---|---|---|---|
| 0 | SELECT STATEMENT | | 14 | 98 | 3 | (0) | 00:00:01 |
| 1 | 0 | TABLE ACCESS FULL | EMP | 14 | 98 | 3 | (0) | 00:00:01 |

Predicate Information (identified by operation id):
-----
1 - filter("EMP"."DEPTNO" IS NOT NULL)
```

Algne päringuga loogiliselt samaväärne päring (arvestades tabelitele deklareeritud kitsendustega): SELECT empno FROM Emp WHERE deptno IS NOT NULL;

15.12.2017

Teema 7

30

Andmebaasid II 2017 © Erki Eessaar

Sama näide PostgreSQLis

- PostgreSQL oskab tabeli elimineerimise teisendust, kuid selle päringu korral seda ei tee.
- Näide kuidas päringu kirjutaja saaks/peaks ise oskama päringut lihtsustada.

```

scott=# EXPLAIN SELECT empno FROM Emp INNER JOIN Dept USING(deptno);
               QUERY PLAN
-----
Hash Join  (cost=1.14..2.47 rows=14 width=2)
Hash Cond: (emp.deptno = dept.deptno)
-> Seq Scan on emp  (cost=0.00..1.14 rows=14 width=4)
-> Hash  (cost=1.06..1.06 rows=6 width=2)
    -> Seq Scan on dept  (cost=0.00..1.06 rows=6 width=2)
(5 rows)

```

15.12.2017 Teema 7 31

Andmebaasid II 2017 © Erki Eessaar

Tabeli elimineerimine – näide Oracles (2)

- Tabelid:
 - [Dept]-1-----0..*-[Emp]
- Päring:
 - SELECT empno FROM Emp INNER JOIN Dept USING (deptno);
- Andmebaasisüsteem saab aru, et päringu täitmiseks piisab vaid tabeli *Emp* indeksi andmete lugemisest.

ALTER TABLE Emp MODIFY (deptno NOT NULL);

Execution Plan
Plan hash value: 179099197

Id	Operation	Name	Rows	Bytes	Cost (KCP)	Time
0	SELECT STATEMENT		14	56	1	(0) 00:00:01
1	INDEX FULL SCAN	PK_EMP	14	56	1	(0) 00:00:01

Alge päringuga loogiliselt samaväärne päring (arvestades tabelitele deklareeritud kitsendustega):
SELECT empno FROM Emp;

15.12.2017 Teema 7 32

Andmebaasid II 2017 © Erki Eessaar

Tabeli elimineerimine (2)

- Rakendamise eelduseks on **välisvõtme kitsenduste andmebaasis deklareerimine** (FOREIGN KEY kitsendustega).
 - Kuna välisvõtme kitsendused peavad viitama kandidaatvõtmetele, siis on vajalikud ka PRIMARY KEY ja UNIQUE kitsendused.
 - Samuti annavad infot NOT NULL kitsendused.
- Välisvõtme kitsendused tagavad viidete terviklikkuse reegli kehtivuse.

15.12.2017 Teema 7 33

Andmebaasid II 2017 © Erki Eessaar

Tabeli elimineerimine (3)

- Välisvõtme kitsendus peab garanteeritud **kehtima kõigi** tabelis olevate andmete korral – kui kitsendus on osaliselt või täielikult **välja lülitatud**, siis pole teisendus võimalik.
 - Näiteks nii Oracles kui PostgreSQLis saab välisvõtme kitsenduse puhul määrata, et see peab kehtima lisatavate andmete korral, kuid mitte andmebaasis juba olevate andmete korral.

15.12.2017 Teema 7 34

Andmebaasid II 2017 © Erki Eessaar

Tabeli elimineerimine – näide Oracles (3)

Käivitan lause: ALTER TABLE Emp ENABLE NOVALIDATE CONSTRAINT fk_deptno; ja seejärel uuesti päringu

Execution Plan
Plan hash value: 3074306753

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	140	3 (0)	00:00:01
1	NESTED LOOPS		14	140	3 (0)	00:00:01
2	TABLE ACCESS FULL	EMP	14	98	3 (0)	00:00:01
3	INDEX UNIQUE SCAN	PK_DEPT	1	3	0 (0)	00:00:01

Predicate Information (identified by operation id):
3 - access ("EMP"."DEPTNO"="DEPT"."DEPTNO")

Tabelit *Dept* ei elimineerita – loetakse selle indeksit.

15.12.2017 Teema 7 35

Andmebaasid II 2017 © Erki Eessaar

Tabeli elimineerimine – erinevad andmebaasisüsteemid

Tabelite elimineerimise rakendamine

Andmebaasisüsteem	Ei toimu tabeli elimineerimist (Red)	Toimub tabeli elimineerimine (Blue)
Oracle	3	9
MS SQL	5	7
PostgreSQL	10	2

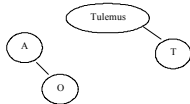
- Pukk, K., 2014. *Tabelite elimineerimise teisenduse rakendamine erinevate SQL-andmebaasisüsteemide poolt*. Bakalaureusetöö. TTÜ Informaatikainstituut.
- 12 lauset, kolm andmebaasisüsteemi
 - Oracle (12.1), MS SQL Server (2012) ja PostgreSQL (9.3).

Kõige rohkem rakendas Oracle, kõige vähem PostgreSQL.

15.12.2017 Teema 7 36

Vastuolude otsimine

- ♦ `SELECT T.nimi FROM Aine A, Oppimine O, Tudeng T WHERE A.aine_kood=O.aine_kood;`
 - Puudub ühendamisoperatsioon
- ♦ Kasutada relatsioonide seosegraafi.



Kontranäide

```
INSERT INTO Oppimine (matrikli_nr,
    aine_kood, regkuup, aktuaalne)
SELECT matrikli_nr, aine_kood, '2001-02-28', 1
FROM Tudeng, Aine
WHERE Tudeng.kursus= 3 AND Aine.nimetus
= 'Filosoofia 2';
```

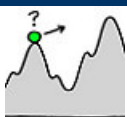
Ühendamisoperatsiooni puudumisest ei tulene automaatselt lause ebakorrektus!

Täitmisplaani optimeerimine

- ♦ Leida täitmisplaan, mis on mingis aspektis parim.
 - Kogu lause täitmiseks kulub võimalikult vähe aega.
 - Lause täitmise alustamiseks kulub võimalikult vähe aega.
 - Lause vahetulemuste salvestamiseks kulub kõige vähem salvestusruumi.
 - Ketta poole tuleb pöörduda võimalikult vähe kordi.
- ♦ Optimeerimine toimub nii loogilise kui ka füüsilise täitmisplaani koostamisel.

Optimeerija

- ♦ Täitmisplaani optimeerimisega tegeleb andmebaasisüsteem.
- ♦ Seda tegevust võib assisteerida inimkasutaja, kes kirjutab "optimaalse" süntaksiga lauseid ja suunab andmebaasisüsteemi tööd.
 - Ideaalses andmebaasisüsteemis poleks seda vaja.



Optimeerija (2)

- ♦ Tulemuseks ei pruugi olla *parim võimalik* lahendus (globaalne optimum), vaid *piisavalt hea lahendus*, mida otsimiseks tehtud pingutuse tulemusena leiti (lokaalne optimum).

Täitmisplaani optimeerimise strateegiad

- ♦ **Staatiline optimeerimine** – viiakse läbi üks kord.
 - *Eelised*: Lause täitmiseks kuluv aeg väheneb plaani koostamise võrra.
 - *Puudused*: Värskeimat andmebaasi statistikat ei võeta arvesse.
- ♦ **Dünaamiline optimeerimine** – iga kord kui lause käivitatakse.
 - *Eelised*: Saab arvesse võtta värsket andmebaasi statistika.
 - *Puudused*: Lause täitmiseks kuluv aeg pikeneb plaani koostamise võrra.

Täitmisplaani optimeerimise strateegiad (2)

♦ Hübriidne optimeerimine.

- Üldjuhul kasutatakse salvestatud täitmisplaani.
- Uus plaan näiteks kui
 - lauses viidatud tabelite ridade arv muutub oluliselt,
 - väärtuste jaotus lauses viidatud veergudes muutub oluliselt,
 - muutuvad andmebaasisüsteemi juhtparameetrite väärtused, mis mõjutavad lause täitmist.

15.12.2017

Teema 7

43



Täitmisplaani optimeerimise strateegiad (3)

♦ Adaptiivne optimeerimine.

- Täitmisplaani koostamine ja täitmine *sulavad kokku*.
- Kuna andmeid on palju, neid tuleb nagu kosest juurde ja andmed võivad paikneda hajutatult, siis ei saa olla kindel, et plaan millega täitmist alustati on parim plaan, millega lõpetada.
- Andmebaasisüsteem peab oskama *käigult* plaani muuta.

15.12.2017

Teema 7

44

Optimaalse tegevuste järjekorra valimine – näite algandmed

- ♦ `SELECT * FROM Tootaja t, Osakond o WHERE t.osakonna_nr=o.osakonna_nr AND (t.ametikoht='Juhataja' AND o.linn='London');`
- Tabelis *Tootaja* on **1000** rida.
- Tabelis *Osakond* on **50** rida.
- Tabelis *Tootaja* on **50** juhataja andmed (üks iga osakonna kohta).
- Tabelis *Osakond* on 5 osakonna andmed, mis asuvad Londonis.

15.12.2017

Teema 7

45

Loogiline täitmisplaan 1

$\sigma(\text{ametikoht}='Juhataja' \wedge (\text{linn}='London'))$
 (Tootaja.osakonna_nr=Osakond.osakonna_nr)(Tootaja x Osakond)

Variant (1) alamtegevused:

Jrk nr	Kirjeldus	Ketta poole tehtavate pöörduste arv	Tulemuseks saadud tabelis olevate ridade arv
1	Otsikorutise loomine <i>Tootaja</i> ja <i>Osakond</i> põhjal.	$1000 + 50 = 1050$	$1000 * 50 = 50\ 000$
2	Otsikorutise tulemus kirjutamine kettale.	$1000 * 50 = 50\ 000$	
3	Tingimusele vastavate ridade otsimine tegevuse nr 2 tulemusest.	$1000 * 50 = 50\ 000$	5
KOKKU		101 050	

15.12.2017

Teema 7

46

Loogiline täitmisplaan 2

$\sigma(\text{ametikoht}='Juhataja' \wedge (\text{linn}='London'))$
 (Tootaja (Tootaja.osakonna_nr=Osakond.osakonna_nr) Osakond)

Variant (2) alamtegevused

Jrk nr	Kirjeldus	Ketta poole tehtavate pöörduste arv	Tulemuseks saadud tabelis olevate ridade arv
1	Equijoini loomine <i>Tootaja</i> ja <i>Osakond</i> põhjal.	$1000 + 50 = 1050$	1000
2	Uhendamise tulemus kirjutamine kettale.	1000	
3	Tingimusele vastavate ridade otsimine tegevuse nr 2 tulemusest.	1000	5
KOKKU		3050	

15.12.2017

Teema 7

47

Loogiline täitmisplaan 3

$\sigma(\text{ametikoht}='Juhataja' \wedge (\text{Tootaja}))$
 (Tootaja.osakonna_nr=Osakond.osakonna_nr)
 $\sigma(\text{linn}='London') \wedge (\text{Osakond})$

Variant (3) alamtegevused

Jrk nr	Kirjeldus	Ketta poole tehtavate pöörduste arv	Tulemuseks saadud tabelis olevate ridade arv
1	Tingimusele: $\text{ametikoht}='Juhataja' \wedge \text{vastavate ridade otsimine tabelist } Tootaja$	1000	50
2	Otsingu tulemuse kirjutamine kettale.	50	
3	Tingimusele: $\text{linn}='London' \wedge \text{vastavate ridade otsimine tabelist } Osakond$	50	5
4	Otsingu tulemuse kirjutamine kettale.	5	
5	Tegevuste 2 ja 4 tulemusel saadud tabelite ühendamine.	$50 + 5$	5
KOKKU		1160	

15.12.2017

Teema 7

48

Optimaalse tegevuste järjekorra valimine

- ♦ **Heuristilise algoritmi** põhiooneks on asjaolu, et nende korral pole tavaliselt midagi teoreetiliselt tõestatud.
- ♦ Heuristiline algoritm põhineb harilikult mingil pealtnäha kavalal ja heal võttel (võtetel) ning annab vastuse suhteliselt kiiresti, kuid võib ka mõnikord oma otsuses eksida.

Heuristilised lause töötlemise strateegiad

- ♦ **Unaarsed operatsioonid** kõigepealt.
 - *Piirangu* operatsioonid nii varakult kui võimalik.
 - Kõige piiravamad piirangu operatsioonid täita kõige varem.
 - *Projektsiooni* operatsioonid nii varakult kui võimalik.
- ♦ Kui lause täitmiseks tuleb kõigepealt läbi viia *otsekorrutise* leidmise operatsioon ning seejärel ühendamiseks vajalik *piirangu* operatsioon, siis tuleb see asendada *ühendamise* operatsiooniga.



Heuristilise lause töötlemise strateegiad

- ♦ $(A \text{ JOIN } B) \text{ WHERE kitsendus } A\text{-le AND kitsendus } B\text{-le} \equiv (A \text{ WHERE kitsendus } A\text{-le}) \text{ JOIN } (B \text{ WHERE kitsendus } B\text{-le})$
- ♦ Kasuta **laiska hindamist** – hoidu sama vahetulemuse mitmekordsest arvutamisest ja hoiu seda mälus.
 - Mittekorreleeruv alampäring.
 - Deterministlik funktsioon.

Funktsioonidest

- ♦ **Puhas funktsioon**
 - a) Tulemus sõltub **ainult argumentidest**
 - Samade argumentidega alati sama tulemus
 - b) Funktsioonil **pole kõrvaleffekte**
 - Ei muuda väliste muutujate väärtuseid (nt andmeid tabelites), ei väljasta infot väljundisse (nt RAISE käsuga)
- **Deterministlik funktsioon**
 - Peab rahuldama ainult tingimust a)

Funktsioonidest (2)

- ♦ PostgreSQL ja Oracle võimaldavad märkida kasutaja-definieeritud funktsioone deterministlikeks.
- ♦ See ei muuda funktsiooni käitumist, vaid annab andmebaasisüsteemile **lisainfot**.
- ♦ Eeldatakse, et sellise tähistusega funktsioon on *puhas funktsioon*.

Kontranäide

```
SELECT T.perenimi, O.* FROM Tudeng T INNER
JOIN Oppimine O ON T.matrikli_nr=O.matrikli_nr
WHERE f_keskmise_hinne(T.matrikli_nr) >3.0
AND O.aine_kood LIKE 'IDU%';
```

- ♦ **f_keskmise_hinne(varchar)** – keskmise hinde arvutamise funktsioon.
 - Ei ole deterministlik.
 - Seda võiks käivitada võimalikult väike arv kordi, sest see peab lugema palju andmeid.

Kontranäide – parim tegevuste järjekord

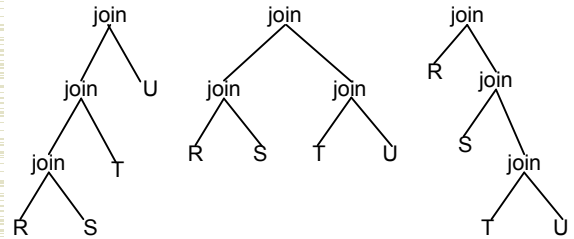
- ♦ Leia IDU ainete õppimised.
- ♦ Ühenda tulemus tabeliga *Tudeng*, leidmaks neid õppinud üliõpilaste matrikli numbrid.
- ♦ Piira tulemust üliõpilase keskmise hinde alusel.

15.12.2017

Teema 7

55

Ühendamise järjekorra valimine – näiteid



15.12.2017

Teema 7

56

Ühendamise järjekorra valimine (2)

- ♦ Vaadata läbi kõik puud (võib võtta liiga palju aega).
- ♦ Vaadata läbi alamosa puudest (kuidas valida?).
- ♦ Kasutada heuristikat.
 - Enne ühendamist rakenda lauses sisalduvat tulemust piiravad tingimused.
 - Valida järjekord selliselt, et ühendamise operatsioonides osaleks võimalikult vähe ridu.

15.12.2017

Teema 7

57

Tegevuste järjekorra muutmine – vaade

```
CREATE VIEW Idu_ained
AS SELECT aine_kood, nimetus, ainepunkte
FROM Aine WHERE aine_kood LIKE 'IDU%'
WITH CHECK OPTION;
```

15.12.2017

Teema 7

58

Tegevuste järjekorra muutmine – päring

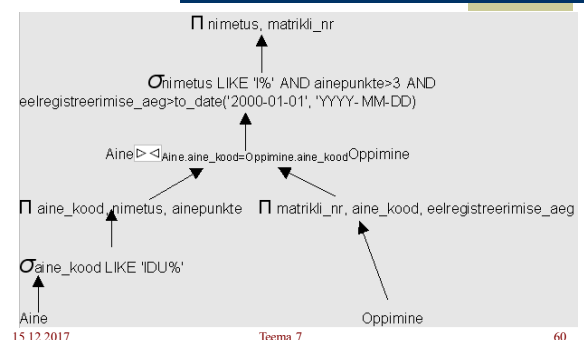
```
SELECT Idu_ained.nimetus,
       Oppimine.matrikli_nr
FROM Idu_ained INNER JOIN Oppimine ON
     Idu_ained.aine_kood=Oppimine.aine_kood
WHERE Idu_ained.nimetus LIKE 'T%' AND
       Idu_ained.ainepunkte>3 AND
       Oppimine.eelregistreerimise_aeg>
       to_date('2000-01-01', 'YYYY-MM-DD');
```

15.12.2017

Teema 7

59

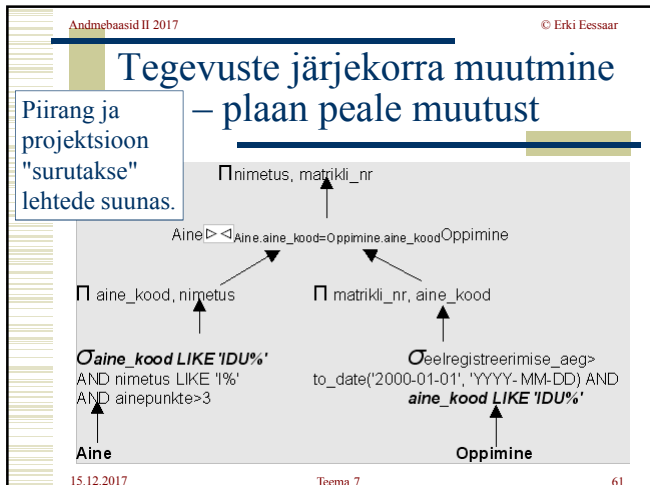
Tegevuste järjekorra muutmine – plaan enne muutust



15.12.2017

Teema 7

60



Andmebaasid II 2017 © Erki Eessaar

Füüsilise täitmisplaani koostamine ja optimeerimine

- Andmebaasisüsteem loob selle loogilise täitmisplaani põhjal.
- Relatsioonialgebra operatsiooni täitmiseks võib andmebaasisüsteem kasutada erinevaid sisemise taseme operatsioone.
- Tuleb valida *optimaalseim* sisemise taseme operatsioonide komplekt.

15.12.2017 Teema 7 62

Andmebaasid II 2017 © Erki Eessaar

Füüsilise täitmisplaani näide (PostgreSQL)

QUERY PLAN

Nested Loop (cost=0.00..0.01 rows=1 width=193)
 Join Filter: ("outer".aine_seisund_id = "inner".aine_seisund_id)
 -> **Seq Scan** on aine a (cost=0.00..0.00 rows=1 width=29)
 Filter: (aine_kood = 'IDU3381'::bpchar)
 -> **Seq Scan** on aine_seisund a1s (cost=0.00..0.00 rows=1 width=172)
 Filter: ((nimetus)::text = 'kinnitatud'::text)

15.12.2017 Teema 7 63

Andmebaasid II 2017 © Erki Eessaar

Füüsilise täitmisplaani koostamine ja optimeerimine (2)

- Sama lause täitmine
- samade andmete põhjal,
- kuid erinevate füüsiliste täitmisplaanide alusel,
- annab leitud / muudetud ridade osas alati sama tulemuse,
- kuid lause täitmiseks kuluv aeg võib sõltuvalt kasutatud plaanist olla erinev.

15.12.2017 Teema 7 64

Andmebaasid II 2017 © Erki Eessaar

Sama ülesande erinevad lahendused – sama täitmisplaan

scott=# EXPLAIN SELECT DISTINCT deptno FROM Emp;
 QUERY PLAN
 HashAggregate (cost=1.20..1.36 rows=16 width=2)
 Group Key: deptno
 -> Seq Scan on emp (cost=0.00..1.16 rows=16 width=2)
 (3 rows)

scott=# EXPLAIN SELECT deptno FROM Emp GROUP BY deptno;
 QUERY PLAN
 HashAggregate (cost=1.20..1.36 rows=16 width=2)
 Group Key: deptno
 -> Seq Scan on emp (cost=0.00..1.16 rows=16 width=2)
 (3 rows)

PostgreSQL 10 füüsiline täitmisplaan

Antud juhul saab andmebaasisüsteem aru, et need kaks lauset on samaväärsed ja koostas neile ühesuguse täitmisplaani. See ei pruugi alati nii olla – sama SQL ülesande erinevatele lahendustele võib andmebaasisüsteem pakkuda erinevaid täitmisplaanid ja nende lausete täitmise kiirus on seega erinev.

15.12.2017 Teema 7 65

Andmebaasid II 2017 © Erki Eessaar

Sama ülesande erinevad lahendused – erinevad täitmisplaanid

scott=# EXPLAIN ANALYZE SELECT Count(DISTINCT empno) AS cnt FROM Emp;
 QUERY PLAN
 Aggregate (cost=1.20..1.21 rows=1 width=8) (actual time=0.221..0.221 rows=1 loops=1)
 -> Seq Scan on emp (cost=0.00..1.16 rows=16 width=2) (actual time=0.084..0.089 rows=16 loops=1)
 Planning time: 0.762 ms
 Execution time: 0.365 ms
 (4 rows)

scott=# EXPLAIN ANALYZE SELECT Count(*) AS cnt FROM (SELECT DISTINCT empno FROM Emp) AS foo;
 QUERY PLAN
 Aggregate (cost=1.56..1.57 rows=1 width=8) (actual time=0.054..0.054 rows=1 loops=1)
 -> HashAggregate (cost=1.20..1.36 rows=16 width=2) (actual time=0.042..0.047 rows=16 loops=1)
 Group Key: emp, empno
 -> Seq Scan on emp (cost=0.00..1.16 rows=16 width=2) (actual time=0.018..0.023 rows=16 loops=1)
 Planning time: 0.223 ms
 Execution time: 0.126 ms
 (6 rows)

PostgreSQL 10 füüsiline täitmisplaan

15.12.2017 Teema 7 66

Andmebaasid II 2017 © Erki Eessaar

Kuidas valida parim täitmisplaan?

LOW COST

- Valida täitmisplaan, millel on kõige madalam **maksumus**.
- Üks *võimalik* maksumuse kriteerium: ketta poole pöörduste arv.
 - Mida väiksem on lause täitmispalani maksumus, seda vähem tuleb selle järgi lugeda kettal olevaid plokkide.
- Maksumuse arvutamiseks on vaja **statistikat** tabelites ja indeksites olevate andmete kohta.
- Statistikat peab regulaarselt **värskendama**.

15.12.2017 Teema 7 67

Andmebaasid II 2017 © Erki Eessaar

Hinnangud operatsioonide tagastatavate ridade arvule

- Selleks, et valida parimad sisemise taseme operatsioonid ja nende kõige parem järjekord, hindab andmebaasisüsteem *statistikat* appi võttes, milline on **iga operatsiooni tulemuseks olev ridade arv** (*cardinality*).

```

scott=# EXPLAIN SELECT DISTINCT deptno FROM Emp;
               QUERY PLAN
-----
HashAggregate  (cost=1.18..1.21 rows=3 width=2)
  Group Key: deptno
    -> Seq Scan on emp  (cost=0.00..1.14 rows=14 width=2)
        (3 rows)
  
```

15.12.2017 Teema 7 68

Andmebaasid II 2017 © Erki Eessaar

Tabeli statistika näiteid

- nTuples(R)** – ridade arv tabelis R.
- bFactor(R)** – tabeli R ridade arv, mis mahub ühte plokkide.
- nBlocks(R)** – tabeli R salvestamiseks kuluv plokkide arv.

15.12.2017 Teema 7 69

Andmebaasid II 2017 © Erki Eessaar

Veeru statistika näiteid

- nDistinct_A(R)** – tabeli R veerus A olevate erinevate väärtuste arv.
- min_A(R), max_A(R)** – tabeli R veerus A olev minimaalne ja maksimaalne väärtus.
- SC_A(R)** – tabeli R veeru keskmine ridade arv, mis rahuldab päringus tingimust A=x, kus A on veerg tabelis R ja x on päringus kasutatav väärtus.

15.12.2017 Teema 7 70

Andmebaasid II 2017 © Erki Eessaar

Indeksi statistika näiteid

- nLevels_A(I)** – indeksile I vastava indeksipuu tasemete arv.
- nLfBlocks_A(I)** – indeksile I vastavas indeksipuu olevate lehtede arv.

```

      1
     / \
    2   3   4
  
```

nLevels_A(I) = 2
nLfBlocks_A(I) = 3

15.12.2017 Teema 7 71

Andmebaasid II 2017 © Erki Eessaar

Oracle 11g Release 2 vaikimisi statistilised väärtused – näiteid

- nBlocks(R)=100**
- nLevels_A(I)=1**
- nLfBlocks_A(I)=25**

15.12.2017 Teema 7 72

Täitmisplaanide maksumuste arvutamise näide

- Algandmed:
 - Tabel T – $n\text{Blocks}(R)=200$
 - Unikaalne indeks tabeli T veerul $v1$ – $n\text{Levels}_A(I)=4$
- SELECT * FROM T WHERE $v1=:v\text{äärtus}$;
 - Indeksil põhinev otsing.
 - Peale kõigi indeksi tasemete läbimist tuleb lugeda tabeliplokk, mis sisaldab otsitavat rida.
 - Maksumus**= $n\text{Levels}_A(I)+1=5$
 - Kui päringule saab vastata ainult indeksi põhjal, siis maksumus on: $n\text{Levels}_A(I)$
 - Tabeli läbiskaneerimisel põhinev otsing.
 - Maksumus**= $n\text{Blocks}(R) = 200$ (Tuleb lugeda kõiki plokkide)

Meetodid relatsioonialgebra operatsioonide läbiviimiseks

- Sorteerimisel põhinevad meetodid.
- Räsiväärtuse kasutamisel põhinevad meetodid.
- Indeksi kasutamisel põhinevad meetodid.

Meetodid relatsioonialgebra operatsioonide läbiviimiseks (2)

- Keerukuse põhjal võib meetodeid jagada.
 - Andmete *ühekordsel* lugemisel põhinevad meetodid.
 - Andmete *kahekordsel* lugemisel põhinevad meetodid.
 - Andmete *kolme- või rohkema kordsel* lugemisel põhinevad meetodid.

Kuidas anda sisend relatsioonialgebra operatsioonile?

- Relatsioonialgebra operatsiooni tulemus saab olla **sisendiks** järgnevale operatsioonile.
- Võimaldab kirjutada keerukaid **avaldisi**.
- Kuidas anda operatsioonile sisend?
 - Moodustada **täielikult** sisendiks olevad relatsioonid ja anda need korraga sisendiks.
 - Vajadusel võib süsteem neid relatsioone (vahetulemusi) kettale kirjutada e **materialiseerida** (aeganõudev!).



Konveiertöö (pipelining)

- Eelmiste operatsioonide tulemuseks olevad read suunatakse **jooksvalt** järgmise operatsiooni sisendisse.
- Kui ei ole tegemist kokkuvõttefunktsiooni (nt *Count*) väärtuse arvutamisega, saab järgmine operatsioon hakata jooksvalt tulemusi väljastama.

Otsuseid, kuidas anda sisend, teeb andmebaasisüsteem!
See ei puuduta relatsioonilise avaldise kirjutajat!

Piirang

- SELECT * FROM Aine WHERE **aine_kood**='IDU3381';
- Alternatiivid.
 - Ainult tabeliplokkide** lugemine:
 - read pole otsingus kasutatud veeru (*aine_kood*) järgi füüsilisel tasemel sorteeritud – kõigi tabeli kasutuses olevate plokkide lugemine ja kõigi ridade läbivaatamine (tabeli täielik läbiskaneerimine),
 - read on füüsilisel tasemel sorteeritud – osade plokkide lugemine (kahendotsing).
 - Indeksi** kasutamine, et leida üles vajalikke ridu sisaldavad **tabeliplokkid**.
 - Ainult indeksi** kasutamine, et päringule vastata.

Tabeli täielik läbiskaneerimine (Oracle)

- Loetakse kõiki tabeli plokkide kuni **kõrgveemärgini**.
- Korraka loetakse mitu plokki – määrab parameetri **DB_FILE_MULTIBLOCK_READ_COUNT** väärtus.
- Loetud ploki kustutatakse vaikumisi kiiresti mälust
 - pannakse andmepuhvri *Least Recently Used* piirkonda.
- Järgnevate käskudega saab kõrgveemärki **alandada** ja sellega vähendada loetavate plokkide arvu:
 - ALTER TABLE ... ENABLE ROW MOVEMENT;
 - ALTER TABLE ... SHRINK SPACE;

15.12.2017

Teema 7

79

Millal tabel täielikult läbiskaneerida? (Oracle kirjandus)

- Indeks ei kata kõiki soovitud veerge (st ei piisa ainult indeksi lugemisest) ja päring toob välja suure hulga tabelis olevaid ridu (erinevatel andmetel vähemalt 20%, 35% või 50% ridadest).
 - Indeksi kasutamisel tuleb tõenäoliselt lugeda ühte ja sama tabeliplokki **korduvalt** ning kokkuvõttes **kasvab plokkide lugemiste arv** ja väheneb töökiirus.

15.12.2017

Teema 7

80

Millal tabel täielikult läbiskaneerida? (2)

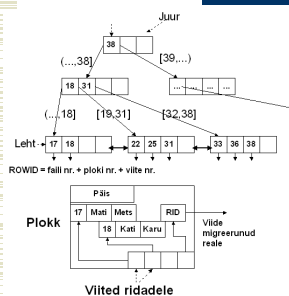
- Tabeli ridade arv on suhteliselt **väike**.
 - Indeksi kasutamine ei vähenda loetavate plokkide hulka.
- Päringu tingimuses kasutatud veerul on B-puu indeks, kuid päringu tingimuses kasutatud väärtustel on **madal selektiivsus** – st paljudes ridades on sellised väärtused.

15.12.2017

Teema 7

81

Indeksil põhineva otsingu meetodid Oracles



Index unique scan:

- SELECT * FROM Isik WHERE isik_id=22;
- Loetakse üks plokk igalt indeksi tasemelt.

Index-range scan:

- SELECT * FROM Isik WHERE isik_id BETWEEN 22 AND 33;

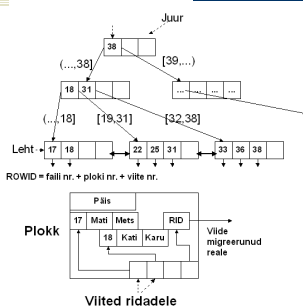
B-puu indeks

15.12.2017

Teema 7

82

Indeksil põhineva otsingu meetodid Oracles (2)



Index full scan:

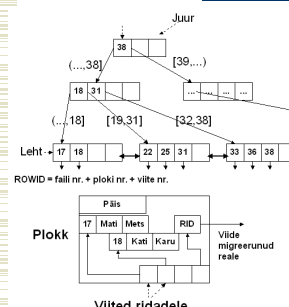
- SELECT isik_id FROM Isik ORDER BY isik_id;
- Indeksipuu *lehtede*ks olevate plokkide *ükshaaval* lugemine.
- Lisaks loetakse ploki, mida läheb vaja esimese leheni jõudmiseks.

15.12.2017

Teema 7

83

Indeksil põhineva otsingu meetodid Oracles (3)



Index fast-full scan:

- SELECT Count(isik_id) arv FROM Isik;
- Kõigi indeksi plokkide lugemine (ühe I/O operatsiooniga võib lugeda mitu plokki).

15.12.2017

Teema 7

84

Index-skip scan

- ◆ Olemas tabel: t(a, b, c, d, e);
- ◆ CREATE INDEX t_idx ON t(a, b, c);
- ◆ Päringud, mis tavaliselt kasutaksid seda indeksit:
 - SELECT * FROM t WHERE a=:a;
 - SELECT * FROM t WHERE a=:a AND b=:b;
 - SELECT * FROM t WHERE a=:a AND b=:b AND c=:c;
- ◆ Päring, mille täitmiseks kasutatakse seda indeksit tänu *skip-scanile*:
 - SELECT * FROM t WHERE b=:b;

15.12.2017

Teema 7

85

Index-skip scan (2)

- ◆ Eeldame, et veerus *a* on kolm võimalikku väärtust (1, 2 ja 3). Sellisel juhul täidab andmebaasisüsteem kontseptuaalselt tegelikult lause:

```
SELECT a, b, c, d, e FROM t WHERE b=:b AND a=1
UNION
SELECT a, b, c, d, e FROM t WHERE b=:b AND a=2
UNION
SELECT a, b, c, d, e FROM t WHERE b=:b AND a=3;
```

Parim kasutada, kui liitindeksi esimeses veerus vähe erinevaid väärtuseid.

15.12.2017

Teema 7

86

Index-join

Indeksid:

- ◆ idx1(a); idx2(b,c)

```
SELECT a, b, c FROM t WHERE a=:a AND b=:b;
```

- ◆ Süsteem saab päringule vastata **ainult indeksite põhjal**, ilma tabeli plokkide lugemata, ühendades kokku *idx1* ja *idx2* kasutades *hash join* algoritmi.

15.12.2017

Teema 7

87

Ainult indeksil põhinev otsing

- ◆ Päringule saab vastata ainult indeksi põhjal ilma tabeli poole pöördumata.
 - Analoogiline näide raamatu lugemisest – tee indeksi põhjal kindlaks, kas raamatus räägitakse viiendast normaalkujust või mitte.
- ◆ Oracle lisavõimalus: *Index Organized Table* (IOT) – tabel ongi indeks.

15.12.2017

Teema 7

88

Ühendamine

- ◆ SELECT Tudeng.perenimi, Oppimine.*
FROM Tudeng INNER JOIN Oppimine ON
Tudeng.matrikli_nr=Oppimine.matrikli_nr;

15.12.2017

Teema 7

89

Sisemise taseme algoritmid ühendamise läbiviimiseks

- ◆ *Nested loop join*
 - Erijuhuseks *cluster join* Oracles. Kasutatakse kui ühendatavad tabelid on koondatud klastrisse.
- ◆ *Hash join*
- ◆ *Merge join (sort-merge join)*
- ◆ http://viktor.ld.ttu.ee/animatsioonid/animation_join_algorithms/

15.12.2017

Teema 7

90

Brute force nested loop join

```
for (each row in Tabel1) { /*Väline tsükkel*/
  for (each row in Tabel2) { /*Sisemine tsükkel*/
    if (Tabel1 join column matches Tabel2 join column)
      then
        pass
      else
        fail
    end if;
  }
}
```

Brute force nested loop join efektiivsus

♦ Lähteandmed:

- Tabel1 – 100 rida
- Tabel1 – plokis 1 rida – 100 plokki
- Tabel2 – 10000 rida
- Tabel2 – plokis 10 rida – 1000 plokki

♦ Plokkide lugemise arv:

- Tabel1 – väline tabel, Tabel2 – sisemine tabel
 - $100/1 + 100 \cdot (10\,000/10) = \mathbf{100\,100}$
- Tabel2 – väline tabel, Tabel1 – sisemine tabel
 - $10000/10 + 10000 \cdot (100/1) = \mathbf{1\,001\,000}$

Järeldus

- ♦ **Välimiseks tabeliks** tuleb valida tabel, kus peale WHERE klauslis olevate tingimuste rakendamist on **vähem** plokkide kui sisemises.
- ♦ Töökiirus oleks *palju parem*, kui sisemise tabeli ühendamises osaleval veerul oleks **indeks**.
 - Põhjus, miks indekseerida välisvõtit!

Nested loop join based on index lookup

```
for (each row in Tabel1) { /*Väline tsükkel*/
  lookup value in the index in table Table 2 {
    if (found) then
      pass /*Oleme leidnud kokkulangeva väärtuste paari*/
    else
      fail
    end if;
  }
}
```

Väike kõrvalepõige

- ♦ Teine põhjus, miks indekseerida välisvõtit:
 - [Parent]-1-----0...*-[Child]
 - Kui rea kustutamisel Parent tabelist tuleb kaskaadselt kustutada read Child tabelis, siis ilma indeksita peaks andmebaasisüsteem läbi vaatama kõik read Child tabelis (*full table scan*).
 - Kui välisvõti oleks indekseeritud, siis saab kustutavate ridade otsimiseks kasutada indeksit.
 - Sama kehtib ka Parent tabeli primaarvõtme muutuse kaskaadsel ülekandmisel Child tabelisse.

Millal kasutada nested loop joini?

- ♦ Välise tabeli ühendatavate ridade hulk on palju väiksem võrreldes sisemise tabeli omaga.
 - Välises tabelis ei pruugi olla vähem ridu kui sisemises.
 - Välisest tabelist pärit ridade hulk peab *peale piiravate tingimuste rakendamist* olema väiksem, kui sisemisest tabelist pärit ridade hulk peale piiravate tingimuste rakendamist.
- ♦ Kasutatakse palju tingimusi, mis piiravad ühendatavate ridade arvu.
- ♦ Ühendatavad tabelid on väikesed.
- ♦ *Equijoin* või *non-equijoin*.

Andmebaasid II 2017 © Erki Eessaar

Merge join

```

sort(Tabel1)
sort(Tabel2)
} Ettevalmistav faas
get first row(Tabel1)
get first row(Tabel2)
for (until no more rows in tables) {
  if (join-column in Tabel1 < join-column in Tabel2) then
    get next row (Tabel1)
  elseif (join-column in Tabel1 > join-column in Tabel2)
    then
    get next row (Tabel2)
  elseif (join-column in Tabel1 = join-column in Tabel2)
    then
    pass
    get next row (Tabel1)
    get next row (Tabel2)}

```

Table 1

b
d
a
c

Table 2

f
a
d
g
b

Samm 1

Table 1

a
b
c
d

Table 2

a
b
d
f
g

Samm 2

a	a	pass(a)
b	b	pass(b)
c	d	
d	d	pass(d)
f		
g		

15.12.2017 Teema 7 97

Andmebaasid II 2017 © Erki Eessaar

Merge join (2)

- ♦ Toimub nii *Tabel1* kui ka *Tabel2* täielik läbiskaneerimine (*full table scan*).
- ♦ Nii *Tabel1* kui ka *Tabel2* tuleb sorteerida.
 - Töökiirus kannatab, kui kogu tabel ei mahu sorteerimiseks mõeldud mälupiirkonda.

15.12.2017 Teema 7 98

Andmebaasid II 2017 © Erki Eessaar

Millal kasutada *merge joini*?

- ♦ Välise tabeli ja sisemise tabeli ühendatavate ridade hulk palju ei erine.
- ♦ Lisaks ühendamisele on päringus ka mõni ridade sorteerimist nõudev operatsioon (*DISTINCT*, *GROUP BY*, *ORDER BY*).
- ♦ Andmed on juba andmefaili tasemel sorteeritud.
- ♦ *Equijoin* või *non-equijoin*.

15.12.2017 Teema 7 99

Andmebaasid II 2017 © Erki Eessaar

Hash join

- ♦ `SELECT A.aine_kood, A.nimi, O.eelregistreerimise_aeg, O.matrikli_nr FROM Aine A NATURAL JOIN Oppimine;`

15.12.2017 Teema 7 100

Andmebaasid II 2017 © Erki Eessaar

Hash join (2)

Räsiväärtused

1. Tabeli *Aine* (väiksem ja sisemine tabel) põhjal mällu räsitabel
2. Tabeli *Oppimine* (suurem ja väline tabel) ridadele vaste otsimine

asdr1231
asdr1233
asdr1431
asdf1231

asdf3423
asdg1232
asdi2331
asdj2431

aseg2123
asch1222
axe13331
aysj3331

asdi2331

TAF3151 → Räsifunktsioon

väärtus ühendamise tingimuses olevast veerust

15.12.2017 Teema 7 101

Andmebaasid II 2017 © Erki Eessaar

Millal kasutada *hash joini*?

- ♦ Välise tabeli ja sisemise tabeli ühendatavate ridade hulk palju ei erine.
- ♦ Sisemine tabel mahub korraga muutmällu (*hash area*).
- ♦ Ainult *equijoin*.
- ♦ Sisemise tabeli ühendamises osalevas veerus on väärtused ühtlaselt jaotunud
- ♦ Sisemise tabeli ühendamises osalev veerg sisaldab unikaalseid väärtuseid

15.12.2017 Teema 7 102

Andmebaasid II 2017 © Erki Eessaar

Võrdlus

- Vaja teha **eeltööd**:
 - Merge join, hash join
- Tulemuse **esimeste ridade**ni jõudmine on kiirem:
 - Nested loop join

15.12.2017 Teema 7 103

Andmebaasid II 2017 © Erki Eessaar

Näide (PostgreSQL)

15.12.2017 Teema 7 104

Andmebaasid II 2017 © Erki Eessaar

Näide (PostgreSQL) (2)

```
1 SELECT * FROM Emp INNER JOIN Dept USING (deptno) ORDER BY deptno;
```

Merge join, kuna tulemus on vaja sorteerida ühendamises osaleva veeru alusel.

15.12.2017 Teema 7 105

Andmebaasid II 2017 © Erki Eessaar

Näide (PostgreSQL) (3)

- Väikesed tabelid
- Üsna sarnase suurusega
- Equijoin
- Dept põhjal loodud räsitabel mahub mällu

Ühendamiseks hash join algoritm. Dept on sisemine tabel.

15.12.2017 Teema 7 106

Andmebaasid II 2017 © Erki Eessaar

Outer join (välisühendamine)

```
SELECT Tudeng.*, Oppimine.aine_kood,
       Oppimine.tulemus
FROM Tudeng LEFT JOIN Oppimine ON
       Tudeng.matrikli_nr=Oppimine.matrikli_nr;
```

15.12.2017 Teema 7 107

Andmebaasid II 2017 © Erki Eessaar

Outer join (2)

```
for x in (select * from tudeng)
loop
    found_record=false
    for y in (select * from oppimine where
              oppimine.matrikli_nr=x.matrikli_nr)
    loop
        found_record=true
        väljasta ühendatud rida
    end loop
    if (not found_record) then
        väljasta rida ikkagi koos NULLidega tabelist Oppimine pärit väljades
    end if
end loop
```

Kasutage päringus välisühendamise operaatorit vaid siis, kui seda on sisuliselt vaja. Vastasel juhul sunniti süsteemi tegema üleliigset tööd, mis lõpptulemuse saavutamiseks pole sisuliselt vajalik.

15.12.2017 Teema 7 108

Andmebaasid II 2017 © Erki Eessaar

Alampäring

```
SELECT * FROM Tabel1 T1
WHERE T1.col1 IN
  (SELECT col2 FROM Table T2);
```

Täitmisplaani koostamise programm võiks aru saada, et need päringud on ekvivalentsed ning koostama sama täitmisplaani.

```
SELECT * FROM Tabel1 T1
WHERE EXISTS
  (SELECT * FROM Table T2 WHERE
    T1.col=T2.col);
```

15.12.2017 Teema 7 109

Andmebaasid II 2017 © Erki Eessaar

Algoritm alampäringu täitmiseks – ilma indeksita

```
for (each row in Tabel1) { /*Väline tsükkel*/
  for (each row in Tabel2) { /*Sisemine tsükkel*/
    if (Tabel1.col1 matches Tabel2.col2) then
      pass /*Oleme leidnud kokkulangeva
väärtuste paari*/
      EXIT LOOP; /*Tsükli töö võib
katkestada*/
    }
  }
```

15.12.2017 Teema 7 110

Andmebaasid II 2017 © Erki Eessaar

Algoritm alampäringu täitmiseks – indeksiga

```
for (each row in Tabel1) { /*Väline tsükkel*/
  lookup value in the index in table Table 2 {
    if (found) then
      pass /*Oleme leidnud kokkulangeva väärtuste
paari*/
      EXIT LOOP
    end if;
  }
}
```

15.12.2017 Teema 7 111

Andmebaasid II 2017 © Erki Eessaar

SQL lausete ümberkirjutamine

- ♦ Sama ülesannet saab enamasti lahendada paljude erinevate SQL lausetega.
- ♦ Andmebaasisüsteem täidab mõnda lauset kiiremini kui teist.
 - Põhjus – kahele samaväärsele lausele koostatakse erinev füüsiline täitmisplaan.
- ♦ Erinevates andmebaasisüsteemides võib lause täitmise kiirus olla erinev.
- ♦ Järgnevad soovitused võivad, kuid ei pruugi, anda tulemusi.

15.12.2017 Teema 7 112

Andmebaasid II 2017 © Erki Eessaar

SQL lausete ümberkirjutamine (2)

- ♦ Ülesanne: Millistel töötajatel on palk 199170USD?
- ♦ 4 andmebaasisüsteemi
- ♦ 12 erinevat SQL lauset, mis etteantud tabelite põhjal ülesande lahendavad
- ♦ Erinevate SQL lausete kasutamine omas mõju vastuse leidmise kiirusele.

Tabel 3 – erinevad täitmisplaanid originaalpäringus 20 miljoni reaga kasutaja poolt indeksitunud tabelis

	JOIN	IN	ANY1	IN2	ANY2	EXISTS	COUNT	INNER	SOME1	SOME2	FULL	J	NAT.	J
Postgres 9.1	1	2	2	3	3	2	4	1	2	3	5	1		
Postgres 9.3	1	1	3	3	1	4	1	1	2	5	1			
Oracle 11g	6	6	6	6	6	7	7	6	6	6	8	6		
Oracle 12c	9	9	9	9	9	10	10	9	9	9	11	9		

Kui mitmes lahtris sama väärtus, siis sama täitmisplaan.

Männil, M., 2014. *Mõnede SQL-andmebaasisüsteemide võimekusest SQLi keelelise liiasuse silumisel*. Magistritöö, TTÜ Informaatikainstituut. [WWW] <https://digi.lib.ttu.ee/171952>

Tabel 4 – päringute kestused sekundis originaalpäringus 20 miljoni reaga kasutaja poolt indeksitunud tabelis

	JOIN	IN	ANY1	IN2	ANY2	EXISTS	COUNT	INNER	SOME1	SOME2	FULL	J	NAT.	J
Postgres 9.1	0.13	0.13	0.13	251.15	251.30	0.15	275.09	0.13	0.13	250.61	0.14	0.14		
Postgres 9.3	0.10	0.10	0.10	207.79	207.83	0.14	224.81	0.15	0.09	208.13	0.14	0.09		
Oracle 11g	0.16	0.16	0.17	0.16	0.16	0.14	0.15	0.16	0.17	0.10	108.87	0.16		
Oracle 12c	0.12	0.11	0.12	0.13	0.12	0.12	0.13	0.13	0.12	0.12	0.12	0.12		

15.12.2017 Teema 7 113

Andmebaasid II 2017 © Erki Eessaar

Ära kasuta üleliigseid tingimusi

```
SELECT * FROM Aine WHERE aine_kood
BETWEEN 'IDU0120' AND 'IDU9870'
AND aine_kood='IDU3381';
```

```
SELECT * FROM Aine WHERE aine_kood
IN ('IDU3381', 'IDU3382', 'IDU3381');
```

15.12.2017 Teema 7 114

Jälgi alamtingimuste järjekorda

- Enamik andmebaasisüsteeme hakkab kontrollima tingimuse täidetust **vasakult-paremale**.
 - AND – kõige kitsendavam tingimus kõige esimene
 - OR – kõige üldisem tingimus kõige esimene
- Oracle 12c Release 1 Enterprise Edition
 - Kui mitu alamtingimust, mis seotud ANDidega või mitu alamtingimust, mis seotud ORidega, siis pole vasakult paremale kontroll garanteeritud.

15.12.2017

Teema 7

115

Jälgi alamtingimuste järjekorda – näide

- Võimalik, et halvem:
 - SELECT * FROM Isik WHERE **perenimi**='Tamm' AND pikkus=215;
- Võimalik, et parem:
 - SELECT * FROM Isik WHERE **pikkus**=215 AND perenimi='Tamm';
 - SELECT * FROM Isik WHERE **perenimi**='Tamm' OR pikkus=215;

Eeldus – isikuid, kelle perenimi on "Tamm" on palju rohkem kui 215 cm pikki isikuid.

15.12.2017

Teema 7

116

◇ operaatori kasutamine

- Halvem (umbes 87% Iiri Vabariigi elanikest on katoliiklased):
SELECT * FROM Iirimaa_elanikud WHERE usutunnistus◇'Katoliiklane';
- Parem (võib sundida kasutama indeksit veerul *usutunnistus*, kui see on loodud):
SELECT * FROM Iirimaa_elanikud WHERE usutunnistus>'Katoliiklane' OR usutunnistus<'Katoliiklane';

15.12.2017

Teema 7

117

Vähenda tabelis tehtavate otsingute arvu

- Leia õppeasutused, kus on vähemalt üks õppekava ning kõikidel õppekavadel on nominaalne õppeaeg üle 100 nädala või on nominaalne õppeaeg määramata.
- Väljasta õppeasutuste arvulised identifikaatorid.

15.12.2017

Teema 7

118

Vähenda tabelis tehtavate otsingute arvu (2)

- SELECT oppeasutus_id FROM Oppeasutus WHERE oppeasutus_id NOT IN (SELECT **oppeasutus_id** FROM Oppekava WHERE **oppe_nadala**<100) AND oppeasutus_id IN (SELECT oppeasutus_id FROM Oppekava);
 - Plokkide mälust lugemiste arv: 38
- SELECT oppeasutus_id FROM Oppekava GROUP BY oppeasutus_id HAVING Min(oppe_nadala)>100 OR Min(oppe_nadala) IS NULL;
 - Plokkide mälust lugemiste arv: 25

15.12.2017

Teema 7

119

Välgi võimalusel sorteerimist

- Sorteerimist võivad (kuid sõltuvalt süsteemist alati ei pruugi) kasutada:
 - DISTINCT, UNION, INTERSECT, EXCEPT ja GROUP BY
- Halvem:
 - SELECT aine_kood, nimetus FROM Aine WHERE aine_kood LIKE 'IDU%' **UNION** SELECT aine_kood, nimetus FROM Aine WHERE ainepunkte>3;
- Parem:
 - SELECT aine_kood, nimetus FROM Aine WHERE aine_kood LIKE 'IDU%' OR ainepunkte>3;

15.12.2017

Teema 7

120

Väldi võimalusel sorteerimist (2)

- ♦ Halvem:


```
SELECT aine_kood, Avg(tulemus) AS keskm
FROM Oppimine
GROUP BY aine_kood
HAVING aine_kood='IDU0230';
```
- ♦ Parem:


```
SELECT 'IDU0230' AS aine_kood, Avg(tulemus) AS
keskm
FROM Oppimine
WHERE aine_kood='IDU0230';
```

15.12.2017

Teema 7

121

Küsi vaid neid andmeid, mida kasutad

- ♦ Halvem:
 - `SELECT * FROM Aine WHERE aine_kood LIKE 'IDU%';`
- ♦ Parem:
 - `SELECT aine_kood, ainepunkte FROM Aine WHERE aine_kood LIKE 'IDU%';`
- ♦ Kui `SELECT *` päring vaate põhjal, siis ei saa andmebaasisüsteem ilmselt rakendada *tabelite elimineerimise* tehnikat.

15.12.2017

Teema 7

122

Täida mitme lause asemel üks – halb

- ♦ `SELECT Count(*) count1 FROM Aine WHERE ainepunkte<2;`
- ♦ `SELECT Count(*) count2 FROM Aine WHERE ainepunkte BETWEEN 2 AND 3;`
- ♦ `SELECT Count(*) count3 FROM Aine WHERE ainepunkte>3;`

15.12.2017

Teema 7

123

Täida mitme lause asemel üks – parem lahendus

```
SELECT COUNT (CASE WHEN ainepunkte < 2
THEN 1 ELSE NULL END) count1,
COUNT (CASE WHEN ainepunkte BETWEEN
2 AND 3
THEN 1 ELSE NULL END) count2,
COUNT (CASE WHEN ainepunkte > 3
THEN 1 ELSE NULL END) count3
FROM Aine;
```

Üle võrgu liigub vähem infot, suhteliselt vähem aega kulub lausete töötlemisele

15.12.2017

Teema 7

124

Täida mitme lause asemel üks – halb (2)

- ♦ Iga olemi jaoks eraldi päring
 - `SELECT nimetus FROM Aine WHERE aine_kood='IDU3381';`
 - `SELECT nimetus FROM Aine WHERE aine_kood='IDU0220';`
- ♦ Sama olemi atribuutide jaoks eraldi päringud
 - `SELECT nimetus FROM Aine WHERE aine_kood='IDU0220';`
 - `SELECT ainepunkte, loenguid FROM Aine WHERE aine_kood='IDU0220';`
- ♦ Varanduse otsimine
 - `SELECT hindamisviis_id, nimetus FROM Aine WHERE aine_kood='IDU0220';`
 - `SELECT nimetus FROM Hindamisviis WHERE hindamisviis_id=1;`

15.12.2017 • /*Väärtuse 1 sain teada eelmise päringuga*/

Teema 7

125

Täida mitme lause asemel üks – parem lahendus (2)

- ♦ `SELECT aine_kood, nimetus FROM Aine WHERE aine_kood IN ('IDU3381', 'IDU0220');`
- ♦ `SELECT nimetus, ainepunkte, loenguid FROM Aine WHERE aine_kood='IDU0220';`
- ♦ `SELECT h.nimetus AS hindamisviis, a.nimetus AS aine FROM Hindamisviis AS H INNER JOIN Aine AS A USING (hindamisviis_id) WHERE aine_kood='IDU0220';`

15.12.2017

Teema 7

126

Lihtsusta lauset

- ♦ Kirjuta samaväärne, kuid lihtsama süntaksiga lause.
- ♦ Halvem:
 - `SELECT ainepunkte FROM (SELECT ainepunkte, aine_kood FROM Aine WHERE aine_kood IN (SELECT aine_kood FROM Oppimine WHERE aine_kood IS NOT NULL)) WHERE aine_kood LIKE 'Z%';`
- ♦ Pareim:
 - `SELECT ainepunkte FROM Aine WHERE aine_kood LIKE 'Z%' AND aine_kood IN (SELECT aine_kood FROM Oppimine WHERE aine_kood IS NOT NULL);`

15.12.2017

Teema 7

127

Lihtsusta lauset (2)

- ♦ Kustuta osakonnad, kus on vähemalt üks töötaja.
- ♦ Halvem:
 - `DELETE FROM Dept WHERE deptno IN (SELECT deptno FROM Dept INTERSECT SELECT deptno FROM Emp);`
- ♦ Pareim:
 - `DELETE FROM Dept WHERE deptno IN (SELECT deptno FROM Emp);`

15.12.2017

Teema 7

128

Põhjused, miks päringu täitmisel ei kasutata indeksit

- ♦ Päring `SELECT Count(*) AS arv FROM T;` ei kasuta tabelile T veerule v loodud indeksit.
 - v on mittekohustuslik (lubatud NULLid)
- ♦ Päringu otsingutingimuses ei kasutata veergu, mis on liitindeksis esimene.
 - Indeks I(x, y)
 - `SELECT * FROM T WHERE y=<literaal>;`

15.12.2017

Teema 7

129

Näide (PostgreSQL)

```

experiment_views=# CREATE TABLE hcv AS SELECT * FROM health_care_visit;
SELECT 1000000
experiment_views=# ALTER TABLE hcv ADD CONSTRAINT pk_hcv PRIMARY KEY (health_care_visit_id);
ALTER TABLE
experiment_views=# ALTER TABLE hcv ADD CONSTRAINT uk_hcv UNIQUE(patient_id, facility_id, postal_address_id, from_d
ate);
ALTER TABLE
experiment_views=# VACUUM ANALYZE;
VACUUM
experiment_views=# EXPLAIN SELECT Count(*) AS cnt FROM hcv WHERE patient_id BETWEEN 10000 AND 20000;
QUERY PLAN
Aggregate  (cost=4.45..4.46 rows=1 width=0)
->  Index Only Scan using uk_hcv on hcv  (cost=0.42..4.44 rows=1 width=0)
    Index Cond: ((patient_id >= 10000) AND (patient_id <= 20000))
(3 rows)
Täitmisaja: 1.7 ms

experiment_views=# EXPLAIN SELECT Count(*) AS cnt FROM hcv WHERE facility_id BETWEEN 10000 AND 20000;
QUERY PLAN
Aggregate  (cost=27418.66..27418.67 rows=1 width=0)
->  Seq Scan on hcv  (cost=0.00..27177.00 rows=96665 width=0)
    Filter: ((facility_id >= 10000) AND (facility_id <= 20000))
(3 rows)
Täitmisaja: 360 ms

experiment_views=# CREATE INDEX idx_hcv ON hcv (facility_id);
CREATE INDEX
experiment_views=# VACUUM ANALYZE;
VACUUM
experiment_views=# EXPLAIN SELECT Count(*) AS cnt FROM hcv WHERE facility_id BETWEEN 10000 AND 20000;
QUERY PLAN
Aggregate  (cost=3285.63..3285.64 rows=1 width=0)
->  Index Only Scan using idx_hcv on hcv  (cost=0.42..3040.61 rows=98009 width=0)
    Index Cond: ((facility_id >= 10000) AND (facility_id <= 20000))
(3 rows)
Täitmisaja: 22 ms
15.12.2017

```

Teema 7

130

Näide (PostgreSQL) (2)

- ♦ **1** – süsteem **kasutab** UNIQUE kitsenduse alusel automaatselt loodud indeksit `ak_hcv`, sest `patient_id` on selles indeksis **esimene veerg**.
- ♦ **2** – süsteem **ei kasuta** UNIQUE kitsenduse alusel automaatselt loodud indeksit `ak_hcv`, sest `facility_id` **ei ole** selles indeksis **esimene veerg**.
- ♦ **3** – luuakse B-puu indeks, kus `facility_id` on esimene veerg. Süsteem **kasutab** seda päringu täitmisel, sest `facility_id` on selles indeksis **esimene veerg**.

15.12.2017

Teema 7

131

Põhjused, miks päringu täitmisel ei kasutata indeksit (2)

- ♦ Otsingutingimuses on veerule rakendatud funktsioon.


```
SELECT * FROM Temperatuurid
WHERE f_Fahrenheit_Celsius(temp)>40;
```
- ♦ Otsingutingimuses on veerule rakendatud funktsioon.


```
SELECT * FROM Oppimine
WHERE (lopetamise_kuupaev-
eelregistreerimise_kuupaev)=30 AND
eelregistreerimise_kuupaev>to_date('21/01/2002','DD/
MM/YYYY');
```

15.12.2017

Teema 7

132

Tingimuses on funktsiooni väljakutse

- ♦ Lahendus 1 – funktsioonil põhinev indeks:
 - `CREATE INDEX idx_f_celsius_fahrenheit ON Temp(f_Fahrenheit_Celsius(temp));`
- ♦ Lahendus 2 – lause ümberkirjutamine:
 - `SELECT * FROM Oppimine WHERE lopetamise_kuupaev=eelregistreerimise_kuupaev+30 AND eelregistreerimise_kuupaev>to_date('21/01/2002','DD/MM/YYYY');`

15.12.2017

Teema 7

133

Põhjused, miks päringu täitmisel ei kasutata indeksit (3)

- ♦ Indeksit ei kasutata, kui WHERE klauslis ei lange kasutatud literaali andmetüüp kokku veeru andmetüübiga.
 - Käivitav lause:
 - `SELECT * FROM T WHERE indekseeritud_veerg=5;`
 - Tegelikult täidetav lause:
 - `SELECT * FROM T WHERE to_number(indekseeritud_veerg)=5;`

15.12.2017

Teema 7

134

Põhjused, miks päringu täitmisel ei kasutata indeksit (4)

- ♦ Indeksit ei kasutata, sest andmebaasisüsteemi arvates **pole see otstarbekas**.
 - Indeksit ei kasutata, sest tabeli/indeksi kohta kogutud statistika on vananenud ja vananenud info alusel leiab andmebaasisüsteem, et parem on indeksit mitte kasutada.
 - Statistika on värske, kuid optimeerimismoodulisse sisse ehitatud reeglid soovivad indeksit mitte kasutada.

15.12.2017

Teema 7

135

Indeksi uuesti ülesehitamine (Oracle)

- ♦ Kui tabelis T toimub palju andmete muudatusi/kustutamisi, siis võib tabeliga seotud indeksis I olla palju kasutamata ruumi.
- ♦ Kui tabelis T on palju andmeid, siis tuleks indeksit I regulaarselt uuesti üles ehitada, et muuta seda kompaktsemaks (hõlmab vähem plokkide) ja optimeerijale atraktiivsemaks.
- ♦ `ALTER INDEX Big_index REBUILD NOLOGGING PARALLEL;`

15.12.2017

Teema 7

136

Töökiiruse parandamine Oracles – veel võimalusi

- ♦ Täitmisplaani vaatamine ja salvestamine
- ♦ Salvestatud täitmisplaanide taaskasutamine
 - Fikseeritakse lause täitmisplaan mingi hetke seisuga ja antakse korraldus seda alati kasutada
- ♦ Paralleeltöö
 - Lause jagamine alamosadeks, nende paralleelne täitmine ja tulemuste ühendamise
- ♦ Automaatne töökiiruse analüüs ja parandamine
- ♦ *Dynamic sampling*
 - *Hard parse* käigus statistika täpsustamine

15.12.2017

Teema 7

137

Töökiiruse parandamine Oracles – veel võimalusi (2)

- ♦ Vihjed
- ♦ Histogrammid
- ♦ Tsoonikaartid (*zone maps*)
- ♦ Hetktõmmised ja käivitatud päringute automaatne ümberkirjutamine, et need hakkaksid kasutama hetktõmmist.
- ♦ Süsteemikataloogi tabelid, milles andmed andmete ja süsteemi kasutuse kohta.

15.12.2017

Teema 7

138

Adaptiivne optimeerimine (alates Oracle 12c)

- ◆ Kui valitud plaani alusel lause täitmisel ilmneb, et tegelik andmehulk erineb oluliselt plaani koostamisel arvesse võetud andmehulkadega (statistika oli ebatäpne), siis
- ◆ Oracle on võimeline lause täitmiseks kasutatavat täitmisplaani mõnda alamosa jooksu pealt muutama
 - näiteks asendama *nested loop join* algoritmi *hash join* algoritmi kasutamisega.

15.12.2017

Teema 7

139

Adaptiivne statistika (alates Oracle 12c)

- ◆ *Dynamic Sampling* edasiarendus – kui täitmisplaani koostamisel ilmneb statistika puudulikkus, siis
- ◆ Oracle on võimeline statistikat jooksupealt täiendama.

15.12.2017

Teema 7

140

Automaatne reoptimeerimine (alates Oracle 12c)

- ◆ Oracle jätab koostatud täitmisplaani meelde, et sama lause järgnevatel täitmistel seda taaskasutada.
- ◆ Kui lause täitmise järel selgub, et nt andmete hulk erineb oluliselt eeldatust, siis
- ◆ Oracle asendab järgnevate käivituste jaoks täitmisplaani uue plaaniga arvestades ka eelmistel käivitustel kogutud infot.

15.12.2017

Teema 7

141

SQL*Plus käsud

- ◆ SET TIMING ON;
 - Näidatakse lause täitmiseks ja tulemuse väljastamiseks kulunud aega.
- ◆ SET AUTOTRACE ON;
 - Lause täidetakse ja lisaks näidatakse lause hinnangulist (ingl *estimated*) e ennustatavat täitmisplaani ning statistikat
- ◆ SET TIMING OFF;
- ◆ SET AUTOTRACE OFF;

15.12.2017

Teema 7

142

Täitmisplaani salvestamine

- ◆ EXPLAIN PLAN lause.
 - Erinevalt AUTOTRACEist ei tingi lause tegelikku täitmist.
 - Tulemuseks samuti *hinnanguline* plaan.
- ◆ Plaan salvestatakse *PLAN_TABLE* süsteemsesse tabelisse.
- ◆ Plaani nägemiseks päring selle tabeli põhjal koos väljundi formaatimisega.

15.12.2017

Teema 7

143

Tegelik täitmisplaan

- ◆ Võib erineda AUTOTRACE või EXPLAIN PLAN abil nähtavast ennustatavast plaanist.
 - Ennustatava plaani koostamisel pole kogu vajaminevat lähteinformatsiooni.
- ◆ Tegeliku plaani vaatamiseks tuleb kasutada paketi *dbms_xplan* olevat funktsiooni *display_cursor*.
 - Käivitatakse peale uuritava lause täitmist ja loeb tegeliku plaani muutmälust (kursori vahemälust).

15.12.2017

Teema 7

144

Trace fail

- ♦ Võimaldab uurida süsteemis mingi perioodi vältel käivitatud lausete tegelikke täitmisplaane ja täitmise statistikat.
- ♦ `ALTER SESSION SET SQL_TRACE=TRUE;`
- ♦ Moodustatakse *trace fail*, mis sisaldab muuhulgas andmeid:
 - SQL lause täitmiseks kulunud aeg,
 - kettalt toimunud lugemised,
 - töödeldud ridade arv.

15.12.2017

Teema 7

145

Salvestatud täitmisplaani taaskasutamine

- ♦ Vanemad Oracle versioonid:
 - OUTLINE objektide loomine
 - Staatiline optimeerimine. Eksisteerib oht, et täitmisplaan pole aja jooksul enam optimaalne.
- ♦ Viimased Oracle versioonid
 - *SQL Plan Management*
 - Võimaldab SQL lause jaoks defineerida aktsepteeritavate täitmisplaanide hulga (*baseline*) ja seda paremate plaanide ilmnedes dünaamiliselt täiendada.
 - Hübriidne optimeerimine.

15.12.2017

Teema 7

146

Vihjed

- ♦ Täitmisplaani koostajat saab vihjetega suunata.
- ♦ Optimeerijat võib sundida:
 - kasutada reeglipõhist optimeerijat,
 - kasutada mingit kindlat indeksit,
 - kasutada mingit ühendamise järjekorda,
 - ...
- ♦ Probleemid.
 - Kas administraator on ikka targem/kavalam kui andmebaasisüsteem?
 - Andmete/serveri seadete muutudes ei pruugi vihje olla enam sobiv.

15.12.2017

Teema 7

147

Vihjed – näited

- ♦ Sunni kasutama indeksit:
 - `SELECT /*+ INDEX(sugu_idx)*/ nimi FROM Tudeng WHERE sugu='M';`
- ♦ *Tabel1* on ühendamisoperatsioonis väline tabel:
 - `SELECT /*+ORDERED */ * FROM Tabel1 T1 INNER JOIN Table2 T2 ON T1.col=T2.col;`
- ♦ *Tabel 2* on ühendamisoperatsioonis väline tabel:
 - `SELECT /*+ORDERED */ * FROM Tabel2 T2 INNER JOIN Table1 T1 ON T1.col=T2.col;`

15.12.2017

Teema 7

148

Histogrammid

- ♦ Annavad infot väärtuste *ebatühtlase* jaotumise kohta veerus.
- ♦ Mõne väärtuse puhul oleks otsimiseks kasulik kasutada indeksit, mõne puhul mitte.
- ♦ Loomiseks kasutada DBMS_STATS paketti kuuluvaid rutine.

15.12.2017

Teema 7

149

Histogrammid (näide)

- ♦ Tabel *Aine* veerg *ainepunkte*. Veerul indeks.
- ♦ Jaotus:

0.5 AP	50
1 AP	5
1.5 AP	100
2.5 AP	125
3 AP	40
3.5 AP	400
4 AP	25
4.5 AP	20

Võiks kasutada indeksit:

```
SELECT * FROM Aine
WHERE ainepunkte=1;
```

Võiks kasutada tabeli *täielikku* läbiskaneerimist:

```
SELECT * FROM Aine
WHERE ainepunkte=3.5;
```

15.12.2017

Teema 7

150

Andmebaasid II 2017 © Erki Eessaar

Histogrammid (näide) (2)

- exec

```

dbms_stats.gather_table_stats
('c##tud1','aine',
method_opt=>'FOR
COLUMNS ainepunkte SIZE
10');

```
- SELECT endpoint_number,
endpoint_value FROM
user_tab_histograms WHERE
column_name='AINEPUNKTE'
ORDER BY endpoint_number;

ENDPOINT_NUMBER	ENDPOINT_VALUE
89	1
154	2
218	3
300	4
301	5

15.12.2017 Teema 7 151

Andmebaasid II 2017 © Erki Eessaar

Histogrammid (näide) (3)

Buketide otspunktid

Võimalikud väärtused veerus

Igas buketis võimalikult ühepalju väärtuseid
Mida kitsam buket, seda rohkem väärtuseid vahemikus
Mida laiem buket, seda vähem väärtuseid vahemikus

15.12.2017 Teema 7 152

Andmebaasid II 2017 © Erki Eessaar

Paralleeltöö

- Lause jagamine alamosadeks ja nende täitmine erinevate keskprotsessorite (CPU-de) poolt.
- Kasulik rakendada lausete juures, mille täitmine võtab kaua aega.
- Lausete tüübid, mille puhul saab paralleeltööd rakendada:
 - SELECT,
 - CREATE INDEX,
 - INSERT, UPDATE, DELETE.

15.12.2017 Teema 7 153

Andmebaasid II 2017 © Erki Eessaar

Paralleeltöö – näide

```

SELECT S.name
FROM Supplier S,
Supplier_parts Sp, Parts P
WHERE S.s_no=Sp.s_no
AND Sp.p_no=P.p_no
AND S.city='London'
AND Sp.qty>200
AND P.color='red'
AND P.weight<25;

```

- SELECT P.p_no FROM Parts P WHERE P.color='red' AND P.weight<25;
- SELECT Sp.p_no, Sp.s_no FROM Supplier_parts Sp WHERE Sp.qty>200;
- SELECT S.s_no, S.name FROM Supplier S WHERE S.city='London';

15.12.2017 Teema 7 154

Andmebaasid II 2017 © Erki Eessaar

Automaatne töökiiruse analüüs ja parandamine

- Automatic Optimizer Statistics Collection
- Automatic Workload Repository (AWR)
- Automatic Database Diagnostic Monitor
 - Analüüsib AWR kogutud andmeid
- Automatic SQL Tuning Advisor
- Active Session History
- Automatic Shared Memory Management

15.12.2017 Teema 7 155

Andmebaasid II 2017 © Erki Eessaar

Dynamic sampling

- Lause täitmisklaani koostamise käigus loeb andmebaasisüsteem kasutatavaid tabeleid/indekseid, et statistikat täpsustada.
- Põhjuseks, et vajalik statistika puudub või ei ole värske.
- 11 erinevat taset (0 – üldse ei kasutata; 10 – statistika täpsustamiseks loetakse terve tabel/indeks) (parameeter optimizer_dynamic_sampling).

15.12.2017 Teema 7 156

Oracle tõlkeraamistik (alates Oracle 12c)

- ♦ *Oracle Translation Framework*
- ♦ Idee: Oracle andmebaasis saab defineerida **tõlked**, mis asendavad sissetuleva andmekäitluskeele lause mõne teise lausega.
- ♦ Rakendustes sisalduvaid SQL lauseid poleks vaja ümber kirjutada.
 - Eriti kasulik olukorras, kui tegemist mõne teise osapoole tehtud rakendusega.

15.12.2017

Teema 7

157

Oracle tõlkeraamistik (alates Oracle 12c) (2)

- ♦ Millal kasutada?
 - Üleminek *mitte-Oracle andmebaasi* kasutamiselt *Oracle andmebaasi* kasutamisele.
 - Mitte-Oracle SQL dialektis laused asendada Oracle dialektis SQL lausetega
 - Kui Oracle andmebaasi kasutavates rakendustes sisaldub *töökiiruse* mõttes mitteoptimaalseid SQL lauseid, siis asendada need paremini kirjutatud lausetega.

15.12.2017

Teema 7

158

PostgreSQL

- ♦ Päringu täitmisplaani koostamisel arvestatakse päringu ümberkirjutamise reegleid (RULE).
- ♦ Füüsilise täitmisplaani vaatamiseks EXPLAIN lause.
- ♦ Tabeliga seotud statistika kogumiseks ANALYZE lause:
 - ANALYZE Oppimine;

15.12.2017

Teema 7

159



PostgreSQL (2)

- ♦ "Prügi" kustutamine – ridade muutmise/kustutamise käigus tekkinud ridade versioonide lõplik kustutamine – VACUUM lause:
 - VACUUM; --kogu andmebaas
 - VACUUM ANALYZE Oppimine;
 - /*Kindel tabel, lisaks peale prügikoristust selle tabeli statistika kogumine*/
 - Ridade versioonid tekkivad sellest, et PostgreSQL kasutab multiversioon konkurentsjuhtimist.

15.12.2017

Teema 7

160

PostgreSQL (3)

- ♦ Statistika kogumist ja prügikoristuse protseduuride läbiviimist automatiseerib *autovacuum* protsesside kogum (autovacuum deemon).
 - *Deemon* on tagaplaanil jooksev programm, mis teostab teatud ettemääratud operatsioone kindlate ajavahemike tagant või vastuseks mingitele sündmustele. (Vallaste, 2000–2016, <http://www.vallaste.ee/>)



15.12.2017

Teema 7

161

PostgreSQL (4)

- ♦ EXPLAIN ANALYZE .. – täitmisplaani genereerimine, lause täitmine ja plaani koostamise ning täitmisaja vaatamine.
- ♦ Kui lauset on sessiooni jooksul vaja korduvalt käivitada, siis saab lause *dekompositsiooni* viia läbi ükskord (PREPARE lause) ning järgnevatel lause käivitustel **samas sessioonis** saab süsteem dekompositsiooni tulemust *taaskasutada* (EXECUTE lause).

15.12.2017

Teema 7

162

PostgreSQL (5)

- ♦ Automaatne täitmisklaanide logimine – *Autoexplain*.
- ♦ Eelised
 - Kui rakenduse lähtekood pole avalik, siis puudub info, milliseid lauseid see võib see põhimõtteliselt käivitada.
 - Saame teada, milliseid plaane andmebaasisüsteem *tegelikult* kasutas.
- ♦ Puudused
 - Logi kasvab kiiresti.
 - Muudab lausete täitmise aeglasemaks.

15.12.2017

Teema 7

163

PostgreSQL (6)

- ♦ **\timing** käsk psqlis, et vaadata lause täitmisaega
- ♦ **Süsteemikataloogi** põhjal saab teha päringuid, et vaadata tabelite/indeksite kohta kogutud statistikat:
 - `SELECT relname AS nimi, relkind AS tyyp, reltuples AS ridade_arv, relpages AS plokkide_arv FROM pg_class WHERE relname='oppekava';`
 - `SELECT * FROM pg_stats WHERE tablename='oppekava';`
 - `pg_stats` – süsteemikataloogi vaade

15.12.2017

Teema 7

164

PostgreSQL (7)

- ♦ Veel *näiteid* küsimustest, millele saab **süsteemikataloogi** põhjal vastuse.
 - Milline on erinevate skeemiobjektide ja kogu andmebaasi suurus?
 - Millal viimati statistikat värskendati?
 - Millised indeksid võivad olla üleliigsed, st süsteem ei kasuta neid päringute täitmiseks?
 - Millised on viimatised serveris andmebaaside klastris käivitatud SQL laused?

15.12.2017

Teema 7

165

PostgreSQL – kuidas mõjutada täitmisklaanide koostamist?

- ♦ Lause kirjutaja saab sundida andmebaasisüsteemi kasutama lause täitmisel kindlat tabelite ühendamise järjekorda:
 - `SET join_collapse_limit=1;`
 - Kasuta süntaksi, kus ühendamise tingimus `FROM` klauslis.
 - `SELECT * FROM a JOIN (b JOIN c ON (b.ref = c.id)) ON (a.id = b.id);`
 - ♦ Ühendatakse **b** ja **c** ning selle ühendamise tulemusega ühendatakse **a**

15.12.2017

Teema 7

166

PostgreSQL – kuidas mõjutada täitmisklaanide koostamist? (2)

- ♦ Juhtparameetrid, mille väärtuseid saab muuta nii süsteemi kui sessiooni tasemel
 - Näide: `SET enable_seqscan = off;`
 - Kui võimalik, ära kasuta tabeli täielikku läbiskaneerimist.
- ♦ Võimalik muuta erinevate operatsioonide maksumuse hinnanguid.
- ♦ Võimalik muuta veeru kohta kogutava statistika hulka.

15.12.2017

Teema 7

167

Näide (PostgreSQL) (1)

```
EXPLAIN ANALYZE SELECT * FROM Objects WHERE object_id BETWEEN 222 AND 333;
```

```
QUERY PLAN
```

```
Index Scan using pk_object on objects (cost=0.00..371.59 rows=119
width=42) (actual time=0.027..0.053 rows=8 loops=1)
  Index Cond: ((object_id >= 222) AND (object_id <= 333))
  Total runtime: 0.106 ms
(3 rows)
```

Indeksipuud kammides leiti tingimusele vastavate ridade aadressid ja loeti plokid, mis neid ridu sisaldavad. Kasutati *index range scan* (vahemikskaneerimise) meetodit.

15.12.2017

Teema 7

168

Andmebaasid II 2017 © Erki Eessaar

Näide (PostgreSQL) (2)

```
EXPLAIN ANALYZE SELECT *
FROM Status WHERE status_id IN (SELECT status_id FROM Objects);
```

QUERY PLAN

```
Nested Loop IN Join (cost=0.00..13.62 rows=4 width=26) (actual
time=35.222..41.029 rows=3 loops=1)
-> Seq Scan on status (cost=0.00..1.04 rows=4 width=26) (actual
time=13.012..13.022 rows=4 loops=1)
-> Index Scan using object_status_idx on objects (cost=0.00..371.30
rows=119 width=4) (actual time=6.992..6.992 rows=1 loops=4)
Index Cond: ("outer".status_id = objects.status_id)
Total runtime: 51.977 ms
(5 rows)
```


Iga tabelist *Status* loetud reale otsitakse indeksi abil vastet tabelist *Objects*.

SeqScan – Sequential Scan.
Vaste Oracle operatsioonile *Full Table Scan*.
Tähendab kõigi tabeli andmeid sisaldavate plokkide lugemist.

15.12.2017 Teema 7 169

Andmebaasid II 2017 © Erki Eessaar

Andmebaasioperatsiooni aegluse võimalikud põhjused



- ♦ Viga selles, kuidas rakendus suhtleb andmebaasiga
- ♦ Andmebaasi tabelite disain
- ♦ Indeksid
- ♦ Andmebaasisüsteemi juhtparameetrite väärtused
- ♦ Regulaarselt käivituvad tegevused, mis koormavad serverit (pakktööd – *batch job*)
- ♦ Riistvara (sh arvutivõrk)

15.12.2017 Teema 7 170

Andmebaasid II 2017 © Erki Eessaar

Täpsemad põhjused – näiteid

- ♦ SQL lausetes ei kasutata *bind variable*
- ♦ Päring välistab indeksi kasutamise
- ♦ On käivitatud mittevajalik andmebaasi funktsionaalsus (nt ridade versioonide hoidmine, auditeerimine)
- ♦ Tabelite ja indeksite statistika on vananenud
- ♦ Sageli kasutatav indeks ei mahu muutmällu

15.12.2017 Teema 7 171

Andmebaasid II 2017 © Erki Eessaar

Täpsemad põhjused – näiteid (2)

- ♦ Veerus on andmed ebaühtlaselt jaotunud, kuid puudub histogramm
- ♦ Üleliigseid veerge hõlmav indeks
- ♦ Vigane indeks, mis tuleks värskendada (*rebuild*)
- ♦ Puudub indeks veerul, kus see võiks olla
- ♦ Rakendus küsib andmeid, mida sellel tööks tegelikult vaja ei lähe

15.12.2017 Teema 7 172

Andmebaasid II 2017 © Erki Eessaar

Põhitõed

- ♦ Süsteemi **arhitektuur** on väga oluline ja sellest sõltub iga konkreetse süsteemi õnnestumine või ebaõnnestumine.
- ♦ Süsteemi disainer peab tundma kasutatavate tarkvaraliste ja riistvaraliste **vahendite peensusi**.
- ♦ **Andmete liiasus** võib parandada andmete lugemise kiirust, kuid vähendab andmete muutmise kiirust.

15.12.2017 Teema 7 173

Andmebaasid II 2017 © Erki Eessaar

Põhitõed (2)

- ♦ Süsteeme tuleb **testida jõudluse** seisukohast arvestades **reaalse kasutajate hulga**, **reaalse andmehulgaga**, **reaalsete andmetega** ning **reaalsuses kasutatavate tarkvara seadistuse parameetrite väärtustega**.
- ♦ Kui andmebaasisüsteem ei kasuta andmekäitluskeele lause täitmiseks **optimaalset täitmisplaani**, siis võib lause täitmine võtta liiga palju aega.

15.12.2017 Teema 7 174

