UNIVERZITET U BEOGRADU · MATEMATIČKI FAKULTET

MATΦ
University of Belgrade
Faculty of Mathematics

Uvod u relacione baze podataka

Nina Radojičić Matić

2026/2027

NAPOMENA O MATERIJALIMA

Slajdovi su nastali korišćenjem materijala **prof. dr Saše Malkova** za kurs „Uvod u relacione baze podataka” na Matematičkom fakultetu Univerziteta u Beogradu, kao i ostale literature navedene u nastavku.

Za početak: gde ste danas već koristili bazu podataka?

Studentski servis
prijava ispita, ocene, raspored

Plaćanje karticom
stanje računa, transakcije

Društvene mreže
profili, objave, poruke

Navigacija i prevoz
mape, linije, polasci

Onlajn kupovina
katalog, korpa, porudžbine

Muzika i video
biblioteka, istorija, preporuke

Šta ako bismo radili bez baze podataka?

Zamislite studentsku službu koja podatke o studentima, predmetima i prijavama ispita čuva u odvojenim Excel fajlovima.

Ponavljanje

Adresa studenta upisana je u tri različita fajla.

Nekonzistentnost

Promenjena je u jednom fajlu, a u druga dva nije. Koja je tačna?

Istovremeni rad

Dvoje studenata u istoj sekundi prijavljuje poslednje mesto u terminu.

Pad sistema

Nestane struje usred upisa ocena. Šta je sačuvano?

Ovlašćenja

Ko sme da vidi, a ko da menja ocene?

REŠENJE

Sistem za upravljanje bazama podataka

— o tome je ovaj kurs.

Šta je baza podataka?

ŠIRA DEFINICIJA

Baza podataka je **strukturirana kolekcija podataka** koja se čuva u računarskom sistemu.

KONKRETNIJA DEFINICIJA

Baza podataka je **softverski sistem** koji se stara o određenoj kolekciji podataka i kome pristupaju drugi softverski proizvodi radi izvođenja različitih operacija nad tim podacima.

Štaviše, baza podataka je računarski sistem — ali ćemo se tokom kursa uglavnom baviti njegovim softverskim elementima, pa je prethodna definicija dovoljno dobra.

Sistem za upravljanje bazama podataka

Formalno, terminom **baza podataka** označavamo samo kolekciju podataka, dok softverski sistem koji radi sa podacima nazivamo **sistem za upravljanje bazama podataka** (SUBP).



Jedan SUBP može da radi sa više baza podataka.

Uloga baze podataka i SUBP-a

Staranje o kolekciji podataka — korisnicima i drugim softverskim proizvodima omogućava:

- uopšteno definisanje strukture podataka
- održavanje podataka (dodavanje, menjanje, brisanje)
- staranje o integritetu podataka
- pretraživanje i čitanje podataka
- administriranje podataka i same baze
- istovremenu upotrebu od strane više korisnika
- pristup sa različitih računarskih sistema
- ...i za sve to garantuje određene uslove **pouzdanosti**

Relacione baze podataka — tema kursa

Najzastupljenija vrsta baza podataka	Praktično sveprisutna tehnologija.
Relativno univerzalne	Uopšten način upotrebe za veliki broj primena i visok nivo standardizacije — druge vrste baza teže da ponude što sličniji način upotrebe.
Visok nivo formalizacije	Formalizovane strukture podataka, metode pretraživanja i izračunavanja, pravila integriteta i pravila projektovanja.
Najviši nivo pouzdanosti	Upravo zahvaljujući nivou formalizacije.
Relativno visok nivo performansi	Efikasan rad i sa velikim količinama podataka.

Imaju i slabosti — o njima ćemo kasnije.

Zašto učimo baze podataka?

Široko su rasprostranjene

Gotovo svaka ozbiljna aplikacija — veb, mobilna, bankarska, naučna — oslanja se na bazu podataka.

Znanje koje ne zastareva

Relacioni model postoji od 1970. (E. F. Codd), a SQL je i danas standardni jezik za rad sa podacima.

Primena u profesionalnoj praksi

Poznavanje baza podataka i jezika SQL neophodno je u razvoju serverskih aplikacija, analizi podataka, nauci o podacima, mašinskom učenju i administraciji sistema.

Drugačiji način razmišljanja

U SQL-u opisujete **šta** želite da dobijete, a ne **kako** — to je deklarativno programiranje.

Aplikacije se menjaju, a podaci ostaju — često su vredniji od softvera koji ih koristi.

Zašto baš Db2?

Tamo gde je SQL nastao	SQL je razvijen u IBM-u 1970-ih (projekat System R); Db2 je IBM-ov relacioni SUBP iz te tradicije, na tržištu od 1983.
Za velike i kritične sisteme	Koristi se u bankama, osiguranju, javnoj upravi i drugim sistemima gde je pouzdanost presudna.
Bogat i strog SQL	Rekurzivni upiti, MERGE, OLAP funkcije, strogi tipovi — dobro za učenje formalnih koncepata.
Dostupan za rad kod kuće	Besplatno izdanje Db2 Community Edition, na primer u Docker kontejneru.

Naučeno se prenosi: PostgreSQL, Oracle, SQL Server i MySQL koriste isti relacioni model i SQL, uz manje razlike u dijalektu.

Šta ćete znati na kraju kursa?

- pisati složene SQL upite i naredbe nad realnim bazama podataka
- razumeti formalne osnove relacionog modela
- projektovati šemu baze podataka i normalizovati je
- razumeti transakcije, konkurentnost, oporavak i sigurnost podataka
- proceniti kada relaciona baza nije najbolje rešenje

Plan kursa

01	**Upitni jezik SQL**	tabele · upiti · ažuriranje · definisanje podataka
02	**Relacioni model podataka**	strukturni, manipulativni i integritetni deo
03	**Elementi projektovanja baza podataka**	nivoi projektovanja · zavisnosti · normalne forme
04	**Elementi implementiranja baza podataka**	fizička struktura · transakcije · sigurnost
05	**RBP i savremeni razvoj softvera**	slabosti · proširenja · nerelacione baze

Plan kursa — detaljnije (1/2)

01

Upitni jezik SQL

- Intuitivan koncept tabele
- Upiti
- Ažuriranje podataka
- Definisanje podataka i elemenata fizičke strukture

02

Relacioni model podataka

- Prethodni modeli podataka i kontekst nastajanja
- Formalno uvođenje relacionog modela
- Strukturni, manipulativni i integritetni deo modela

03

Elementi projektovanja BP

- Arhitektura relacionih BP
- Konceptualni, logički i fizički nivo
- Logičko projektovanje: funkcionalne i višeznačne zavisnosti, normalne forme, normalizacija

Plan kursa — detaljnije (2/2)

04

Elementi implementiranja BP

- Fizička struktura baze podataka
- Transakcije, katanci, konkurentnost, oporavak
- Sigurnost podataka
- Optimizacija

05

RBP i savremeni razvoj softvera

- Slabosti RBP
- Savremena proširenja RBP
- Kada relacione baze nisu idealno rešenje
- Nerelacione baze podataka

⚠

U zavisnosti od okolnosti, u planu može da bude nekih promena.

Obaveze studenata

Trenutni plan — ukupno najviše **100 poena**.

20	40	40
Kolokvijum	**Praktični deo ispita**	**Teorijski deo ispita**

⚠ Obaveze studenata biće naknadno fiksirane. Pratite obaveštenja na stranici kursa: `bazepodataka.matf.bg.ac.rs/UvodURelacioneBazePodataka.html`

Obaveze studenata: prisustvo i ponašanje

PRISUSTVO NA ČASOVIMA

Poželjno, ali nije obavezno

- ✓ časovi su za one koji žele da nauče
- ✓ povremeno popisivanje prisutnih
- ✓ nema negativnih poena za nedolaženje

PRISTOJNO PONAŠANJE

Jeste obavezno

Kome se ne radi, ne sme da ometa druge.

Literatura za kurs

C. J. Date	**An Introduction to Database Systems**, VIII izdanje, Addison Wesley, 2004.
G. Pavlović-Lažetić	**Osnove relacionih baza podataka**, drugo izdanje, Matematički fakultet, 1999.
S. Malkov	**Baze podataka** (skripta)
M. Maljković, I. Tanasijević	**Relacione baze podataka** (skripta) `bazepodataka.matf.bg.ac.rs/rbp/2024/rbp_skripta_2024.pdf`

IBM	**Dokumentacija za Db2 11.5**
	`onlajn: ibm.com/docs/en/db2/11.5`
	`PDF: ibm.com/support/pages/node/627743`

Po potrebi će za određene teme biti sugerisana i dodatna literatura.

VAŽNO!

Prezentacije sa predavanja nisu literatura za učenje.

One mogu da služe samo kao podsetnik ili sažeti uvod.

TEMA 1

Uvod u upitni jezik SQL

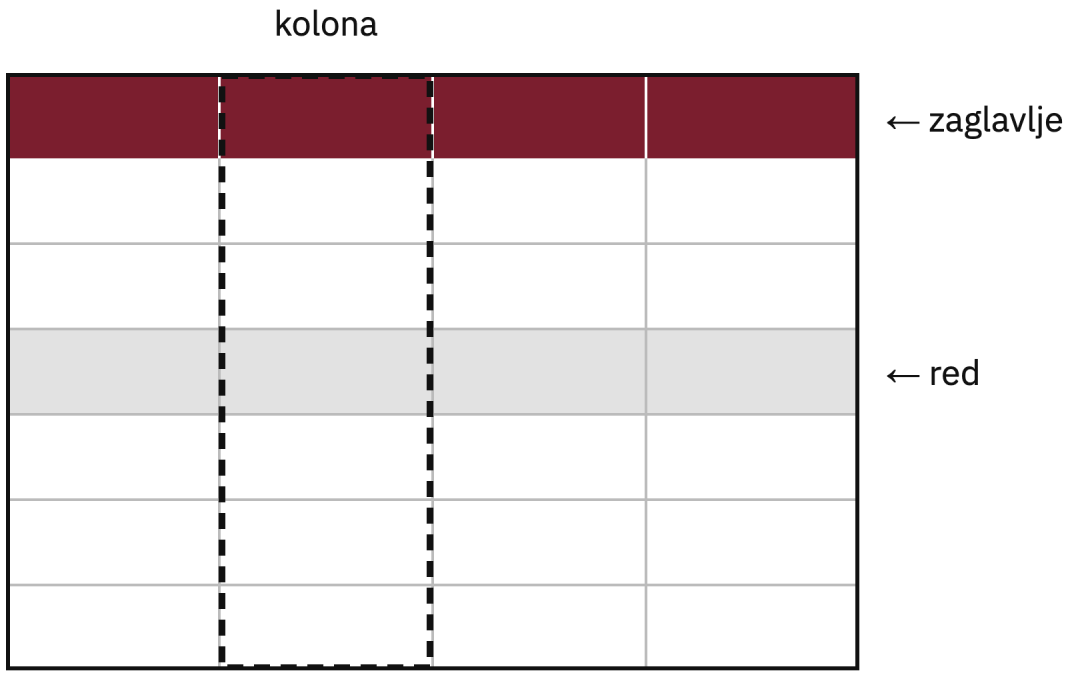
Najpre intuitivno, kroz primere — zatim formalno: podupit, pun upit i upitna naredba.

Najpre intuitivno

</> Rad u terminalu, .sql datoteci ili DataGrip-u

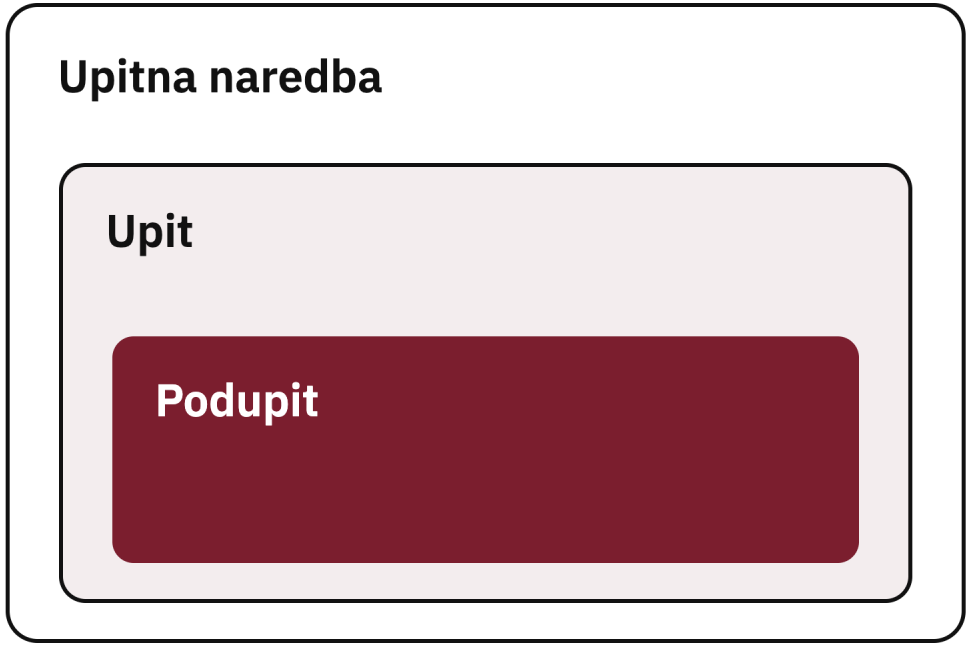
Tabela kao intuitivan koncept: kolone sa imenima i tipovima, redovi sa vrednostima.

Upiti čitaju, izdvajaju i kombinuju redove tabela.



Upitna naredba, upit i podupit

Naredbe za čitanje (izdvajanje) podataka nazivamo **upitnim naredbama**. One čitaju podatke iz postojećih tabela i daju rezultat u obliku privremene tabele — **izračunavaju nove tabele na osnovu postojećih**.



Upitna naredba je naredba SUBP.

Upit je izraz koji čini glavni deo upitne naredbe.

Podupit je manje uopšten upitni izraz, koji se može koristiti kao deo većeg upita ili većeg podupita.

Danas je razlika između upita i podupita mala:

- u podupitima često ne mogu da se koriste skupovne operacije;
- neke stvari, poput uređivanja, nema smisla navoditi u podupitima;
- većina upita, navedena u zagradama, može se koristiti kao podupit.

Podupit

```
(5)   SELECT …
(1)   FROM …
(2)   [WHERE …]
(3)   [GROUP BY …
(4)     [HAVING …]]
(6)   [ORDER BY …]
(7)   [OFFSET …]
(8)   [FETCH …]
      [ISOLATION …]
```

Podupit je komponenta punog upita.

Broj ispred klauzule označava **redosled izračunavanja**.

Klauzule u uglastim zagradama su opcione.

Iako se piše prvi, **SELECT** se izračunava tek peti — posle FROM, WHERE, GROUP BY i HAVING.

Podupit sa ORDER BY, OFFSET ili FETCH

Takav podupit ne može da se koristi na svim mestima:

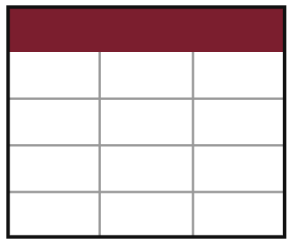
- mora da se navodi **u zagradama**
- klauzula **ORDER BY** može da bude ignorisana

Puna sintaksa je opisana u dokumentaciji.

Primer oblika:

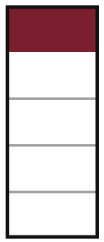
```
SELECT …
FROM ( SELECT …
       FROM …
       ORDER BY …
       FETCH FIRST 5 ROWS ONLY ) AS t
```

Upiti prema složenosti rezultata



Tabelarni upit

Izračunava **tabelu** — najopštija vrsta upita.



Vektorski upit

Izračunava **jednu kolonu**.



Skalarni upit

Izračunava **jednu vrednost**.

Osnovna upitna naredba je **SELECT**. Pre nje upoznajemo naredbu **VALUES** za zapisivanje konstantnih tabela (literala), a kasnije i **WITH** i druge složenije oblike upita.

VALUES — literalni tabelarni izrazi

Poseban oblik podupita su *konstantni* ili *literalni* tabelarni izrazi.

```
VALUES (1, 'jedan'),
       (2, 'dva'),
       (3, 'tri')
```



1	2
1	jedan
2	dva
3	tri

Puna sintaksa je opisana u dokumentaciji.

Proverite razumevanje

1. Koja je razlika između baze podataka i SUBP-a?

2. Kojim redom se izračunavaju klauzule SELECT, FROM i WHERE?

3. Šta izračunava skalarni, a šta vektorski upit?

Proverite razumevanje — odgovori

1. Koja je razlika između baze podataka i SUBP-a?

Baza podataka je kolekcija podataka; SUBP je softverski sistem koji njome upravlja. Jedan SUBP može da radi sa više baza.

2. Kojim redom se izračunavaju klauzule SELECT, FROM i WHERE?

FROM (1) → WHERE (2) → … → SELECT (5). Zato se alijas iz SELECT ne može koristiti u WHERE.

3. Šta izračunava skalarni, a šta vektorski upit?

Skalarni — jednu vrednost; vektorski — jednu kolonu; tabelarni — celu tabelu.

TEMA 1 · ČAS 2

Tipovi, izrazi i spajanje tabela

Ugrađeni tipovi i konverzije · izrazi · spajanje tabela · podupiti

Tipovi i izrazi

Tipovi

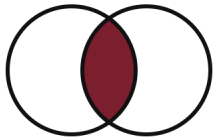
- Ugrađeni tipovi
- Konverzije tipova

Izrazi

- Zapisivanje konstanti
- Posebni registri
- Operatori
- Rad sa vremenskim tipovima
- Rad sa niskama
- Predikati **IN**, **BETWEEN**, **LIKE**
- Izraz **CASE**
- Brojanje redova
- Funkcije **COALESCE** i **NULLIF**

Detalji i primeri — u dokumentu sa primerima za Temu 1.

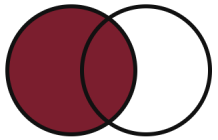
Spajanje tabela



Unutrašnje

`INNER JOIN … ON`

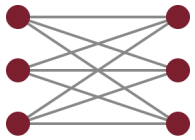
samo parovi redova koji zadovoljavaju uslov



Spoljašnje

`LEFT / RIGHT / FULL OUTER JOIN`

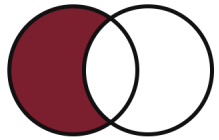
zadržavaju se i redovi bez para



Unakrsno

`CROSS JOIN`

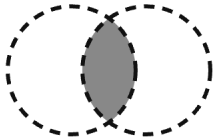
svi parovi redova (dekartov proizvod)



„Spajanje“ razlike

`LEFT JOIN … WHERE … IS NULL`

redovi prve tabele bez para u drugoj



Implicitno

`FROM A, B WHERE …`

uslov spajanja se navodi u WHERE

Podupiti

Vrsta	Rezultat	Gde se koristi
Skalarni	jedna vrednost	u izrazima i poređenjima
Vektorski	jedna kolona	IN, ANY, SOME, ALL
Tabelarni	tabela	EXISTS, FROM

Nekorelisani

Ne zavise od spoljašnjeg upita — dovoljno ih je izračunati jednom.

Korelisani

Pozivaju se na kolone spoljašnjeg upita — konceptualno se izračunavaju za svaki njegov red.

Proverite razumevanje

1. Po čemu se razlikuju unutrašnje i levo spoljašnje spajanje?

2. Sa kojim predikatima se koriste vektorski podupiti?

3. Šta je korelisani podupit?

Proverite razumevanje — odgovori

1. Po čemu se razlikuju unutrašnje i levo spoljašnje spajanje?

Levo spoljašnje spajanje zadržava i redove leve tabele bez para — kolone desne tabele su tada NULL.

2. Sa kojim predikatima se koriste vektorski podupiti?

IN, ANY, SOME i ALL.

3. Šta je korelisani podupit?

Podupit koji koristi kolone spoljašnjeg upita; konceptualno se izračunava za svaki red spoljašnjeg upita.

TEMA 1 · ČAS 3

Grupisanje, skupovne operacije i upiti

Podupiti (nastavak) · kolonske funkcije · skupovne operacije · pun upit · upitna naredba

Kolonske funkcije i grupisanje

`COUNT` `SUM` `AVG` `MEDIAN` `MAX` `MIN` …

Grupisanje redova

`GROUP BY`

kolonske funkcije se izračunavaju za svaku grupu posebno

Restrikcija po grupama

`HAVING`

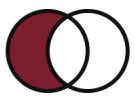
uslov se primenjuje na grupe, a ne na pojedinačne redove

Operatori grupisanja

`GROUPING SETS, ROLLUP, CUBE`

više nivoa grupisanja u jednom upitu

Skupovne operacije

OPERACIJA	BEZ PONAVLJANJA	SA PONAVLJANJEM
 **Unija**	`UNION`	`UNION ALL`
 **Razlika**	`EXCEPT`	`EXCEPT ALL`
 **Presek**	`INTERSECT`	`INTERSECT ALL`

Pun upit

Pun upit može da bude:

- podupit
- izraz **VALUES**
- skupovne operacije nad upitima

```
{ <podupit> | (<pun_upit>) | <izraz_VALUES> }
[ <skupovna_operacija>
    { <podupit> | (<pun_upit>) | <izraz_VALUES> } ]*
[ ORDER BY … ]
[ OFFSET … ]
[ FETCH … ]
[ ISOLATION … ]
```

Upitna naredba

U osnovi pun upit, sa opcionim pomoćnim **imenovanim tabelarnim izrazima** (privremenim pogledima) i podešavanjima režima rada.

Sintaksa

```
[ WITH <imenovan_izraz>
    [, <imenovan_izraz>]* ]
<pun_upit>
[ FOR { READ | FETCH } ONLY ]
[ OPTIMIZE FOR <n> { ROWS | ROW } ]
[ … ]
```

Oblik primera

```
WITH t AS (
    SELECT … FROM …
)
SELECT …
FROM t
FOR READ ONLY
```

Podupit, upit i upitna naredba — razlike

Nivo	Šta dodaje u odnosu na prethodni	Primeri
Podupit	osnovni upitni izraz: izvor podataka, uslovi, grupisanje	SELECT, FROM, WHERE, GROUP BY, HAVING
Upit	skupovne operacije nad podupitima; uređivanje i ograničavanje broja redova rezultata	UNION, EXCEPT, INTERSECT, ORDER BY, OFFSET, FETCH
Upitna naredba	pomoćni imenovani tabelarni izrazi; način korišćenja rezultata (ne menja rezultat)	WITH, FOR READ ONLY, OPTIMIZE FOR



I podupit može da ima ORDER BY, OFFSET i FETCH, ali uz ograničenja: mora biti u zagradama, a ORDER BY može biti zanemaren.

Proverite razumevanje

1. Koja je razlika između WHERE i HAVING?

2. Po čemu se razlikuju UNION i UNION ALL?

3. Čemu služi WITH u upitnoj naredbi?

Proverite razumevanje — odgovori

1. Koja je razlika između WHERE i HAVING?

WHERE izdvaja redove pre grupisanja; HAVING izdvaja grupe nakon grupisanja.

2. Po čemu se razlikuju UNION i UNION ALL?

UNION uklanja ponovljene redove, a UNION ALL ih zadržava.

3. Čemu služi WITH u upitnoj naredbi?

Definiše imenovane pomoćne tabelarne izraze (privremene poglede) koje upit dalje koristi.

TEMA 1 · ČAS 4

Pogledi i menjanje podataka

Rekurzivni upiti · pogledi · INSERT, UPDATE, DELETE, MERGE · CREATE, ALTER, DROP

Rekurzivni upiti

Imenovani tabelarni izraz (WITH) koji se **poziva na samog sebe**.

Početni deo daje prve redove; rekurzivni deo iz već dobijenih računa nove — sve dok ima novih redova. Delovi se spajaju sa UNION ALL.

Primene: hijerarhije (npr. preduslovi predmeta), putevi u grafovima, generisanje nizova.

```
WITH brojevi(n) AS (
    -- početni deo
    SELECT 1 FROM sysibm.sysdummy1
  UNION ALL
    -- rekurzivni deo
    SELECT n + 1 FROM brojevi
    WHERE n < 10
)
SELECT n FROM brojevi
```

Pogledi

Pogled je **imenovani upit** koji se koristi kao tabela — „virtuelna tabela”.

Ne čuva sopstvene podatke: izračunava se iz osnovnih tabela pri svakoj upotrebi.

Koristi se za pojednostavljivanje složenih upita i za ograničavanje pristupa podacima.

```
CREATE [OR REPLACE] VIEW <ime>
    [( <lista_imena_kolona> )]
AS <upit>
…

DROP VIEW <ime>
```

Menjanje podataka: INSERT i DELETE

Dodavanje novih podataka

```
INSERT INTO <naziv_tabele>
    [( <lista_kolona> )]
<upit>
```

<upit> je najčešće izraz VALUES (...), ali može biti i bilo koji drugi upit.

Brisanje postojećih podataka

```
DELETE FROM <naziv_tabele>
[ WHERE <uslov> ]
```

⚠ Bez WHERE brišu se **svi** redovi tabele!

Menjanje podataka: UPDATE

```
UPDATE <naziv_tabele>
SET { <pojedinačne_dodele> | <grupna_dodela> }
[ WHERE <uslov> ]
```

Pojedinačne dodele

`<kolona> = <izraz> [, <kolona> = <izraz>]*`

Grupna dodela

`( <lista_kolona> ) = <upit>`

Menjanje podataka: MERGE

Objedinjuje dodavanje, menjanje i brisanje: redovi izvora se uparuju sa redovima ciljne tabele.

```
MERGE INTO <ciljna_tabela>
USING <upit>
ON <uslov_povezivanja>
{ WHEN [NOT] MATCHED
    [AND <uslov>] THEN <akcija> }+
[ ELSE IGNORE ]
```

<akcija> ima jedan od oblika:

- `UPDATE SET …`
- `DELETE …`
- `INSERT [(…)] VALUES …`

Sa … su označene dodatne opcije.

Menjanje podataka kroz poglede

INSERT, UPDATE i DELETE mogu se izvršiti i nad (ažurljivim) pogledom — tada menjaju osnovnu tabelu.

```
CREATE VIEW <ime> AS <upit>
[ WITH [ LOCAL | CASCADED ] CHECK OPTION ]
```

CHECK OPTION

Sprečava izmene posle kojih red ne bi zadovoljavao uslov pogleda — red ne sme da „nestane” iz pogleda.

CASCADED ili LOCAL

CASCADED (podrazumevano) proverava i uslove pogleda na kojima se pogled zasniva; LOCAL samo one koji i sami imaju CHECK OPTION.

Pravljenje objekata baze podataka

`CREATE <vrsta_objekta> …`

pravi novi objekat

`ALTER <vrsta_objekta> …`

menja postojeći objekat

`DROP <vrsta_objekta> …`

briše objekat

Neke vrste objekata:

`TABLE` `VIEW` `INDEX` `SCHEMA` `SEQUENCE` `…`

Pravljenje tabela

```
CREATE TABLE [ IF NOT EXISTS ] <ime_tabele>
    ( <lista_elemenata> )
[ <opcije> ]
```

Primer

```
CREATE TABLE student (
    indeks INTEGER NOT NULL PRIMARY KEY,
    ime VARCHAR(50) NOT NULL,
    smer_id INTEGER REFERENCES smer
)
```

Element tabele može da bude definicija:

- kolone
- primarnog ključa
- jedinstvenog ključa
- stranog ključa
- drugog uslova integriteta

Menjanje strukture i brisanje tabele

Menjanje strukture

```
ALTER TABLE <ime_tabele>
{ ADD <element>
| ALTER <element>
| DROP <element> }*
[ <opcije> ]
```

```
ALTER TABLE student ADD email VARCHAR(100)
```

Brisanje tabele

```
DROP TABLE <ime_tabele>
```

⚠ Briše i strukturu tabele i sve njene podatke.

Proverite razumevanje

1. Šta se dešava ako se DELETE izvrši bez WHERE?

2. Koja je razlika između DELETE FROM t i DROP TABLE t?

3. Čemu služi WITH CHECK OPTION kod pogleda?

Proverite razumevanje — odgovori

1. Šta se dešava ako se DELETE izvrši bez WHERE?

Brišu se svi redovi tabele; sama tabela ostaje, prazna.

2. Koja je razlika između DELETE FROM t i DROP TABLE t?

DELETE briše redove, a DROP TABLE briše i tabelu — njenu strukturu i sve podatke.

3. Čemu služi WITH CHECK OPTION kod pogleda?

Sprečava izmene kroz pogled posle kojih red više ne bi zadovoljavao uslov pogleda.

Literatura za temu

Dokumentacija za Db2 11.5

`onlajn: ibm.com/docs/en/db2/11.5`

`PDF: ibm.com/support/pages/node/627743`

Primeri i okvirna sintaksa za sva tri časa — u dokumentu „Tema 1: Uvod u SQL — primeri sa objašnjenjima” i u kratkom podsetniku najvažnijih primera.