1. predavanje (1. ciklus; 24.02.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred1.pdf)

Grupa: 2.R1
Predavači: prof. dr. sc. Zoran Skočir i dr. sc. Marko Banek
Asistent: mr. sc. Damir Jurić

- **Organizacijski (poslovni) sustav** – složeni sustav koji sadrži tehničke i humane podsustave (npr. knjižnica, sveučilište, ...)
- **Informacija** – sadržaj koji primatelju opisuje nove činjenice (npr. informacija: prosjek svih ocjena studenta Marka Horvata na studiju na FERu u ovom trenutku je 4.62)
- **Podatak** – skup simbola (znakova) (npr. podatak: 4.62)
- **Informacijski sustav** – dio svakog organizacijskog sustava čija je svrha prikupljanje, obrada, pohranjivanje i distribucija informacija koje su potrebne za upravljanje organizacijskim sustavom ili nekim njegovim podsustavom (ne mora koristiti suvremenu tehnologiju i središnji dio mu čini baza podataka)
- **Baza podataka (J. Martin, 1979)** – skup međusobno povezanih podataka, pohranjenih zajedno uz isključenje bespotrebne zalihosti (redundancije) koji mogu zadovoljiti različite primjene. Podaci su pohranjeni na način neovisan o programima koji ih koriste. Prilikom dodavanja novih podataka, mijenjanja i pretraživanja postojećih podataka primjenjuje se zajednički i kontrolirani pristup. Podaci su strukturirani tako da služe kao osnova za razvoj budućih primjena.
- **Entitet** – bilo što što posjeduje suštinu ili bit i posjeduje značajke s pomoću kojih se može razlučiti od svoje okoline (nešto o čemu želimo prikupljati i pohranjivati podatke)
- **Atribut** – entitet posjeduje nekakva svojstva ili atribute koji ga karakteriziraju (izbor atributa koje ćemo pratiti ovisi o namjeni informacijskog sustava) (neformalna definicija: imenovani stupac relacije)
- **Skup entiteta** – slični entiteti (entiteti kojima se promatraju ista svojstva) svrstavaju se u skupove entiteta (atribut entiteta → atribut skupa entiteta)
- **Domena i vrijednost atributa** – za svaki entitet svaki atribut poprima određenu vrijednost iz određenog skupa vrijednosti koji predstavlja domenu tog atributa (neformalna definicija: domena je skup dopuštenih vrijednosti atributa)

2. predavanje (1. ciklus; 26.02.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred1.pdf)

- **Identifikatori (ključevi) skupa entiteta** – skupovi atributa čije vrijednosti jednoznačno određuju entitet u promatranom skupu entiteta (dakle ne postoje dva entiteta s kompletno istim vrijednostima tog atributa). Ključevi mogu biti prirodni (jedan ili skup atributa pojedinog entiteta) ili umjetni (šifra).
- **Model podataka** – formalni sustav koji koristimo kod modeliranja baza podataka. Sastoji se od skupa objekata (osnovnih elemenata baze podataka), skupa operacija koje se provode nad objektima i skupom integritetskih ograničenja.
- **Relacijski model podataka**
  - *Elementi skupa objekata* u relacijskom modelu su relacije
  - *Relacija* – skup n-torki (neformalna definicija: imenovana dvodimenzionalna tablica)
  - *Shema relacije* – obuhvaća naziv relacijske sheme i skup atributa (prazna tablica)
  - *N-torka* – redak relacije
  - Neke od *operacija* u relacijskom modelu podataka su: unija, razlika, presjek, projekcija, ...
  - Neki od *integritetskih ograničenja* su: domenski integritet, entitetski, referencijski
- **ER model podataka** (obilato će se raditi u 3. ciklusu)
  - Postrelacijski model (zadržava dobra svojstva relacijskog modela)
  - Objekti ER modela su entiteti i njihove međusobne veze
- **Arhitektura baze podataka**: 3 razine
  - *Konceptualna (logička) shema (razina)* – sadrži opis svih entiteta i veza, atributa, domena i integritetskih ograničenja i može se opisati korištenjem modela podataka (relacijski, ER, ...)

  - *Interna (unutarnja) shema (razina)* – opisuje detalje fizičke strukture pohrane i metode pristupa podacima, tj. kako su podaci pohranjeni (organizacija podataka na disk) i koje se metode koriste za pristup podacima + postoji fizička nezavisnost podataka
  - *Eksterna (vanjska) shema (razina)* – opisuje „pogled" na dio baze podataka koji je namjenje specifičnoj grupi korisnika i podskup je logičke sheme + postoji logička nezavisnost podataka
  - Jedna baza podataka ima maksimalno jednu konceptualnu, jednu internu i (najčešće) više eksternih shema
  - Shema baze podataka se rijetko mijenja dok se sadržaj ili instanca mijenja često
- **Sustav za upravljanje bazom podataka (SUBP)** – programski sustav koji omogućava upravljanje podacima u bazi podataka
  - temelji se na modelu podataka pa ih dijelimo na: hijerarhijske, mrežne, relacijske, ...
  - sakriva od korisnika detalje fizičke pohrane podataka i osigurava logičku i fizičku nezavisnost podataka
  - omogućava definiciju i rukovanje podacima putem *DDL*a (*data definition language*; omogućava imenovanje i opis entiteta, atributa veza i pripadnih ograničenja integriteta i pravila sigurnosti, koristi se za definiranje nove sheme baze podataka ili modificiranja postojeće te obavljanje DDL operacija rezultira izmjenom sadržaja rječnika podataka) i *DML*a (*data manipulation language*; omogućava korištenje skupa operacija za rukovanje podacima u bazi podataka)
  - obavlja funkciju zaštite podataka (integritet, pristup, autorizacija, kontrola paralelnog pristupa, obnova u slučaju razrušenja) i optimiziranje upita
- **Zašto SUBP a ne samo datoteke?** – 6 razloga
  - *Jednostavan pristup podacima* – u datotečnom sustavu podaci su raspršeni po datotekama koje mogu biti različitih formata te je za dobivanje odgovora na netipična pitanja potrebno dotjerivati stare ili pisati nove programe
  - *Upravljanje zalihostima i nekonzistentnošću* – isti podaci mogu biti pohranjeni u više datoteka
  - *Transakcijska obrada* – SUBP ima ugrađenu podršku za transakcijsku obradu što je vrlo vrlo teško ostvariti u datotečnom sustavu
  - *Složeni odnosi među podacima* – SUBP omogućuje predstavljanje različitih odnosa među podacima, definiranje novih odnosa kad se oni pojave te jednostavan dohvat i izmjenu međusobno povezanih podataka
  - *Istovremeni pristup više korisnika* – SUBP omogućuje većem broju korisnika pristup bazi podataka u isto vrijeme ali pri tome i nemogućava paralelni pristup istom podatku
  - *Autorizirani pristup* – nije svim korisnicima omogućen pristup svim podacima u bazi
- **Uloge osoba u životnom ciklusu baze podataka**
  - *Projektanti baze podataka* – razgovaraju s korisnicima kako bi saznali njihove želje te oblikuju strukturu baze podataka
  - *Analitičari sustava i programeri aplikacija* – pomoću zahtjeva korisnika analitičari pišu specifikacije za razvoj aplikacije za pristup bazi podataka a programeri na temelju tih specifikacija izrađuju programe koje testiraju, dokumentiraju i održavaju
  - *Administratori baze podataka* – instaliraju i održavaju bazu podataka, organiziraju, nadziru i optimiziraju korištenje baze te su odgovorni za pristup bazi podataka
  - *Korisnici* – pristupaju bazi tako da postavljaju upite, mijenjaju podatke i izrađuju izvještaje
    - korisnici koji povremeno pristupaju koristeći upitni jezik (npr. mi)
    - korisnici koji često koriste bazu postavljajući standardne upite (npr. službenici u bankama, turističkim agencijama, ...)
    - sofisticirani korisnici koji su dobro upoznati s bazom podataka i koriste ju na složeniji način (npr. inženjeri, znanstvenici, poslovni analitičari, ...)

3. predavanje (1. ciklus; 03.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred2.pdf)

- **Relacijski model podataka (dr. Edgar Frank Codd, 1970)** – najzastupljeniji i glavni cilj mu je postavljanje temelja za rješavanje semantike, konzistentnosti i redundancije podataka, tj. njihova normalizacija

  - matematička definicija relacije *R* kaže da relacija *R* definirana nad skupovima $D_1, D_2, D_3, ..., D_n$ te da je ona podskup Kartezijevog produkta skupova nad kojima je definirana ($D_1, D_2, D_3, ...D_n$)
    $R \subseteq D_1 \times D_2 \times D_3 \times ... \times D_N$
- **Svojstva relacije**
  - jedinstveno ime unutar sheme baze podataka
  - atributi imaju jedinstvena imena unutar relacije te jedan atribut može poprimiti vrijednost iz samo jedne domene
  - ne postoje dvije jednake n-torke (dva jednaka retka) u relaciji (tablici)
  - redoslijed atributa i n-torki u relaciji je nebitan
- **Formalna definicija relacijske sheme** – $A_1, A_2, A_3, ..., A_n$ su zadani atributi. Relacijska shema *R* (intenzija) je imenovani skup atributa $R=\{A_1, A_2, A_3, ..., A_n\}$ ili $R=A_1A_2A_3...A_n$ [npr. *MJESTO={pbr, nazMjesto, sifZup}*].
  - Poredak atributa u shemi nije bitan te se shema relativno rijetko mijenja.
- **Formalna definicija n-torke** - $R=\{A_1, A_2, A_3, ..., A_n\}$ je relacijska shema, a $D_1, D_2, D_3, ..., D_n$ su domene atributa $A_1, A_2, A_3, ..., A_n$. N-torka *t* je skup parova oblika *atribut: vrijednostAtributa* definirana na relacijskoj shemi *R* pri čemu je $v_1 \in D_1$, $v_2 \in D_2$, $v_3 \in D_3$, ..., $v_n \in D_n$ (npr. t_1={*matBr:* 1234, *ime:* Iva, *prez:* Novak}
  - Pojednostavljena notacija izgleda ovako $t=<v_1, v_2, v_3, ..., v_n>$, ali se onda poredak vrijednosti atributa *v* mora odgovarati poretku atributa *A* u relacijskoj shemi *R* (npr. $t_1=<1234, Iva, Novak>$).
- **Formalna definicija relacije** - $R=\{A_1, A_2, A_3, ..., A_n\}$ je relacijska shema, a $D_1, D_2, D_3, ..., D_n$ su domene atributa $A_1, A_2, A_3, ..., A_n$. Relacija r (malo slovo; 'r'), tj. instanca relacije definirana na shemi relacije R je skup n-torki koje su definirane na relacijskoj shemi R (veliko slovo; 'R') (npr. *student(STUDENT)={{matBr:102, prez: Novak, ime: Ivan},{matBr: 135, prez: Ban, ime: Marko}}*).
  - Instanca relacije *r* za razliku od sheme relacije *R* predstavlja trenutnu vrijednost i često se mijenja (pri unosu/brisanju/izmjeni podataka)
  - Pojednostavljeni prikaz relacije znači vizualizacija relacije tablicom *student(STUDENT)*.

matBr	Prez	Ime
102	Novak	Ivan
135	Ban	Marko

- **Stupanj (degree) relacije** – broj atributa (stupaca) relacije (tablice) (oznaka: *deg(student)=3*)
- **Kardinalnost (cardinality) relacije** – broj n-torki (redaka) relacije (tablice) (oznaka: *card(student)=2*)
- **Shema baze podataka** – skup relacijski shema $R = \{R_1, R_2, R_3, ..., R_n\}$
  - napomena: relacijske sheme u istoj shemi baze podataka moraju imati različita imena
- **Instanca baze podataka** – skup instanci relacija $r = \{r_1(R_1), r_2(R_2), r_3(R_3), ..., r_n(R_n)\}$ definirana na shemi baze podataka R
- **Operacije (relacijska algebra):** $\cup$ unija, $\cap$ presjek, $\setminus$ razlika, $\div$ dijeljenje, π projekcija, σ selekcija, $\times$ Kartezijev produkt, ρ preimenovanje, $\rhd\lhd$ spajanje
  - proceduralnost – karakteristika relacijske algebre a ona navodi redoslijed operacija koje se provode nad operacijama
- **Predikatni račun** – filtriranje n-torki s obzirom na njihove vrijednosti atributa
  - $r = \{ t \mid F(t)\} \rightarrow t$ je varijabla koja predstavlja n-torke, a rezultat *r* je skup n-torki za koje je vrijednost predikata *F* istina
  - predikatni račun je neproceduralan
- **SQL (Structured Query Language)** – temeljen na relacijskom modelu podataka, predikatnom računu i relacijskoj algebri te nastao na temelju jezika SEQUEL i proglašen standardnim jezikom za relacijske sustave
  - objekti u SQL-u su tablice a ne relacije
  - poredak atributa u tablici zna biti značajan i postoji mogućnost pojavljivanja istih n-torki

- *CREATE* i *DROP DATABASE* – kreiranje i brisanje nove instance baze podataka (1 SUBP može istovremeno upravljati s više baza podataka)

```
CREATE DATABASE studAdmin;          DROP DATABASE knjiznica;

CREATE DATABASE knjiznica;
```

- *CREATE* i *DROP TABLE* – kreiranje i brisanje relacije (kreiranje prazne) te mogućnost definiranja integritetskih ograničenja

```
CREATE TABLE mjesto (                DROP TABLE mjesto;
        pbr         INTEGER
,       nazMjesto   CHAR
,       sifZup      SMALL INT
);
```

- *INSERT INTO* – upisivanje novih n-torki u relaciju. Napomena: poredak n-torki nemora biti u skladu s redoslijedom upisa.

```
INSERT INTO mjesto
        VALUE (42000,  'Varaždin',  7) ;
INSERT INTO mjesto
        VALUE (52100, 'Pula', 4);
INSERT INTO mjesto
        VALUE (42230, 'Ludbreg', 7);
```

- *SELECT* – naredba za dohvat podataka iz relacije te naredba koje će se najčešće koristiti

```
SELECT * FROM mjesto
        WHERE sifZup = 7 ;
```

- *UPDATE* – izmjena vrijednosti atributa u relaciji

```
UPDATE mjesto
        SET nazMjesto =  'VARAŽDIN'
        WHERE pbr = 42000;
```

- *DELETE* – brisanje n-torki iz relacije.

```
DELETE FROM mjesto
        WHERE sifZup = 7 ;
```

(1) `DELETE FROM mjesto ;` (2) `DROP TABLE mjesto;`

Razlika (1) i (2) je u tome što *(1)* ostavlja praznu tablicu dok *(2)* briše i praznu tablicu, tj. shemu relacije.

- **Relacijska algebra**
  - *Unarne operacija* – projekcija, selekcija, preimenovanje, agregacija, grupiranje
  - *Binarne operacije* – skupovske operacije (unija, presjek, razlika; mogu se obavljati isključivo nad unijski kompaktibilnim relacijama), Kartezijev produkt, dijeljenje, spajanje
  - $r_3 = r_1 \cup r_2$ – obavljanje operacije ne utječe na operande nego uvijek nastaje nova relacija, tj. skup relacija je zatvoren s obzirom na operacije relacijske algebre što nam omogućava da se rezultati jedne operacije upotrebe kao operandi druge operacije te tako možemo formirati složenije izraze [npr. $r_5 = (r_1 \cup r_2) \times (r_3 \rhd\lhd r_4)$]
  - *Unijska kompaktibilnost* – dvije relacije su unijski komapktibilne ukoliko vrijedi da su relacije istog stupnja i ako su korespodentni atributi definirani nad istim domenama (poredak i imena atributa u relacijama koje uspoređujemo nisu bitna). Notacija *imeRelacije.imeAtributa* se često koristi kada je potrebno razlikovati atribute različitih relacija (npr. *mjesto.pbr* u relaciji *mjesto* jednako je kao i da smo napisali samo *pbr*)

- *Unija* – rezultat operacije $r_1 \cup r_2$ je relacija čije su n-torke elementi r_1 ili r_2 ili obje relacija. N-torke koje su elementi i r_1 i r_2 u rezultatu se pojavljuju samo jednom jer je relacija skup n-torki). Vrijedi pravilo komutacije ($r_1 \cup r_2 \equiv r_2 \cup r_1$)
- *Presjek* – rezultat operacije $r_1 \cap r_2$ je relacija čije su n-torke elementi relacije r_1 i r_2 . Također vrijedi komutacija
- *Razlika* – rezultat operacije $r_1 \setminus r_2$ je relacija čije su n-torke elementi relacije r_1 a nisu elementi r_2. Komutativnost naravno ne vrijedi
- *Napomena uz uniju, presjek i razliku:* Kao imena atributa u rezultatu se koriste imena atributa prvog operanda ako se imena atributa operanada razlikuju
- *Dijeljenje* – ovom operacijom se nećemo ozibljno baviti i ne treba razbijati glavu. Ukoliko nije jasno iz same definicije bacite oko na primjer u predavanjima i trebali bi shvatiti. Neka su zadane relacije *r(R)* i *s(S)* i neka je $S \subseteq R$. Rezultat operacije $r \div s$ je relacija sa shemom $P = R \setminus S$ a n-torka $t_r(P)$ se pojavljuje u rezultatu ako i samo ako za n-torku $t_r \in r$ vrijedi da se $t_r(P)$ u relaciji r pojavljuje u kombinaciji sa svakom n-torkom $t_s \in s$
- *Projekcija (SELECT DISTINCT)* – vertikalni presjek, tj. odabir po stupcima. Sastoji se od dva koraka, prvi je izdvajanje vertikalnog stupca a drugi je eliminacija duplikata [npr. s = $\pi_{B, C}(r)$]. *deg(r) = 4*, *deg(s) = 2*, *card(s)* $\leq$ *card(r)* zbog eliminacije duplikata. Predavači kažu da će sigurno biti u međuispitima pa da pazimo da ne zaboravio ključnu riječ «DISTINCT».

  SELECT *SELECT List* FROM mjesto ;

  - SELECT List je dio SELECT naredbe koji određuje koji "stupci" će se pojaviti u rezultatu. U toj listi se mogu naći samo atributi koji se nalaze u dosegu naredbe SELECT, tj. samo atributi relacije koja je navedena u FROM dijelu naredbe
  - ' * ' na mjestu ' SELECT List ' označava da se pojave svi "stupci" (atributi) iz tablice (relacije)

    SELECT * FROM mjesto ;

  - Kao što smo rekli projekcija se sastoji od dva koraka. Sa listom za selekciju smo napravili prvi korak i sad još moramo eliminirati duplikate a da bi to napravili koristimo «dodatak» DISTINC naredbi SELECT [npr. $\pi_{B, C}(r)$].

    SELECT DISTINCT B
    , C
    FROM r ;
- *Selekcija (WHERE CONDITION)* – horizontalni presjek, tj. odabir po retcima. Predikat (formula, uvjet, condition) *F* sastoji se od operanada (imena atributa iz sheme relacije i konstante) i operatora (usporedbe: $\leq, \geq, \neq, <, >, =$ i logičkih: $\wedge, \vee, \neg$). Obavljanjem $\sigma_F(r)$ dobiva se relacija sa shemom *R* koja sadrži one n-torke relacije *r* za koje je vrijednost predikata *F* istina (true).

  SELECT SELECT List FROM table
  [WHERE Condition]

- *Kartezijev produkt (FROM student, predmet, ...)* – prva binarna operacija koju radimo. Zadane su relacije *r(R)* i *s(S)* pri čemu je $R \cap S = \emptyset$ (u *R* i *S* nema atributa (stupaca) sa istim imenima, ukoliko ima onda se prije operacije kartezijevog produkta treba napraviti operacija preimenovanja). Obavljanjem operacije $r \times s$ dobiva se relacija *p(P)* pri čemu vrijedi $P = R \cup S$. N-torke relacije *p* dobivaju se ulančavanjem (spajanjem) svake n-torke iz relacije *r* sa svakom n-torkom iz relacije *s*. $deg(p) = deg(r) + deg(s)$ i $card(p) = card(r) \cdot card(s)$.

  SELECT *
  FROM student, predmet;

- *Preimenovanje* – postoje preimenovanje relacija (1) i preimenovanje relacije i atributa (2). U (1) operacijom preimenovanja $\rho_s(r)$ dobiva se relacija *s* koja ima jednaku shemu i sadržaj kao i relacija *r*. U (2) operacijom preimenovanja $\rho_{s(B1, B2, B3, ..., Bn)}(r)$ dobiva se relacija *s* čija shema

umjesto atributa relacije *r* sadrži atribute *B1, B2, B3, ..., Bn* a sadržaj relacije *s* jednak sadržaju relacije *r*.

```
SELECT      sifZupanija AS sifraZ
      ,     nazZup AS nazZ
      FROM zupanija;
```

≡

```
SELECT      sifZupanija sifraZ
      ,     nazZup nazZ
      FROM zupanija;
```

- ključna riječ AS može se ispustiti

4. predavanje (1. ciklus; 05.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred2.pdf)

- **Spajanje uz uvjet (Θ spajanje)** – operacijom spajanje ($r \underset{F}{\triangleright\!\triangleleft} s$) dobiva se nova relacija koja sadrži n-torke iz $r \times s$ za koje je vrijednost predikata *F* istina (true), odnosno $r \underset{F}{\triangleright\!\triangleleft} s = \sigma_F(r \times s)$. Kao i do sada presjek relacija *R* i *S* je prazan skup, a ukoliko nije to se rješava preimenovanjem. Čest greška u isptima je da studenti miješaju spajanje uz uvjet i prirodno spajanje. $\deg(r \triangleright\!\triangleleft s) = \deg(r) + \deg(s)$. Primjer:

$$linija \underset{dolet \geq udaljenost}{\triangleright\!\triangleleft} zrakoplov$$

```
SELECT *
      FROM linija, zrakoplov -- kartezijev produkt
      WHERE dolet >= udaljenost; -- selekcija
```

≡

```
SELECT *
      FROM linija JOIN zrakoplov
            ON dolet >= udaljenost;
```

- **Spajanje uz uvjet i selekcija** - $\sigma_{udaljetnost > 4000}(linija \underset{dolet \geq udaljenost}{\triangleright\!\triangleleft} zrakoplov)$

```
SELECT *
      FROM linija JOIN zrakoplov
            ON dolet >= udaljenost
WHERE udaljenost > 4000;
```

≡

```
SELECT *
      FROM linija, zrakoplov
      WHERE dolet >= udaljenost
            AND udaljenost > 4000;
```

- **Spajanje uz uvijet i projekcija** - $\Pi_{tip} linija \underset{dolet \geq udaljenost}{\triangleright\!\triangleleft} zrakoplov)$

```
SELECT DISTINCT tip
      FROM linija JOIN zrakoplov
            ON dolet >= udaljenost
```

≡

```
SELECT DISTINCT tip
      FROM linija, zrakoplov
      WHERE dolet >= udaljenost
```

- **Spajanje s izjednačavanjem (Equi-join)** – poseban oblik spajanja pri kojemu se kao Θ operator koristi isključivo operator jednakost (=). Ukoliko u relacijama postoje istoimeni atributi samo u relacijskoj algebri je potrebno izvršiti preimenovanje prije spajanja. Primjer:

$$mjesto \underset{sifZup = sifZup2}{\triangleright\!\triangleleft} \rho_{zupanija(sifZup2, nazZup)} zupanija$$

```
SELECT      mjesto.*
      ,     zupanija.sifZup AS zupanija.sifZup2
      ,     zupanija. nazZup
      FROM mjesto JOIN zupanija
            ON sifZup = sifZup2;
```

mjesto

pbr	nazMjesto	sifZup
42000	Varaždin	7
52100	Pula	4
42230	Ludbreg	7

zupanija

sifZup	nazZup
7	Varaždinska
4	Istarska

- **Prirodno spajanje (Natural join)** – obavlja se na temelju jednakih vrijednosti atributa. Ukoliko nema jednakih atributa, tj. $R \cap S = \varnothing$ onda je rezultat prirodnog spajanja r(R) i s(S), $r \rhd\lhd s$, jednak rezultatu kartezijevog produkta r×s. $\deg(r \rhd\lhd s) = [\deg(r) + \deg(s)] - brojIstihAtributa$. Primjer: (vidi tablicu iznad) $mjestouZupaniji = mjesto \rhd\lhd zupanija$

```
SELECT mjesto.*, zupanija.nazZup
        FROM mjesto JOIN zupanija
                ON mjesto.sifZup = zupanija.sifZup;
```

mjestouZupaniji

pbr	nazMjesto	sifZup	nazZup
42000	Varaždin	7	Varaždinska
52100	Pula	4	Istarska
42230	Ludbreg	7	Varaždinska

  - Česta greška u ispitima: Koja je razlika u pisanju SQL naredbi za prirodno spajanje i spajanje s izjednačavanjem?
    -
- **Agregacija** – postoji relacija r(R) i neki atribut A∈ R. Neka je $\mathcal{AF}$ jedna od agregatnih funkcija (COUNT, MIN, MAX, SUM, AVG). Rezultat operacije $G_{\mathcal{AF}(A)}(r)$ je relacija stupnja i kardinalnosti 1 pri čemu je vrijednost atributa A određena primjenom agregatne funkcije $\mathcal{AF}$ nad vrijednostima atributa u svim n-torkama relacije *r*. Naziv rezultatne operacije i atributa nije definiran operacijom pa se koristi u kombinaciji s preimenovanjem. Primjer: $\rho_{prosjek(prosjOcj)}(G_{AVG(ocjena)}(ispit))$

```
SELECT AVG (ocjena) AS prosjOcj
        FROM ispit;
```

ispit

mbrStud	akGod	nazPred	ocjena
100	2005	Matematika	3
101	2005	Matematika	5
102	2005	Matematika	2
103	2006	Matematika	3
100	2004	Fizika	5
101	2006	Fizika	5
102	2006	Fizika	2
100	2005	Vjerojatnost	4

prosjek

prosjOcj
3.625

sumOcj
14

prosjek2

nazPred	prosjOcj
Matematika	3.25
Fizika	4
Vjerojatnost	4

  - moguće je odjedno izračunati i više agregatnih vrijednosti te će onda stupanj rezlutatne relacije biti jednak broju korištenih agregatnih funkcija. Primjer:

$$\rho_{rezultat(broj1,broj2,broj3)}(G_{AVG(ocjena),MIN(težina),SUM(brIspita)}(test))$$

  - DISTINCT se također može koristiti zajedno sa agregatnim funckijama (samo u SQL-u) te se onda u obzir uzimaju samo različite vrijednosti atributa. Primjer: (vidi tablicu iznad)

```
SELECT SUM (DISTINTC ocjena) AS sumOcj
        FROM ispit;
```

- **Agregacija i grupiranje** – ovo nam neće biti jasno ni nakon polaganja predmeta (bar 30% studenata) i treba puno vježbati te se to inače jako puno koristi u bazama podataka. Najviše problema ćemo imati sa „GROUP BY“ koji cijeli uvijek mora biti u SELECTListi. Primjer: (vidi tablicu iznad) Za svaki predmet ispisati proječnu ocjenu. $\rho_{prosjek2(nazPred,prosjOcj)}({}_{nazPred}G_{AVG(ocjena)}(ispit))$
  - grupirati po *nazPred*
  - za svaku grupu izračunati *AVG(ocjena)*
  - za svaku grupu formirati po jednu n-torku s vrijednošću atributa *nazPred* i izračunatim prosjekom
  - obaviti operaciju preimenovanja

```
SELECT      nazPred
      ,     AVG (ocjena) AS prosjOcj
FROM ispit
GROUP BY nazPred;
```

5. predavanje (1. ciklus; 10.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred3.pdf)

- **NULL vrijednosti** – poseban oblik prikazivanja informacije koje nedostaju (trenutno nisu poznate, uopće ne postoje ili postoje ali do njih nije moguće doći). Neovisne su od tipa podataka kojeg predstavljaju i SQL-u se koristi kontanta NULL. Način na koji se prikazuje korisniku ovisi o programskom alatu koji koristi.
- **SQL Izrazi** – sastoji se od imena atributa, kontanti, operatora i zagrada te se mogu koristiti u listi za selekciju, u uvjetu u WHERE dijelu naredbe i drugdje.
- **NULL vrijednost u izrazima** – prilikom korištenja i unarnih operatora ∈ (+, -) i binarnih ∈ (+, -, *, /) operatora s NULL vrijednosti rezultati izraza također poprimaju NULL vrijednost.
- **NULL vrijednost u uvjetima usporedbe** – prilikom korištenja NULL vrijednosti u uvjetima usporedbe vrijede sljedeća pravila
  - ako niti jedan operand nije NULL onda je rezultat logička vrijednost istina (true) ili laž (false)
  - ako su jedan ili oba operanda NULL onda je rezultat logička vrijednost nepoznato (unknown)
- **NULL vrijednost i operacija selekcije** – obavljanjem operacije $\sigma_F(r)$ dobiva se relacija koja sadrži samo one n-torke za koje je vrijednost predikata isključivo istina (true). Znači svugdje gdje se pojavljuje laž (false) ili nepoznato (unknown) ne ulazi u rezultat
- **SQL operatori** IS NULL **i** IS NOT NULL – rezultat je uvijek ili istina (true) ili laž (false). Nije dopušteno koristiti standardne operatore usporedbe ($<, \leq, =, \neq, \geq, >$) sa „konstantom“ NULL
- **Trovalentna logika** – tablica istinitosti u prisustvu logičke vrijednosti *unknown*.

AND	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

OR	true	unknown	false
true	true	true	true
unknown	true	unknown	unknown
false	true	unknown	false

NOT	
true	false
unknown	unknown
false	true

- **NULL vrijednost i skupovi** – postoji skup $S \in \{1, 2, 3, NULL\}$

  - *Kardinalnost skupa* je neodređena (može biti i 3 ili 4 jer NULL može poprimiti i od vrijednosti 1, 2 ili 3).
  - Narušena je *definicija skupa* jer se može pojaviti 1 ili više jednakih vrijednosti.
  - *Logička vrijednost* izraza $NULL \in S, 4 \in S$ je unknown.
  - SUBP ne razlikuje NULL vrijednosti pa se kao konvencija koristi sljedeći *način rukovanja NULL vrijednostima*
    - dopuštena je pojava samo 1 NULL vrijednosti u skupu
    - element e je kopija jednog od elemenata u skupu ako vrijednost *e* nije NULL a u skupu već postoji element *e* ili ako vrijednost elementa *e* je NULL a u skupu već postoji NULL
- **Kopija n-torke** – ako su vrijednosti korespodentnih atributa n-torki ili jednake ili su obje NULL (neformalna definicija)
- **Unija, presjek i razlika s NULL vrijednosti** – to su skupovske operacije te treba voditi računa o definiciji kopije n-torke prilikom usporedbe elemenata (vidi slajd 20 za primjer)
- **Projekcija i NULL vrijednost** – također treba pri eliminaciji voditi računa o definiciji kopije n-torke (vidi slajd 21 za primjer)
- **Kartezijev produkt i NULL vrijednost** – nema nikakvog utjecaja NULL na rezultat Kartezijevog produkta
- **Spajanje uz uvjet, prirodno spajanje i spajanje s izjednačavanjem uz NULL vrijednosti** – paziti na to da se spajaju samo one n-torke za koje je uvjet spajanja isključivo istina (true) (vidi slajd 23 i 24 za primjer)
- **Agregacija i NULL vrijednost** (vidi slajd 26 za primjer)
  - *ako su sve vrijednosti* za koje se izračunava agregatna funkcija NULL vrijednosti ili ako agregatna funkcija izračunava za prazan skup onda je rezultat agregatne funkcije COUNT nula a ostalih NULL
  - *ako se među vrijednostima* za koje se agregatna funkcija izračunava nalaze vrijednosti koje nisu NULL onda agregatna funkcija izračunava tako da se NULL vrijednosti zanemaruju
  - *COUNT** – broji n-torke zanemarujući njihov sadržaj (dok COUNT broji n-torke u kojima vrijednost atributa nije NULL, primjer na slajdu 27)
- **Grupiranje i NULL vrijednost** – treba paziti na kopije n-torke i grupirati tako da u istu grupu ulaze one n-torke čije su X-vrijednosti međusobno jednake ako grupiramo prema atributima iz skupa X (vidi primjer na slajdu 28)

6. predavanje (1. ciklus; 12.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred3[2].pdf)
– download slajdova (http://www.fer.hr/_download/repository/pred4.pdf)

- **Vanjsko spajanje**
  - *Lijevo vanjsko spajanje (Left outer join)* - $*\triangleright\triangleleft$
    - n-torke „lijeve“ relacije za koje ne postoj n-torka u „desnoj“ relaciji se kao vrijednost atributa postavljaju NULL vrijednosti

```
SELECT mjesto.*, zupanija.* -- moze i SELECT *
        FROM mjesto LEFT OUTER JOIN zupanija
                ON sifZupMj=sifZup;
```

  - *Desno vanjsko spajanje (Right outer join)* – $\triangleright\triangleleft*$
    - n-torkama „desne“ relacije za koje ne postoji n-torke u „lijevoj“ relaciji će se kao vrijednost atributa postaviti NULL vrijednosti
    - SQL – ključna riječ RIGHT OUTER JOIN
  - *Puno vanjsko spajanje (Full outer join)* – $*\triangleright\triangleleft*$
    - sve n-torke iz obje relacije će se pojaviti u rezultatu spajanja a za one koje nemaju svoj par će vrijednosti atributa biti postavljenje na NULL
    - SQL – ključna riječ FULL OUTER JOIN
  - *Prirodno vanjsko spajanje*
    - kod *vanjskog spajanja uz uvjet i vanjskog spajanja s izjednačavanjem* u shemi rezultata se pojavljuju svi atributi obje relacije

        - kod *prirodnog lijevog vanjskog spajanja* iz sheme rezultata se izbacuju istoimeni atributi desnog operanda jer ionako mogu poprimiti ili vrijednosti jednake vrijednostima korespodentnih atributima lijevog operanda ili NULL vrijednosti
        - kod *prirodnog desnog vanjskog spajanja* iz sheme rezultata se izbacuju istoimeni atributi lijevog operanda
        - kod *punog vanjskog spajanja* iz sheme rezultata izbacuju se istoimeni atributi obje relacije te se primjenjuje operacije preimenovanja atributa
- **SQL (Structure Query Language)** – standardni jezik relacijskih baza podataka koji objedinjuje funkcije jezika DDL i DML, razvoj započeo 70ih godina (IBM San Jose Research Laboratory, California, USA)
    - proizvođači imaju pravo i to koriste te često *ugrađuju i svoje DDL i DML naredbe* te tako programski kod postaje neprenosiv između različitih SQL sustava i tako se otežava usuglašavanje oko budućih standarda
    - *SQL je neproceduralan* što nam govori da je naredbom dovoljno opisati što se želi dobiti kao rezultat i nije potrebno definirati kako do rezultata doći (SUBP pronalazi najefikasniji način)
- **SQL vrste objekata** – baza podataka (database), relacija (table), atribut (column), virtualna tablica (view), sinonim (synonym), integritetsko ograničenje (constraint), indeks (index), pohranjena procedura (stored procedure), varijabla u pohranjenoj proceduri (SPL variable), okidač (trigger)
- **SQL identifikatori (imena objekata**) – formiraju se iz slova, znaka '_' i znamenki tako da prvi znak (od 128 značajnih) mora biti slovo ili znak '_'
- **SQL rezervne riječi** – „neosjetljiv“ na razliku u velikim i malim slovima kad se radi o rezerviranim riječima (SELECT, WHERE, UPDATE, DELETE, GROUP BY, ...) i identifikatorima ali ako se radi o nizovima znakova onda je „osjetljiv“ (case sensitive)
- **SQL format naredbi** – jezik slobodnog formata kao i C
- **SQL komentari**
    - *blok komentari* – protežu se kroz više redova, označava se kao i u C-u (/* i */)
      ```
      /* ovo je komentar koji se
                  proteže kroz više redaka teksta */
      ```
    - *linijski komentar* – do kraja reda (--)
      ```
      -- ovo je komentar koji se proteže do kraja retka
      ```

7. predavanje (1. ciklus; 17.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred4.pdf)

- **SQL tipovi podataka**
    - *INTEGER* – cijeli broj pohranjen u 4 bajta u dvojnom komplementu. Raspon brojeva određen je intervalom $[-^{2n-1}, 2^{n-1}+1]$ ali pošto se u broj -2^{n-1} pohranju NULL vrijednost u stvarnosti je raspon brojeva $[-2^{n-1}+1, 2^{n-1}+1]$
    - *SMALLINT* – cijeli broj pohranjen u 2 bajta
    - *CHAR(m)* – znakovni niz (string) s unaprijed definiranom maksimalnom znakova m koja označava oktet a ne broj znakova. Napomena: Koriste se jednostruki navodnici ('Ana').
    - *NCHAR(m)* – isto kao i CHAR(m) samo omogućava ispravno leksikografsko uređenje nizova znakova koji sadrže nacionalne kodne stranice (character set). Znakovi 'ž', 'ć', ... zauzimaju po 2 okteta
    - *REAL* – odgovara tipu podataka *float* u jeziku C (IEEE-754 format u jednostrukoj preciznosti)
    - *DOUBLE PRECISION* – odgovara tipu podataka *double* u jeziku C (IEEE-754 format u dvostrukoj preciznosti)
    - *DECIMAL(m,n)* – *m* predstavlja ukupan broj znakova koj je ≤ 32, a *n* predstavlja broj znamenki iza decimalne točke ($n \leq m$). Razlikuje se od *float* i *double* tipa podatka u jeziku C jer nema numeričke pogreške
    - *DATE* – podaci ovog tipa se uvijek prikazuju u obliku datuma (npr. 18.11.2006.) a interno predstavlja broj proteklih dana od 31.12.1899. Omogućava korištenje operacija zbrajanja i oduzimanja.

- **NULL vrijednost –** način na koji se prikazuje ovisi o korištenom programskom alatu. Interno se pohranjuje drugačije od bilo koje druge dopuštene vrijednosti (nije 0, nije 0.0, nije prazan niz, ...) i za korisnika je potpuno nevažan
- **Projection Clause**
  - *ALL* – isto kao i da nema ništa
  - *DISTINCT* – briše duplikate n-torki (stvara pravu relaciju u kojoj ne smije biti jednakih n-torki)
  - *FIRST max* – rezultat je relacija s *max* brojem n-torki ali se nezna koje će se dobiti kao prvih *max* n-torki
- **Izraz (Expression)** – redoslijed obavljanja operacija u složenim izrazima određuje se prema istim pravilima kao i u programskom jeziku C dok se konverzija tipova podataka prilikom evaulacije obaljva prema sličnim pravilima kao i u C-u.
  - *Unarni operatori:* +, -
  - *Binarni operatori:* +, -, *, /, || (ulančavaje nizova znakova, konkatenacija)
- **Funkcije (function expression)**
  - *ABS (num_expression)* – računa apsolutnu vrijednost izraza i mora biti numerički tip podataka a rezultat ovisi o tipu ulaznog argumenta
  - *MOD (dividend, divisor)* – računa ostatak cjelobrojnog dijeljenja djeljenika (*dividend*) i djelitelja (*divisor*). Rezultat funkcije je cijeli broj a djeljenik i djelitetelj su numerički tipovi podataka (INTEGER, DECIMAL, FLOAT, ...)
  - *ROUND (expression[, rounding_factor])* – zaokružuje vrijednost izraza (*expression*) i ukoliko se ne navede *rounding_factor* uzima se da je on jednak 0. Tip podatka rezultata ovisi o tipu podatka ulaznog argumenta
  - *SUBSTRING (source_string FROM start_position [FOR lenght])* – vraća podniz zadanog niza te ukoliko se *lenght* ne navede vraća se podniz koji počinje na *start_position* a završava gdje i niz *source_string*
  - *UPPER (expression)* – sva mala slova ('a' – 'z') zamjenjuje odgovarajućim velikim slovima ('A' – 'Z')
  - *LOWER (expression)* – sve velika slova zamjenjuje odgovarajućim malim slovima
  - *TRIM (source_expression)* – vraća niz znakova koji nastaje tako da se s početka i kraja niza *source_expression* izbace sve praznine
  - *CHAR_LENGHT (expression)* – vraća broj znakova u zadanom nizu uključujući i praznine
  - *OCTET_LENGHT (expression)* – vraća broj bajtova zadanog niza uključujući i praznine
  - *USER* – vraća login korisnika koji je trenutno prijavljen za rad s bazom podataka
  - *TODAY* – vraća današnji datum dobiven iz operacijskog sustava
  - *MDY (month, day, year)* – vraća varijablu tipa DATE, tj. izračunava datum iz tri INTEGER varijable koje predstavljaju mjesec (cijeli broj u intervalu [1,12]), dan (0 < cijeli broj < broj dana u određenom mjesecu) i godinu (četveroznamenkasti cijeli broj)
  - *DAY (date_expression*) – vraća redni broj dana u mjesecu za zadani datum
  - *MONTH (date_expression)* – vraća redni broj mjeseca za zadani datum
  - *YEAR (date_expression)* – vraća redni broj godine za zadani datum
  - *WEEKDAY (date_expression)* – vraća redni broj dana u tjednu za zadani datum
- **Funkcije i NULL vrijednosti** – ukoliko se kao jedan ili više argumenata funkcije zada NULL vrijednost rezultat funkcije će također biti NULL vrijednost
- WHERE **Clause** – vrijednost svake n-torke iz relacije table se uvrštavaju u *Condition* i ako je dobiveni sud istinit n-torka se pojavljuje u rezultatu
- **Condition** – uvjet (condition) se sastoji od operanada (imena atributa iz relacije table i konstante) i operatora (operatori usporedbe: $<, \leq, =, <>, \geq, >$ i logički operatori: AND, OR i NOT)
- **Comparison condition**
  - *wildcardovi '%*' koji zamjenjuje bilokoju kombinaciju znakova (0 i više znakova) i '_' koji zamjenjuje točno jedan znak
  - kod BETWEEN intervali su uključeni
- **Conditional expression**

  - *1. oblik* sličan *if – else if – else* naredbi u C-u
  - *2. oblik* sličan *switch – case – default* naredbi u C-u
- **Unija (UNION)** – UNION izbacuje duplikate, a UNION ALL ne izbacuje

8. predavanje (1. ciklus; 19.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred4.pdf)

- FROM **Clause** – ukoliko se obavljaju operacije spajanja i selekcije, uvjete spajanja navodimo u ON dijelu naredbe, a uvjete selekcije u WHERE dijelu SELECT naredbe
  - *klasična sintaksa (Table Refrence)* može se koristiti za operacije Kartezijevog produkta, spajanja uz uvjet i spajanja s izjednačavanjem te prirodno spajanje (uvjeti spajanja se navode u WHERE dijelu SELECT naredbe)
  - *ANSI sintaksa (ANSI Joined Table)* može se koristiti još i za operaciju vanjskog spajanja uz uvjet, s izjednačavanjem i prirodno vanjsko spajanje
    - logički promatrano *redoslijed spajanja je s lijeva na desno*, tj. prvo se spoje prve dvije relacije pa se onda njihov rezultat spoji s trećom, itd...
    - ako se ne koristi vanjsko spajanje *redoslijed je* nebitan, dok ako se koristi onda je *itekako važan*
  - preimenovanje relacija unutar upita radi se pomoću allias imena i ključne riječi AS koja se može ispustiti te je vidljivo samo unutar upita, tj. ne mijenja se stvarno ime relacije
- **Paralelno spajanje**

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
013	Ban	10000	10000

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

mbr	prez	nazMjestoR	pbrStan	nazMjestoS
103	Ban	Zagreb	10000	Zagreb

mjestoR

pbr	nazMjesto
10000	Zagreb
21000	Split

mjestoS

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT      mbr, prez
      ,     pbrRod, mjesto.nazMjesto AS nazMjestoR
      ,     prbrStan, mjesto.nazMjesto AS nazMjestoS
FROM student, mjesto
WHERE       student.pbrRod = mjesto.pbr
      AND   student.pbrStan = mjesto.pbr;
```

- *neispravno rješenje* jer istovremeno vrijednosti atributa pbrRod i pbrStan iz relacije *student* su jednake vrijednosti atributa pbr iz relacije *mjesto*

```
SELECT      mbr, prez
      ,     pbrRod, mjesto.nazMjesto AS nazMjestoR
      ,     prbrStan, mjesto.nazMjesto AS nazMjestoS
FROM        student
      ,     mjesto AS mjestoR
      ,     mjesto AS mjestoS
WHERE       student.pbrRod = mjesto.pbr
      AND   student.pbrStan = mjesto.pbr;
```

- *ispravno rješenje* je koristeći AS napraviti da se samo u upitu (ne radi se kopija relacije) relacija mjesto nalazi u dvije različite uloge

- **Refleksivno spajanje** – pojedine n-torke iz relacije povezane su s drugim n-torkama iste relacije

orgjed

sifOrgjed	nazOrgjed	sifNadorgjed
1	Uprava	NULL
2	Odjel A	1
3	Odjel B	1
4	Pododjel X	2
5	Pododjel Y	2
6	Pododjel Z	3

nazNadorgjed
NULL
Uprava
Uprava
Odjel A
Odjel A
Odjel B

- Kako u tu relaciju dodati stupac (atribut) s nazivom *nazNadorgjed* u kojemu će pisati nazivi nadzornih organizacijskih jedinica određenog organizacijskog odjela?
  - *radi se o spajanju relacije same sa sobom* i javlja se isti problem da relacija *orgjed* ima dvije funkcije i da se dva puta mora pojaviti u upitu

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

  - *LEFT OUTER JOIN* zato da bi se pojavila i n-torka Uprava, a pomoću ključne riječi AS radimo u upitu duplikat relacije *orgjed* jer ju dva puta moramo koristiti

- GROUP BY **Caluse** – tu se navodi jedan ili više atributa relacija navedenih u FROM dijelu naredbe te nije dopušteno korištenje izraza ili zamjenskih naziva (display_label)

```
SELECT      nazPredmet AS naziv
      ,     AVG (ocjena)
FROM ispit
GROUP BY naziv;
```

- HAVING **Clause** – Kako u rezultatu prikazati samo one grupe koje zadovoljavaju neki uvjet?

  - *u Condition dijelu* koji se navodi u HAVING dijelu naredbe dopušteno je u izrazima izvan agregatnih funkcija koristiti samo one atribute koji su navedeni u GROUP BY dijelu naredbe

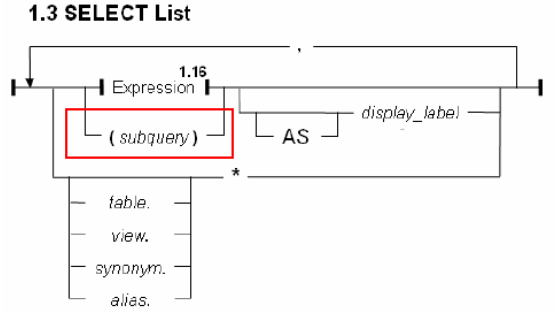
```
SELECT      nazPredmet AS naziv
      ,     AVG (ocjena) AS prosjek
FROM ispit
GROUP BY naziv
HAVING matbr > 104;
```

- ORDER BY **Clause** – koristi se za sortiranje rezultata upita te ukoliko se na navede smjer sortiranja podrazumjeva se uzlazni (ASC) smjer sortiranja
  - mogu se koristiti i *izrazi koji nisu navedeni* u listi za selekciju
  - jedino mjesto u SELECT naredbi u kojem je *dopušteno referencirati se* na zamjensko ime (display_label)
  - u jednoj SELECT naredbi može biti *najviše jedan ORDER BY* (ukoliko se koristi UNION ORDER BY ide iza zadnje SELECT naredbe)
  - SQL standardi zahtjeva da se NULL vrijednost pri sortiranju uvijek smatra ili manjim ili većim od svih drugih vrijednosti (IBM Informix NULL treitra *kao uvijek manju vrijednost* od ostalih)
- **Redoslijed obavljanja dijelova SELECT naredbe** – FROM → WHERE → GROUP BY → HAVING → DISTINCT → UNION → ORDER BY

9. predavanje (1. ciklus; 24.03.2009.)

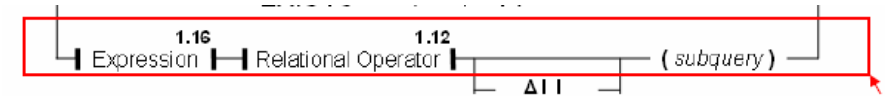
– download slajdova (http://www.fer.hr/_download/repository/pred5.pdf)

- **Podupiti** – upit koji je ugrađen u neki drugi upit te se taj drugi upit u koji je ugrađen podupit naziva vanjski upit (*outer querry*).
  - *može se ugraditi u* listu selekcije (SELECT List) vanjskog upita, u uvjet (*Condition*) u WHERE dijelu vanjskog upita i u uvjet (*Condition*) u HAVING dijelu vanjskog upita.
  - *može sadržavati* sve do sada spomenut SELECT naredbe osim ORDER BY
  - *u vanjski upit se može ugraditi* više podupita u koje se opet može ugraditi više podupita, itd...
  - Banek kaže da je to *najteži dio gradiva definitivno* ☺
- **Skalarni podupit** – najjednostavniji podupit čiji je rezultat jedna jednostavna vrijednost (skalar) (npr. podatak tipa *cijeli broj*, *niz znakova*, *datum*, itd...), tj. relacija kardinalnosti 1 i stupnja 1 i vrijednost atributa dotične n-torke se u vanjskom upitu koristi kao skalarna vrijednost
- **Podupit u listi za selekciju (**SELECT List**)** – npr. *Ispisati podatke o svim vozilima ali uz njih ispisati podatak o najvećoj težini tereta*




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

- **Podupit u** WHERE **dijelu naredbe** – ako se koristi ovako dolje navedeni oblik uvjeta (točka 1.18 u službenom podsjetniku) u WHERE ili HAVING dijelu naredbe tada je dopušteno koristiti isključivo skalarne podupite (*npr. Ispisati podatke o studentima koji su rođeni u mjestu Ludbreg, a stanuju u mjestu Varaždin*)

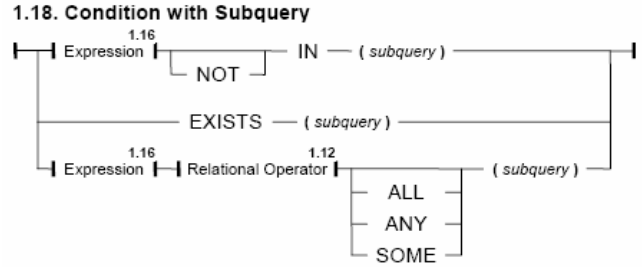



```
SELECT *
        FROM vozilo
        WHERE pbrRod = (SELECT pbr FROM mjesto WHERE nazMjesto = 'Ludbreg') AND
                pbrSt = (SELECT pbr FROM mjesto WHERE nazMjesto = 'Varaždin');
```

- **Podupiti u HAVING dijelu naredbe** – koriste se na isti način kao i u WHERE dijelu naredbe
- **Korelirani upit** – podupit koji je koleriran s vanjskim upitom, tj. u podupitu se koriste atributi iz vanjskog upita. Najčešće se korelirani podupit mora izvršiti za svaku n-torku iz vanjskog upita.
  Npr. Ispisati podatke o strojevima koji su ukupno korišteni više od dopuštenog broja radnih sati.

```
SELECT oznStr, dopBrSati
        FROM stroj
        WHERE dopBrSati < (SELECT SUM(brSatiRada) FROM radStroja WHERE oznStr = stroj.oznStr);
```

  - *rezultat upita ovisi o* vrijednostima atributa vanjskog upita, tj. za svaku n-torku se dobiva drugačiji rezultat podupita
  - *upit se (logički promatrano) obavlja na sljedeći način:*
    - vanjski upit uzima jednu n-torku iz relacije *stroj* te na temelju sadržaja te n-torke i sadržaja relacija *RadStroja* u podupitu se izračunava sumu sati rada dotičnog stroja
    - ukoliko je uvjet usporedbe zadovoljen, testirana n-torka se pojavljuje u rezultatu
    - postupak se ponavlja za svaku n-torku relacije stroj
  - *korištenje atributa vanjskog upita u podupitu:*
    - u vanjskom upitu se nemogu koristi atributi podupita
    - ukoliko se imena atributa vanjskog upita podudaraju s imenima atributa podupita onda se ime navedeno u podupitu odnosi na ime atributa navedeno u podupitu, a ime navedeno u vanjskom upitu na ime atributa u vanjskom upitu
    - ukoliko je potrebno razrješiti dvosmislenost dovoljno je preimenovati relaciju u vanjskom upitu ili u podupitu
- **Jednostupčani upit (*Single-column subquerry*)** – rezultat je relacija stupnja 1 a kardinalnosti ≥ 0, a koriste se u WHERE ili HAVING dijelu vanjskog upita (ne koriste se u listi za selekciju).
  Npr. Ispisati podatke o studentima čije je prezime različito od svih prezimena nastavnika.




```
SELECT *
        FROM stud
        WHERE prez <> ALL (
                        SELECT prez
                        FROM nastavnik);
```

Npr. Ispisati podatke o dvoranama čiji je kapacitet veći od broja studenata u barem jednoj od grupa.

```
SELECT *
        FROM dvorana
        WHERE kapacitet > SOME (SELECT brSt FROM grupa);
```

Npr. Ispisati podatke o studentima koji su bilo koji predmet položili tijekom akademske godine 2005.

```
SELECT *
        FROM student
        WHERE mbr IN (SELECT mbr FROM ispit WHERE akGod = 2005 AND ocjena > 1);
```

Npr. Ispisati podatke o studentima koji u akademskoj godini u kojoj su upisali studij nisu položili niti jedan ispit.

```
SELECT *
        FROM student
        WHERE NOT EXISTS (SELECT mbr FROM ispit WHERE ispit.mbr = student.mbr
                                                AND akGod = akGodUpis
                                                AND ocjena > 1;
```

Npr. Ispisati naziv(e) predmeta s najvećim prosjekom.

```
SELECT predmet
        FROM ispit
        GROUP BY predmet
        HAVING AVG (ocjena) >= ALL (SELECT AVG(ocjena) FROM ispit GROUP BY predmet);
```

- **Presjek** – npr. Ispisati *studente* koji su položili i *Matematiku* i *Programiranje*

```
SELECT *
        FROM polozioMatem
        WHERE EXISTS
        (SELECT * FROM polozioProgr WHERE polozioProgr.mbr = polozioMatem.mbr);
```

- **Razlika** – npr. *Studenti* koji su položili *Matematiku* ali nisu položili *Programiranje*

```
SELECT *
        FROM polozioMatem
        WHERE NOT EXISTS
        (SELECT * FROM polozioProgr WHERE polozioProgr.mbr = polozioMatem.mbr);
```

THE END 1. CIKLUSA ☺

10. predavanje (1. ciklus; 26.03.2009.)

– download slajdova (http://www.fer.hr/_download/repository/pred5.pdf)

- na 1. satu smo vježbali zadatka sa prošlih MI, a na 2. satu smo obradili slajdove 45 – 66 ali pošto to gradivo ne ulazi u gradivo 1. MI onda sad neću pisati taj sažetak nego ću to napraviti u 1. tjednu 2. ciklusa kad će to biti potrebno