

## 1.UVOD

### 1) Kaj je SUPB?

Sistem za upravljanje s podatkovnimi zbirkami ali podatkovnimi bazami, je množica programov, namenjenih ustvarjanju, vzdrževanju, nadzoru dostopa do podatkov v podatkovnih zbirkah.

### 2) Datoteke proti SUPB?

Datoteko je treba poskenirat, prebrati vsak zapis. Z eno datoteko se da hitro, z več datotekami imamo ogromno dela. Kadar izpade elektrika, lahko izgubimo vse podatke. SUPB nudi zaščito pred izpadom sistema, kontrolira dostop uporabnikov. Zaščiti podatke pred sistemskimi napakami.

### 3) Kaj je podatkovni model?

Logični načrt podatkovne baze. Zbirka konceptualnih gradnikov za opis podatkov. Povezana zbirka konceptov, namenjenih opisovanju in manipulaciji s podatki. Poznamo relacijski( tabela : stolpci-kaj narediti, vrstice-več možnosti, kaj narediti)

### 4) Kaj je podatkovna neodvisnost?

Ločimo dve vrsti. 1 Fizična neodvisnost- Na kakšen način so podatki zapisani. Dobimo podatek in želimo shraniti podatek brez spremembe strukture podatka. 2. Logična neodvisnost- Želimo naredimo logično spremembo, da se obstoječi podatek zlige z novim in ne uporabniku ne bi bilo treba preveč razmišljati o tem.

### 5) Opišite stopnje abstrakcije v SUPB.

Nivoji abstrakcije:

- 1.Pogledi:Kako uporabnik vidi podatke(npr. direktor vidi podatke o svoji firmi)
- 2.Konceptualna shema: imamo atribut(ime) in tip atributa(string), definira logično strukturo.
- 3.Fizična shema: podatke vidimo tako kot so predstavljeni, opisuje uporabljeni neurejene datoteke in indeksne datoteke.

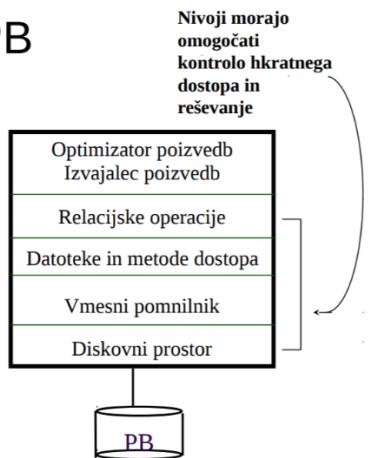
### 6) Kaj je transakcija?

Atomarna(se izvedejo vse ali nobena) sekvenca akcij za branje in pisanje. Vsaka transakcija, ki se izvrši celotno mora pustiti PB v konsistentnem stanju(omejitve nad podatki niso kršene). Je operacija ali niz operacij, s strani enega uporabnika ozr. upr. Programa, ki pišejo ali berejo v podatkovno bazo. Lahko se zaključi uspešno- COMMIT, ali neuspešno ABORT. (NPR pišejo se obresti, medtem ko se še računa se akcija še ni zaključila)

## 7) Opišite strukturo SUPB.

### Struktura SUPB

- Tipični SUPB ima nivojsko arhitekturo.
- Slika ne prikazuje komponent za kontrolo hkratnega dostopa in recovery.
- To je samo ena izmed bolj pogostih arhitektur; vsak sistem ima svoje variacije.



OPB-Uvod

## 2. Relacijski podatkovni model

Nterica = relacija

Null vrednost- neznana npr. ocena ni bila vnešena

### 1) Kakšen je model relacijske baze podatkov?

Ima dva dela.

Instanca : tabela, ki ima vrstice in stolpce. #vrstic = kardinalnost, #stolpcov = stopnja. – Shema : določa ime relacije ter imena in tipe vseh stolpcev.

### 2) Kaj je omejitev integritete?

Osnovno pravilo. Za vsak atribut določimo lastnosti, ki se preverjajo ob spremnjanju relacij. Manj možnosti za napake pri vnosu. 1.primarni ključ., tuji ključ, superključ, kandidatni ključ...

Osnovane so na pomenu okolja(logično razmišljanje).

### 3) Opišite pojem primarnega ključa.

Množica atributov, ki enolično določa vsako vrstico relacijo. Superključ-če primarnemu ključu dodamo atribut{sid, pid}. Kandidatni ključ-imamo več identifikatorjev ključev(dva identifikatorja npr identifikator vpisne št in emše), zato izberemo enega in ga postavimo kot primarni ključ. Ostali atributi pravimo da so unikatni.

### 4) Kaj je tuji ključ?

Definirajo strukturo relacijo. Iz neke tabele pokažemo na primarni ključ druge relacije. Množica atributov neke relacije, ki referencira zapise druge relacije. Izbrana množica atributov mora ustrezati primarnem ključu druge relacije. Neke vrste “logični kazalec”.

### 5) Kaj je referenčna integriteta?

Identifikator študenta, ki ne obstaja. Ni visečih referenc. HTML in internetni modeli nimajo referenčnih integritet. Zagotavlja! : Pobrišejo se pripadajoči zapisi v relaciji, ne dovoli brisanje zapisa relacije (NO ACTION) npr. Studenti, na katerega se referencirajo zapisi v tabeli (CASCADE) npr. Vpis . Vrednost izbrisanih npr.sid se v tabeli VPIS postavi na privzeto vrednost. Izbrisani zapisi se postavijo na null vrednost(SET NULL/SET DEFAULTneznana ali nenavedena).

**Referenčna integriteta** je lastnost podatkov, ki navaja, da so vsi njeni sklici veljavni. V kontekstu [relacijskih baz podatkov](#) zahteva, da mora referenčna vrednost obstajati , če se vrednost enega atributa (stolpca) [relacije](#) (tabele) sklicuje na vrednost drugega atributa (bodisi v isti ali drugi relaciji). [\[1\]](#)

Da referenčna integriteta ostane v relacijski bazi podatkov, lahko kateri koli stolpec v osnovni tabeli, ki je razglašen za tudi ključ, vsebuje samo ničelne vrednosti ali vrednosti iz primarnega ključa nadrejene tabele ali ključa kandidata. <sup>[2]</sup> Z drugimi besedami, kadar se uporablja vrednost tujega ključa, se mora v nadrejeni tabeli sklicevati na veljavni obstoječi primarni ključ.

## 6) Opišite relacijsko algebro.

Bolj proceduralen jezik, uporaben za predstavitev plana izvajanja poizvedb.

Osnovne operacije:

- Selekcija ( $\delta$ ) Izbere podmnožico n-teric iz relacije.
- Projekcija ( $\Pi$ ) Izbere določene stolpce relacije.
- Produkt ( $\times$ ) Omogoča kombiniranje dveh relacij.
- Razlika ( $-$ ) N-terice iz prve in ne iz druge relacije.
- Unija ( $\cup$ ) N-terice iz obeh relacij.

Dodatne operacije – Presek, Stik,

**Deljenje**— Poišči vse mornarje, ki so rezervirali vse ladje.

Naj ima A dva atributa x in y; B pa samo en atribut y:  $-A/B = -A/B$  vsebuje vse n-terice x (mornarji) tako da za vsako n-terico y (ladja) v B, obstaja n-terica xy v A.

Primer deljenja A/B

sno	pno				
s1	p1				
s1	p2	pno			
s1	p3	p2			
s1	p4		p4		
s2	p1			p1	
s2	p2			p2	
s3	p2			p4	
s4	p2				
s4	p4				

*A*

pno
p2

*B1*

pno
p2

*B2*

pno
p4

*B3*

sno
s1

*A/B1*

sno
s2

*A/B2*

sno
s3

*A/B3*

OPB, Algebra

Preimenovanje. – Niso nujne, so pa ZELO (!) koristne. Vsaka operacija vrne relacijo kot rezultat. – Operacije se lahko sestavljajo – funkcionalni jezik.

## 7) Opišite relacijski račun.

Je deklarativen jezik, uporabniki definirajo vprašanja tako, da zapišejo kaj želijo in kako naj sistem poišče rezultat.

Relacijska kompletnost- Algebra in varni izrazi relacijskega računa imajo enako izrazno moč.

Povpraševalni jezik (npr. SQL) lahko izrazi vsako vprašanje, ki ga lahko izrazimo z relacijsko algebro ali računom

Niso varni: npr. {  $S \mid \neg (S \in \text{Mornarji})$  }

- Dva jezika: N-terični relacijski račun (TRC) in Domenski relacijski račun (DRC).
  - Izrazi vsebujejo spremenljivke, konstante, primerjalne operacije, logične operacije in kvantifikatorje.
    - TRC: Spremenljivke so omejene na n-terice.
    - DRC: Spremenljivke so omejene na domene atributov.
- Atomični izrazi:  $<$ ,  $>$ ,  $=$ ,  $p \vee q$ ,  $p \text{ in } q$
- Uporaba kvantifikatorjev:  $\exists X$  in  $\forall X$
- TRC in DRC so podmnožice predikatnega računa.
  - Izraze teh jezikov imenujemo formule. N-terico, ki je odgovor na vprašanje dobimo tako, da prostim spremenljivkam priredimo konstante tako, da je vrednost formule enaka true.

## 8) Predstavite osnovno skladnjo poizvedbenega jezika SQL.

Deljenje: NOT EXIST (VSE LADJE- except VSE REZERVACIJE VSEH MORNARJEV)

```
SELECT [DISTINCT] seznamizbire  
FROM seznam-relacij  
WHERE pogoj-izbire
```

## 9) Opisite koncept pridruževanja v SQL.

JOIN- če želimo v rezultatu kombinerati stolpce iz različnih tabel, izvedemo JOIN. V FROM navedemo katere tabele želimo povezati, v sklopu WHERE tabele povežemo preko stolpcev. Pri sklicevanju na stolpce, definiramo iz katere tabele je stolpec, ker imamo lahko stolpce enakih imen.

## 10) Opisite funkcije agregacije v SQL.

V celotni relaciji štejemo ozr. štejemo nad celotno n-terico

COUNT- preštejemo št. Stolpcev, \*-odstrani null vrednosti

SUM-seštevek vred. V določenem stolpcu !uporaba distinct brez enakih vrednosti

AVG-povprečje vrednosti posameznega stolpca !uporaba distinct brez enakih vrednosti

MIN, MAX, najmanjša vrednost, največja vrednost (z vgnezdeno zanko) !distinct nima vpliva

GRUPIRANJE- grupiranje nad n-tericami po enem atributu.

**GROUP BY:**

Skupine po ocenah,

Poisci starost najmlajšega mornarja v skupinah, ki pripadajo ocenam in vsebujejo vsaj dva takšna mornarja. Zanimajo nas samo mornarji, ki so stari več ali enako 18.

```
SELECT M.ocena, MIN(M.star)
AS minstar
FROM Mornarji M
WHERE M.star >= 18
GROUP BY M.ocena
HAVING COUNT(*) > 1
```

Odgovor:

ocena	minstar
3	25.5
7	35.0
8	25.5

OPB, SQL

Sailors instance:

mid	mime	ocena	star
22	novak	7	45.0
29	pevec	1	33.0
31	kranjc	8	55.5
32	andrej	8	25.5
58	petelin	10	35.0
64	tom	7	35.0
71	kos	10	16.0
74	tom	9	35.0
85	miha	3	25.5
95	janez	3	63.5
96	egon	3	25.5

Poisci starost najmlajšega mornarja v skupinah, ki pripadajo ocenam in vsebujejo vsaj dva takšna mornarja. Zanimajo nas samo mornarji, ki so stari več ali enako 18.

ocena	star
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
3	25.5
3	63.5
3	25.5
10	35.0

OPB, SQL

ocena	star
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

ocena	minstar
3	25.5
7	35.0
8	25.5

- Izbere pravilne n terice
- Uredi po velikosti ocene
- Filtriramo: Izbere oceno vsaj 2 ali več in min starost teh izbranih ocen

## HAVING:

Pogoj nad skupinami npr. vsaj dva mornarja  
ANY-vsaj en da ima tak pogoj, EVERY-vsi stem pogojem.

## 11) Opišite poizvedbeni jezik QBE.

Osnovan na domenskem relacijskem računu!, podoben SQL- grupiranje na enak princip. Enostaven za enostavna vprašanja – Neroden za kompleksna vprašanja.

P. za izpis ; \_N za spremenljivko ; AO naraščajoči vrsti red prvi izpis AO(1), drugi izpis AO(2) ;

### And/Or vprašanja

- Imena mornarjev, ki so starejši od 30 ali mlajši od 20:

Sailors	sid	sname	rating	age
		P.		> 30
		P.		< 20

- Imena mornarjev, ki so mlajši od 30 in starejši od 20:

Sailors	sid	sname	rating	age
	_Id	P.		< 30
	_Id	P.		> 20

- Imena mornarjev, ki so mlajši od 30 in imajo rating > 4:

Sailors	sid	sname	rating	age
	_Id	P.	> 4	< 30

OPB, QBE

### Duplikati

- Ena vrstica vsebuje P: duplikati niso izločeni privzeto; izločenje dosežemo z UNQ.

Sailors	sid	sname	rating	age
UNQ.		P.		< 30

- Več vrstic z P: duplikati so izločeni privzeto! lahko preprečimo izločenje duplikatov z ALL.

Sailors	sid	sname	rating	age
ALL.	_Id	P.		< 30
	_Id	P.		> 20

OPB, QBE

## Stiki

- Barve ladij z imenom "Interlake", ki so jih rezervirali mornarji za dan 8/24/96 in so starejši od 25 :

Sailors	sid	sname	rating	age
	_Id			> 25
Reserves	sid	bid	day	
	_Id	_B	'8/24/96'	
Boats	bid	bname	color	
	_B	'Interlake'	P.	

OPB, QBE

## Stik

- Imena in starost mornarjev, ki so rezervirali ladjo, ki jo je rezerviral mornar z sid = 22:

Sailors	sid	sname	rating	age
	_Id	P.		P.
Reserves	sid	bid	day	
	22	_B		
	_Id	_B		

OPB, QBE

## Agregacija

- QBE podpira **AVG, COUNT, MIN, MAX, SUM**
  - Nobena operacija ne eliminira duplikatov, razen COUNT
  - Imamo tudi **AVG.UNQ** ...

Sailors	sid	sname	rating	age
	G.	G.P.AO	_A	P.AVG._A

- Stolci z G. so *group-by* polja; vse n-terice iz skupine imajo isto vrednost izbranih atributov.
  - Opcija uporaba .AO uredi odgovore.
  - Vsek stolpec z P. mora imeti G. ali agregacijsko operacijo.**

OPB, QBE

## Ločeni pogoji

- Ločene pogoje uporabimo za izražanje zvez med več relacijami oz. stolpci različnih relacij.
- Lahko izražamo pogoje za skupino n-teric podobno stavku HAVING v SQL:

Sailors	sid	sname	rating	age	CONDITIONS
				G.P.	_A

♦ Izražanje pogojev z AND in OR:

Sailors	sid	sname	rating	age	CONDITIONS
	P.			_A	20 < _A AND _A < 30

OPB, QBE

Grupiranje po sname in rating. Izpišemo rating in uredimo rezultat po naraščajočem vrstnem redu po oceni. Sprem A predstavlja starost mornarja in za določene skupine imena mornarja in rating mornarja izpišemo povprečje.

## Poišči mornarje, ki so rezervirali vse ladje

- Vprašanje z deljenjem
- Potrebovali bomo agregacijske operacije (in update).

Sailors	sid	sname	rating	age
	P.G._Id			
Reserves	sid	bid	day	CONDITIONS
	_Id	_B1		COUNT._B1=COUNT._B2
Boats	bid	bname	color	
	_B2			

- Kako lahko spremenimo vprašanje, da izpiše imena mornarjev, ki so rezervirali vse ladje?

OPB, QBE

Za vsakega mornarja prestejemo koliko ladij je rezerviral in primerjamo z vsemi ladjami.

Pri ladjah bid postavimo B2 in pod conditions preštejemo št ladij, ki jih primerjamo z rezervacijami. Izpišemo sid mornarjev ki jih grupiramo in jih povežemo z rezervacijami, (st vseh rezervacij, ki jih je rezerviral mornar)

Izračunamo št. Vseh rezervacij za posameznega mornarja in primerjamo z vsemi ladjami. Kjer je enako se izpiše.

### Vstavljanje zapisov

- Vstavljanje enega zapisa:

Sailors	sid	sname	rating	age
I.	74	Janice	7	14

- Vstavljanje več zapisov (rating je null v spodnjih zapisih):

Sailors	sid	sname	rating	age
I.	Id	N		A

CONDITIONS				
A > 18 OR				
_N LIKE 'C%'				

Students	sid	name	login	age
	Id	N		A

OPB, QBE

Najprej izberemo sid mornarjev in študentov. Nato vse študente prepišemo v mornarje.

### Brisanje in update

- Pobriši vse rezervacije za mornarje, ki imajo rating < 4

Sailors	sid	sname	rating	age
	Id		< 4	

Reserves	sid	bid	day
D.	Id		

- Povečaj za eno starost mornarja, ki ima sid = 74

Sailors	sid	sname	rating	age
	74			U..A+1

OPB, QBE

### Omejitve pri popravljanju zapisov

- Ne moremo mešati I., D. in U. v enem primerku tabele
- I., D. in U. ne moremo kombinirati z P. in G.
- Ne moremo vstavljati, popravljati ali spremenjati n-teric z vrednostmi iz stolpcev iz ostalih n-teric iste tabele.

Sailors	sid	sname	rating	age
		john		A
		joe		U..A+1

Naj popravimo starosti vseh Joe?  
Starost katerega Johna naj uporabimo?

## 3. Podatkovni model ER

1) Opišite podatkovni model ER.

Značilna predstavitev realnega sveta. Za predstavitev se uporablja grafična notacija, za boljši pregled nad strukturo podatkov. Preslikava med realnim svetom in podatkovnim modelom je nazorna, zato omogoča dobro komunikacijo z uporabniki, ki jih lažje vključimo v proces.

## 2) Opišite, kdaj naj se uporablja entiteta, razmerje in atribut v idejni zasnovi relacijskega DBMS.

Entiteta: Objekt iz realnega sveta, ki ga lahko ločimo od ostalih objektov. Predstavljena je z množica atributov(lastnosti).

Razmerje: definira povezavo med dvemi ali več entitetami, ki lahko pripadajo različnim množicam.

Atribut: lastnosti entitet, vsak atribut ima določene vrednosti.

## 3) Predstavite koncept kardinalnosti o odnosu.

Števnost, z njo povemo v kolikih različnih razmerjih npr. R lahko sodeluje entiteta iz množice E. Poznamo min, max in glede na to zapišemo razmerje(N,N) (1,1)..(1,N)

## 4) Kaj je šibka entiteta?

Entitetna množica E je šibka če je njen obstoj iz entitete množice E pogojen z obstojem iz mn F. Da množica E obstaja samo ob obstoju F(to pa imenujemo močna entiteta). Šibko entiteto identificiramo s pomočjo primarnega ključa močne entitete.

## 5) Predstavite Chenovo notacijo modela ER.

Minimalne vrednosti so izpuščene in števnosti glede na (min, max) zamenjani. Označujemo z oznako 1 proti N (mnogo).

## 6) Predstavite konstrukte za modeliranje hierarhije generalizacije / specializacije entitet.

Odnos tip-nadtip: generalizacija: Dvema tipoma(študent, upokojenec) priredimo nadtip član. Hkrati je tip študent in član.

Odnos tip-podtip: specializacija: Lastnosti atributov(moški, ženska) pripadajo nadtipu(član).

## 7) Kako modelirati združevanje(agregacijo) subjektov?

Razmerje, ki povezuje množice razmerij. Omogoča, da obravnavamo množice razmerij kot entitetne množice, ki lahko sodelujejo v drugih razmerjih.

## 8) Predstavite pravila za prevajanje modela ER v model relacijske baze podatkov.

PRAVILA:

- Za vsak entitetni tip kreiramo eno relacijsko shemo. Ime relacije naj bo ime entitetnega tipa (priporočilo).

- Atributi relacije so atributi entitetnega tipa.
- Opcijske atribute prevedemo v attribute, v katerih dovolimo vrednosti 'NULL' oz. neobvezna polja.
  - Ključ entitetnega tipa postane primarni ključ pripadajoče relacije.
  - Tuji ključi entitetnega tipa postanejo tuji ključi relacije.

## 4. Relacijski jeziki poizvedb

1) Predstavite osnovno skladnjo poizvedbenega jezika SQL.

2) Predstavite osnovno sintakso QBE.

Uporabniki definirajo vprašanja s primeri tabel ali skeleti.

3) Opisite koncept stika v SQL in QBE.

Definiran z ujemajočima spremenljivkama: npr mornarji, ki so rezervirali ladjo 25 in so starejši od 25

4) Predstavite vgnezdene poizvedbe v SQL.

```
Where M.mid IN (SELECT R.mid  
                  FROM Rezervacije R  
                  WHERE R.lid = 103)
```

Stavek WHERE lahko vsebuje SQL poizvedbo(lahko tudi From, having). Vgnezdene zanke, za vsakega mornarja preveri pogoj poizvedbe, ki vsebuje logično poizvedbo

Stavek WHERE EXISTS- Primerjava s prazno množico

UNIQUE-pomeni iskanje mornarjev, ki imajo največ eno rezervacijo ladje

*5) Opišite vlogo stavkov, ki [ne] obstajajo in [niso] edinstveni v SQL.*

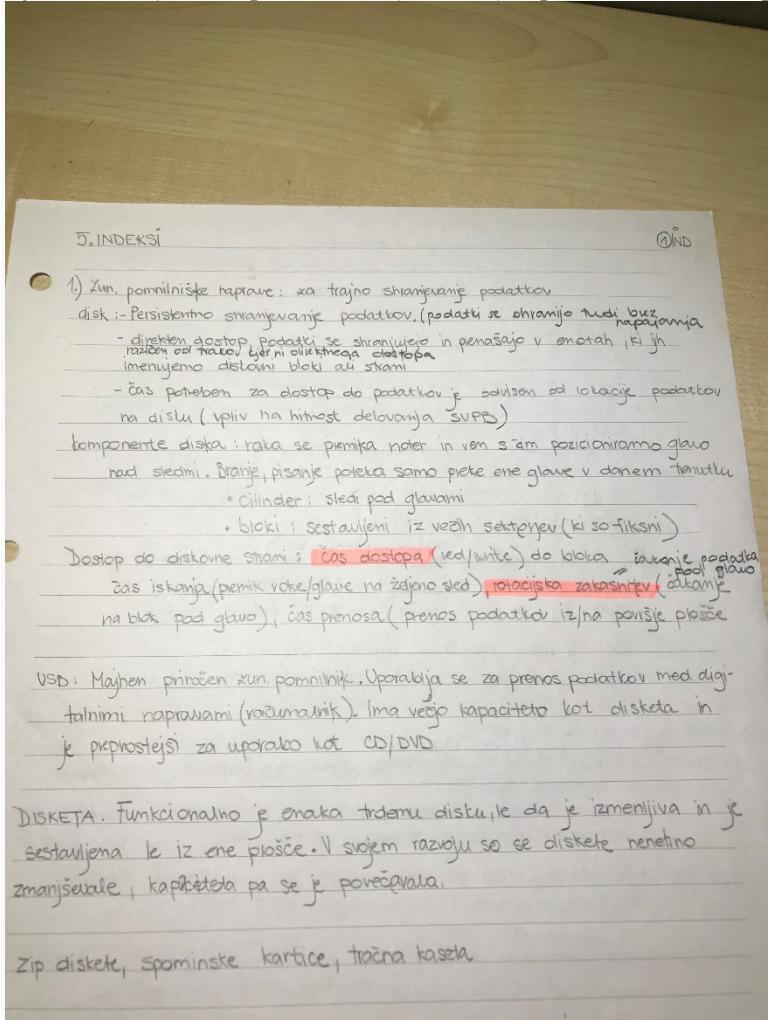
NOT IN- except, NOT EXIST, NOT UNIQUE-ne izpiše

*6) Kako izvajati delitev (operacija RA) v SQL in QBE?*

*7) Opišite uporabo funkcij GRUPIRANJA v SQL in QBE.*

Podpira AVG, COUNT, MIN, MAX, SUM. Stolpci z G. so group-by polja in imajo že vrednost izbranih atributov(vsak stolpec mora imeti agregacijsko operacijo.)

## 1) Poimenujte in opišite nekaj zunanjih pomnilniških naprav



## 2) Kakšna je organizacija datotek? Katere alternative imamo?

1. Neurejene datoteke; zapisi niso urejeni, se vrstijo naključno. Hitro skeniranje, vstavljanje, brisanje, počasno iskanje, za skeniranje celotne datoteke (najenostavnejša datotečna struktura): implementirane s seznamom: vsaka stran vsebuje 2 kazalca in podatke.

Implementirane z direktorijem strani-imamo kazalce, ki so shranjeni v direktoriju, vsak kazalec kaže na svojo stran.

2. Sortirane: vrste zapisov glede na iskalni ključ, hitro iskanje, počasno brisanje.

3. indeksi: pohitri na osnovi iskalnih ključev (vsaka podmnožica je lahko iskalni ključ relacije), ni enak primarnemu ključu. Indeks podpira učinkovito iskanje podatkovnih vpisov  $k^*$  z vred ključa

3) Opišite koncepte iskalni ključ, vnos podatkov, vnos indeksa:

Iskalni ključ je lahko poljubna podmnožica polj na dani relaciji, ni enak primarnemu lahko pa zahtevamo da je indeks unikaten.

4) Kakšne so možnosti za podatkovni vpis  $k^*$ ?

Sestavljen iz ključa in kazalca na zapis (rid-npr >15 in to so vsi ključi 17, 18 20 ).

1. Podatkovni zapis z vred. Ključa k (indeks je datotečna organizacija, shranjuje množico zapisov, lahko je urejena ali neurejena, sortirana datoteka, en indeks nad podatki-ker je shranjen zapis).

Neglede na to kateri indeks izberemo pa lahko z indeksi gradimo B+ drevesa, razpršilne indekse.

2. Ključ in identifikator zapisa (če želimo kreirati več različnih indeksov ki vsebujejo kazalce zapisov., boljše kot v primeru 1, če imamo veliko podatkovnih zapisov in če so ključi majhni)

3. Ključ in identifikator seznama (imamo precej več dela, ker so zapisi variabilne dolžine, iskalni ključi fiksne dolžine)

5) Kaj je primarni / sekundarni indeks? Kaj je gosti / redki indeks?

Primarni-če vsebuje primarni ključ. Sekundarni- če vsebuje poljubni ključ

Gosti-Za vsak zapis osnovne datoteko, obstaja zapis v indeksni datoteki. Vrednosti podatkovnih elementov, ki tvorijo ključ zapisa se ujemajo z enakimi podat. Elementi v zapisu osnovne datoteke. Kazalec index. Zapisa kaže na polje v katerem se nahaja osnovni zapis.

Redki-Pogoji za uporabo: osnovna datoteka je zaporedna +primarni indeks, nahaja se po en zapis s kazalcem na vsako skupino polj (omogočajo hiter dostop). Indeks vsebuje manj zapisov kot osnovna datoteka. Zapis kjer nastopata v paru kazalec na skupino polj in vrednost ključa zadnjega zapisa v skupini.

6) Predstavite indekse ISAM.

Drevesna struktura, sčasoma drevo postane neuravnovezeno. Statičen , sestavljen iz indeksnih zapisov-notr vozlišča in listi-podatkovnih vpis. Za izboljšanje na začetku, ko gradimo pustimo prazen prostor v straneh listov. Vsako vozlišče vsebuje dva vpisa, ni kazalcev, razen pri prelivnih straneh

7) Predstavite B + drevesni indeks.

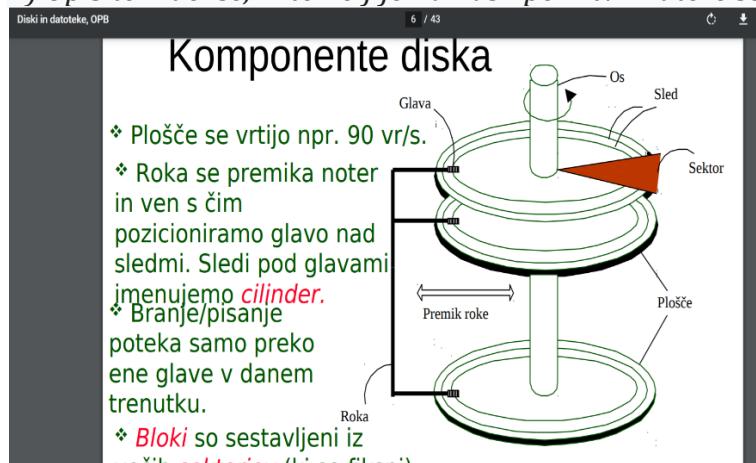
Najbolj razširjen indeks. Uravnoteženo drevo, indeksne strani so 50% napolnjene. Iskanje se začne v korenju: primerjava ključev vodi do lista s podatkovnimi vpisi. Listi vsebujejo podatkovne vpise (urejeni, prev, next), notranje strani imajo indeksne vpise oz. podatkovne zapise

### 8) Opiši vstavljanje in brisanje drevesnih operacij B +.

Vstavljanje-Pošči list L, vstavi podatkovni vpis v list: če je prost je ok, če ni: je potrebno razcepiti v L (v L in novo vozlišče L2), enakomerno porazdeli vpise in prepiši ključ gor, vstavi indeksni vpis, ki kaže na L2 v starša od L.

Brisanje- začni pri korenju in poišči list z vpisom. Izbriši vpis- če je L vsaj polovico poln potem končaj. Če vsebuje L d-1 vpisov-poskusite porazdeliti vpise, da si jih sposodiš od sosedov. Če porazdelitev ne uspe, zlij L in sosedja. V primeru zlitja moramo zbrisati vpis iz starša od L. Zlivanje se lahko nadaljuje vse do korena. Se zniža višina drevesa.

### 9) Opišite indekse, ki temeljijo na hash polnitvi. Katere so alternative?



Linearni razpršilni indeks:

Razcepljanje v rundah. 1 runda se neha ko se vsi začetni skupki razcepljeni. Naslednja runda ima 2x več skupkov. Uporabimo kazalec next = 0. Razcepimo tistega na kateri kaže kazalec next. Nivo predstavlja trenutni skupek.

Statični razpršilni indeks:

Na iskalnem ključu uporabimo razpršilno funkcijo. Razvije se dolg seznam prelivnih strani in poslabša učinkovitost podatkovne strukture.

Razširljiv razpršilni indeks:

Dinamični, direktorij kazalcev na skupine (prestavljam strani skupkov, št skupkov se podvoji s podvojitvijo direktorija, razcepimo stran ki je napolnjena, ni prelivnih strani).

## 6. EVALUACIJA RELACIJSKIH POIZVEDB

## *1) Kaj je metoda dostopa? Kakšne načine dostopa poznate?*

Način pridobivanja n-teric s tabele(dostop z indeksom, ki se ujema z atributi v pogoju izbire. Da prebere najmanj strani z diska

Načini:

1.Pregled datoteke

2.S pomočjo indeksa: povezane ali nepoveze.

Drevesni: se ujema z atributi v pogoju izbire, v primeru da so atributi predpona izbranega ključa.

Razpršilni indeks: Pogoj vsebuje vse atrubute indeksa. Pogoj je sestavljen iz samih operacij enakosti.

## *2) Opišite skupne tehnike, ki se uporabljajo za evaluacijo relacijskih operacij.*

1.Indeksiranje: Uporaba pogojev iz stavka WHERE za izbiro majhnega št. N teric pri selekciji in stikih.

2.Iteracija: Včasih hitreje pogledamo vse n-terice, čeprav je na razpolago indeks(včasih je hitreje izvesti iteracijo po podatkovnih vpisih indeksa namesto na sami tabeli.)

3.Particijr: Velkokrat koristi porazdeliti probelm na več enakih delov. S tem zamenjamo izvajanje časovno potratnih operacij s podobnimi operacijami nad manjšim številom n-teric.

## *3) Predstavite splošni zunanji algoritmom za sortiranje z zlivanjem. Kakšna je zapletenost algoritma z zlivanjem?*

Temelji na dveh vgnezdenih zankah. 1. zanka se sprehodi čez vse nteric tabele R, druga se sprehodi čez vse n-terice tabele S. Za vsak par( kartezijski produkt) pogledamo če se par ujema v ključih in ga damo v rezultat.

Cena:  $M(\text{koliko strani preberemo da pregledamo celotno relacijo } R) + pr(\text{št. N terk na strani}) \times \text{sšt strani}$ .

Prvo je treba najprej zun strani prebrt, nato notranje.

## *4) Kako izvesti operacijo selekcije ?*

Pogoj izbire je preveden v konjunktivno normalno obliko.

Osnovni pristop: Poišče najbolj selektivne metode dostopa(da preberemo čimmanj blokov z diska).Selektivnost pogoja: Delež relacije, ki je rezultat selekcije z danim pogojem.

## *5) Predstavite metode za izvedbo projekcije.*

Najdražja operacija: odtramitev duplikatov(Distinct), npr. sortiranje po mid in lid in odstranimo duplike. Uporaba razpršilnega indeksa nad mid in lid dobimo particijo, funkcijo h1(preberi vsako množico v dinamičnem pomnilniku, sortiraj tabelo v dim pomnilniku in odstrani duplike). V primeru prevelike particije-množice, ponovi postopek z razpršilno funkcijo h2.

## *ALGORITMI*

### *6) Opišite stik z vgnezdeno zanko z indeksom, stik z drevesnim indeksom, stik z vgnezdeno zanko po blokih.*

1. Stik z vgnezdeno zanko z indeksom: Preberemo vse n terke relacije R, za vsako n terko mormo dostopat preko indeksa do relacije S. 1. v primeru da uporabimo razpršilni indeks v povprečju preberemo 1-2 strani. 2 v primeru uporabe B+ drevesa uporabimo 2-4 strani. Preberemo še pod. Zapise (v primeru povezanega- 1 stran, nepovezanega- 1stran + vse n terke).

2. Stik z vgnezdeno zanko po blokih: za vsako stran R2 preberemo vsako stran R1 in izpišemo pare n-teric. Cena: Vse zun. Strani + št. Zun blokov \* vse notranji bloki. Št zun. Blokov = št vseh stranih zun.tabele deljimo z (B-2(DOLŽINA VRSTE)) ozr. strani blokov.

### *7) Opišite stik z zlivanjem*

1. Sortiramo obe tabeli. Stik zlivanje parov, ki se ujemajo s ključi shranimo kot rezultat. 2.(branje vsake posebaj) Najprej preberemo R dokler je n terka  $\geq$  S nterki. Potem preberemo vse ključe dokler S n terka  $\geq$  R terki. Nato so ključi R terk = S terkam. Nadalujemo da najdemo vse enake ključe iz n terk. Na koncu je potrebno prebrati še obe tabeli.

### *8) Opišite stik z razpršilnim indeksom.*

Uporabimo razpršilno funkcijo h, tako da razpršimo obe relaciji na B-1 particij. Preberemo eno particijo in vse zapise razpršimo z raz. Fun. H2. Beremo pripadajoče particije iz druge relacije, ki se bojo joinale in jih ponovno razpršimo z raz. Funk. H2. Te vsebujejo n terke ki jih joinamo skupaj.

# 7.Optimizacija poizvedbe

- 1) Kako oceniti plan poizvedbe?
- 2) Predstavi ekvivalentne operacije relacijske algebri?
- 3) Kako pridobiti vse enakovredne izraze poizvedbe za dano poizvedbo, izraženo v relacijski algebri?(Prostor rešitev)
- 4) Opišite postopek za oceno alternativnih načrtov poizvedb.
- 5) Opišite poizvedbe nad eno relacijo

4. OPTIMIZACIJA POIZVEDBE

1) OCENA PLANA : za vsak plan je potrebno določiti ceno :

- potrebno je določiti ceno za vsako operacijo v dnevnu ; cena operacij je odvisna od kardinalnosti vhodnih relacij :
- za vsak stik predpostavimo neodvisnost med predikati
- cene so različne pri vsakem pregledu

2) Ekvivalenze operacij Relacijske algebri :

- omogočajo : načrtovanje ekvivalentnih zapisov izrazov RA , različni vrsti red stikov, spuščanje selekcije in projekcije proti listom,  $\text{selec. in cross} \rightarrow \text{stik}$
- zapis določa delo operacij (Program)
- z pregledom vseh ekvivalentnih izrazov RA načrtujemo vsa dneva operacij

3) Prostor rešitev : prostor ekvivalentnih poizvedb .

- lastnosti RA omogočajo transformacije poizvedbe
  - transf. poizvedba vrne isti rezultat
  - transf. poizvedba parnoč spremembu plana
  - ponovno je potrebno določiti algoritme za implementacijo operacij
  - izberemo izraz , ki se najhitje izvede in uporabi najmanj prostora
- izogibamo se karteziskim produktom in najslabšim rešitvam.

4.) Alternativni plani : uporaba ekvivalentnih pravil nad izrazi

- dobimo logično ekvivalentne izraze , različna implementacija & cena.
- samo nad levo usmerjenimi plani

Dva osnovna primera : 1. plan nad eno relacijo ; 2. plan nad večimi relacijami  
optimalna metoda dostopa      permutacija stikov

5) Poizvedba nad eno relacijo : kombinacije selekcije, projekcije in agregacijskih operacij ; ni stikov !

- pregledajo se vse možne metode dostopa ( brez , z indeksi )
- izberemo metodo z najnižjo ceno

## 6) Definicija naštevanja levo-usmerjenih planov

## 7) Opišite postopek za naštevanje levo-usmerjenih planov.

• METODE DOSTOPA

- BREZ INDEKSOV
  - Osnovni postopek je pregled vseh zapisov tabele npr. Mornarji in sprotno filtriranje zapisov (selekcija + projekcija)
  - rezultat se uvedi v skladu z atributi relacije Group-by
  - zapis rezultata se generira iz ene skupine mornarjev ki ustreza pogojem operacije having
  - agregacijske funk. se izvršijo nad skupinami
- DOSTOPI:
  - Z indeksom 1. x 1 ind.: če je na razpolago več indeksov, vsak indeks predstavlja eno metodo dostopa, izberemo enega, ki zahteva najmanj VI
  - 2. z večimi : če več indeksov tipa 2,3 ustreza pogoju izber
    - preberemo rezultate večih indeksov in naredimo presel, sortiramo izbrane id-je za optimalen dostop
    - projekcije in dodatne selekcije sledijo po branju zapisov
  - 3. dostop s sortiranim indeksom: seznam atributov ustreza za grupiranje sortiranega indeksa
    - uvojenost indeksa uporabimo za grupiranje, vse selekcije lahko naredimo, po tem ko preberemo n-telice preko indeksa
  - 4. Dostop samo z indeksom:
    - kadar imamo indeks, ki vsebuje vse atribute v vprašanju
    - podatkovni vpis = zapisu, zato ga ni potrebno preberati
    - naredimo sel., proj in sortiramo končne zapise za grupiranje
    - funkcionalna tudi, kadar atributi izbere niso vsebnani v atributih indeksa
    - filtriramo n-telice in dostopamo samo do izbranih podatkovnih zapisov

6.) Naštevanje levo usmerjenih planov: - uporabljam se samo levo usmerjena dlevesa razlikujejo se v : vrstnem redu relacij, v metodah za dostop do relacij, algoritmih za izvajanje stike

7.) Postopek za leve usmerjene plome :

1. optimizator sistema R pregleda vse možne stike, kjer se  $\sigma$  in  $\Pi$  spusti k relacijam
2. Lastnosti algoritma : št. planov raste s št. relacij
  - možna je generiranje planov, ki "realizirajo" "celoude"
  - vmesnih rezultatov ni potrebno zapisati v začasne datoteke

- NAŠTEVANJE z N prehodi za N relacij  
Prehod 1, 2, N
- DINAMIČNO PROGRAMIRANJE : za vsako podmnožico sestavljenico iz K relacij je potrebno dlaniti samo najcenejši plan.

plane iz N relacij sestavljamo iz planov N-1 tako da dodajamo še en stik

(4.2)

### Prehod 1 :

- naštevamo vse plane z eno relacijo iz FROM dela SELECT stanka
- plan nad eno relacijo je delen levo-usmerjen plan
- za vsako relacijo A identificiramo pogoje izber, ki se ne nanašajo na atribut t, ki so selekcije, ki so potekle pred vsemi stiki
- izločimo attribute, ki niso potrebni po branju podatkovnega zapisa
- optimalna metoda dostopa ; ujemnost izkoristimo za GROUP BY operacijo

### Prehod 2 :

- naštevamo vse plane z dveimi relacijami
- prej izračunane optimalne metode dostopa do ene relacije uporabimo kot zun. relacijo, vse ostale za notranje
- npr. A = zun. B = notr. ; nato identificiramo: selekcije nad atributi, ki se tičejo samo B in se izvršijo pred stikom; selekcije nad atributi, ki definirajo stik; selekcije nad atributi drugih relacij, ki se lahko izvršijo po stiku
- pridemo ceno sortiranja, v primeru da se ne izvrši sortiranje z živanjem.

### Prehod 3 :

- Generiramo vse plane s tremi relacijami
- vzame optimalne plane iz prehoda 2 in jih uporabimo kot zunanj poizvedbo

### Pri preostalih prehodih :

- grupiranje v bloku se izvrši na koncu
- včasih je potrebno postribeti za pravilno ujemnost izhoda stikov, da lahko izvedemo grupiranje.

## 8. TRANSAKCIJE/KONTROLA VZPOREDNOSTI

- 1) Kaj je transakcija?
- 2) Pojasnite WW RW konflikt.
- 3) Kakšen je konflikt zaporedno uredljivosti?
- 4) Opišite stroge in nestroge dvofazne protokole zaklepanja.
- 5) Kaj se zgodi, ko je transakcija prekinjena?(abort)

8. TRANSAKCIJE

1) Transakcija je abstrakten pogled SUPB na uporabniški program  
(sekvenca ukazov za branje in pisanje)

Lastnosti: A: atomarnost, C: konsistentnos, I: izolacija, D: trajnost

2) Anomalijske prekrivajočih transakcij:  
WR : branje nepotjenih podatkov      R(A) W(A)      R(A) W(A) R(B) W(B) C  
RW : nepotjavno branje      R(A) R(B) W(B) C      R(A) W(A) C  
WW : prepis nepotjenih podatkov      W(A) W(A) W(B) C      W(A) W(B) C

3) Zaporedna uredljivost: najprej se izvede  $T_1$  in  $T_2$   
- izvajanje dveh zaporednih transakcij nam da drugačne rezultate

4.) Zaklep: nobena druga transakcija ne bo dostopala do objekta.  
- protokol zaklepanja: množica pravil, ki se izdrži vsata transakcija, za zaporedno izvajanje

\* STROGO 2FZ: Vsaka transakcija mora pridobiti S(delen) zaklep na objektu pred branjem in X(ekskluzivni) zaklep na objektu pred pisanjem  
- vsi zaklepi se sprostijo ob koncu transakcije  
- zaklepanje omogoča samo varno izvajanje, dopušča tudi prekrivanje akcij

\* 2FZ: sprosti zakljenjene objekte kadarkoli, vendar po sprostitvi istih objektov ni moč ponovno zakleniti

5.) Ko je transakcija prekinjena:  
- če se prekine, so vse akcije izničene  
- večina sistemov se izogne kaskadnim prekrivanjim s sprostitvijo zakljenjenih objektov tik pred prekrivitvijo  
- SUPB zapisuje kompleten dnevnik vseh popravkov:  
- prekinjene transakcije lahko izničimo, uporablja se enak mehanizem za vzpostavitev konsistentnega stanja po sistemski napaki

6) Opišite metode preprečevanja zastojev.(preprečevanje, detekcija)

7) Kaj je zaklepanje indeksa? Kako se izvaja?

8) Opišite optimistično kontrolo vzporednosti.

6.) SMRTNI OBJEM :

-Preprečevanje smrtnega dejema

•določi prioritete glede na časovne zamitke zaznamke . 2 politiki :

-počakaj-umri : če ima  $T_i$  višjo prioriteto potem  $T_i$  počaka na  $T_j$  drugače prekine izvajanje

-Rani-čakaj : če ima  $T_i$  višjo prioriteto , potem  $T_j$  prekine izvajanje, sicer  $T_i$  počaka

- v primeru, da transakcija ponovno starta je potrebno zagotoviti , da ima originalno časovno označo

-Odkrivjanje smrtnega dejema :

• Kreiraj graf-čakalni

-vozlišča so transakcije, obstaja povezava od  $T_i$  do  $T_j$  , če  $T_i$  čaka na  $T_j$  za sproščanje zaklepja.

•periodično preveri ali ima čakalni graf cikle

7.) Zaklepanje indeksa :

- Pri sekvenčnem skenirjanju je potrebno zagotoviti, da se ne sime dodajati zapisov tabeli

•  $T_1$  mora zakleniti vse strani, kot tudi dalečko , da bi onemogočili dodajanje

zapisov z rating=1

-če imamo index za atribut rating z alternativo(2) potem bi  $T_1$  moralna zakleniti indeksne strani z indeksnimi vpisi z rating=1

-če ni zapisov z rating=1 , potem mora  $T_1$  zakleniti indeksne strani kjer bi bili takšni podatki, če bi obstajali

! enostaven algoritmom za zaklepanje dleves;

ISKANJE : začni pri korenju , na poti do lista vedno zakleni otroka z S zaklepom in odkleni starša

STAVI/BRISI : začni pri korenju in potuj navzdol , da pridobiš X zaklep po potrebi i Ko je otrok zaklenjen preveč, če je varen:

-če je otrok varen sprosti ~~vse~~ zaklep na starsih

VARNO VOZLISČE : Vozlišče, ki spremeni spremembbo ne bo sišlo navzgor.

Vstavljanje : vozlišče ni polno

brisanje : vozlišče ni pol prazno

8.) Optimistična kontrola vzporednosti :

- zaklepanje je konzervativni pristop , ki prepreči konflikte . Slabe lastnosti :

Slabe lastnosti :

- dodatna koda za zaklepanje , preverjanje smrtnega dejema , zasičenost z zaklepi za vse pogosto uporabljene objekte

- če so konflikti redki, lahko pridobimo na vzporednosti tako , da ne zaklepavajo  
namesto tega izvaj preverjanje konflikte pred izvajanjem transakcij protjerovanja

-optimistična kontrola je hitrejša od zaklepanja , če je malo konfliktov.

## 9.Obnovitev zrušitve

### *1) Pojasnite lastnosti ACID relacijskega DBMS.*

A: Atomarnost: izvršijo se vse akcije ali nobena. C: konsistentnost- če je transakcija in celotna pb konsistentna potem je po transakciji pb konsistentna. I- izolacija: izvajanje ene transakcije je izolirano od izvajanja drugih transakcij. D : ohranjevanje: če transakcija potrdi, so vse spremembe stalne. Upravljalnik obnavljanja: generira atomarnost in ohranjevanje.

### *2) Razložite kompromise pri "kraji" strani iz vmesnega pomnilnika in "prisili", da se strani shranijo na disk.*

T1 želi prebrati podatkovni objekt vendar je delovni pomnilnik poln z drugimi transakcijami. Torej T1 počisti nekaj pomnilnik, s tem da neko drugo stran delovnega pomnilnika brcne v stabilno shrambo. To je nevarno, ker nevemo, če je bilo tisto brcnjeno že potrjeno 'kraja'. Prisilno pomeni, da so vse prizadete strani potisnene v stabilno shrambo. Na vsako stran se lahko zapiše več transakcij in to upočasni sistem.

### *3) Opišite protokol WAL (Write-Ahead Logging). Pisanje dnevnika v naprej*

1. Generira atomarnost- Mora shraniti dnevniški zapis preden se pripadajoča stran zapiše na disk.
2. Generira ohranitev- mora zapisati vse dnevniške zapise za transakcije pred potrditvijo.

### *4) Opišite, kateri podatki so shranjeni in kje so shranjeni za izvajanje protokola WAL.*

Vsek dnevniški zapis ima dnevniško sekvenčno številko. Vsaka podatkovna stran vsebuje LSN strani. Updejt, Abort, End

### *5) Predstavite faze analize, ponovitve in razveljavitve obnove po nesreči.« Obnovitev po zrušitvi: celotna slika»*

Začnemo od kontrolne točke.

Tri faze: 1-Odkrijemo katera transakcija je potrdila od kontrolne točke naprej in katera ni uspela (to analiziramo).

2-Obnovitev: REDO faza: ponovimo zgodovino tako da rekonstruiramo stanje ob zrušitvi: ponovno ovrednotimo vse popravke.

3-UNDO: učinki neuspelih transakcij: razvrsti dejanja transakcij, ki niso bila potrjena, tako da baza odraža le dejanja potrjenih transakcij.

## 10. LOGIČNO NAČRTOVANJE

*1) Zakaj se lahko redundanca pojavi v relacijskih zbirkah podatkov? «problem redundancy»*

Anomalija update: pri posodabljanju se ustvari kopija, podatki niso več enaki. Anomalija insert: nekaterih podatkov ni mogoče shraniti, če se ne shranijo drugi. Anomalija delete: če izbrišemo ene podatke lahko tudi druge(nevemo če smo jih).

*2) Kaj je funkcionalna odvisnost? 3) Kako lahko razmišljamo o funkcionalnih odvisnostih?*

Funkcionalna odvisnost (FD) določa razmerje enega atributa do drugega atributa v sistemu za upravljanje baz podatkov (DBMS). Funkcionalna odvisnost pomaga ohranljati kakovost podatkov v bazi podatkov. Funkcionalno odvisnost X od Y predstavlja  $X \rightarrow Y$ . funk odvis. igra ključno vlogo pri iskanju razlike med dobrim in slabim oblikovanjem baze podatkov.

*4) Kaj je namen normalizacije relacije? Kaj je normalna oblika?*

Relacija je v določeni normalni obliki: Izognemo se nekaterim problemom. Normalna oblika določa lastnosti shem. Normalne oblike povedo ali je potrebno relacijo restrukturirati.

### *5) Predstavite Boyce-Coddovo normalno obliko odnosa.*

Če so vse netrivialne odvisnosti odvisne od ključa. Če imamo A za superključ(npr.  $A \rightarrow B$  ).

### *6) Predstavite tretjo normalno obliko (3NF) relacije.*

Relacija je v tretji normalni obliki 3NO, če: • je v drugi normalni oblik 2NO in • nima tranzitivnih odvisnosti. Tranzitivna odvisnost je funkcionalna odvisnost med atributi, ki niso del ključa. Tranzitivne odvisnosti odpravimo tako, da relacijo razbijemo na več novih relacij. Relacija je avtomatsko v 3NO, če je v 2NO in je njen ključ sestavljen iz vseh ali vseh razen enega atributa sheme.

### *7) Kaj je dekompozicija z brezizgubnim stikom?*

Dekompozicija pomeni zamenjavo R z dvema ali več relacijama, kjer velja vsaj ena odvisnost. Relacijo R razgradimo v dve novi relaciji in če velja  $R1 R2 = R$ ., Kar pomeni da mora biti vsak atribut R v R1 ali v R2.

### *8) Kaj je ohranjanje funkcijске odvisnosti?*

Definirana na osnovi projekcije množice odvisnosti: preverjanje ob vstavljanju in spremjanju. Unija projekcij mora pokrivati vse odvisnosti. Če razstavimo relacijo R v relaciji R1 in R2, morajo biti vse odvisnosti R del R1 ali R2 ali pa jih je mogoče izpeljati iz kombinacije FD R1 in R2.

### *9) Predstavite algoritem dekompozicije v BCNO.*

Vsaka relacija razcepi relacijo in algoritom zaključi brezizgubni stil(ker velja vsaj ena odvisnost) koraki algoritma tvorijo drevo(rekonstrukcija).

### *10) Predstavite algoritem dekompozicije v 3NF.*

Algoritem prej zaključi kot pri BCNO. Ta pristop ne zagotavlja ohranitev odvisnosti. Enostavna dekompozicija z brezizgubnim stikom in z ohranitvijo odvisnosti.

## **11.FIZIČNO NAČRTOVANJE**

## *1) Opišite fizično načrtovanje. Katere informacije so vključene?*

Delovna obremenitev vključuje vse poizvedbe, ki jih generira aplikacija. Opis delovne obremenitve vključuje naslednje: 1. Seznam poizvedb (z njihovo pogostostjo kot ocena vseh poizvedb / posodobitev). 2. seznam posodobitev in njihove frekvence. 3. Cilji uspešnosti za vsako vrsto poizvedb in posodobitev. Za vsako poizvedbo v obsegu dela moramo identificirati: -Do katerih odnosov je mogoče dostopati. -Kateri atributi se obdržijo (v določbi SELECT). -Pri katerih atributih je izbran join ali selekcija pogoje in kako selektivni so ti pogoji. Za vsak update: -kateri atributi imajo na njih izražene pogoje join ali selekcije (v klavzulo WHERE) in kako selektivni so ti pogoji. -Vrsta posodobitve (INSERT, DELETE ali UPDATE) in posodobljena relacija. -Za UPDATE celijskih vrednosti so polja spremenjena zarad posodobitev.

## *2) Opišite izbiro indeksa za učinkovito izvajanje izbrane operacije.*

## *3) Opišite selekcijo indeksov za stike.*

Ko razmišljate o pogoju pridružitve(join): Hash indeks na notranji strani je zelo dober za Index Gnezdene zanke. Zgrajeno drevo B + v stolpcih za združevanje dobro za Razvrsti-Združi(sort-merge).

## *4) Pojasnite, zakaj in kako je treba izbrati alternativno razgradnjo relacije.*

denormaliziramo-torej se lahko odločimo zbirko relacij, dobljeno z razgradnjo iz večjega razmerje, z izvirnim (večjim) razmerjem, obstaja ogromno število alternativnih dekompozicij, ki jih je mogoče izdelati za določen sistem. Ker število komponent v sistemu narašča, se število načinov razgradnje povečuje eksponencialno. Naša definicija razgradnje(decomposition): Odnos se nadomesti z zbirko odnosov, ki so projekcije.

## *5) Pojasnite, kdaj je smiselna denormalizacija.*

Kadarkoli normalizirate podatke, morate vedno uskladiti konkurenčne cilje zmanjšanja odvečnosti podatkov z dodatnim delom, ki bo ustvarjeno, ko boste morali izvleči podatke iz baze. Elementi, kot

so mesto, država in poštna številka, se večinoma prenašajo odvečno. Razmislite o primeru poskusa dodelitve mesta stranki, vendar mesto ni opredeljeno v mestni tabeli. Najprej morate vzdrževati mestno tabelo. Zdaj bi to, kar bi trajalo samo eno operacijo, trajalo dve. Zmanjšanje zahtevnosti poizvedb in izboljšanje uspešnosti poizvedb sta dobra razloga za denormalizacijo. Ker kršite pravilo, bi to morala biti izjema, ne pa norma

## *6) Kaj je horizontalna dekompozicija? Zakaj se uporablja?*

Včasih lahko relacijo zamenjate z zbirkо relacij, ki so selekcije. -Vsako novo razmerje ima isto shemo kot original, vendar podmnožica vrstic. -Skupaj novi odnosi vsebujejo vse vrstice originala. Nove relacije so navadno ločene(disjoint).

## *7) Predstavite nekaj primerov poizvedb, ki jih je treba prilagoditi ali prepisati, da bi dosegli učinkovitost. «usmerjanje konceptualne sheme»*

Če poizvedba teče počasneje, kot je bilo pričakovano, preverite, ali je indeks treba znova zgraditi ali pa je statistika prestara. Včasih DBMS ne izvaja načrta ki ste ga imeli v mislih, Skupna območja Šibkosti: Izbori, ki vsebujejo ničelne vrednosti. Izbori, ki vključujejo aritmetične ali string izraze. Izbori, ki vključujejo pogoje ALI Pomanjkanje funkcij vrednotenja, kot so strategije samo za indeks ali nekatere metode združevanja(join) ali slaba ocena velikosti.