

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

1. (5 оноо) Нэгэн ерөнхий боловсролын сургуулийн сурагчдийн мэдээлэл дараах бүтэцтэй өгөгдлийн санд байв.

Highschooler(ID int, name text, grade int);

Friend(ID1 int, ID2 int);

Likes(ID1 int, ID2 int);

Дараах нөхцөлүүд биелэх тохиолдол ажиллах триггер бич.

Нөхцөл, үүнд: (1) А сурагч В-д дуртай байсан, (2) А сурагч С сурагчид дуртай болж төлөвөө өөрчлөх, (3) В, С сурагчид найзууд байсан. Энэ тохиолдолд В, С-н найзын харилцааг цуцлах триггер бич. Найзын харилцаа тэгш хэмтэй гэдгийг мартаж болохгүй.

БОДОЛТ:

```
create trigger examTgr_Q01
on Likes
after update
as
begin
    if exists(select*from Friend, (select ID2 from deleted) a, (select ID2 from inserted) b
        where (Friend.ID1 = a.ID2 or Friend.ID2 = a.ID2) and (Friend.ID1 = b.ID2 or Friend.ID2 = b.ID2) and a.ID2 <> b.ID2)
    begin
        delete from Friend where ID1 = (select ID2 from deleted) and ID2 = (select ID2 from inserted);
        delete from Friend where ID2 = (select ID2 from deleted) and ID1 = (select ID2 from inserted);
    end
end
go
```



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

```
create trigger examTgr_Q01
on Likes
  after update
as
begin
  if exists(select*from Friend, (select ID2 from deleted) a, (select ID2 from inserted) b
    where (Friend.ID1 = a.ID2 or Friend.ID2 = a.ID2) and (Friend.ID1 = b.ID2 or Friend.ID2 = b.ID2) and a.ID2 <> b.ID2)
  begin
    delete from Friend where ID1 = (select ID2 from deleted) and ID2 = (select ID2 from inserted);
    delete from Friend where ID2 = (select ID2 from deleted) and ID1 = (select ID2 from inserted);
  end
end
go
```

Тайлбар: А сурагч В-д дуртай байсан, В, С сурагчид найзууд байсан, харин А сурагч С сурагчид дуртай болж төлөвөө өөрчлөхөд В, С-н найзын харилцааг цуцалдаг триггер бичих тул Likes хүснэгт дээр after insert буюу insert хийсний дараа хэрэгждэг байхаар тухайн триггериийг зохион байгуулсан. Мөн if exists(...) зааврыг ашиглан нөхцлүүдийг тооцсон бөгөөд тооцохдоо update trigger хэрэгжихэд автоматаар үүсэх deleted, inserted гэх tmp хүснэгтүүдийг ашигласан. Deleted хүснэгтээс авсан ID2 нь В сурагчийн ID байна. Харин inserted хүснэгтээс авсан ID2 нь С сурагчийн ID байх юм. Тухайн хүснэгтүүдийн ID1 нь А сурагчийн ID байх бөгөөд В, С сурагчдыг найзууд гэдгийг мэдэхийн тулд Friend хүснэгтийн ID1 – д В or С , ID2 – д С or В сурагчийн ID байгаа эсэхийг шалгасан мөн өөрчлөгдсөн ID2 баганын ID өмнөхөөс ялгаатайг шалгасан. Тухайн нөхцөлүүд биелэж байвал deleted хүснэгтэд байх В сурагчийн ID – г , inserted хүснэгтэд байх С сурагчийн ID – г ашиглан тухайн сурагчдын найзын харилцааг цуцална.



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

Шалгасан жишээ:

```
select*from Likes where ID1 = 1689-- 1689 likes 1709
select*from Friend where ID1 = 1709 or ID2 = 1709--1709 friends 1247
```

Results		Messages	
ID1	ID2		
1689	1709		
ID1	ID2		
1689	1709		
1709	1247		
1709	1689		
1247	1709		



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

Шалгасан жишээ:

```
update Likes set ID2 = 1247 where ID1 = 1689;  
select*from Likes where ID1 = 1689-- 1689 likes 1247  
select*from Friend where ID1 = 1709 or ID2 = 1709--1709 not friends 1247
```

Results

Messages

ID1	ID2
1689	1247

ID1	ID2
1689	1709
1709	1689



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт


2. Гар утасны компани хэрэглэгчдийнхээ утасны төлбөрийг хянах зорилгоор доорх бүтэцтэй өгөгдлийн санг ашигладаг.

CUSTOMER(SSN, Name, Surname, PhoneNum, PlanID) PRICINGPLAN(PlanID, PricePerSecond) PHONECALL(SSN, Date, Time, CalledNum, Seconds) BILL(SSN, Month, Year, amount)

A. (3 оноо). Дуудлага бүрийн дараа(PhoneCall хүснэгтэд шинэ мөр нэмэгдэх)тухайн хэрэглэгчийн утасны төлбөрийг шинэчлэх триггер бичнэ үү.


БОДОЛТ:

```
--2.A
create trigger examTgr_Q02a
on PhoneCall
    after insert
as
begin
    update Bill set month1 = getdate().month, year1 = getdate().year,
    amount = amount + (select P.PricePerSecond * (select seconds from inserted)
                        from PRICINGPLAN P, CUSTOMER C
                        where C.PlanID = P.PlanID and C.SSN = (Select SSN from inserted))
    where SSN = (select SSN from inserted);
end
go
```



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

```
--2.A
create trigger examTgr_Q02a
on PhoneCall
    after insert
as
begin
    update Bill set month1 = getdate().month, year1 = getdate().year,
    amount = amount + (select P.PricePerSecond * (select seconds from inserted)
                        from PRICINGPLAN P, CUSTOMER C
                        where C.PlanID = P.PlanID and C.SSN = (Select SSN from inserted))
    where SSN = (select SSN from inserted);
end
go
```



Тайлбар: Дуудлага бүрийн дараа (PhoneCall хүснэгтэд шинэ мөр нэмэгдэхэд) тухайн хэрэглэгчийн утасны төлбөрийг шинэчлэх триггер бичих тул PhoneCall хүснэгтэд insert хийсний дараа буюу insert after хэрэгжидэг байхаар үүсгэсэн. Bill буюу төлбөрийн хүснэгтэд хэрэглэгчийн тухайн сарын мэдээлэл үүсгэгдсэн байгаа ингэж үзсэн бөгөөд уг өгөгдөл дээр шинэчлэлт хийсэн болно. Ингэхдээ getdate() функцаар системийн цагийг авч, month болон year функцаар системийн цагаас харгалзан сар болон жилийг авсан. Мөн төлбөрийн хэмжээг бодохдоо insert – ээр орж ирсэн буюу inserted tmp хүснэгтэд байгаа өгөгдлийн секундийг тухайн хэрэглэгчийн Pricingplan хүснэгтэд дэх 1 секундийн төлбөрөөр үржүүлж олсон. Update - ийн нөхцөл дээр tmp inserted хүснэгтэд байгаа дуудлагын мэдээлэл дэх тухайн хэрэглэгчийн SSN(Social Security Number) ингэснээр зөвхөн tmp insert хүснэгтэд байгаа хэрэглэгчийн төлбөрийн мэдээлэл шинэчлэгдэх юм.

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт


2. Гар утасны компани хэрэглэгчдийнхээ утасны төлбөрийг хянах зорилгоор доорх бүтэцтэй өгөгдлийн санг ашигладаг.

CUSTOMER(SSN, Name, Surname, PhoneNum, PlanID) PRICINGPLAN(PlanID, PricePerSecond) PHONECALL(SSN, Date, Time, CalledNum, Seconds) BILL(SSN, Month, Year, amount)

В. (2оноо). Хэрэв хэрэглэгч 103 руу залгасан бол дуудлагын хөлсийг 0 үнээр тооцоолох триггер бичнэ үү.


БОДОЛТ:

```
--2.B
create trigger examTgr_Q02b
on Phonecall
    after insert
as
begin
    if(103 = (select CalledNum from inserted))
    begin
        update Bill set month1 = month(getdate()), year1 = year(getdate()), amount = amount + 0 where SSN = (select SSN from inserted);
    end
end
go
```



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

```
--2.B
create trigger examTgr_Q02b
on Phonecall
    after insert
as
begin
    if(103 = (select CalledNum from inserted))
    begin
        update Bill set month1 = month(getdate()), year1 = year(getdate()), amount = amount + 0 where SSN = (select SSN from inserted);
    end
end
go
```



Тайлбар:). Хэрэв хэрэглэгч 103 руу залгасан залгасан буюу Phonecall хүснэгт рүү insert хийгдсэний дараа хэрэгжих триггер бичих тул after insert гэж тохируулсан. Insert хийгдэхэд үүсдэг tmp inserted хүснэгтээс CalledNum – ийн мэдээллийг авч 103 эсэхийг шалгасан. Хэрэв 103 байгаа тохиолдолд Bill хүснэгтэд update хийгдэх бөгөөд тухайн хэрэглэгчийн төлбөрийн хэмжээ нэмэгдэхгүй бөгөөд 0 – ээр нэмэгдэнэ. Харин сар болон жилийн мэдээллийг шинэчилсэн бөгөөд getdate() функцаар системийн цагийг авч, month болон year функцаар сар жилийн тасдаж авсан. Зөвхөн тухайн дуудлага хийсэн хэрэглэгчийн мэдээллийг шинэчлэх тул нөхцөл дээр inserted хүснэгтийн SSN –ийг Bill –ийн SSN-д утгийн олгосон болно.

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

3. Bid(item, price) хүснэгт өгөгдөв. Bid хүснэгт доорх 2 мөртэй: (phone,10) and (laptop,20). Бид доорх 2 транзакшныг зэрэг ажиллуулав:

T1

S1: update Bid set price = price + 5

S2: insert into Bid values (desktop,30)

T2

S3: select sum(price) as s from Bid

S4: select max(price) as m from Bid

A. (2оноо). T1-н isolation level нь serializable байв. 2 транзакшин 2-уул коммит хийгджээ. T2 транзакшин isolation level repeatable read бол, T2-р буцаагдах боломжтой хосуудыг олно уу, тайлбарла.



Тайлбар:

A. Буцаагдах боломжтой хос утгууд: (70, 30), (30, 20);

□ T2 транзакшин нь T1 транзакшины serializable тусгаарлалтын түвшинд ямар нэг байдлаар нөлөөлж чадахгүй учир нь T2 нь зөвхөн уншилт хийдэг тран юм. Харин T1 нь T2 – ийн repeatable read буюу phantom tuple –ээс бусдын зөвшөөрдөггүй тусгаарлалтын түвшинд нөлөөлж, бохир уншилт болон non-repeatable read үүсгэх боломжтой. Боломжит хувилбарууд:

□ **T1;T2** – T1 тран – ыг бүрэн ажиллаж дууссаны дараа T2 уншилт хийх буюу сериал байдлаар ажиллах тул ямар нэг асуулдал үүсэхгүй. Буцах утга: (70, 30)

□ **T2;T1** – Буцах утга: (30, 20)

□ **T1:T2 = S1;S3;S4;S2** – T2 тран-ны S3, S4 нь баталгаажаагүй өгөгдөл уншиж, бохир уншилт үүсэх тул ингэж ажиллах боломжгүй. Учир нь repeatable read тусгаарлалтын түвшин нь бохир уншилтыг зөвшөөрдөггүй.

□ **T2:T1 = S3;S1;S2;S4** – T2 транд non-repeatable read үүсэх тул мөн ингэж ажиллах боломжгүй. Учир S3, S4 нь нэг хүснэгтийн ижил багана дээрээс уншилт хийж байгаа тул голоор орж тухайн багана дээр өөрчлөлт хийхэд non-repeatable read үүсэх юм. Мөн голоор insert хийснээр phantom tuple үүснэ. Гэхдээ үүнийг repeatable read тусгаарлалтын түвшин зөвшөөрдөг. Мөн T2 – д бохир уншилт үүсэхгүй учир нь T2 нь баталгаажсан өгөгдөл уншина.

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

3. Bid(item, price) хүснэгт өгөгдөв. Bid хүснэгт доорх 2 мөртэй: (phone,10) and (laptop,20). Бид доорх 2 транзакшныг зэрэг ажиллуулав:

T1

S1: update Bid set price = price + 5

S2: insert into Bid values (desktop,30)

T2

S3: select sum(price) as s from Bid

S4: select max(price) as m from Bid

Б. (Зоноо). T1-н isolation level нь read uncommitted байв. 2 транзакшин 2-уул коммит хийгджээ. T2 транзакшин isolation level нь read committed бол, T2-р буцаагдах боломжтой хосуудыг олно уу, тайлбарла.



Тайлбар:

Б. Буцаагдах боломжтой хос утгууд: (70, 30), (30, 20), (30, 30);

□ T2 транзакшин нь T1 транзакшины read uncommitted тусгаарлалтын түвшинд ямар нэг байдлаар нөлөөлж чадахгүй учир нь T2 нь зөвхөн уншилт хийдэг тран юм. Харин T1 нь T2 – ийн read committed буюу бохир уншилтаас бусдын зөвшөөрдөг тусгаарлалтын түвшинд нөлөөлж, бохир уншилт үүсгэх боломжтой. Боломжит хувилбарууд:

□ **T1;T2** – T1 тран – ыг бүрэн ажиллаж дууссаны дараа T2 уншилт хийх буюу сериал байдлаар ажиллах тул ямар нэг асуулдал үүсэхгүй. Буцах утга: (70, 30)

□ **T2;T1** – Буцах утга: (30, 20)

□ **T1:T2 = S1;S3;S4;S2** – T2 тран-ны S3, S4 нь баталгаажаагүй өгөгдөл уншиж, бохир уншилт үүсэх тул ингэж ажиллах боломжгүй. Учир нь read committed тусгаарлалтын түвшин нь бохир уншилтыг зөвшөөрдөггүй.

□ **T2:T1 = S3;S1;S2;S4** – T2 транд non-repeatable read үүснэ. Учир S3, S4 нь нэг хүснэгтийн ижил багана дээрээс уншилт хийж байгаа тул голоор орж тухайн багана дээр өөрчлөлт хийхэд non-repeatable read үүсэх юм. Мөн голоор insert хийснээр phantom tuple үүснэ. Гэхдээ эдгээрийг read committed тусгаарлалтын түвшин зөвшөөрдөг бөгөөд T2-д бохир уншилт үүсэхгүй тул ингэж ажиллах боломжтой. Буцаах утга: (30, 30)

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

4. (5оноо)Бүхэл тоон хүснэгтR(x) байв. Эхлэх үед R дахьнийт тооны нийлбэр 1000, бөгөөд 10, 20, 30 гэсэн утгууд R байдаггүй байг. Доорх гурван оюутны транзакшин зэрэг ажиллав. Бүгд REPEATABLE READ тусгаарлалтын түвшинтэй байв. Амарын тран-ы буцааж болох бүх утгуудыг олж дэлгэрэнгүй тайлбарла (тайлбар ойлгомжгүй, дутуу бол оноо хасагдана).

Амарын транзакшин доорх байдалтай байв.

```
select sum(x) from R;  
commit;
```

Билгүүний транзакшиныг доор харуулав.

```
insert into R values (10);  
insert into R values (20);  
insert into R values(30);  
commit;
```

Заяагийн транзакшиныг доор харуулав.

```
delete from R where x=30;  
delete from R where x=20;  
commit;
```



Тайлбар: Буцаах боломжтой бүх утгууд => 1000, 1060, 1010, 1040

Эхний ээлжинд сериал байдлаар ажиллах үед Амарийн траны буцах утгуудыг олъё. Сериалаар ажиллах үед ямар нэг бохир уншилт үүсэх боломжгүй юм. Үүнд:

Амар; Билгүүн; Заяа == Заяа; Амар; Билгүүн => 1000

Билгүүн; Амар; Заяа == Заяа; Билгүүн; Амар => 1060

Билгүүн; Заяа; Амар == 1010

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

Амарын Транзакшин:

```
select sum(x) from R;  
commit;
```

Билгүүний
Транзакшин:

```
insert into R values (10);  
insert into R values (20);  
insert into R values (30);  
commit;
```

Заяагийн Транзакшин:

```
delete from R where x=30;  
delete from R where x=20;  
commit;
```



Тайлбар: REPEATABLE READ тусгаарлалтын түвшин нь бохир уншилт болон non-repeatable read – ийг зөвшөөрдөггүй. Харин зөвхөн phantoms tuple – ийг зөвшөөрдөг. Мөн Амарын тран нь зөвхөн уншилт хийж байгаа учраас Билгүүн болон Заяа тран-д ямар нэг асуудал үүсгэх боломжгүй. Харин Билгүүн болон Заяагийн тран нь Амарын тран – д ямар нэг асуудал(бохир уншилт болон non-repeatable read) үүсгэх магадлалтай юм. Мөн билгүүний тран нь зөвхөн бичилт хийнэ, харин Заяагийн тран нь зөвхөн устгал хийх юм. Delete нь эхлээд уншаад, дараа нь устгадаг. Тийм учраас Билгүүний тран мөн Заяагийн тран – д мөн бохир уншилт үүсгэх магадлалтай. Амарын тран Билгүүний эсвэл Заяагийн тран-ы statement – үүдийн голоор орж ажиллах боломжгүй учир нь тэгэх үед Амарын тран-д бохир уншилт үүснэ. Дараагийн нэг тохиолдол нь Заяагийн 2 delete – ийн голоор Амарын тран бүхлээрээ орж ажиллах юм. Энэ тохиолдолд Заяагийн тран нь бохир уншилт хийхгүй бөгөөд 2 delete statement нь уншилт хийдэг гэдгээс голоор нь insert хийгдэх үед phantoms tuple үүснэ гэхдээ үүнийг repeatable read isolation level зөвшөөрөх тул ингэж ажиллах боломжтой юм. Заяа: Билгүүн; Амар => 1040 олдоно. Эндээс Амарын тран-ы буцаах болох утгууд: 1000, 1010, 1040, 1060 болно.



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

5. Мэдрэгч төхөөрөмжүүдээс ирсэн уншилт (хугацааны тодорхой агшинд мэдрэгч орчиноо унших) бүрийг дараах хүснэгтэд хадгалдаг: Temps(sensorID, time, temp)

Уг хүснэгтийн [sensorID, time] хослол unique ба доорх асуулга өгөгдөв: `Select * from Temps Where sensorID = 'sensor541' and time = '11:15:06'`

Дараах тохиолдуудад дээрх асуулгын хандалт хийх мөрийн тоо (хамгийн муу тохиолдолд)-г олж, тайлбарла. Индекс рүү хандалтыг тоолохгүй. А-зөвхөн sensorID дээр индекстэй (1 оноо); В- sensorID, time баганууд дээр тус тусад нь индекстэй (1 оноо); С-(sensorID, time) багануудын хослол дээр multi-attribute индекстэй. (1 оноо)

Нэмэлт: Хүснэгт Temps дахин давтагдахгүй 70ш sensorID –тай, sensorID бүр яг 20 уншилт хийсэн, хугацааны агшин бүрд 15 уншилт хийсэн байна.

Тайлбар:

А – Асуулгын хандалт хийх мөрийн тоо буюу үйлдэл: 20

❑ SensorID = 'sensor541' буюу тэнцүү байх гэсэн нөхцөл шалгаж байгаа учраас hash tables – ээр ажиллаж, нэг үйлдлийн дотор тухайн sensorID – г олно. Иймд тухайн sensorID – д харгалзах 20 мөрөөр гүйж, time = '11:15:06' нөхцөлийг шалгаж үзнэ. Иймд асуулгын хандалт хийх мөрийн тоо буюу үйлдэл 20 болно.

В – Асуулгын хандалт хийх мөрийн тоо буюу үйлдэл: 15

❑ SensorID = 'sensor541' буюу тэнцүү байх гэсэн нөхцөл шалгаж байгаа учраас hash tables – ээр ажиллаж, нэг үйлдлийн дотор тухайн sensorID – г олно. Мөн үүнээс гадна time = '11:15:06' буюу тэнцүү байх гэсэн нөхцөл шалгаж байгаа учраас hash tables – ээр ажиллаж, нэг үйлдлийн дотор тухайн time – г олно. Иймд тухайн нэг үйлдлээр олсон sensorID – д харгалзах 20 мөрөөр гүйх бөгөөд асуулгын хандалт хийх мөрийн тоо (хамгийн муу тохиолдолд)-г олбол 15 болно.

С – Асуулгын хандалт хийх мөрийн тоо буюу үйлдэл: 1

❑ SensorID = 'sensor541' буюу тэнцүү гэж нөхцөл шалгаж байгаа учраас hash tables – ээр ажиллаж, нэг үйлдлийн дотор тухайн sensorID – г олно. Мөн үүний дараагаар тухайн 20 мөрөөс time = '11:15:06' буюу тэнцүүгээр нөхцөл шалгаж байгаа учраас hash tables – ээр ажиллаж, нэг үйлдлийн дотор тухайн time – г олно. Иймд асуулгын хандалт хийх мөрийн тоо 1 болно.

Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

6. (4оноо) Холбоост өгөгдлийн сангийн хурдыг сайжруулах аргуудын талаар тайлбарлан бичнэ үү.

Хариулт: 1. Индекс ба зэрэгцээ хандалтын оновчлол (Optimizing indexes and concurrency)

- Индекслэл болон зэрэгцээ хандалт өгөгдлийн сангийн гүйцэтгэлийн хурдыг сайжруулах үндсэн механизм.
- Багана, баганууд болон багануудын хослол дээр индекслэл үүсгэж болно.
- Хүснэгтийн бүх мөрөөр гүйдэггүй. Тодорхой нэг болон хэд хэдэн баганад индекслэлийг оновчтой үүсгэснээр хурдыг хожих боломжтой.
- Индекслэл нь тухайн өгөгдлийг дахин өгөгдлийн санд мод бүтэц үүсгэн хадгалж байдаг.
- Өөрчлөлт ихтэй хүснэгтэд тохиромжгүй. Учир тухайн нэг багана өөрчлөлт ороход мөн тухайн баганын индекслэлийн мод бүтцэд өөрчлөлт ордог. Ингэснээр хурд өмнөхөө ч илүү муудах эрсдэлтэй.
- Мөн өөрчлөлт ихтэй өгөгдлийн сангийн тусгаарлалтын түвшинг сулруулах нь өгөгдлийн санд огт байхгүй өгөгдөл унших гэх зэрэг асуудал үүсэх эрсдэлтэй байдаг.
- Индекслэлийн суурь өгөгдлийн бүтэц
 - Balanced trees(B tree, B+ tree)
 - Бүх төрлийн нөхцөл (жиших, тэнцүү, завсар) дээр ажиллана.
 - Хурдыг $\log(N)$ эрэмбээр хождог.
 - Hash tables
 - Зөвхөн тэнцүү байх гэсэн нөхцөл дээр л ажиллана.
 - Нэг үйлдлээр шууд тухайн мөрд очдог.



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

6. (4оноо) Холбоост өгөгдлийн сангийн хурдыг сайжруулах аргуудын талаар тайлбарлан бичнэ үү.

Хариулт: 2. Өгөгдлийн сангийн бүтцийн оновчлол (Optimizing database structure)

- Өгөгдлийн сангийн хурдыг сайжруулах үндсэн механизм.
- Өгөгдлийн сангийн бүтэц/бүрэлдэхүүн, дискэнд хэрхэн хадгалагддаг:
 1. **Өгөгдлийн сан бүр** нэг болон түүнээс олон **файл групп/file group** – тэй байж болно.
 2. **Файл групп/file group** нь өгөгдлийн хэрхэн хаана байршуулахыг зааж өгөх арга зам.
 3. **Файл групп/file group** бүр нэг болон түүнээс олон **файл/file** – тай байна.
 4. **Файл/file** бүр **Extents** – ээс тогтоно.
 5. **Extents** нь page – ээс бүрддэг.
- Файл групп:
 - Файлуудын логик/хийсвэр бүлэглэл юм.
 - Өгөгдлийг өөр өөр байрлалд хадгалах боломжийг олгодог.
 - Файл групп бүр өөр өөр дискэн дээр байрлаж болно.
 - Файл группыг бусдаас хамааралгүйгээр backup хийж болно.
 - Нэг хүснэгтийг нэг файл группт хадгалж болно.
 - Хүснэгтийг нэг файл дээр хадгалж чадахгүй. Өөрөөр хэлбэл шууд файлыг хүснэгттэй холбож чадахгүй. Харин хүснэгтийг нэг файл групп дээр хадгалж, тухайн файл группын файлын тоог зааж өгөх замаар хүснэгтийг файльтай холбож болно.
 - Файл группыг ашиглаж, өгөгдлийг striping буюу хуваалт хийх боломжийг олгодог.
 - Өөрчлөлт их ордог хүснэгтүүдийг өөр хурдтай дискэн дээр хадгалах, бусад дискэнд индекслэл үүсгэх боломжийг олгодог.
- Data file – ийг stripping(RAID 0, 1, ...) хийх замаар мөн хурдыг хожиж болно. Data file internals: Auto Grow and Auto Shrink
- Extent:
 - Page – үүдийг үр ашигтай буюу хурдтайгаар зохицуулахад ашиглагддаг.
- Page: Өгөгдлийг хадгалах үндсэн нэгж (SQL server)
 - Дискийн оролт гаралтын үйлдлүүд page – ийн түвшинд хийгддэг.



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

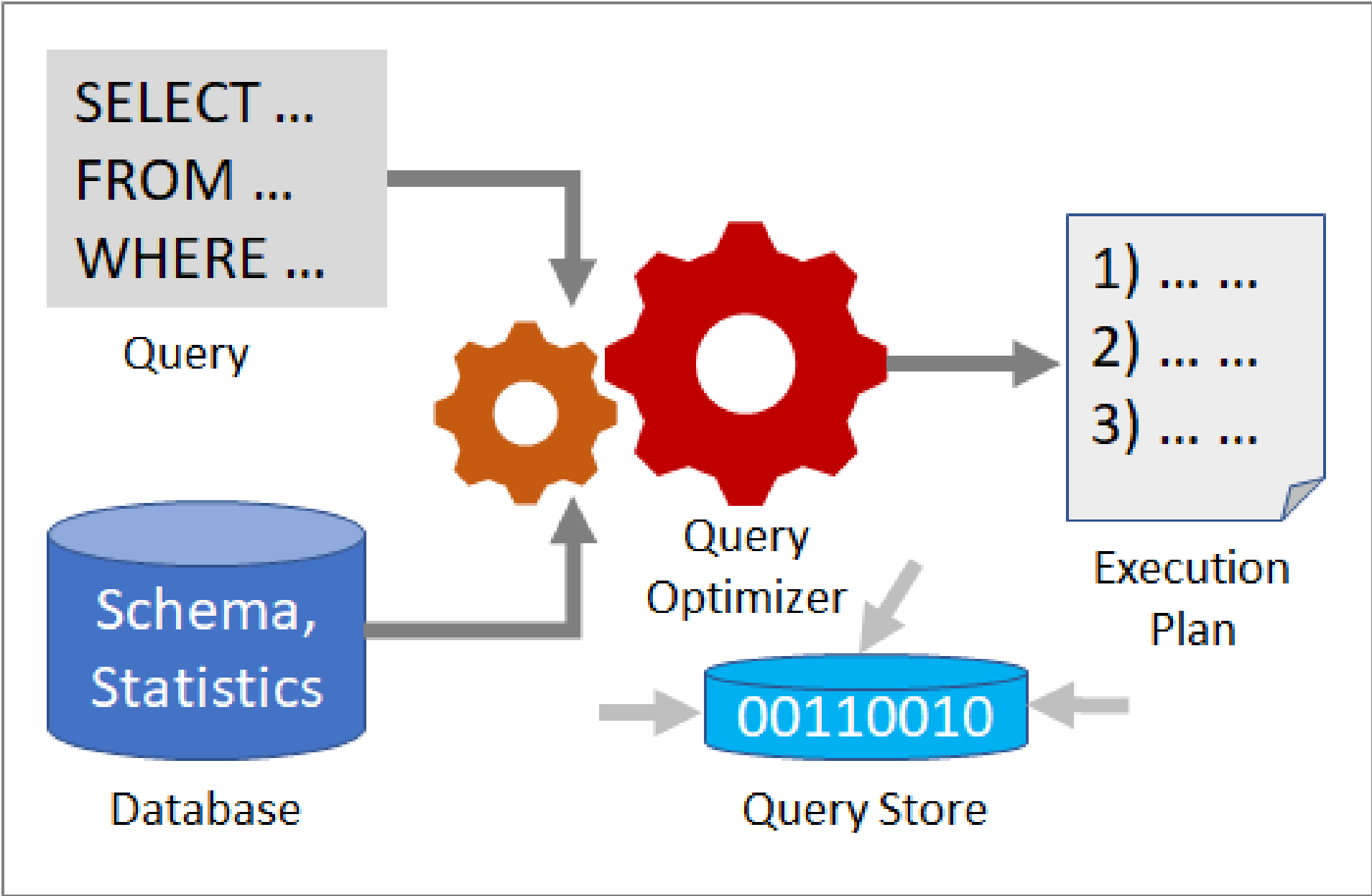
6. (4оноо) Холбоост өгөгдлийн сангийн хурдыг сайжруулах аргуудын талаар тайлбарлан бичнэ үү.

Хариулт: 3. Асуулга буюу query-ийн оновчлол (Optimizing queries)

- Өгөгдлийн сангийн хурдыг сайжруулах үндсэн механизм.
- Бага өгөгдөл дээр ажиллах тусам сайн. Аль болох бага өгөгдөл ажиллэх асуулга бичих чухал.
- Query optimizer нь тухайн асуулга бүрийн хувьд ямар төлөвлөгөөгөөр ажиллахыг шийдэж өгөх ёстой.
 - Үүнийг logical query tree – г ашиглан хийдэг.
 - Төлөвлөгөөнүүдээс хамгийн байж болох шилдэг төлөвлөгөөг олохыг хичээдэг.



STATEMENT CLAUSE	PROCESSING SEQUENCE
SELECT	8
TOP	11
DISTINCT	9
FROM	1
JOIN	3
ON	2
WHERE	4
GROUP BY	5
WITH	6
HAVING	7
ORDER BY	10



Өгөгдлийн сангийн програмчлал – Эцсийн шалгалт

7. (3 оноо) MapReduce програмчлалын загвар, түүний талаар бич.

-давуу талуудыг тооч,

-түүний бүрэлдэхүүн (map, group by key, reduce) –ийг тайлбарла.



Хариулт:

▪ Давуу талууд

- Scalability/Өргөтгөх чадвар
- Cost-effective solution/Зардал хэмнэлттэй шийдэл
- Flexibility/Уян хатан
- Fast/Хурдан
- Security and Authentication/Аюулгүй байдал ба баталгаажуулалт
- Parallel processing/Зэрэгцээ боловсруулалт
- Availability and resilient nature/Бэлэн байдал, өсөх чадвар
- Simple model of programming/Програмчлалын энгийн загвар
- Use in computation problems/Тооцооллын асуудлуудад ашигладаг
- Memory Requirements Less/Санах ойн шаардлага бага

Хариулт:

▪ Бүрэлдэхүүн:

- Map – Ажилчин/worker зангилаа/node бүр map функцийг дотоод өгөгдөл дээр ашиглаж, мөн гаралтыг түр хадгалах санд бичнэ. Мастер зангилаа нь илүүдэл оролтын өгөгдлийн зөвхөн нэг хуулбарын боловсруулалтыг баталгаажуулдаг.
- Group by key/Shuffle - Ажилчин зангилаа нь гаралтын түлхүүр дээр үндэслэн өгөгдлийг дахин хуваарилдаг (produced by the map function), тухайлбал нэг түлхүүрт хамаарах бүх өгөгдөл ижил ажилчин зангилаа дээр байрлуулдаг.
- Reduce - Ажилчин зангилаа гаралтын өгөгдлийн болон түлхүүрийн бүлэг бүрийг зэрэгцээгээр боловсруулдаг.

Эх сурвалж: <https://en.wikipedia.org/wiki/MapReduce>