Napredne baze podataka

Priprema za ispit

Tehničko veleučilište u Zagrebu
Informatičko-računarski odjel

•Sadržaj

1. Očuvanje integriteta

- Integritet baze podataka
  - Korektnost / istinitost podataka (dopuštene vrijednosti)
  - Konzistencija /ispravnost podataka (međusobna suglasnost)

I. Opća pravila integriteta
   1. Pravilo entitetskog integriteta
   2. Pravilo referencijskog integriteta

II. Korisnička pravila integriteta
   3. Pravilo domenskog integriteta
   4. Pravilo odnosnog integriteta

1. Očuvanje integriteta

- 1. Pravilo entitetskog integriteta
  - Vrijednost primarnog ključa kao cjeline ne smije biti jednaka NULL vrijednosti.
  - Jednostavan / složen primarni ključ
  - Osigurava se: pomoću oznake PRIMARY KEY

- 2. Pravilo referencijskog integriteta
  - Ako u relacijskoj shemi R postoji strani ključ koji odgovara primarnom ključu rel. sheme S, tada svaka vrijednost stranog ključa u relaciji r(R) mora biti:
    - ili jednaka vrijednosti primarnog ključa neke n-torke iz relacije s(S)
    - ili jednaka NULL vrijednosti
  - Osigurava se: pomoću oznake FOREIGN KEY

1. Očuvanje integriteta

- 3. Pravilo domenskog integriteta
  - Definira domenu atributa - specificira skup vrijednosti koje atribut smije poprimiti
  - Osigurava se: djelomično osiguran samom definicijom tipa podataka

- 4. Pravilo odnosnog integriteta
  - Određuju se dozvoljeni odnosi među pojedinim atributima
  - Osigurava se: programski / okidačima

2. Zaštita od neovlaštenog korištenja podataka

- Baza mora biti zaštićena od nedopuštenih radnji
- Osnovni vid zaštite: ograničenje fizičkog pristupa do samog računala i njegovih terminala
- Softwareski vid zaštite koji se ugrađuje u DBMS

A. Identifikacija korisnika
   A. Korisničko ime
   B. Lozinka
B. Dodjeljivanje ovlaštenja (dozvola)

2. Zaštita od neovlaštenog korištenja podataka

- Pristupanje podacima preko Interneta
  - Internet – baziran na IP protokolu – komunikacija preko paketa
  - Svaki IP paket dodjeljuje se jednoj vrsti Internet usluge – identificira se preko PORTa
  - Svaka usluga na Internetu ima svoj broj porta (FTP 20/21, SSH 22, telnet 23, mail 25, www 80, …)
  - MySQL -> 3306

- Kako zaštiti bazu podataka na Internetu ?
  - Sasvim isključiti mrežne mogućnosti DBMSa
  - Dopustiti da samo lokalni programi pristupaju DBMSu (localhost)
  - Dopustiti da samo računala unutar lokalne mreže pristupaju DBMSu
  - Dopustiti nekima, ali uz identifikaciju (korisnik/lozinka)
  - Koristiti šifriranu komunikaciju (ssl/ssh, ...)

2. Zaštita od neovlaštenog korištenja podataka

Dodjeljivanje ovlaštenja

- GRANT
  - Dodjeljivanje samo onih dozvola koje su potrebne *korisniku* nad odgovarajućim skupom podataka (odgovarajuća granulacija)
- Korisnik: ‘***ime_korisnika’@’lokacija’***
- Granulacija – pristup podacima na odgovarajućoj razini:
  - Sustav
  - Baza
  - Tablica
  - Atribut
  - Pohranjeni zadatak

2. Zaštita od neovlaštenog korištenja podataka

Dodjeljivanje ovlaštenja

```
GRANT vrsta_privilegije
    ON skup_podataka
    TO 'ime_korisnika'@'lokacija'
    [IDENTIFIED BY lozinka]
```

- Primjer:

  GRANT SELECT, INSERT, DELETE, DROP
  ON autoradionica.nalog
  TO 'admin1'@'localhost'
  IDENTIFIED BY 'Loz1nka';

- Primjer:

  REVOKE SELECT
  ON autoradionica.nalog
  FROM 'admin1'@'localhost';

3. Kontrola istovremenog pristupa podacima (transakcije)

- TRANSAKCIJA
  - jedinica posla provedena nad bazom podataka koja mora biti:
    - **provedena u cijelosti**

      ILI
    - **ne smije biti provedena uopće**
  - Više međusobno povezanih radnji nad podacima koje čuvaju konzistenciju i korektnost pohranjenih podataka
  - Provodi bazu iz jednog konzistentnog stanja u drugo

- Svojstva transakcija
  - **A**
  - **C**
  - **I**
  - **D**

•3. Kontrola istovremenog pristupa podacima (transakcije)

```
SET AUTOCOMMIT = 0;
BEGIN;
        UPDATE radnik
        SET koefPlaca=koefPlaca+0.5
        WHERE prezimeRadnik = 'Kruljac' AND imeRadnik='Petar';
SAVEPOINT tocka;
UPDATE radnik
        SET koefPlaca=koefPlaca-0.5
        WHERE prezimeRadnik = 'Parlov' AND imeRadnik='Dino';
ROLLBACK TO SAVEPOINT tocka;
SELECT * FROM radnik WHERE (prezimeRadnik = 'Kruljac' AND
imeRadnik='Petar')
        OR (prezimeRadnik = 'Parlov' AND imeRadnik='Dino');
COMMIT;
SET AUTOCOMMIT = 1;
```

•3. Kontrola istovremenog pristupa podacima (transakcije)

```
SET AUTOCOMMIT = 0;
BEGIN;
        UPDATE radnik
        SET koefPlaca=koefPlaca+0.5
        WHERE prezimeRadnik = 'Kruljac' AND imeRadnik='Petar';
SAVEPOINT tocka;
UPDATE radnik
        SET koefPlaca=koefPlaca-0.5
        WHERE prezimeRadnik = 'Parlov' AND imeRadnik='Dino';
ROLLBACK TO SAVEPOINT tocka;
SELECT * FROM radnik WHERE (prezimeRadnik = 'Kruljac' AND
imeRadnik='Petar')
        OR (prezimeRadnik = 'Parlov' AND imeRadnik='Dino');
COMMIT;
SET AUTOCOMMIT = 1;
```

Što radi gornji kod?

•3. Kontrola istovremenog pristupa podacima (transakcije)

- Napisati transakciju koja će kvaru s nazivom „Izmjena amortizera“ postaviti vrijednost atributa *sifKvar* na 100. Uz to, potrebno je promijeniti vrijednosti odgovarajućeg atributa (*sifKvar*) u tablicama *nalog* i *rezervacija* na novu vrijednost. Potrebno je potvrditi navedene promjene.

```
SET AUTOCOMMIT = 0;
BEGIN;
        SELECT sifKvar INTO @a FROM kvar
                WHERE nazivKvar='Izmjena amortizera';
        UPDATE kvar SET sifKvar=100
                WHERE nazivKvar='Izmjena amortizera';
        UPDATE nalog SET sifKvar=100 WHERE sifKvar=@a;
        UPDATE rezervacija SET sifKvar=100 WHERE sifKvar=@a;
COMMIT ;
SET AUTOCOMMIT = 1;
```

- **3. Kontrola istovremenog pristupa podacima (transakcije)**

  Serijabilnost (serijalizabilnost)

  - Neka se u višekorisničkoj bazi podataka izvodi nekoliko transakcija paralelno tako da se pojedini dijelovi tih transakcija izvode vremenski izmiješano
  - Ako je konačni učinak njihovog izvođenja isti kao da su se one izvršavale serijski (sekvencijalno), kažemo da se radi o serijabilnom (ili serijalizabilnom) izvršavanju transakcija
  - Primjer: rezervacija avionskih karata sa dvije različite lokacije
  - Rješenje: **zaključavanje podataka i 2PL protokol**

4. Zaključavanje podataka

- KLJUČ ZA PISANJE/IZMJENU - WRITE LOCK
- KLJUČ ZA ČITANJE - READ LOCK
- UNAPRIJEDIVI KLJUČ – PROMOTABLE LOCK

	Read	Promotable	Write	Nije zaključano
Read	OK	OK	Greška	OK
Promotable	OK	Greška	Greška	OK
Write	Greška	Greška	Greška	OK

- Granulacija zaključavanja:
  - Baza podataka
  - Tablica (relacija)
  - Redak (n-torka)
  - Indeks

5. Oporavak baze podataka

- Baza podataka se može naći u nekonzistentnom stanju zbog prekida transakcije, pogrešnog rada transakcije, ili nekih drugih hardwareskih ili softwareskih razloga

- Do pogrešnog rada transakcije može doći zbog:
  - logičke greške u samoj transakciji
  - softwareske greške u DBMS-u ili operacijskom sustavu
  - hardwareske greške u računalu
- Do prekida transakcije može doći:
  - zbog odustajanja od same transakcije
  - kao posljedica kontrole istovremenog izvršavanja više transakcija
  - zbog softwareske greške u DBMSu ili operacijskom sustavu
  - hardwareske greške u računalu

5. Oporavak baze podataka

- DBMS mora omogućiti oporavak baze, tj. njen povratak iz nekonzistentnog u blisko konzistentno stanje

- To se postiže:
  - rezervnom kopijom baze podataka
  - žurnal datotekom
    - Odmotavanje unatrag
    - Odmotavanje unaprijed

6. Pohranjene rutine

- Pohranjene procedure (stored procedures)
  - Pozivaju se naredbom CALL
  - Ne vraćaju razultat direktno (pomoću naredbe RETURN)
  - Mogu vratiti rezultat preko parametara (varijabli)
- Pohranjene funkcije (stored functions)
  - Izvršavaju funkciju i vraćaju rezultat pomoću naredbe RETURN

6.1. Procedure

Napisati proceduru koja za zadani naziv županije, preko parametra vraća iznos najmanje i najveće plaće radnika koji živi u toj županiji. Potrebno je napisati primjer poziva procedure te ispisa rezultata.

```
DELIMITER ??
DROP PROCEDURE IF EXISTS iznosiPlace ??
CREATE PROCEDURE iznosiPlace(IN zup VARCHAR(50),
        OUT minimum INT, OUT maksimum INT)
BEGIN
SELECT MAX(koefplaca*iznosOsnovice), MIN(koefplaca*iznosOsnovice)
        INTO maksimum , minimum
                FROM radnik JOIN mjesto ON pbrStan=pbrMjesto
                NATURAL JOIN zupanija
                WHERE nazivZupanija=zup;
END ??
DELIMITER ;

CALL iznosiPlace('Grad Zagreb',@a,@b);
SELECT @a AS najmanja, @b AS najveca;
```

6.2. Funkcije

Napisati funkciju koja za zadanog radnika vraća koliko je klijenata registriralo vozilo u županiji u kojoj radi zadani radnik. Šifru radnika funkcija mora primiti preko globalne varijable. Potrebno je napisati primjer poziva funkcije.

```
DROP FUNCTION IF EXISTS brojiReg;
DELIMITER //
CREATE FUNCTION brojiReg() RETURNS INT
DETERMINISTIC
BEGIN
        DECLARE vrati, zadanaZupanija INT;
        SELECT sifZupanija INTO zadanaZupanija
                FROM zupanija NATURAL JOIN mjesto JOIN radnik ON pbrStan=pbrMjesto
                WHERE sifRadnik=@zadaniradnik;
        SELECT COUNT(*) INTO vrati
                FROM klijent JOIN mjesto ON pbrReg=pbrMjesto NATURAL JOIN zupanija
                WHERE sifZupanija=zadanaZupanija;
        RETURN vrati;
END;
//
DELIMITER ;

SET @zadaniradnik=122;
SELECT brojiReg();
```

6.2. Funkcije

- Napisati funkciju koja će primiti podatak o radniku (*sifRadnik*). Funkcija mora vratiti:
  - 1 ako radnik ima plaću veću od prosječne plaće
  - 0 ako radnik ima plaću jednaku iznosu prosječne plaće
  - -1 ako radnik ima plaću manju od prosječne plaće

```
DELIMITER %%
DROP FUNCTION IF EXISTS klasificiraj %%
CREATE FUNCTION klasificiraj (zadaniRadnik INT) RETURNS INT
DETERMINISTIC
        BEGIN
                DECLARE k INT DEFAULT NULL;
                DECLARE placaRadnika, prosjecnaPlaca DECIMAL(6,2) DEFAULT NULL;
                SELECT (koefPlaca*iznosOsnovice) INTO placaRadnika
                        FROM radnik WHERE sifRadnik=zadaniRadnik;
                SELECT AVG(koefPlaca*iznosOsnovice) INTO prosjecnaPlaca FROM radnik;
                IF placaRadnika>prosjecnaPlaca THEN RETURN 1;
                ELSEIF placaRadnika=prosjecnaPlaca THEN RETURN 0;
                ELSE RETURN -1;
                END IF;
        END %%
DELIMITER ;
Primjer poziva funkcije:
SELECT klasificiraj(285);
```

6. Pohranjeni zadaci

Kursori & kontrola toka

- Kursor
  - Služi za procesiranje pojedinih n-torki koje je baza vratila kao rezultat upita
  - Uzastopno dohvaćanje n-torki iz dobivenog seta rezultata (prilikom svake iteracije petlje)
  - Kada ne koristiti kursore?
  - `DECLARE imeKursora CURSOR FOR select_izraz;`
- Condition
  - Služi za definiranje uvjeta prilikom kojeg je potrebno preusmjeriti tok koda
- Handler
  - Služi za definiranje radnji koje se izvršavaju u slučaju pojave koja zadovoljava uvjet definiran condition-om
  - CONTINUE / EXIT / UNDO

6. Pohranjeni zadaci

Kursori & kontrola toka

Tablici radnik dodati novi atribut *brNaloga* inicijalno postavljen na -1. Napisati proceduru koja će u novostvoreni atribut upisati koliko je naloga radnik obradio. Potrebno je obraditi sve radnike, te vratiti broj obrađenih zapisa.

```
ALTER TABLE radnik ADD brNaloga INT DEFAULT -1;
DROP PROCEDURE IF EXISTS brojiNaloge1;
DELIMITER //
CREATE PROCEDURE brojiNaloge1()
BEGIN
        DECLARE kraj BOOL DEFAULT FALSE;
        DECLARE sifra, n INT DEFAULT NULL;
        DECLARE dohvaceno, obradeno INT DEFAULT 0;
        DECLARE k CURSOR FOR SELECT sifRadnik FROM radnik;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;

        OPEN k;
        /*SELECT FOUND_ROWS() INTO dohvaceno;*/
```

6. Pohranjeni zadaci

Kursori & kontrola toka

```
FETCH k INTO sifra;
                    IF kraj=TRUE THEN LEAVE vrti;
                    END IF;
                    SELECT COUNT(*) INTO n FROM nalog NATURAL JOIN radnik
WHERE sifRadnik=sifra;
                    UPDATE radnik SET brNaloga=n WHERE sifRadnik=sifra;
                    SET obradeno=obradeno+1;
          END LOOP;
          CLOSE k;
          SELECT obradeno AS obradeno_zapisa;
END
//
DELIMITER ;
CALL brojiNaloge1();
```

6. Pohranjeni zadaci

- U tablicu *radnik* dodati novi atribut *brNaloga* tipa int. Napisati proceduru koja će pozivajući funkciju sa **slidea br. 21** popuniti atribut *brNaloga* za sve radnike. Procedura vraća broj obrađenih zapisa.

```
ALTER TABLE radnik ADD brNaloga INT;
DELIMITER &&
DROP PROCEDURE IF EXISTS klasificirajSve &&
CREATE PROCEDURE klasificirajSve (OUT i INT)
BEGIN
      DECLARE sif, tmp INT DEFAULT 0;
      DECLARE error BOOL DEFAULT FALSE;
      DECLARE k CURSOR FOR SELECT sifRadnik FROM radnik;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET error=TRUE;
      SET i=0;
      OPEN k;
      vrti:LOOP
           FETCH k INTO sif;
           IF error THEN LEAVE vrti;
           END IF;
           SELECT klasificiraj(sif) INTO tmp;
           UPDATE radnik SET brNaloga=tmp WHERE sifRadnik=sif;
           SET i=i+1;
      END LOOP;
      CLOSE k;
END &&
DELIMITER ;
          Primjer poziva procedure i ispisa rezultata:
CALL klasificirajSve(@a);
SELECT @a;
```

6. Pohranjeni zadaci

- Napisati proceduru koja će za svakog radnika u novostvoreni atribut satiNaNalozima računati kolika je ukupna suma ostvarenih sati rada za tog radnika po svim nalozima. Potrebno je izračunati vrijednost atributa satiNaNalozima samo za one radnike koji imaju koeficijent plaće veći od 1.

```
ALTER TABLE radnik ADD satiNaNalozima INT;
DELIMITER &&
DROP PROCEDURE IF EXISTS sumirajSatePremaKoef &&
CREATE PROCEDURE sumirajSatePremaKoef ()
BEGIN
      DECLARE sif, suma INT DEFAULT 0;
      DECLARE error BOOL DEFAULT FALSE;
      DECLARE k CURSOR FOR
      SELECT sifRadnik, SUM(OstvareniSatiRada)
           FROM radnik NATURAL JOIN nalog WHERE koefPlaca>1 GROUP BY sifRadnik;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET error=TRUE;
      OPEN k;
      petlja:LOOP
           FETCH k INTO sif, suma;
           IF error THEN LEAVE petlja;
           END IF;
           UPDATE radnik SET satiNaNalozima=suma WHERE sifRadnik=sif;
      END LOOP;
      CLOSE k;
END &&
DELIMITER ;
Poziv procedure:
CALL sumirajSatePremaKoef();
```

7. Okidači

- objekt baze podataka koji se asocira s tablicom i aktivira kada se dogodi neki događaj nad tablicom (insert, update, delete)
- prema vremenu – 2 podkategorije
  - BEFORE – kada?
  - AFTER – kada?

- Mehanizam rada okidača?

- Tablica **NEW**
- Tablica **OLD**

•7. Okidači

Napisati okidač koji će se pokrenuti prilikom unosa novog zapisa u tablicu *rezervacija*. Okidač će provjeriti polje *satServis* i ako je unesen podatak 0 postavit će mu vrijednost na 10.

```
DROP TRIGGER rezervac;
DELIMITER $$
CREATE TRIGGER rezervac BEFORE INSERT ON rezervacija
FOR EACH ROW
BEGIN
        IF NEW.satServis=0 THEN SET NEW.satServis=10;
        END IF;
END;
$$
DELIMITER ;
```

7. Okidači

- Napisati okidač koji će prilikom brisanja zapisa iz tablice *županija* provjeriti je li ta županija referencirana u tablici *mjesto*. Ako jest, treba postaviti vrijednost tog atributa (*sifZupanija*) u tablici *mjesto* na NULL.

```
DROP TRIGGER IF EXISTS tZup;
CREATE TRIGGER tZup BEFORE DELETE ON zupanija
      FOR EACH ROW
            UPDATE mjesto SET sifZupanija=NULL
                  WHERE sifZupanija=old.sifZupanija;
/*može i bez BEGIN i END*/

Aktiviranje okidača:
DELETE FROM zupanija WHERE sifZupanija=3;
```

8. Spojnost

- $link = mysql_connect ("host", "username", "password") Or die ("Neuspjesno spajanje na server");
- Mysql_select_db ("baza“,$link)
- $upit=mysql_query(“SELECT * FROM klijent”);
- Mysql_error()
- Mysql_num_rows
- Mysql_result
- Mysql_fetch_array

•8. Spojnost

- Koristeći ranije izrađenu proceduru, ispisati za zadano mjesto iznos najmanje i najveće plaće radnika koji živi u tom mjestu.
- Dokument: **min_max.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Radnici koji popravljaju kvar</title>
</head>

<body>
        <form action="racunaj.php" method="post">
                Unesite naziv mjesta:<br />
                <input type="text" name="mj" />
        <input type="submit" value="dohvati" />
    </form>
</body>
</html>
```

8. Spojnost

- Dokument: **racunaj.php**

```
<?php

$link_server=mysql_connect("localhost", "root", "root") or die ("Ne
mogu se spojiti na poslužitelj");

$link_baza=mysql_select_db("autoradionica", $link_server) or die ("Ne
mogu se spojiti na bazu");

mysql_query("SET CHARACTER SET cp1250", $link_server);

$mjesto=$_POST['mj'];


$upit1="CALL iznosiPlace('$mjesto',@a,@b);";
$rezultat1=mysql_query($upit1, $link_server) or die (mysql_error());

$upit2="SELECT @a AS najmanja, @b AS najveca;";
$rezultat2=mysql_query($upit2, $link_server) or die (mysql_error());
```

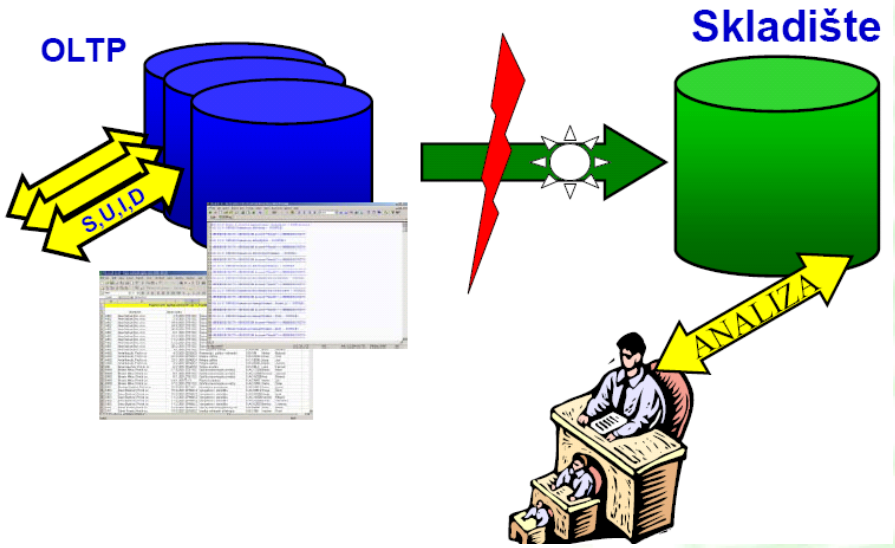
•8. Spojnost

```
$n=mysql_num_rows($rezultat2);
echo <<< ispis
        <table border="1">
                <tr>
                        <th style="width:150px;">Najveća plaća</th>
                        <th style="width:150px;">Najmanja plaća</th>
                </tr>
ispis;

for($i=0;$i<$n;$i++)
        {$row=mysql_fetch_array($rezultat2);
                echo <<< EOT
                        <tr>
                                <td> {$row['najveca']} </td>
                                <td> {$row['najmanja']} </td>
                        </tr>
EOT;
}
echo ("</table>");
mysql_close($link_server);
?>
```

9.1. Skladišta podataka

- Analitička obrada podataka organizira se u zasebnoj bazi nazvanoj skladište podataka

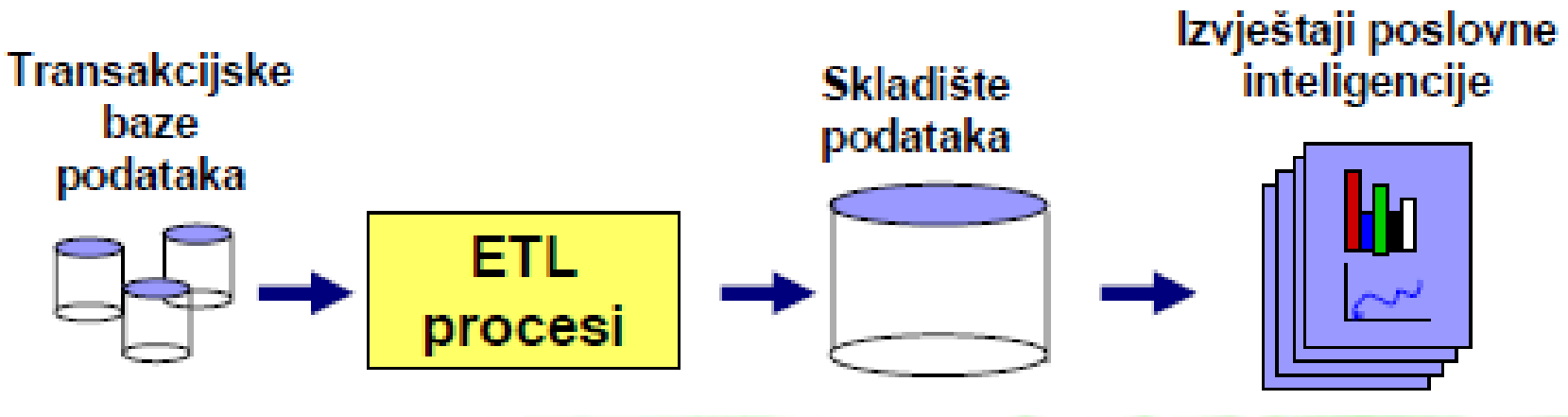


- Razlike između transakcijske baze podataka i skladišta podataka?
- Povijesni podaci, agregacija, nepromjenjenost podataka, bez NF,...

- Primjeri: ISVU skladište (prosječna ocjena,...), trgovina (ukupan broj

•Kako se generira skladište?

- **ETL postupak**
  - **Extraction** - Vađenje podataka i izbor (potpunih!) podataka iz raznorodnih najčešće transakcijskih izvora podataka
  - **Transformation** - Transformacija podataka u odgovarajući format, uključujući čišćenje, agregaciju i filtriranje
  - **Loading** - Učitavanje transformiranih i ujedinjenih podataka



9. Dubinska analiza podatka

- Netrivijalni postupak identifikacije valjanih, novih, potencijalno korisnih i prije svega razumljivih načela u podacima
- Cilj: pronaći nove i neočekivane zakonitosti među podatcima
- Medicinske aplikacije, osiguravajuća društva, E-trgovina, financijske aplikacije
- Regresija, razvrstavanje, grupiranje, asocijativna pravila