**1.Да се опишат операциите од реладиона алгебра развиена за релациона бази на податоци и за секоја од нив да се даде пример!**

Релационата алгебра е формален јазик за манипулација со релациона база на податоци. Она што ја прави релационата алгебра моќна и ефикасна е што операциите се засновани на математички операции и функции.

Еве неколку од најосновните операции во релационата алгебра:

1. Проекција (π): Операцијата на проекција го избира само потребниот дел од релацијата, односно само одредени колони. На пример, ако имаме релација "Личности" со колони "Име", "Презиме" и "Адреса", операцијата на проекција π(Име, Презиме) би го вратила само делот од релацијата кој содржи само колоните "Име" и "Презиме".
2. Селекција (σ): Операцијата на селекција го избира делот од релацијата кој задоволува одредени услови, односно филтрира редови според некои критериуми. На пример, ако имаме релација "Студенти" со колони "Име", "Презиме" и "Година\_студирање", операцијата на селекција σ(Година\_студирање > 2) би ги вратила само студентите кои студираат повеќе од две години.
3. Обединение (∪): Операцијата на обединение ги комбинира две релации со иста структура и ги враќа сите редови без дупликати. На пример, ако имаме две релации "Студенти\_група1" и "Студенти\_група2" со истата структура, операцијата на обединение Студенти\_група1 ∪ Студенти\_група2 ќе ги врати сите студенти од двете групи без дупликати.
4. Разлика (-): Операцијата на разлика ги враќа редовите кои се присутни во една релација, но не и во другата. На пример, ако имаме релации
5. "Студенти\_положиле\_испит" и "Студенти\_

**2.Да се направи споредба помеѓу релациона алгебра и релационо сметање!**

Релациона алгебра и релационо сметање се две различни пристапи за манипулација со релациона база на податоци.

Еве неколку точки на споредба помеѓу нив:

1. Математичка основа: Релациона алгебра е формален математички јазик кој се базира на алгебрата за множества и операции како проекции, селекции, обединувања и разлики. Наспроти тоа, релационо сметање е основано на математичката логика и представува практична техника за извршување на операции со релациони табели.
2. Цел: Релационата алгебра се користи за формулирање и изразување на прашања и операции над релациите. Со помош на алгебрата, можеме да дефинираме и применуваме операции за да добиеме резултати според нашите потреби. Релационо сметање, на друга страна, се фокусира на извршување на конкретни операции и прашања врз податоците.
3. Имплементација: Релациона алгебра е апстрактен јазик и не се фокусира на конкретни начини на извршување. Она што ја прави интересна е што операциите можат да се извршуваат на различни начини, како на пример преку SQL (Structured Query Language). Релационо сметање, на друга страна, се однесува на конкретните техники и алгоритми за извршување на операциите во релационите бази на податоци.
4. Флексибилност: Релационата алгебра обезбедува флексибилност и ефикасност во манипулацијата со податоци. Со користење на различни операции и комбинации, можеме да ги дефинираме и изразиме различни прашања и манипулации со податоците. Релационо сметање, исто така, овозможува извршување на различни операции, но се фокусира пок

**3. Кои се основните функции на SQL и кои наредби тие ги содржат?**

SQL е стандарден јазик за управување со релациони бази на податоци. Секој SQL јазик се состои од неколку основни функции и наредби кои овозможуваат креирање, читање, ажурирање и бришење на податоци во базата. Еве некои од најосновните функции и наредби во SQL:

1. SELECT: Користена за читање или избор на податоци од базата. Наредбата SELECT овозможува специфицирање на табели и колони од кои сакате да добиете податоци. Може да се комбинира со други наредби и функции за филтрирање, сортирање и групирање на податоците.
2. INSERT: Користена за вметнување на нови податоци во табелата. Со наредбата INSERT, можете да наведете вредности за колоните во табелата или да вметнете податоци од друга табела.
3. UPDATE: Користена за ажурирање на постоечки записи во табелата. Со наредбата UPDATE, можете да ги промените вредностите на една или повеќе колони во постоечките записи.
4. DELETE: Користена за бришење на записи од табелата. Наредбата DELETE може да се користи за бришење на еден или повеќе записи што ги задоволуваат одредени услови.
5. WHERE: Користена за филтрирање на податоците базирани на одредени услови. WHERE наредбата се користи во комбинација со SELECT, UPDATE и DELETE наредбите за да се зададат услови за избор, ажурирање или бришење на податоци.
6. JOIN: Користена за спојување на податоци од две или повеќе табели базирани на заеднички клучеви. JOIN наредбата ви овозможува да ги комбинирате податоците од поврзани табели во еден резултатен сет.
7. GROUP BY: Користена за групирање на податоците базирани на одредени колони. GROUP BY наредбата се користи за креирање на групи од под

**4.Дадени се реалциите**

**AKTER(AkterID,ImeAK,PrezimeAK,DatRg,DrzavaRag)**

**FILM(FilmID,ImeFilm,GodPrimiera,ImeRez,PrezimeRez,Zanr)**

**GLUMI(AkterID,FilmID,Uloga,Honorar)**

**a)Преку операции од реалациона алгебра да се прикаже име и презиме на актерите кои играле во филмот пред 1999 година и имале хонорар поголем од 50.000 парични единици** .

SELECT Akter.ImeAK, Akter.PrezimeAK

FROM AKTER

JOIN GLUMI ON AKTER.AkterID = GLUMI.AkterID

JOIN FILM ON GLUMI.FilmID = FILM.FilmID

WHERE FILM.GodPrimiera < 1999

AND GLUMI.Honorar > 50000;

**б)Преку операцијија од релациона алгебра да се прикаже име и презиме на актерите родени пред 1950 година кои имале главна улога во филмови чија примена биле пред 1990 година**.

SELECT Akter.ImeAK, Akter.PrezimeAK

FROM AKTER

JOIN GLUMI ON AKTER.AkterID = GLUMI.AkterID

JOIN FILM ON GLUMI.FilmID = FILM.FilmID

WHERE YEAR(AKTER.DatRg) < 1950

AND FILM.GodPrimiera < 1990

AND GLUMI.Uloga = 'Главна улога';

**в)Користејки SQL наредби за секој актер да се прикаже неговото име и презиме висината на максималниот хонорар што тој го добил за фулмот чија примена се одржала помеќу 1990 и 1995 година**

За да се прикаже името и презимето на секој актер, заедно со висината на неговиот максимален хонорар за филмови од 1990 до 1995 година, може да се користат следниве SQL наредби:

SELECT AKTER.ImeAK, AKTER.PrezimeAK, MAX(GLUMI.Honorar) AS MaksimalenHonorar

FROM AKTER

JOIN GLUMI ON AKTER.AkterID = GLUMI.AkterID

JOIN FILM ON GLUMI.FilmID = FILM.FilmID

WHERE FILM.GodPrimiera BETWEEN '1990-01-01' AND '1995-12-31'

GROUP BY AKTER.AkterID, AKTER.ImeAK, AKTER.PrezimeAK;

Во оваа наредба, ги спојуваме табелите **AKTER**, **GLUMI** и **FILM** со користење на JOIN операциите базирани на соодветните клучеви. Потоа, поставуваме услови во WHERE класата за да ги филтрираме филмовите кои имале примена помеѓу 1990 и 1995 година.

GROUP BY класата се користи за да се групираат податоците според **AkterID**, **ImeAK** и **PrezimeAK**, и MAX функцијата се користи за да се избере максималниот хонорар за секој актер.

Оваа наредба ќе резултира со прикажување на името и презимето на секој актер, заедно со неговиот максимален хонорар за филмови чија примена се одржала помеѓу 1990 и 1995 година.

**г)Користејќи SQL наредби за секој актер од филмовите да се прикаже минималниот и максималниот исплатен хонорар , вкупниот број на актери кои глумеле во секој од филмовите и вкупната сума исплатена за сите актери**

За да се прикажат минималниот и максималниот исплатен хонорар, вкупниот број на актери кои глумеле во секој од филмовите и вкупната сума исплатена за сите актери во филмовите, може да се користат следниве SQL наредби:

SELECT FILM.ImeFilm, MIN(GLUMI.Honorar) AS MinimalenHonorar, MAX(GLUMI.Honorar) AS MaksimalenHonorar, COUNT(DISTINCT GLUMI.AkterID) AS BrojNaAteri, SUM(GLUMI.Honorar) AS VkupnoIspaltno

FROM FILM

JOIN GLUMI ON FILM.FilmID = GLUMI.FilmID

GROUP BY FILM.FilmID, FILM.ImeFilm;

Во оваа наредба, ги спојуваме табелите FILM и GLUMI со користење на JOIN операциите базирани на соодветните клучеви. Потоа, преку GROUP BY клазата ги групираме податоците според FilmID и ImeFilm.

Во SELECT наредбата, користиме MIN функцијата за да го најдеме минималниот хонорар, MAX функцијата за да го најдеме максималниот хонорар, COUNT(DISTINCT) функцијата за да го пресметаме вкупниот број на различни актери во секој филм, и SUM функцијата за да го пресметаме вкупниот исплатен хонорар за сите актери.Оваа наредба ќе резултира со прикажување на името на филмот, минималниот и максималниот исплатен хонорар, вкупниот број на актери кои глумеле во филмот и вкупната сума исплатена за сите актери во филмовите.

**д)Користејќи SQL наредби,за секоја актер да се прикаже неговото име и презиме и името на филмот во кој го глумел улогата која ја имал и хонорарот ко го заботил**

За да се прикаже името и презимето на секој актер, името на филмот во кој го глумел и улогата која ја имал, заедно со хонорарот кој го заработил, може да се користат следниве SQL наредби:

SELECT AKTER.ImeAK, AKTER.PrezimeAK, FILM.ImeFilm, GLUMI.Uloga, GLUMI.Honorar

FROM AKTER

JOIN GLUMI ON AKTER.AkterID = GLUMI.AkterID

JOIN FILM ON GLUMI.FilmID = FILM.FilmID;

Во оваа наредба, ги спојуваме табелите **AKTER**, **GLUMI** и **FILM** со користење на JOIN операциите базирани на соодветните клучеви.

Со SELECT наредбата, ги избираме полињата **ImeAK** и **PrezimeAK** од табелата **AKTER**, полето **ImeFilm** од табелата **FILM**, полето **Uloga** од табелата **GLUMI**, и полето **Honorar** од табелата **GLUMI**.

Оваа наредба ќе резултира со прикажување на името и презимето на секој актер, името на филмот во кој го глумел, улогата која ја имал и хонорарот кој го заработил за секој запис во табелата **GLUMI**.

**ѓ)Користејки SQL наредби да се излистаат имињата и презимињата на актерите кои се од САД и и имаат главна машка улога како и името на фимот**

За да се излистаат имињата и презимињата на актерите кои се од САД, имаат главна машка улога и да се прикаже името на филмот, може да се користат следниве SQL наредби:

SELECT AKTER.ImeAK, AKTER.PrezimeAK, FILM.ImeFilm

FROM AKTER

JOIN GLUMI ON AKTER.AkterID = GLUMI.AkterID

JOIN FILM ON GLUMI.FilmID = FILM.FilmID

WHERE AKTER.DrzavaRag = 'САД' AND GLUMI.Uloga = 'главна машка';

Во оваа наредба, ги спојуваме табелите AKTER, GLUMI и FILM со користење на JOIN операциите базирани на соодветните клучеви.

Во WHERE клазата поставуваме услови за да филтрираме актерите кои се од САД (DrzavaRag = 'САД') и имаат главна машка улога (Uloga = 'главна машка').

Оваа наредба ќе резултира со прикажување на имињата и презимињата на актерите кои се од САД, имаат главна машка улога, и името на филмот во кој го имале таа улога, според записите во табелите AKTER, GLUMI и FILM.

**5.Кои аномалични се јавуваат при одржување(ажурирање) на релационата база податоци(РБП) ?**

Основни операции за одржување(ажурирање) на РБП:

* Додавање на нова торка во ралцијата
* Отстранување на некоја торка од реалцијата
* Промена на вредноста на некој атрибут во релацијата

**6.Дадена е релација**

**PRODAZBA\_AVTOMOBIL(SifraAVR,DatProdazba,SifProdLice,Provizija) Претпоставуваме дека е еден модел на автомобил може да го продат продавачи(примарниот клуч е SifraAvt,SifProdLice).Други зависности се :DatProdazba-Popust,SifprodLice-Provizija Дали релацијата во 1НФ,2НФ ИЛИ 3НФ.Како целосно би се нормализирала?**

Според информациите што ги дадовте, релацијата "PRODAZBA\_AVTOMOBIL" не е во трета нормална форма (3NF). За да ги нормализираме релацијата, треба да ги идентификуваме функционалните зависности и да ги разделиме во посебни релации.

Според претпоставките, имаме следните функционални зависности:

1. SifraAVR -> DatProdazba, Popust

2. SifProdLice -> Provizija

За да ја нормализираме релацијата, треба да создадеме две нови релации:

1. Релација "AVTOMOBIL" со атрибути SifraAVR, DatProdazba и Popust.

2. Релација "PRODAVACH" со атрибути SifProdLice и Provizija.

Првата нормална форма (1NF) веќе е задоволена бидејќи немаме повторување на групи атрибути.

Во втората нормална форма (2NF), треба да се осигураме дека секој атрибут зависи целосно од клучот. Во нашиот случај, клучевите се SifraAVR и SifProdLice, и сите атрибути зависат од нив.

Во третата нормална форма (3NF), треба да се отстрани зависноста на неклучните атрибути помеѓу нив. Во овај случај, немаме такви зависности.

Така, нормализираната релација би изгледала:

Релација "AVTOMOBIL" (SifraAVR, DatProdazba, Popust)

Релација "PRODAVACH" (SifProdLice, Provizija)

Оваа нормализација ги задоволува претпоставените зависности и овозможува поефикасно управување со податоците.

**6.2 Како се менува големината на нормализираните, а како на ненормализираните дадотеки?**

Големината на нормализираните и ненормализираните датотеки може да се разликува во зависност од нивната структура и организација. Еве неколку аспекти кои можат да влијаат на големината на датотеките:

1. Нормализација: Нормализираните датотеки често можат да бидат помали од ненормализираните, особено кога имаат многу повторувања на податоци. Нормализацијата го разделува податочниот модел на посебни релации, што може да резултира во повеќе мали датотеки наместо една голема.
2. Дупликати: Ненормализираните датотеки можат да содржат повторување на податоци во истата датотека, што може да ги зголеми нивните големини. Нормализацијата обично ги отстранува овие дупликати преку разделување на податоците во посебни релации, што резултира во помали датотеки.
3. Денормализација: Денормализацијата е противоположен процес на нормализацијата и може да резултира во поголеми датотеки.

4. Индексирање: Индексите се користат за брз пристап до податоците и може да зголемат големината на датотеките. Ненормализираните датотеки можат да содржат индекси за брз пристап до релевантните податоци. Со нормализација, индексите може да бидат поедноставени или сведени на минимум, што може да резултира во помали датотеки.Ова се само некои фактори кои можат да влијаат на големината на датотеките. Различните фактори и изборите што ги правите во дизајнот

**7. Дадени се релациите**

**VRABOTEN(VrabotenID,ImeVr,DatRag,Kategorija,TelVr,AdresaVr)**

**VOZILO(VoziloSif,Marka,Model,GodProizvodstvo)**

**UPRAVUVA(VrabotenID,VoziloSif,Relacija,PaTrosok)**

**a)Преку операциите од релационата алгебра да се прикаже име и презиме на бравотените кои управубале возило приведени пред 1999 година на релација Битола-Скопје**

SELECT ImeVr, PrezimeVr

FROM VRABOTEN

JOIN UPRAVUVA ON VRABOTEN.VrabotenID = UPRAVUVA.VrabotenID

JOIN VOZILO ON UPRAVUVA.VoziloSif = VOZILO.VoziloSif

WHERE Relacija = 'Битола-Скопје'

AND GodProizvodstvo < 1999;

Оваа SQL наредба ќе прикаже имињата и презимињата на вработените кои управуваат возила приведени пред 1999 година на релацијата Битола-Скопје, со користење на релациите VRABOTEN, UPRAVUVA и VOZILO.

**б)Преку операциите од релационата алгебра да се прикаже име и презиме на вработени родени пред 1960 година кои управувале возило Golf од моделот Polo произведено пред 2000 година**.

SELECT ImeVr, PrezimeVr

FROM VRABOTEN

JOIN UPRAVUVA ON VRABOTEN.VrabotenID = UPRAVUVA.VrabotenID

JOIN VOZILO ON UPRAVUVA.VoziloSif = VOZILO.VoziloSif

WHERE YEAR(DatRag) < 1960

AND Model = 'Polo'

AND Marka = 'Golf'

AND GodProizvodstvo < 2000;

Оваа SQL наредба ќе прикаже имињата и презимињата на вработените родени пред 1960 година кои управуваат возило Golf од моделот Polo произведено пред 2000 година, со користење на релациите VRABOTEN, UPRAVUVA и VOZILO.

**в)Користејќи SQL наредби за секој вработен да се прикаже име и презиме и висината на патниот трошок што го добил управувајќи возило поизведено помеѓу 1994 и 1999 година**.

SELECT VRABOTEN.ImeVr, VRABOTEN.PrezimeVr, UPRAVUVA.PatTrosok

FROM VRABOTEN

JOIN UPRAVUVA ON VRABOTEN.VrabotenID = UPRAVUVA.VrabotenID

JOIN VOZILO ON UPRAVUVA.VoziloSif = VOZILO.VoziloSif

WHERE VOZILO.GodProizvodstvo BETWEEN 1994 AND 1999;

Оваа SQL наредба ќе прикаже имињата и презимињата на секој вработен, заедно со нивната висина на патниот трошок (PatTrosok), која го добил управувајќи возило што е произведено помеѓу 1994 и 1999 година. Ова се постигнува со користење на релациите VRABOTEN, UPRAVUVA и VOZILO и со спојување на нивните соодветни полиња.

**г)Користејќи SQL наредби за секое од возило да се прикаже минималниот и максималниот примарен трошок вкупниот број на вработени кои го управувале секое од возилота и вкупната сума успратени патни прошоци**

SELECT

VOZILO.VoziloSif,

MIN(UPRAVUVA.PatTrosok) AS MinPrimenTrosok,

MAX(UPRAVUVA.PatTrosok) AS MaxPrimenTrosok,

COUNT(DISTINCT UPRAVUVA.VrabotenID) AS BrojNaVraboteni,

SUM(UPRAVUVA.PatTrosok) AS VkupenPatniProshok

FROM

VOZILO

JOIN

UPRAVUVA ON VOZILO.VoziloSif = UPRAVUVA.VoziloSif

GROUP BY

VOZILO.VoziloSif;

Оваа SQL наредба ќе прикаже информации за секое од возилата. За секое возило, ќе се прикаже минималниот (MinPrimenTrosok) и максималниот (MaxPrimenTrosok) примарен трошок, вкупниот број на вработени (BrojNaVraboteni) кои го управувале тоа возило и вкупната сума на испратени патни прошоци (VkupenPatniProshok). Ова се постигнува со користење на релациите VRABOTEN, UPRAVUVA и VOZILO и со спојување на нивните соодветни полиња. Со помош на функциите **MIN**, **MAX**, **COUNT** и **SUM**, се извршуваат агрегатни операции за добивање на соодветните информации за секое возило. Резултатот се групира по полето VoziloSif со помош на **GROUP BY** наредбата.

**д) Користејќи SQL наредби за секој вработен да се прикаже неговото име и презиме името на возилото што тој го управувал релацијата на која возел и патниот трошок што му бил исплатен**

SELECT

VRABOTEN.ImeVr,

VRABOTEN.PrezimeVr,

VOZILO.Marka,

VOZILO.Model,

UPRAVUVA.Relacija,

UPRAVUVA.PatTrosok

FROM

VRABOTEN

JOIN

UPRAVUVA ON VRABOTEN.VrabotenID = UPRAVUVA.VrabotenID

JOIN

VOZILO ON UPRAVUVA.VoziloSif = VOZILO.VoziloSif;

Оваа SQL наредба ќе прикаже информации за секој вработен. За секој вработен, ќе се прикаже неговото име (ImeVr) и презиме (PrezimeVr), името на возилото (Marka и Model) што тој го управувал, релацијата (Relacija) на која возел и патниот трошок (PatTrosok) што му бил исплатен. Ова се постигнува со користење на релациите VRABOTEN, UPRAVUVA и VOZILO и со спојување на нивните соодветни полиња. Резултатот ќе биде список со информации за секој вработен и нивните управувања на возила.

**ѓ)Користејќи SQL наредби да се излистаат имињата и презимињата на вработените кои имаат возачка категорија Ц или Д и управуваат зовило од марка Mercedes на релацијата Битола-Прилеп!**

SELECT VRABOTEN.ImeVr, VRABOTEN.PrezimeVr

FROM VRABOTEN

JOIN UPRAVUVA ON VRABOTEN.VrabotenID = UPRAVUVA.VrabotenID

JOIN VOZILO ON UPRAVUVA.VoziloSif = VOZILO.VoziloSif

WHERE VRABOTEN.Kategorija IN ('Ц', 'Д')

AND VOZILO.Marka = 'Mercedes'

AND UPRAVUVA.Relacija = 'Битола-Прилеп';