[BAZE]

//2. PREZENTACIJA - RELACIJSKI MODEL PODATAKA
RELACIJSKI MODEL PODATAKA
- Objekti u relacijskom modelu podataka su relacije
- Relacija je imenovana 2d tablica - skup n-torki definiranih na relacijskoj shemi R
  - u jednoj relaciji ne postoje dvije jednake n-torke
  → Atribut - imenovani stupac
    - ima jedinstveno ime unutar R
    - može primiti vrijednost iz samo jedne domene
  → Domena - skup dopuštenih vrijednosti atributa
  → n-torka - redak relacije
    - t = {A1:v1, ..., An:vn} == t <v1, ..., vn>

- Relacijska shema - imenovani skup atributa R={A1, A2, ..., An}
  - poredak atributa je nebitan
- Stupanj relacije(degree) - broj atributa/stupaca
- Kardinalnost relacije(cardinality) - broj n-torki/redaka

- Baza podataka - shema - skup relacijskih shema - rijetko se mijenja
  - instanca - skup instanci relacija - često se mijenja

SQL
- kreiranje nove instance baze podataka →
```
CREATE DATABASE imeBaze;
DROP DATABSE imeBaze;
```
- kreiranje relacije →
```
CREATE TABLE mjesto(
      pbr       INTEGER
    , nazMjesto CHAR(30)
    , sifZup    SMALLITN
);
DROP TABLE mjesto;
```
- upis novih n-torki u relaciju →
```
INSERT INTO mjesto
    VALUES (42000, 'Varaždin', 7);
```
- dohvat podataka iz relacije →
```
SELECT * FROM mjesto
    WHERE sifZup = 7;
```
- izmjena vrijednosti atributa →
```
UPDATE mjesto
    SET nazMjesto = 'VARAŽDIN'
    WHERE pbr = '42000';
```
- brisanje n-torki →
```
DELETE FROM mjesto
    WHERE sifZup = 7;
```

RELACIJSKA ALGEBRA
- Predikatni račun r = {t | F(t) }
- Obavljenje operacije ne utječe na operande, rezultat je relacija
- *Unijska kompatibilnost*: 1) relacije su istog stupnja
  2) korespondentni atributi su definirani nad istim domenama
- Skupovske operacije - operandi moraju biti unijski kompatibilni
  kao imena atributa u rezultatu se koriste imena atributa prvog operanda

¤ Unija
```
SELECT * FROM polozioMat
    UNION
SELECT * FROM polozioUpro;
```
¤ Presjek
```
SELECT * FROM polozioMat
    INTERSECT
SELECT * FROM polozioUpro;
```
¤ Razlika
```
SELECT * FROM polozioMat
    EXCEPT
SELECT * FROM polozioUpro;
```

¤ Dijeljenje - n-torka se pojavljuje u rezultatu AKKO za svaku n-torku iz r vrijedi
vrijedi sa se u relaciji r pojavljuje u kombinaciji sa SVAKOM n-torkom iz s

¤ Selekcija - n-torka se pojavljuje u rezultatu AKKO je vrijednost predikata F istina(true)
```
SELECT SELECTList FROM table
    [WHERE Condition];
```
¤ Projekcija - uzima se vertikalni podskup iz relacije r
```
SELECT DISTINCT tenor          - DISTINCT - osigurava da rezultat opet
              , grad             bude relacija (miče dupliće)
    FROM nastup;
```
¤ Kartezijev produkt - spajanje svake n-torke iz r s svakom n-torkom iz s
```
SELECT SELECTList                  SELECT SELECTList
    FROM table1, table2       ili      FROM table1 CROSS JOIN table2;
    [WHERE Condition];
```
- ako r i s imaju neke atribute koji se jednako zovu potrebno ih je preimenovati
```
SELECT    atr1 AS atribut1
        , atr2 atribut2        - AS se smije ispustiti
    FROM tablica;
```

```
    ¤ Natural join - spajanje na temelju jednakih vrijedosti istoimenih atributa
                        - ako nemaju istoimenih atributa onda se to ponaša kao kartezijev produkt
                    SELECT    tablica1.*                            SELECT    tablica1.*
                            , tablica2.atributKojegNemaU1   ili             , tablica2.atr
                        FROM tablica1 JOIN tablica2                 FROM tablica1 NATURAL JOIN tablica2;
                        USING(zajednickiAtribut);

    ¤ Spajanje uz uvjet - kartezijev produkt pa predikat == true
                    SELECT *                                        SELECT *
                        FROM tablica1, tablica2             ili         FROM tablica1 JOIN tablica2
                        WHERE nekiUvjet;                                    ON nekiUvjet;

    ¤ Equi-join - spajanje uz uvjet ali je operator isključivo operator jednakosti(=)
                - razlikuje se od prirodnog spajanja zato što se ovdje istoimeni atributi NE izbacuju
                    SELECT *
                        FROM tablica1, tablica2
                        WHERE atr1 = atribut1;

    ¤ Agregacija - COUNT, SUM, AVG, MIN, MAX, COUNT-DISTINCT, SUM-DISTINCT, AVG-DISTINCT
                - rezultantni atribut nema naziv pa se koristi operator preimenovanja

                    SELECT AVG(ocjena) AS prosjOcj
                        FROM ispit;

        ¤ Agregacija i grupiranje -
                    SELECT    nazPred
                            , akGod
                            , AVG(ocjena) AS prosjOcj
                            , MAX(ocjena) AS maxOcj
                        FROM ispit
                        GROUP BY nazPred, akGod;  - svi atributi koji se nalaze u listi za selekciju
                                                    a nisu argumenti agregatnih funkcija
                                                    *moraju* biti navedeni u GROUP BY dijelu naredbe

    ¤ Left outer join - sve n-torke lijeve relacije će se sigurno pojaviti u rezultatu
                        ako ne postoji tuple iz desne relacija s kojom se može spojiti
                        vrijednosti atributa 'desne' reclacije se pune NULL vrijednostima
                    SELECT lijeva.*, desna.*
                        FROM lijeva LEFT OUTER JOIN desna       - pojavljuju se sve n-torke lijeve tablice
                        ON atr = atribut;

    ¤ Right outer join - isto kao left outer join samo s druge strane
                    SELECT lijeva.*, desna.*
                        FROM lijeva RIGHT OUTER JOIN desna      - pojavljuju se sve n-torke desne tablice
                        ON atr = atribut;

    ¤ Full outer join - sve n-torke iz obje relacije će se sigurno pojaviti u rezultatu
                    SELECT lijeva.*, desna.*
                        FROM lijeva FULL OUTER JOIN desna       - pojavljuju se sve n-torke
                        ON atr = atribut;

    ¤ Natural outer join - izbacuju se istoimeni atributi kod lijevog i desnog, a kod punog
                    sa zadržavaju atributi obje relacije uz potrebno preimenovanje atributa

///3. PREZENTACIJA - NEPOTPUNE INFORMACIJE I NULL VRIJEDNOSTI
NULL VRIJEDNOSTI
    - Način pohrane NULL vrijednosti je nebitan
    - Ako je jedan od operanada u izrazu NULL rezultat je NULL
    - Ako je jedan od operanada u usporedbi NULL rezultat je unknown != true!!
    - Provjera je li nešto NULL
            SELECT * FROM tablica
                WHERE atr IS (NOT) NULL;    - ovo uvijek vraća true/false

    - U skupu je dopuštena jedna pojava NULL vrijednosti
    - Dvije n-torke su kopije ako su vrijednosti korespondentnih atributa iste ili NULL
    - Za obavljanje kartezijevog produkta NULL vrijednosti nemaju utjecaja
    - Agregatne funkcije zanemaruju NULL vrijednosti
        *osim* funkcije COUNT(*) - koja zanemaruje vrijednosti n-torke i samo broji retke


////4. PREZENTACIJA - SQL(1. DIO)
VRSTE OBJEKATA
    - Database, Table, Column, View (Virtualna tablica), Constraint (Integritetsko ograničenje)
        ,Index, Stored Procedure, Trigger

TIPOVI PODATAKA
    → INTEGER - 4 bajta, 2k
    → SMALLINT - 2 bajta
    → CHAR(m) - m je duljina stringa, ako je duljina stringa < m ostalo se puni ' '
    → VARCHAR(m) - varijabilni string, m je max duljina
        --NCHAR(m) i NVARCHAR(m) - nacionalne kodne stranice, deprecated
    → REAL - float
    → DOUBLE PRECISION - double
    → NUMERIC(m, n) == DECIMAL(m, n) - m je preciznost, n je broj znamenki iza dec. točke
    → DATE - datum, mogu se zbrajati i oduzimati
    → TIMESTAMP[(p)] - datum + vrijeme, p je rezolucija <= 6
    → TIME[(p)] - vrijeme (h, min, sec), p je rezolucija
    → INTERVAL[fields][(p)] - interval, ako definicija sadrži i fields i p
            fields mora sadržavati sekunde
```

```
|| konkatenacija
\% bilo koja kombinacija znakova
_ jedan znak
ECSAPE char - taj char poništava wildcard znak neposredno iza njega(pretvara ga u obični '%'/'_')

- Eksplicitna pretvorba tipova podataka
    CAST (expression AS type)           ili expression::type


FUNKCIJE(function expression)
    ▪ ABS(num_expression)
    ▪ MOD(dividend, divisor)
    ▪ ROUND(expression [, rounding_factor]) - rounding_factor default = 0
    ▪ SUBSTRING(source_string FROM start_position [FOR length]) - length default = do kraja
    ▪ UPPER(expression)
    ▪ LOWER(expression)
    ▪ TRIM(expression)
    ▪ CHAR_LENGTH(expression) - broj znakova *ne* uključujući trailing spaces
    ▪ OCTET_LENGTH(expression) - broj byte-ova uključujući trailing spaces
    ▪ CURRENT_DATE - DATE dobiven od OS-a
    ▪ CURRENT_TIME - TIME *s vremenskom zonom*
    ▪ CURRENT_TIMESTAMP - TIMESTAMP *s vremenskom zonom*
    ▪ CURRENT_USER - login korisnika koji radi
    ▪ EXTRACT(field FROM source) - funkcija vraća redni broj
       godine, mjeseca, dana, tjedna za DATE sadržan u source ili
       sata, minute, sekunde u danu za TIME sadržano u source
       - za field: year, month, day, hour, minute, second, week, dow, doy


    - nećemo koristiti vremenske zone pa ćemo sve svoditi na rezultate bez nje
            CURRENT_TIME ::TIME(x), CURRENT_TIMESTAMP ::TIMESTAMP(x)

    - Intervali - quantity unit[quantity unit] [direction]
                    ↓↓↓↓   ↓↓↓                   ↓↓↓↓
                    broj   ↓↓↓                  ago/ø
                           ↓↓↓
 microsecond, millisecond, second, minute, hour, day, week, month,
               year, decade, century, millennium

FROM Clause
    - ako se obavlja operacija spajanja i selekcija, uvjete spajanja treba navesti u ON dijelu,
       , a uvjete selekcije u WHERE dijelu SELECT naredbe
       - *bitno* za vanjsko spajanje

PARALELNO SPAJANJE
    - pokušavamo spojiti dvije tablice na osnovu više atributa
       pr. tablica student sa mjestom na osnovu mjesta rođenja && mjesta stanovanja
    - za tablicu mjesto definiramo dvije uloge i pomoću operanda preimenovanja tablice
       joj definiramo dvije uloge

       SELECT mbr, prezime
               , pbrRod, mjestoR.nazMjesto AS nazMjestoR
               , pbrStan, mjestoS.nazMjesto AS nazMjestoS
           FROM student
               JOIN mjesto AS mjestoR
                ON student.pbrRod = mjestoR.pbr
               JOIN mjesto AS mjestoS
                ON student.pbrStan = mjestoS.pbr;

REFLEKSIVNO SPAJANJE
    - pojedine n-torke iz relacije su povezane s drugim n-torkama iz *iste* relacije

       SELECT orgjed.sifOrgjed
               , orgjed.nazOrgjed
               , orgjed.sifNadorgjed
               , nadorgjed.nazOrgjed AS nazNadorgjed
           FROM orgjed
               LEFT OUTER JOIN orgjed AS nadOrgjed
                ON orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;

- preimenovanje osobe unutar upita - smije se

HAVING Clause
    - u rezultatu želimo samo one grupe koje zadovoljavaju određeni uvjet
        SELECT nazPredmet AS naziv
               , AVG(ocjena) AS prosjek
           FROM ispit
       GROUP BY nazPredmet
       HAVING AVG(ocjena) > 2;
               ↓↓↓↓↓↓
               u ovaj condition mogu ići ili agregatne funkcije ili atributi koji su dio
                   GROUP BY dijela

    - postavljaju se uvjeti na grupe nastale grupiranjem

ORDER BY Clause
    - sortiranje rezultata upita
    - mogu se koristiti i izrazi koji nisu u listi za selekciju
    - NULL vrijednosti su manje od svih drugih
```

////5. PREZENTACIJA - SQL(2. DIO)
SUBQUERIES
    - Podupit je upit koji je ugrađen u neki drugi upit → outer query
    - Može se ugraditi u: Condition u WHERE dijelu
                            Condition u HAVING dijelu
                            listi za selekciju outer querya
    - Može sadržavati sve osim ORDER BY

SCALAR SUBQUERY
    - Podupit čiji je rezultat jedna jednostavna vrijednost
                                    → relacija deg = 1, card = 1

```
SELECT *
    , (SELECT MAX(tezina)
        FROM teret) AS maxTezina
    FROM vozilo;
```

CORRELATED SUBQUERY
    - Ako se u podupitu koriste atributi za vanjskog upita onda su korelirani
    - Najčešće se korelirani podupit mora fizički izvršiti po jedanput za svaku n-torku iz
          vanjskog upita

```
SELECT oznStr, dopBrSati                        - vanjski upit uzima jednu n-torku
    FROM stroj                                      iz relacije stroj, na temelju te n-torke
    WHERE dopBrSati <                               i sadržaja relacije radStroja u podupitu
        (SELECT SUM(brSatiRada)                     se izračunava suma sati dotičnog stroja
            FROM radStroja                      - ovo se ponavlja za svaku n-torku iz stroj
            WHERE oznStr = stroj.oznStr);
```

    - U podupitu se mogu koristiti atributi vanjskog upita (obratn ne)
    - Ako se imena atributa/relacija poklapaju
        - imena u podupitu se odnose na podupit, imena u vanjskom se odnose na vanjski
    - Ako je potrebno riješiti dvosmislenost dovoljno je jednu od njih preimenovati

SINGLE-COLUMN SUBQUERY
    - Rezultat je relacija stupnja 1, s 0+ redaka
    - Može se ugraditi u: Condition u WHERE dijelu
                            Condition u HAVING dijelu
    - Trebaju se koristiti ključne riječi (NOT) IN, ALL, ANY === SOME

    ! WHERE expression relationalOperator ALL (subquery)
    ! WHERE expression NOT IN (subquery)
        → ako je u rezultatu nekih od ovih subquerya barem jedna NULL vrijednost rezultat selekcije
        NIKAD neće biti true

OPERATOR EXISTS
    - true ako rezultat podupita sadrži barem jednu n-torku

Kada se subquery stavlja u select listu?
    U situacijama kada u SELECT listu treba uključiti izraz čiji su
    uvjeti ili način grupiranja različiti od uvjeta u vanjskom upitu.

NAREDBE ZA IZMJENU SADRŽAJA RELACIJE
    INSERT
        - Uvijek koristi verziju s imenima atributa!
            - oni atributi koji se ne navedu idu na default ili NULL

```
        INSERT INTO tablica(
              atr1
            , atr2)
            VALUES (
              'v1'
            , 'v2');
```

        - SERIAL   - samoinkrementalni integer - super za ID
                    - ako je neki ID brisanjem postao slobodan ne koristi se ponovno

    DELETE
        - Ako se npiše WHERE dio naredbe onda se brišu sve n-torke za koje je uvjet true
            , a ako se ne napiše onda se brišu sve n-torke

```
        DELETE FROM table
            WHERE idAtr = '3';
```

    UPDATE
        - Mijenja vrijednosti atributa postojećih n-torki

```
        UPDATE bodovi
            SET bodMI = bodMI + 10              - ovdje se može koristiti i CASE
            WHERE bodLab + bodMI + 10 <= 100;
```

//////6. PREZENTACIJA - OBLIKOVANJE SHEME BAZE PODATAKA(1.DIO)
- Karakteristike loše koncipirane sheme baze podataka
    (objašnjeno na primjeru narudžbi)
        ▪ redundancija (čije su posljedice)
            ø anomalija unosa
                - ne mogu se unijeti podaci o artiklima koji nisu naručeni
                - svaki put kada se unosi novi podatak o narudžbi porebno je ponovo
                    upisivati adresu i naziv i mjesto dućana - pri tome treba paziti da
                    budu konzistentni za svaki dućan

            ø anomalija izmjene
                - ako neki dućan promijeni adresu potrebno je to mijenjati na više mjesta

            ø anomalija brisanja
                - brisanjem svih narudžbi za artikl se brišu svi podaci o artiklu

        ▪ pojava lažnih n-torki
            - odvojimo artikle u odvojenu tablicu sa stupcima (sifArtikl, nazArtikl, kolicina)
                i ako to prirodno spojimo s tablicom narudzba dobit ćemo neke n-torke koje nisu
                trebale postojati tamo (n-torke viška)

- Kako poboljšati loše koncipiranu relacijsku shemu?
    - proučavanjem značenja podataka (semantike)
    - proučavanjem zavisnosti među podacima
    - uvođenjem ograničenja koja su ovisna o semantici podataka

FUNKCIJSKE ZAVISNOSTI
    - Def: Neka je r relacija sa shemom R i neka su X i Y skupovi atributa X, Y podskup od R.
            Funkcijska zavisnost X → Y vrijedi na shemi R ukoliko u svim dopuštenim stanjima relacije
            r(R) svaki par n-torki t1 i t2 koje imaju jednake X vrijednosti također imaju jednake
            Y vrijednosti odnosno
                                    t1(X) = t2(X) → t1(Y) = t2(Y)

    - FZ proizlaze iz značenja podataka(semantike), a ne iz trenutnog stanja tablice!
    - Ako postoje dvije n-torke s istim jednim, a različitim drugim atributom slijedi da
        FZ prviAtr → drugiAtr ne vrijedi. Ne možemo zaključiti suprotno.

ARMSTRONGOVI AKSIOMI
    Neka je R relacijska shema, neka su X, Y, Z skupovi atributa i neka vrijedi: X, Y, Z <= R.
    A-1 REFLEKSIVNOST -
        Ako je Y <= X tada X → Y
    A-2 UVEĆANJE -
        Ako u shemi R vrijedi X → Y, tada vrijedi i XZ → Y
    A-3 TRANZITIVNOST
        Ako u shemi R vrijedi X → Y i Y → Z tada vrijedi i X → Z

    Pravila koja proizlaze iz ovih aksioma:
        P-1 PRAVILO UNIJE
            Ako u shemi R vrijedi X → Y i X → Z tada vrijedi i X → YZ
        P-2 PRAVILO DEKOMPOZICIJE
            Ako u shemi R vrijedi X → YZ tada vrijedi i X → Y
        P-3 PRAVILO O PSEUDOTRANZITIVNOSTI
            Ako u shemi R vrijedi X → Y i VY → Z tada vrijedi i XV → Z

    PRAVILO O AKUMULACIJI
        Ako u shemi R vrijedi X → VZ i Z → W tada vrijedi i X → VZW

KLJUČ RELACIJE
    - skup atributa koji nedvosmisleno određuje n-torke relacije
    - ima svojstvo da funkcijski određuje atribute u preostalom dijelu relacije i minimalan je
    - Ključevi:
        mogući ključevi (candidate)
        primarni ključ (primary) - odabire se jedan od candidate
        alternativni ključ (alternate) - preostali candidate

STRUKTURA RELACIJE
    - Relacijska shema se sastoji od:
        atributa koji su dio ključa {matBR, OIB}
        atributa iz zavisnog dijela relacije {ime, prezime}

//////7. PREZENTACIJA - OBLIKOVANJE SHEME BAZE PODATAKA(2.DIO)
NORMALIZACIJA
    - Cilj: ukloniti redundanciju i spriječiti pojavu lažnih n-torki
    - NORMALNE FORME
            1) Prva NF
                    Def: Relacijska shema je u 1NF ako:
                                    - domene atributa sadrže samo jednostavne vrijednosti
                                    - vrijednost svakog atributa je samo jedna vrijednost iz domene
                                    - neključni atributi ovise o ključu relacije
                    Norm: × izdvajanjem atributa u posebnu relaciju (nova tablica djeca kojima je sifra matbr roditelja)
                            × promjenom ključa (imenaDjece → imeDjeteta)

            2) Druga NF
                    Def: Relacijska shema je u 2NF ako:
                                    - je u 1NF
                                    - ako je svaki atribut iz zavisnog dijela potpuno funkcijski ovisan o
                                        svakom ključu relacije   ↓↓↓↓
                                                                        Y je potpuno funkcijski ovisan o X ako
                                                                        X → Y i ne postoji pravi podskup od X koji
                                                                        funkciiski određuie Y

Norm. nastaju: • relacijska shema koja sadrži skup atributa koji su bili nepotpuno funkcijski ovisni o ključu i dio ključa o kojem su potpuno funkcijski
• relacijska shema koja sadrži ključ originalne relacije i skup atributa koji su potpuno funkcijski ovisni o ključu

3) Treća NF
Def: Relacijska shema je u 3NF ako:
- je u 1NF
- ako nijedan atribut iz zavisnog dijela nije tranzitivno funckijski ovisan o bilo kojem ključu relacije ↓↓↓↓
Z je tranzitivno ovisan o X ako X → Y
Y !→ X i Y → Z

Norm. nastaju: • relacijska shema koja sadrži skup atributa relacijske sheme OSOBA koji su tranzitivno ovisni o ključu (nazMjesto) te srednji skup atributa uočene tranzitivne zavisnosti (postBr)
• relacijska shema koja sadrži ključ relacijske sheme OSOBA (matBr) i neključne atribute relacijske sheme OSOBA koji nisu tranzitivno ovisni o ključu
- normalizacija na 2NF nije nužna za normalizaciju na 3NF jer se nepotpune FZ mogu promatrati kao tranzitivne NF, *ali* bi ipak trebalo 1→2→3

4) Boyce-Coddova NF

5) Četvrta NF
6) Projekcijsko-spojna NF

POSTUPCI NORMALIZACIJE
× DEKOMPOZICIJA
- Početne relacije se dekomponiraju na temelju uočenih funkcijskih zavisnosti
R se zamjenjuje s R1, R2, .. Rn pri čemu su Ri <= R
- Lossless decomposition - ako je prirodno spajanje svih r1 do rn = r
- Razlaganje relacije bez gubitaka na dvije projekcije:
¤ projekcije imaju zajedničke atribute
¤ zajednički atributi su ključ u barem jednoj od projekcija
× SINTEZA
- Zadan je skup atributa i nad njima skup FZ iz kojih se sintetiziraju R koje zadovoljavaju 3NF

////////9. PREZENTACIJA - FIZIČKA ORGANIZACIJA PODATAKA
UVOD
- Pojam fizičke organizacije podataka:
- stukture primjenjene pri pohrani
- metode pristupa podatcima

- Fizička organizacija ne utječe na na rezultate, ali utječe na učinkovitost

- Glavna memorija - relativno skupa, nedovoljan kapacitet, volatilnost
- Sekundarna memorija - uglavnom se koristi ta, prijelaz glavna ←→ sekundarna se obavlja u blokovima
- sporija negdje 10^5 puta >> vrijeme obrade podataka
- Ciljevi - minimizirati broj UI operacija
- minimizirati prostor za pohranu
- različite metode pristupa na temelju ključa pretrage

- Strukture pohrane: ø heap
ø sorted
ø hash
ø index-sequential
ø B-stablo

NEPOREDANI HEAP
- Zapis ide na bilo koje slobodno mjesto, dohvat podataka je linearan
- Primjena: mala tablica ili u relacijama gdje se zapisi obrađuju slijedno
- Dohvat prema ključu:
¤ primarni / alternativni ključ - θ(n/2) gdje je n broj fizičkih blokova
- O(n) ako zapis ne postoji
¤ ostali ključevi - θ(n)

B-STABLA
B+Stabla - balansirana stabla
ORMARIĆI LADICE KARTICE

- Struktura internog čvora B+Stabla reda n:
- Najviše n kazaljki, a najmanje $\lceil \frac{n}{2} \rceil$
↓↓
ne vrijedi za korijen

- Struktura lista B+Stabla
- Sadrži najviše n-1, a najmanje $\lceil \frac{n-1}{2} \rceil$
vrijednosti ključa i pripadnih kazaljki za zapise
- Zadnji sadrži pointer na sljedeći list

- Algoritam za pronalaženje zapisa - obično pretraživanje stabla
- Algoritmi za dodavanje i brisanje osiguravaju pravilnu popunjenost i balansiraju stablo
  , može se promijeniti dubina stabla. Ako se mijenjaju atributi na temelju kojih se stablo
  gradilo onda je to zapravo brisanje i ponovno upisivanje zapisa.

- Učinkovitost pretrage B+Stabla -
  ¤ red stabla n se odabire tako da jedan čvor stane u jedan fizički blok
    za dohvat jednog čvora je potrebna jedna UI operacija

PR) Za broj n-torki m = 1 000 000, a red stabla = 70, ukupni broj razina je u najlošijem slučaju 4.

1. točno 1 čvor
2. najmanje 2 čvora, najmanje 70 kazaljki
3. najmanje 70 čvorova, najmanje 2450 kazaljki
4. najmanje 2450 čvorova, najmanje 85750 kazaljki
5. najmanje 85 750 čvorova, najmanje 3 001 250 kazaljki

↓↓↓↓

ali trebamo spremiti samo 1 000 000 n-torki

SQL: INDEXI

CREATE INDEX osoba_prez ON osoba(prez) - kreira B stablo osoba_prez za tablicu osoba nad atr prez

CREATE UNIQUE INDEX osoba_prez ON osoba(mbr) - kreira B stablo osoba_mbr za tablicu osoba nad atr mbr

↓↓↓

atributi koji moraju imati jedinstvene vrijednosti u tablice

- Ako već imamo tablicu, a u njoj su ljudi s istim mbr sustav neće kreirati index i javlja grešku.
  Isto vrijedi i ako se nakon kreiranja indexa pokuša dodati osoba s postojećim mbr.

DROP INDEX osoba_prez; - uništavanje indexa

- Nad istom relacijom se može izgraditi više indeksa
- Dohvat prema atributima za koje nismo izgradili indeks pretraživanje ide linearno.

- Treba li izgraditi indeks za sve atribute? NE
  → Indeksi zauzimaju prostor
  → Prilikom izmjene ili brisanja to se mora mijenjati u svim stablima

- Za koje treba izgraditi indeks?
  → za one koji se često koriste za selekciju
  → oni koji se koriste za spajanje relacija
  → za atribute prema kojima se često obavlja sortiranje/grupiranje

- Za koje ne treba graditi indeks?
  → za one koji imaju mali broj različitih vrijednosti npr spol
  → ako u relaciji postoji veliki broj upisa / izmjena / brisanja indeksa
    - preporuča se izbrisati indekse, obaviti operacije, pa ih opet složiti
  → ako relacija ima malo n-torki npr zupanija

SLOŽENI INDEKSI
- Indeksi nad više atributa

CREATE INDEX stud_prez_ime ON stud (prez, ime)

Problem:
¤ može se koristiti za upite (prez, ime) (ime, prez) i (prez), ali NE i za upit (ime)
¤ sortiranje prema (prez, ime), (prez desc, ime desc), ali NE i (prez desc, ime)

##
---------------------------------- DRUGI CIKLUS ----------------------------------
##

//////////11. PREZENTACIJA - ER MODEL BAZE PODATAKA 1. DIO

ER MODEL (Entity-relationship model)
- postrelacijski model
- omogućuje eksplicitni prikaz veza koje u sebi sadrže važne semantičke informacije
- svrhe: dizajn nove baze
  dokumentiranje postojeće baze podataka

Entitet = bilo što što posjeduje neku suštinu
Skup entiteta = grupa sličnih entiteta
Skup veza = metematička relacija između n entiteta
Uloga = funkcija koju skup entiteta obavlja u skupu veza
Atribut = funkcija koja preslikava iz skupa entiteta/veza u skup vrijednosti

TERMINOLOGIJA

Chen:	entitet	skup entiteta	veza	skup veza
	↓↓	↓↓	↓↓	↓↓
Teorey:	instanca entiteta	entitet	instance veza	veza

Stupanj veze = broj entiteta koje povezuje dotična veza
Spojnost veze = ograničenje preslikavanje pojedinačnih entiteta (ONE / MANY)

PRESLIKAVANJE
- međusobni odnos entiteta u vezi

one-to-one, one-to-many, many-to-one, many-to-many

SLABI ENTITETI
- entiteti koji ne mogu postojati sami za sebe
- treba imati "vlasnika"
- entitet može biti egzistencijalno slab, ali ne mora biti indentifikacijski slab
  ,ako je identifikacijski slab mora biti egzistencijalno slab
  ↓↓
  ø kod određivanje indentifikatora nisu im dovoljni vlastiti atributi

Atributi veza : veza sigurno nasljeđuje primarne ključeve entiteta koje spaja

KLJUČEVI VEZA
- povezanost entiteta se opisuje kao odnos među ključevima entiteta
- ključevi veza definirani su pomoću ključeva entiteta koje povezuju i njihove SPOJNOSTI

Kako odrediti ključ veze?
TEOREY-EVA DEFINICIJA:
U vezi koja povezuje entitete
E1, ..., Ek, ... , Em
spojnost = 1 entiteta Ek znači da za svaku vrijednost svih entiteta E1, ..., Em
osim Ek, UVIJEK POSTOJI TOČNO JEDNA VRIJEDNOST Ek

¤ Može se reći da tada vrijedi FZ
$\%\bigcup_{j=1}^{m} K_j \setminus K_k \rightarrow K_k$
gdje su skupovi Kj ključevi entiteta E1, ..., Em

- ako na nekoj nogici piše 1 pr student i mjesto ključ veze je onaj atribut gdje nema brojke 1 - matBrSt

- Ako npr imamo N, N ključ veze je kombinacija ključeva s obje strane - KOMPOZITNI KLJUČ
¤ Ili ako veza ima vlastite atribute ključ veze su atributi stvari koje povezuje
ali! on funkcijski određuje vlastite atribute veze

¤ Međutim, ima slučajeva gdje ključ može sadržavati i druge atribute

DEGENERIRANI ENTITET! - postoji samo da bi posudio ključ vezi

Alternativa: veza postaje slabi entitet

Veza 1:N → preslikavanje u relacijski model
1) svaki entitet i svaku vezu pretvaramo u relacijsku shemu (tablicu)
2) unija relacijskih shema s jednakim ključevima

Veza N:N → preslikavanje u relacijski model
1) svaki entitet i svaku vezu pretvaramo u relacijsku shemu (tablicu)

Refleksivna veza 1:N → relacijski model
1) preimenovati jedan od atributa veze
2) veza 1:N

VLASTITI ATRIBUTI ENTITETA =
- vlastiti atribut entiteta je atribut koji opisuje znanja o entitetu koja se pripisuju isključivo samom entitetu, a nikako u vezi s drugim entitetima
- entitete opisujemo isključivo tim entitetima
- postBrStan nije vlastiti atribut zato što opisuje vezu s mjestom
¤ IZNIMKA - identifikacijski slabi entiteti, osim svojih vlastitih posjeduju i atribute primarnog ključa vlasnika

Redundante veze =
- više puta opisujemo istu informaciju
- petlje nisu uvijek redundante veze, ali ih treba provjeriti

Paralelne veze =
- entiteti povezani na više od 2 načina
- ako imamo dvije veze s istim ključevima to rješavamo preko uloga (preimenovanja)

Homogeno stablo =
- stablasta struktura gdje svaki čvor može imati 0+ djece i 0 ili 1 roditelja
¤ Govoreća šifra: pametna šifra iz koje se isto mogu iščitati informacije
- šifre i identifikatori ne smiju biti govoreće šifre

OBLIKOVANJE ER MODELA
1) definiranje entiteta
2) definiranje veza → ITERATIVAN
3) definiranje atributa entiteta POSTUPAK
4) definiranje atribut veza

MODEL BAZE PODATAKA
Sadrži opise:
- entiteta
- veza
- atributa entiteta
- atributa veza

Karakteristike dobrog modela:
- opisuje suštinu, dobro apstrahira
- sveobuhvatan
- neredundantan
- fleksibilan
- razumljiv

Posebno pazi na:
- različito shvaćanje istih stvari - (korisnicima)
- praćenje promjena u vremenu
- jednakost - uopćavanje

////////////12. PREZENTACIJA - ER MODEL BAZE PODATAKA 2. DIO

Homogena mreža =
- čvor može imati i 0..N roditelja za razliku od stabla
- N:N veza uvijek ostaju odvojene tablice

TERNARNE VEZE
▪N:N:N =
- opisuju *istovremeni* odnos triju elemenata
- ne može se (bez gubitka informacija) zamijeniti trima binarnim vezama
- interpretacija: ne može se opisati jednom rečenicim
" Jedan robot jednu operaciju može raditi nad 0..N modela.
Nad jednim modelom jednu operaciju može raditi 0..N robota.
Jedan robot nad jednim modelom može raditi 0..N operacija. "

TEOREY-EVA DEFINICIJA:
U vezi koja povezuje entitete
E1, ..., Ek, ... , Em
spojnost = 1 entiteta Ek znači da za svaku vrijednost svih entiteta E1, ..., Em
osim Ek, UVIJEK POSTOJI TOČNO JEDNA VRIJEDNOST Ek

¤ Može se reći da tada vrijedi FZ
$\%\bigcup_{j=1}^{m} K_j \setminus K_k \rightarrow K_k$
gdje su skupovi Kj ključevi entiteta E1, ..., Em

▪N:N:1 =
" Jedan student jedan predmet može položiti kod jednog nastavnika.
Kod jednog nastavnika jedan predmet može položiti 0..N studenata.
Jedan student kod jednog nastavnika može polagati 0..N predmeta. "
- u ovom slučaju ključ je kompozitni i dolazi od onih entiteta s N na nogicama (student, predmet)

ø ternarna veza N:N:1 → relacijski model
1) svaki entitet postaje vlastita tablica
2) veza postaje entitet s kompozitnim ključem

▪N:1:1 =
" Jedan student na jednom smjeru smije imati jednog mentora.
Jedan mentor na jednom smjeru smije imati 0..N studenata.
Jedan mentor na s jednim studentom se može naći na jednom smjeru."

- ključ veze:
KOMPOZITNI KLJUČ: onaj gdje je N, i onda možemo izabrati ključ od neke od nogica s 1

▪1:1:1 =
- svaki put kada imamo dva atributa točno znamo treći
efektivno samo jedan entitet

SPECIJALIZACIJA I GENERALIZACIJA
Specijalizacija - slaganje podklasa
Generalizacija - slaganje nadklasa

Ekskluzivna specijalizacija - entitet može biti samo jedna podklasa

→ relacijski model:
specijalizacije nemaju vlastite ključeve

Neekskluzivna specijalizacija - podklase mogu imati presjek

→ relacijski model:
ista osoba može biti zastupljena u više tablica

///////////////14. PREZENTACIJA - PRIVREMENE I VIRTUALNE TABLICE

VRSTE TABLICA:
1) TEMELJNA TABLICA (base table)
- njena shema i sadržaj su TRAJNO pohranjeni u bazi podataka
- obavljanjem naredbe CREATE TABLE se u rječnik podataka pohranjuju metapodaci:
ø naziv tablice
ø nazivi i tipovi atributa
ø integritetska ograničenja
ø ostali metapodaci
- shema i sadržaj su postojani → pohranjeni su na neograničeno vrijeme
- mijenjaju se, ali samo uz eksplicitne naredbe

- SQL-session →
    - kontekst u kojem jedan korisnik obavlja niz SQL naredbi putem jedne veze prema sustavu
        - započinje kada se korisnik spoji sa sustavom
        - završava kada korisnik prekine vezu

2) PRIVREMENA TABLICA (temporary table)
- njena shema i sadržaj su PRIVREMENO pohranjenu u bazi
- stvara se naredbom CREATE TEMP table, uklanja sa DROP TABLE ili krajem sessiona
    - nije moguće definirati ograničenje referencijskog integriteta
- vidljiva je isključivo tijekom jednog sessiona
- koriste se npr za pohranu međurezultata složenih upita

skraćena sintaksa
```
CREATE TEMP TABLE prosjek(sifPred, prosOcj)
    AS
    SELECT sifPred, AVG(ocj) AS prosOcj
    FROM polozenIspit
    GROUP BY sifPred;
```

! sadržaj tablice prosjek se neće promijeniti prilikom dodavanja vrijednosti u položeni ispiti - vodi računa o tom!

3) VIRTUALNA TABLICA(view)
- tablica u kojoj su shema i sadržaj definirani izrazom relacijske algebre čiji su operandi temeljne ili virtualne tablice
- sadržaj se dinamički određuje u trenutku obavljanja operacije nad virtualnom tablicom - ovisi o trenutnom stanju temeljnih tablica

→ rješava problem zastarijevanja podataka
```
CREATE VIEW prosjek(sifPred, prosOcj)
    AS
    SELECT sifPred, AVG(ocj) AS prosOcj
    FROM polozenIspit
    GROUP BY sifPred;
```

- obavljanjem ove naredbe se u riječnik pohranjuje samo definicija tablice
    - ¤ sadržaj se određuje tek za vrijeme izvršavanja upita

uklanja se iz tablice naredbom DROP VIEW nazivVirTablice

- mogu se koristiti na svim mjestima gdje se koriste temeljne tablice

view != temp
- ▪ definicija view je trajno pohranjena u bazi
- ▪ u scopeu svih sessiona je

- postoji i TEMP VIEW u postgreu

ATRIBUTI VIRTUALNE TABLICE:
- ako se nazivi atributa ne zadaju u definiciji određeni su nazivima atributa iz SELECT naredbe
- tipovi podataka proizlaze iz tipova podataka atributa

```
CREATE VIEW zadrani2 (matBr, imeSt, prezSt) => zadani
    vs
CREATE VIEW zadrani1                        => ne zadani
```

- ako se u definiciji koriste izrazi nazive atributa EKSPLICITNO imenovati!!!

MATERIJALIZIRANE VIRTUALNE TABLICE:
- SUBP fizički pohranjuje sadržaj virtualne tablice, kada se promijeni sadržaj neke od temeljnih tablica podaci se automatski osvježavaju
    - ↓→ ovo skoro pa nitko ne radi zapravo

Implementacija virtualnih tablica = modifikacija upita
definicija tablice se ugrađuje u SQL upit umjesto imena tablice pa se zapravo to izvršava

UPDATE, INSERT I DELETE- ne mijenja se sadržaj virtualne tablice, mora se promijeniti sadržaj temeljnih tablica

Problem migrirajućih n-torki:
- n-torka se pojavljuje u virtualnoj tablici ako zadovoljava uvjet iz definicije virtualne tablice
- izmjenom temeljnih tablica n-torke mogu migrirati iz jedne u drugu virtualnu

- ▪ Rješenje!
    WITH CHECK OPTION
    ↓→ za one tablice koje se koriste u naredbama koje mijenjaju podatke jer se tada ne dopušta izmjena ili unos n-torke putem virtualne tablice ukoliko ta n-torka više ne bi pripadala toj tablici putem koje je izmijenjena ili unesena

Neizmjenjive virtualne tablice:
SUBP onda ne može promijeniti sadržaj virtualne tablice nego mora promijeniti sadržaj temeljnih tablica koje se koriste u definiciji te virtualne tablice

- ako je virtualna tablica definirana tako da SUBP nije u stanju
jednoznačno odrediti koje operacije treba obaviti na temeljnim
tablicama, tada je virtualna tablica NEIZMJENJIVA(non-updatable)

Izmjenjive virtualne tablice:
Tablica je IZMJENJIVA ako u glavnom dijelu SELECT definicije koristi
atribute iz samo jedne relacije i pri tome:
ø ne sadrži DISTINCT
ø nema izraze (osim trivijalnih izraza koji sadrže samo ime atributa)
ø izostavljeni atributi nemaju NOT NULL ili imaju DEFAULT vrijednost
ø nema spajanja / unije
ø nema GROUP BY i HAVING

- ova ograničenja se ne odnose na eventualne podupite u WHERE dijelu

///////////////15. PREZENTACIJA - TRANSAKCIJE I OBNOVA

Zadaća SUBP - zaštita podataka
ø zaštita integriteta
ø zaštita pristupa podacima - autorizacija, sigurnost
→ pohranom pravila u rječnik podataka
ø upravljanje istodobnim pristupom podacima
ø obnova u slučaju pogreške ili uništenja BP
→ potpora za upravljanje transakcijama

TRANSAKCIJE
TRANSAKCIJA =
- jedinica rada nad bazom podataka
- sastoji se od niza logički povezanih izmjena
- početak transakcije - BEGIN WORK
- završetak transakcije - COMMIT WORK / ROLLBACK WORK
↓↓ ↓↓
uspješno neuspješno

Upravljanje transakcija - dio sustava koji brine o obavljanju transakcija i osigurava
zadovoljavanje svih poznatih pravila integriteta

```
CREATE PROCEDURE prijenos(s_racuna INTEGER
                        , na_racun INTEGER
                        , iznos DECIMAL (8, 2))
DEFINE pom_saldo DECIMAL(8, 2);
BEGIN WORK;
    UPDATE racun SET saldo = saldo - iznos;
        WHERE br_racun = s_racuna;
    UPDATE racun SET saldo = saldo + iznos;
        WHERE br_racun = na_racun;
    SELECT saldo INTO pom_saldo FROM racun
        WHERE br_racun = s_racuna;
    IF pom_saldo < 0 THEN ROLLBACK WORK;
    ELSE COMMIT WORK;
    END IF;
ENDPROCEDURE
```

!Ako granice transakcije nisu eksplicitno definirane naredbama granice se određuju
implicitno t.d. je svaka SQL naredba jedna transakcija.

Stanja transakcija:
¤ AKTIVNA
¤ DJELOMIČNO ZAVRŠENA
¤ NEISPRAVNA
¤ NEUSPJEŠNO ZAVRŠENA
¤ POTVRĐENA - sve akcije koje je transakcija obavila prije ovog
stanja su privremene, a tek nakon ovog postaje trajna i ne može biti poništena

ACID =>
Atomicity - nedjeljivost
Conistency - baza transakcijom prelazi iz jednog u drugo trajno stanje
Isolation - ako se paralelno obavlja više transakcijema intersekcija
Durability - ako je transakcija potvrđena to mora biti trajno u sustavu

→ POSTIZANJE OVIH SVOJSTAVA
o A i D brine podsustav za upravljanje transakcijama i obnovu baze u slučaju
razrušenja
o C i I brine podsustav za upravljanje istodobnim pristupom

OBNOVA BAZE PODATAKA
Obnoviti bazu podataka - dovesti bazu u najnovije stanje za koje se pouzdano zna da je
bilo ispravno

Pravilo koje omogućuje obnovu = REDUNDANCIJA
- svaki se podatak mora moći rekonstruirati iz nekih drugih informacija
pohranjenih negdje drugdje u sustavu
- mirroring, backup, logical log

Postupak obnove
1) Stvaranje arhivske kopije - periodičko spremanje sadržaja na arhivski medij
2) Svaka izmjena u bazi podataka evidentira se u logičkom dnevniku izmjena
↓↓
stara vrijednost zapisa, nova vrijednost zapisa, korisnik, vrijeme
!!!izmjena se prvo zapiše u log pa se onda izvodi!!!
↓→ ZAŠTO? da se ne dogodi da se provela izmjena i nešto se skrši

- ZABITI da se ne dogodi da se provede izmjena i nešto se skrije

ø omogućavaju:
poništavanje i obnavljanje transakcija

Zapisivanje podataka iz RAM na disk
Kad nastane kvar: ako je baza potpuno uništena učitiva se najnovija arhivska kopija,
pomoću dnevnika izmjena se izvode promjene od zadnje promjene
ako baza nije potpuno uništena, ali je nepouzdana pomoću dnevnika
izmjena se poništavaju izmjene

Tipovi pogrešaka:
ø POGREŠKE TRANSAKCIJA - posljedica (ne)planiranog prekida
→ poništavaju se sve izmjene prekinutih transakcija unatrag
do naredbe BEGIN WORK
a) POGREŠKE KOJE OTKRIVA APLIKACIJA:
postoji eksplicitni ROLLBACK WORK
b) POGREŠKE KOJE NE OTKRIVA APLIKACIJA:
npr pokušaj kršenja integritetskih ograničenja
- program se prekida i SUBP poništava transakciju
, eventualno daljni tijek ovisi o klijentu npr pgadmin

ø POGREŠKA RAČUNALNOG SUSTAVA - baza je nekonzistentna - prekid rada u trenutku
izvođenja transakcija
→ sve transakcije koje su se odvijale u trenutku kvara moraju biti poništene
- pretražuju se one koje imaju BEGIN ali ne i COMMMIT/ROLLBACK - sporo!
- KONTROLNA TOČKA - sustav pohrani kontrolne informacije
~ sadržaj spremnika dnevnika
~ zapisa kontrolne točke u dnevnik
ø lista svih aktivnih transakcija u tom trenutku
ø adresu zadnje naredbe za svaku od tih transakcija
~ adresa zapisa kontrolne točke iz dnevnika ide u restart file
~ sadržaj spremnika baze u bazu
? Kako će sada sustav ponovno pokrenuti bazu?
- naći će zapis zadnje kontrolne točke i listu aktivnih transakcija
u tom trenutku
- stvara dvije liste - za poništavanje i ponovno obavljanje
¤ trenutne transakcije stavlja u listu za poništavanje
¤ kad nađe početak neke transakcije stavlja ju u poništavanje,
kada nađe njen kraj premješta ju u obnavljanje
- SUBP ne prima niti jedan zahtjev dok ovo sve traje
→ održavanje svojstva izdržljivosti - važno je da je očuvan dnevnik i da se
spremnik dnevnika zapisuje na disk prije potvrđivanja transakcije

→ Što ako dođe do pogreške u procesu?
- Učinak ponovnog obavljanja, bez obzira na to koliko se puta pokrenuo,
mora imati isti učinka kao da se pokrenuo samo jednom → analogno za poništavanje

Ponovno obavljanje = forward recovery
Poništavanje = backward recovery

ø KVAR MEDIJA ZA POHRANU - baza je fizički uništena
→ obnova baze pomoću posljednje arhivske kopije
→ pomoću najnovijeg dnevnika obavljaju se transakcije koje su bile provedene od
trenutka arhiviranje

- ako je posljednja arhivska kopija pokvarena uzima se predzanji
set (arhiva_n-1, dnevnik_n-1, dnevnik_n)
- ako je dnevnik pokvaren →↓
=> CIKLIČKA IZMJENA LOGIČIH DNEVNIKA
Dnevnik može biti jako velik - traje između dvije arhive
rj: dijeli se na odsječke, kada se jedan popuni sprema se na
arhivski medij, ALI!!! mora ostati i online za ROLLBACK dok se ne
završe sve transakcije koje se nalaze u njemu

Dodatni problemi -
VREMENSKI PREDUGE TRANSAKCIJE
- zaključa ostale resurse ostalim korisnicima
- timeout warning
PREVELIKE TRANSAKCIJE
- stvara puno izmjena, puni dnevnik bezveze i ako sustav
padne gube se sve izmjene

Mehanizmi za postizanje velike dostupnosti =
(1) mirroring
- postoji zrcalna kopija diska
- promjene se provode istovremeno na primarnom i zrcalnom području
- u slučaju greške korisnik radi na ispravnom od ta dva, pokvareni se obnavlja
na temelju ispravnog
(+) visoka dostupnost
(-) ne može pomoći pri poništavanju transakcija

(2) on-line backup
- arhiviranje se obavlja tijekom rada korisnika
- stanje baze u arhivskoj opiji je konzistentno

(3) incremental backup
- arhiviranje je naporno - cilj je skratiti to
- omogućuje stvaranje arhiva različitih razina

~ razina 0 - kopija čitave baze podataka cca 1 x mjesečno
~ razina 1 - promjene nastale nakon arhive 0 cca 1 x tjedno
~ razina 2 - promjene nastake nakon arhive 1 cca 1 x dnevno

(4) replication
- slično zrcaljenju, ali se obavlja na dva sustava
- sinkrona/asinkrona
- primarna namjena = visoka dostupnost

////////////////16. PREZENTACIJA - UPRAVLJANJE ISTODOBNIM PRISTUPOM
Cilj: podržati multiuser SUBP
↓→ osigurati - maksimalnu dostupnost podacima
- ispravnost tih podataka

Zbog čega je istodobni pristup nužan?
- transakcije koriste U/I i CPU operacije - različiti resursi
- povećava se throughput i utilization, smanjuje se avg response time

ISTODOBNI PRISTUP I TRANSAKCIJA:
- Rezultat transakcije ne smije ovisiti o tome odvijaju li se istodobno i druge transakcije.
- Koja od ACID svojstava su ugrožena prilikom istodobnog pristupa?
Isolation - problem je ako 2+ transakcija pristupaju istom podatku a njihove
se aktivnosti (čitanje/pisanje) isprepliću - kao nezavisnost dretvi iz OS

→ OSIGURAVANJE IZOLACIJE!
Karakteristični problemi istodobnog pristupa
P1 - prljavo čitanje/read uncommited
¤ transakcija čita nepotvrđene podatke druge transakcije
P2 - neponovljivo čitanje/non repeatable read
¤ ponovnim izvršavanjem ISTE SELECT naredbe unutar transakcije se dobiva drukčiji rezultat
¤ to izaziva potvrđena UPDATE
P3 - sablasne n-torke/phantom rows
¤ ponovnim izvršavanjem ISTE SELECT naredbe unutar transakcije se dobiva drukčiji rezultat - ali s drukčijem brojem redaka
¤ to izazivaju potvrđene INSERT/DELETE
P4 - izgubljena izmjena/lost update
pr

```
        A                              B
čita br_mjesta = 20                    /
        /                      čita br_mjesta = 20
br_mjesta = 20 - 3                     /
piše br_mjesta = p = 17        br_mjesta = 20 - 1
        /                  piše br_mjesta = p = 19

        br_mjesta = 19 → TREBAO BI BITI 16
```

Rješenja:
a) PROTOKOL ZASNOVAN NA ZAKLJUČAVANJU
- ekvivalentno semaforima

▪ ključ za pisanje/izmjenu = WRITE LOCK, EXCLUSIVE LOCK
→ transakcija T1 zaključa objekt za pisanje, niti jedna druga transakcija ga ne može zaključati dok ga T1 ne otključa
→ taj ključ postavlja svaka operacija izmjene - INSERT, UPDATE, DELETE

▪ ključ za čitanje = READ LOCK, SHARED LOCK
→ transakcija T1 (naredbom SELECT) zaključa objekt za čitanje
- bilo koja druga ga može zaključati za čitanje, ali ne i za pisanje

- nije dovoljan za serijalizabilnost
TWO-PHASE LOCKING PROTOCOL (2PL):
Pravila >
~ prije obavljanja operacije nad objektom transakcija mora za taj objekt postaviti ključ
~ nakon otpuštanja ključa ne smije više tražiti niti jedan ključ

Faze >
1) faza rasta = pribavljanje ključeva
2) faza sužavanja = otpušanje ključeva -> ovo se uglavnom obavlja jednom operacijom (COMMIT/ROLLBACK)

▪ GRANULACIJA ZAKLJUČAVANJA =
- određeno relativnom veličinom objekta koji se zaključava
- utječe na performanse sustava - finija granulacija poboljšava konkurentnost , ali i povećava troškove postavljanja ključa

▪ RAZINE IZOLACIJA =
- omogućava transakcijama balans između konzistentnosti i istovremenosti tj. ponekad tolerira nekonzistentnost za poboljšane performanse
- definiraju se na razini transakcije i mijenjaju ponašanje transakcije pri postavljanju ključeva za čitanje

¤ read uncommited:
- podaci se čitaju bez zaključavanja i bez provjere jesu li zaključani
- mogu postojati n-torke koje nikad nisu bile potvrđene u bazi

```
                    ¤ read commited
                        - čitaju se isključivo potvrđene n-torke
                        - provjerava se je li trenutno pročitan podatak zaključan za pisanje
                    ¤ repeatable read
                        - osigurava ponovljivo čitanje podataka u okviru transakcije
                        - podatak se zaključava i ostaje zaključan za čitanje do kraja trans.
                        - ne sprječava sablasne n-torke
                    ¤ serializable
                        - čitanjem se podatak zaključava ključem za čitanje i ostaje zaključan do
                        kraja
                        - sprječava probleme:   prljavo čitanje, neponovljivo čitanje
                                                , sablasne-ntorke, izgubljena izmjena

            - može doći do deadlocka, ali SUBP ima neka dodatna pravila za izbjegavanje toga


        b) PROTOKOL KORIŠTENJA VREMENSKIH OZNAKA
            - MVCC - multiversion concurrency control
            - transakciji se dodjeljuje identifikator Tid (timestamp)
            - SUBP održava višestruke fizičke verzije jednog logičkog objekta u bazi
                - istovremeno postoji više verzija istog objekta koje su međusobno povezane
                pokazivačima
            - Garbage collection!
            - kada transakcija:
                - čita iz objekta SUBP čita najnoviju potvrđenu verziju zapisa
                - piše u objekt SUBP stvara novu fizičku verziju tog objekta
            (+) pisanje i čitanje se međusobno ne blokiraju
                samo pisanje blokira pisanje
                transakcije koje samo čitaju, čitaju konzistente snimke bez zaključavanja

            ▪ Definiranje razine izolacije =
                    default
                      ↓↓↓
                read committed == read uncommitted - ne dozvoljavaju prljavo čitanje
                repeatable read == serializable - osim što ne osigurava uvijek serializabilnost

            PostgreSQL transaction snapshot
                ¤ određuje koje će transakcije biti vidljive transakcijama koje su koriste
                - trenutak određivanja snapshota ovisi o razini izolacije:
                    - read commited - na početku svake naredbe
                    - serializable - na početku transakcije
                ¤ za snapshot su potrebne sljedeće informacije:
                    ~ id najranije aktivne transakcije
                    ~ prvi nedodijeljen id
                    ~ id-evi svih trenutno aktivnih transakcija

/////////////////17. PREZENTACIJA - SIGURNOST BAZE PODATAKA
Def ->
    INTEGRITET - operacije nad podacima koje korisnici obavljaju su ispravne tj uvijek rezultiraju
                 konzistentnim stanjem baze podataka
    SIGURNOST - korisnici koji obavljaju operacije nad podacima su ovlašteni za obavljanje
                tih operacija
        ↓→ u oba slučaja:
                ▪ moraju biti definirana pravila koja korisnici ne smiju narušiti
                ▪ pravila se pohranjuju u rječnik podataka
                ▪ SUBP nadgleda rad korisnika - osigurava poštivanje pravila

Oblici narušavanja sigurnosti baze su:
    - neovlašteno čitanje podataka
    - neovlaštena izmjena podataka
    - neovlašteno uništavanje podataka

Autorizacija=   postupak kojim se određenom korisniku dodjeljuje dozvola za obavljanje određenih
                vrsta operacija nad određenim objektima baze podataka
                - podaci o tim dozvolama se pohranjuju u rječnik podataka

        DISKRECIJSKO UPRAVLJANJE PRISTUPOM =>
            - određenom korisniku se eksplicitno dodjeljuje dozvola za obavljanje
            određene operacije nadodređenim objektom
                        <korisnik, objekt, vrsta operacije>
            - kad korisnik horvat pokušava obaviti operaciju čitanja objekta (tablice) predmet
            SUBP provjerava postoji li dozvola oblika <horvat, predmet, čitanje>

        MANDATNO UPRAVLJANJE PRISTUPOM =>
            - primjenjivo u sustavima u kojima se dozvole dodjeljuju na temelju hijerhije

            - svaki objekt dobiva razinu tajnosti (classification level), a svaki korisnik dobiva
            razinu ovlasi (clearance level)

Korisnici u SQL-u =
    ▪ Autentificirani korisnik - pri uspostavljaju sjednice prijavljuje se svojim loginom, te
                                 lozinkom ovjerava svoju autentičnost
                                 - funcija CURRENT_USER vraća id korisnika trenutne SQL sjednice
    ▪ PUBLIC - dozvolu dobivaju svi sadašnji i budući korisnici

    CREATE USER name [[WITH] option [...]]
                          ↓↓↓
                        SUPERUSER - nema ograničenja, loše;                DEF = NOSUPERUSER
                        CREATEDB - ovlast kreiranje baze podataka na SUBP;  DEF = NOCREATEDB
                        CREATEUSER - oblast kreiranja drugih korisnika;     DEF = NOCREATEUSER
                        [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
                        INHERIT | NOINHERIT
```

INHERIT | NOINHERIT

ex CREATE USER bpadmin WITH CREATEDB CREATEUSER PASSWORD 'bpadminPwd';

Objekti i vlasnici objekata u SQL-u
- Vlasnik objekta je korisnik koji je kreirao objekt, implicitno dobiva dozvole za obavljanje SVIH vrsta operacija nad objektom, uključujući dozvole za dodjeljivanje dozvola i uništavanje objekta

SHEME =>
- BP sadrži jednu ili više shema
  ↓→ sheme sadrže tablice, virtualne tablice
- različite sheme mogu sadržavati istoimene tablice
- sheme analogne s:
  -> mapama u datotečnom sustavu
  -> imenskim područjima

Zašto sheme?
~ omogućuju višekorisnički pristup
~ organziraju tablice u logičke grupe kako bi se njima lakše upravljalo (interno, javno, admin)
~ uspostavljaju sustav dozovla

ex
```
CREATE SCHEMA student;
CREATE TABLE student.postavke(
    username TEXT primary key
    , cm_skin TEXT not null
);
SELECT * FROM student.postavke;
DROP SCHEMA student;
DROP SCHEMA student CASCADE;
```

¤ Određivanje sheme SSP (schema search path)
- ako se u SQL naredbi ne upotrijebi puno ime tablice SUBP traži tablicu koristeći SSP
  - koristi se prva pronađena tablica
  - ako se ne pronađe baca error (iako možda postoji tablica tog imena u nekoj drugoj shemi koja nije sadržana u korištenom SSP)

- prva shema u SSP se zove trenutna shema
- ako pri kreiranju novih objekata ne navedemo ime sheme objekt će se kreirati u trenutnoj shemi

- funkcija current_schema() vraća ime trenutne sheme
- trenutni SSP se može dobiti funkcijom SHOW search_path;

VRSTE DOZVOLA =>
(*) Dozvole na razini baze podataka (dbPrivilege)
PostgreSQL:
~ CONNECT - dozvoljava spajanje na bazu
  - public ima tu dozvolu na razini baze
~ CREATE - dozvoljava stvaranje novih shema unutar baze

(*) Dozvole na razini sheme (schemaPrivilege)
~ USAGE - nužan preduvjet za pristupanje objektima u shemi
~ CREATE - dozvoljava stvaranje novih objekata u shemi
  ¤ default ponašanje:
    - korisnik ima pristup samo onim objektima kojima je vlasnik
    - za pristup mu vlasnik sheme treba dodijeliti USAGE
    - za kreiranje objekata dodatno mora dobiti CREATE
    - ima obje te dozvole za public

(*) Dozvole za objekte unutar sheme (tablePrivilege)
~ SELECT[(columnList)]
~ UPDATE[(columnList)]
~ INSERT[(columnList)]
~ DELETE
~ ALL PRIVILEGES

SQL naredbe za dodjeljivanje i ukidanje dozvola:
ex
```
GRANT dbPrivilege ON DATABASE name TO {PUBLIC | userList}
REVOKE dbPrivilege ON DATABASE name FROM {PUBLIC | userList}
```

PostgreSQL → default dozvole korisnika PUBLIC
¤ ključna riječ PUBLIC != shema public!
```
GRANT CONNECT ON DATABASE * TO PUBLIC;
GRANT ALL(USAGE, CREATE) ON SHEMA public TO PUBLIC
```

- PUBLIC nema nikakvu dozvolu na razini tablica u shemi public

Prenosive dozvole = ako se korisniku dozvola dodijeli uz navođenje opcije WITH GRANT OPTION on dobiva mogućnost tu dozvolu koju je dobio dalje dodijeljivati iako nije vlasnik objekta

Ukidanje dozvola = naredba REVOKE
- ako se ukine dozvola korisniku2 koji je dobio WITH GRANT OPTION mora se navesti i CASCADE kako bi se ukinula dozvola svima koji su dozvolu dobili od korisnika2
- ako se CASCADE ne navede REVOKE neće uspjeti ako je korisnik2 dodijelio neku dozvolu

//////////////////19. PREZENTACIJA - OPTIMIRANJE UPITA

Query processing = niz aktivnostipotrebnih da bi se dohvatilo podatke iz BP
Koraci:
    1) parsiranje i translacija
        -pretvorba SQL u relacijsku algebru
        - zamjena virtualnih tablica s temeljnim tablicama koje se koriste u upitu

        ex SELECT *
            FROM filmskazvijezda, glumiu
            WHERE imezvijezda = ime
            AND godina = 2012
            AND spol = 'M';

            ↓→ npr izgradi stablo upita - čvorovi su operacije, grane su relacije

    2) optimiranje
        PLAN IZVOĐENJA
            - odabir operanada i operacija
            - redoslijed operacija
            - metode pristupa podacima
            - metoda spajanja - npr kreiranje privremenog indeksa, sortiranje međurezultata

        ø Plan izvođenja nije jednoznačno određen upitom

        Optimiranje upita => proces odabira najprikladnijeg plana
    3) izvršavanje

Optimizator vodi računa o:
    • N(r) - broj n-torki u tablici
    • sadrži li atribut vl vrijednosti
    • indeksi, redoslijed, cluster
    • dubina B stabla za indekse
    • V(A, r) broj različitih vrijednosti atributa A u tablici r
    • broj jedinstvenih vrijednosti u indeksu
    • distribucija vrijednosti po tablici

    - ne skuplja statistiku baš svih upita
    - postoji naredba kojom se pokreće prikupljanje statistike VACUUM ANALYZE
            ↓→ vlasnik baze je odgovoran to pozvati kada se jako promijeni sadržaj baze
                (masovna brisanja i izmjene) da bi se ažurirao rječnik podataka

Evivalentni izrazi rel algebre - ako nad svakom instancom BP daju isti rezultat

ALGEBARSKE TRANSFORMACIJE =
    • Prirodno spajanje
        ~ komutativno %$r \bowtie s = s \bowtie r$
        ~ asocijativno %$(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$
    • Asocijativne i komutativne operacije %$\times, \cup, \cap$
    • %θ- spajanje nije uvijek asocijativno!, ali može biti

    • Pravila selekcije
        podjela
            ~ AND %$\sigma_{c1 AND c2} = \sigma_{c1}(\sigma_{c2}(r)) = \sigma_{c2}(\sigma_{c1}(r))$
            ~ OR %$\sigma_{c1 OR c2} = \sigma_{c1} \cup \sigma_{c2}$

        POTISKIVANJE SELEKCIJE
            - cilj je obaviti selekciju što prije moguće
            tj primjeni se nad operandima prije samog spajanja

HEURISTIČKA PRAVILA =
    - nije moguće generirati i analizirati baš sve planove izvođenja - reduciranje broja planova
    1) potiskivanje selekcije
    2) kombiniranje selekcije i kartezijevog produkta -> u prirodno spajanje

- procjenjuje se trošak izvršavanje za dobivene planove
    više UI operacija == veći trošak
    veličina međurezultata

Procjena veličina rezultata selekcije
    pretp. jednolika razdioba vrijednosti atributa A

    ▪ Selekcija uz uvjet A=c
        %$N(\sigma_{A=c}(r)) = N(r)/V(A,r)$
        - ako c uopće ne postoji?? pretp da c postoji

    ▪ Selekcija koja uključuje nejednakost
        %$\sigma_{A<c}(r)$
        cca 1/3 n-torki zadovoljava uvjet

    ▪ Složeni uvjeti - algebarske transformacije

    ▪ Neka su X i Y skupovi atributa iz r odnosno s i %$t = r \bowtie s$

        1) %$X \cap Y = \varnothing$
            - spajanje je produkt, nema eliminacije duplikata %N(t) = N(r) * N(s)
        2) %$X \cap Y =$ ključ od r
            - svaka n-torka iz se može spojiti s max jednom ntorkom iz r %N(t) = N(s)

3) %$X \cap Y$ nije prazan skup, nije ključ od r ili s
%N(t) = N(r) * N(s) / max(V(A, r), V(A, s))

Metode pristupa podacima u tablici (access plan) =>
- Čitanje n-torki
  ~ slijednim čitanjem blokova podataka (sequential scan, table scan)
    → kad nema upotrebljivog indeksa ili kad ionako treba pročitati većinu n-torki
  ~ korištenjem indeksa (index-only scan, index scan)
    → index-only scan - ako su svi podaci koji se čitaju dijelovi JEDNOG indeksa
    → index scan - ako se svi potrebi podaci ne mogu naći u indeksu, mora se pristupati podatkovnim blokovima

Spajanje ugnijezđenim petljama:
- tablice koje se spajaju = vanjska i unutarnja tblica
- iz vanjske se čita svaka n-torka (ako je moguće obavi se selekcija + index/index-only search)
- za svaku ntorku iz vanjske tablice se traže ntorke iz unutarnje koje zadovoljavaju uvjet spajanja
  ↓→ za svaku ntorku iz V mora se proći cijela U

Raspršeno spajanje (HASH!):
ø FAZA IZGRADNJE
  → raspršena se tablica izgrađuje za jednu tablicu iz para kojeg treba spojiti
ø FAZA ISPITIVANJA
  → čita se sadržaj druge tablice, za svaku pročitanu ntorku pomoću fje raspršenja, a na temelju vrijednosti atributa iz tablice S prema kojoj se obavlja spajanje, izračuna se džep hash tablice r u kojeg bi trebao ići taj zapis

Optimiranje upita → analiza plana obavljanja PostgreSQL
- EXPLAIN [ANALZYE] [VERBOSE] statement
  ↓ ↓
  statistika dodatne info npr listu izlaznih stupaca za svaki čvor u stablu izvođenja