UNIVERZITET
METROPOLITAN
BEOGRAD

IT250 - BAZE PODATAKA

Arhitektura organizacije podataka-relacioni model baza podataka

Lekcija 02

PRIRUČNIK ZA STUDENTE

IT250 - BAZE PODATAKA

Lekcija 02

ARHITEKTURA ORGANIZACIJE PODATAKA-RELACIONI MODEL BAZA PODATAKA

UVOD

Šta ćemo naučiti u ovoj lekciji?

Većina informacionih sistema danas koristi relacione modele baza podataka. Osnova relacionog modela je matematički koncept poznat kao relacija. Da bismo izbegli složenost apstraktne matematičke teorije, možemo zamisliti relaciju (ponekad nazvanu tabela) kao matricu sastavljenu od redova i kolona. Svaki red u relaciji se naziva torkom. Svaka kolona predstavlja atribut. Relacioni model takođe opisuje precizan skup konstrukcija za manipulaciju podacima, zasnovan na naprednim matematičkim konceptima. Relacija u relacionom modelu je isto što i relacija u matematici, s tom razlikom da su relacije u relacionom modelu podataka vremenski promenljive.

U ovoj lekciji će biti date teorijske, koncepcijske osnove relacionog modela i objašnjena njegova sličnost sa relacijama u matematici. Opisani su osnovni elementi relacionog modela kao što su relacije i ograničenja integriteta relacionog modela: integritet domena, integritet entiteta i referencijalni integritet. Svi opisani koncepti su potkrepljeni primerima.

Deo lekcije se odnosi na prikaz primene relacione algebre u pretraživanju baza podataka. Detaljno, uz primere, su objašnjene operacije relacione algebre: unija, presek, razlika, JOIN itd. kao i mogućnost da se one kombinuju kako bi se dobili kompleksniji upiti za pretraživanje.

Poglavlje 1

Teorijske osnove RM

ŠTA SPADA U TEORIJSKE OSNOVE RELACIONOG MODELA?

Funkcije, Dekartov proizvod skupova, odnosi između skupova

Većina informacionih sistema danas koristi relacione modele baza podataka. U osnovi, postoje dve vrste relacionih modela:

- **klasični relacioni model (RM) i**
- **prošireni relacioni model (RM/T).**

Relacija u relacionom modelu je isto što i relacija u matematici s tom razlikom da su relacije u relacionom modelu podataka vremenski promenljive.

U teorijske, koncepcijske osnove relacionog modela spadaju:

- **Funkcije,**
- **Dekartov proizvod skupova,**
- **Odnosi između skupova**

POJAM FUNKCIJE

Kad god imamo dva skupa E i F među kojima je uspostavljena veza da je svakom x iz E pridružen po jedan element y iz F kažemo da je zadata funkcija f na E sa vrednostima u F

$f : E \mapsto F$, označava da funkcija f elementu $e \in E$ pridružuje element $f(x) \in F$. Skup E je domen ili područje definicije, dok je F kodomen funkcije $f : E \mapsto F$. Podskup $\{f(x) \mid x \in E\}$ skupa F je slika funkcije F.

Svaki učenik ima svoje prezime. Označimo li sa E skup svih učenika nekog razreda, tada i prezimena tih učenika obrazuju potpuno uređen skup; označimo ga sa F. Između elemenata skupa E i elemenata skupa F postoji potpuno određen odnos, jer svakom elementu x koji pripada E, tj. svakom učeniku iz uočenog razreda pripada potpuno određen element y iz skupa F (tj. prezime učenika x). Ovde je svakom elementu x iz E pridružen po jedan element iz F.

Kad god imamo dva neprazna skupa E i F među kojima je uspostavljena veza navedene prirode, tj.svakom x iz E je pridružen po jedan element y iz F, kažemo da je zadata funkcija

f na E sa vrednostima u F. Predlog "na" često zamenjujemo predlogom "sa", pa kažemo da *f* deluje sa E u F. Funkcije označavamo obično malim slovima, npr. *f, g, h*, ...

Definicija funkcije uključuje tri objekta. Skup E (domen), skup F (kodomen) i postupak (propis, radnju, transformaciju) prema kojem je svakom elementu x iz E pridružen jedinstven element f(x) iz F.

Primer 1. Ukoliko stojimo ispred nekog izloga, onda svi predmeti u tom izlogu obrazuju skup E, a njihove cene skup F. Svakom predmetu x iz E pridružen je po jedan element iz F, njegova cena. Tu se radi o funkciji f koja npr. artiklu x iz E pridružuje njegovu cenu f(x) iz F.

ŠTA JE DEKARTOV PROIZVOD?

Dekartov proivod pretstavlja skup svih uređenih parova dva skupa (S' i S'')

U mnogim razmatranjima posmatraju se podskupovi nekog skupa U. Za nas, skup U ima univerzalno značenje i zato se zove univerzalni skup ili univerzum. Jasno je da je univerzalnost skupa U relativna i da varira od problema do problema.

Skup svihpodskupova skupa U se označava sa P(U) i zove**se** partitivni skup**skupa U.** Za skup U je povezan i skup svih funkcija sa U u U. Prema tome, preko postojećih skupova konstruišu se novi skupovi koji su na određen način prateći objekti (skupovi) polaznih skupova.

Neka su A i B dva skupa i neka su a koje pripada A ib koje pripada B zadati elementi. Kažemo da je (a, b) uređen par, ako je element a proglašen prvim, a b drugim u tome paru. Element *a* uređenog para *(a,b)* se zove prva projekcija, prva koordinata, odnosno prva komponenta, a *b* se zove druga projekcija, druga koordinata, odnosno druga komponenta uređenog para *(a, b)*.

Dekartov proizvod skupova predstavlja skup svih uređenih parova dva skupa A i B.

Za proizvoljne neprazne skupove S' i S'' sa $S = S' \times S'' = \{(s', s'') \mid s' \in S', s'' \in S''\}$ označavamo skup svih uređenih parova (s', s'') i zovemo ga Dekartov, direktni ili kombinovani proizvod skupova S' i S'' . Dekartov proizvod se definiše i kada je jedan od skupova S' i S'' prazan. Uzima se $S' \times \varnothing = \varnothing, \varnothing \times S'' = \varnothing, \varnothing \times \varnothing = \varnothing$, za sve skupove S' i S'' .

PRIMER DEKARTOVOG PROIZVODA SKUPOVA

Dekartov proizvod skup pantalona i kaputa

Učenik Marko Marković ima četvoro pantalona: *p – plave, c – crne, b – bele i s – sive pantalone*. Skup njegovih pantalona označimo sa S' = {p, c, b, s} . Isti taj učenik ima tri kaputa: *A – crni, B – beli i P – plavi kaput*. Neka je S'' ={ A, B, P} skup njegovih kaputa. Na koliko se načina on može obući tako da obuče jedan kaput i jedne pantalone? Kombinujmo:

p, A (plave pantalone, crni kaput)

p, B (plave pantalone, beli kaput)

p, C (plave pantalone, plavi kaput) itd.

Tako npr. element (c, B) odgovara mogućnosti da Marko Marković obuče crne pantalone i beli kaput. Kako vidimo ovde ima 12 mogućnosti, a svaka od njih je element Dekartovog proizvoda S' X S'', tj. uređen par.

$$S\prime \times S\prime\prime = \{(a, b) \mid a \in S\prime \wedge b \in S\prime\prime\}$$

PRIMER RELACIJE IZMEĐU SKUPOVA

Uspostavljanje relacije "poznavanja" između dva skupa koji predstavljaju dve različite porodice

Neka je E = {a, b, c, d} društvo od četiri osobe, a F = {e, f, g} drugo društvo (porodica) od tri člana. Između te dve porodice može se uspostaviti odnos "poznavanja", tj. može se dogoditi da pripadnik b porodice E poznaje nijednog, jednog ili više članova porodice F.

Uzmimo ovu mogućnost:

- a poznaje osobe e i f
- b poznaje osobu f
- c poznaje osobe e, f, g
- d ne poznaje ni e, ni f, ni g.

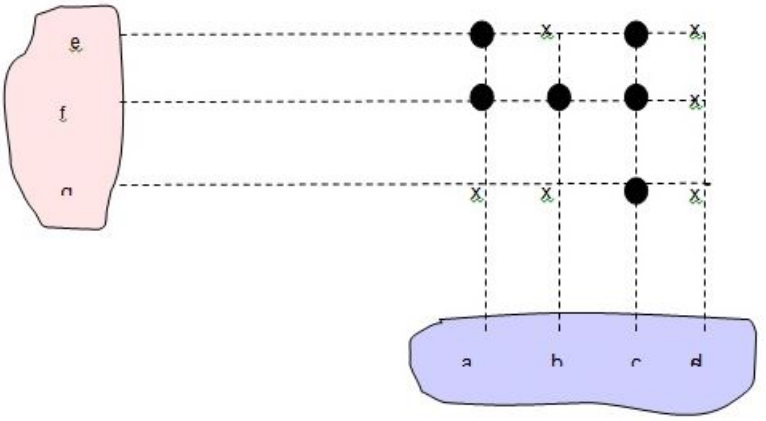
Na ovaj način putem "poznavanja" uočen je, odnosno ustanovljen odnos između skupova E i F koji je prikazan na slici 1 .

Puni kružić je "poznaje" a **x** ne poznaje.
Prirodno je da se posmatra kombinovani proizvod skupova E i F, jer se tu pojavljuju sve mogućnosti. Na slici su prikazani skupovi E i F i njihov proizvod E x F.

Označimo sa **R** skup svih ispunjenih krugova na slici 1.1, tj.

R ={(a, e), (a, f), (b, f), (c, e), (c, f), (c, g)}.

Vidi se da je R podskup direktnog (Dekartovog) proizvoda E x F i to tačno onaj koji sadrži elemente koji su u navedenom odnosu "poznavanja".



Slika 1.1 Skupovi E i F i njihov proizvod E x F [Izvor: NM IT350-2020/2021.]

DEFINICIJA RELACIJE U MATEMATICI

Relacija je svaki podskup Dekartovog proizvoda dva ili više skupa

Neka su E i F skupovi. Svaki podskup Dekartovog proizvoda E x F zove se relacija.

Za element x ∈ E kažemo da je u relaciji R sa y ∈ F, ako i samo ako je (x, y) ∈ R .

Činjenicu (x, y) ∈ R često pišemo u obliku x R y ili y = R (x) i kažemo da x ima *osobinu* da je u relaciji R sa y.

Prva projekcija skupa R zove se područje definicije ili domen relacije R, a druga projekcija skupa R zove se slika relacije R.

Prema definiciji, relacija R iz E u F je podskup od E x F. Time se relacija shvata "geometrijski".

Notaciju relacije lako možemo proširiti i na tri skupa, npr. D_1, D_2 i D_3. Dekartov proizvod ova tri skupa je skup svih uređenih torki (tj. trojki), tako da je prvi element iz D_1, drugi element iz D_2 i treći element iz D_3. Bilo koji podskup Dekartovog proizvoda skupova je relacija. Na primer, pretpostavimo da imamo:
$D_1 = \{1, 3\}$, $D_2 = \{2, 4\}$, $D_3 = \{5, 6\}$
$D_1 \times D_2 \times D_3 = \{(1, 2, 5), (1, 2, 6), (1, 4, 5), (1, 4, 6), (3, 2, 5), (3, 2, 6), (3, 4, 5), (3, 4, 6)\}$.

Bilo koji podskup od $D_1 \times D_2 \times D_3$, je relacija. Npr. R= {(1, 2, 5), (1, 4, 6), (3, 2, 5), (3, 2, 6), (3, 4, 6)}.

Poglavlje 2

Svojstva relacija u relacionom modelu

DEFINICIJA RELACIJE U RELACIONOM MODELU

Polazeći od ovih pretpostavki, relacija u relacionom modelu se može definisati kao skup parova atributa relacije i imena domena.

Pojam i definicija relacije iz matematike iskorišćeni su za definiciju i razvoj relacionog modela (baze) podataka.

Polazna osnova za definisanje relacije u relacionom modelu podataka je da je svaka relacija opisana skupom atributa Ai. Svaki atribut Ai opisuje neku odabranu osobinu klase entiteta koji relacija opisuje. Svakom atributu je pridružen domen, koji predstavlja specifikaciju skupa mogućih vrednosti iz kojeg atribut uzima vrednosti. Uobičajeni način specifikacije domena je preko tipa podataka. Korisno je da svaki domen ima naziv. Domen se može definisati i kao podskup vrednosti nekog drugog domena ili nabrajanjem dopuštenih domenskih vrednosti.

Polazeći od ovih pretpostavki, relacija relacionog modela se može definisati kao skup parova atributa relacije i imena domena.

DEFINICIJA RELACIJE: Neka su A_1, A_2, . . . , A_n atributi sa domenima D_1, D_2, . . . , D_n. Tada je $\{A_1:D_1, A_2:D_2, . . . , A_n:D_n\}$ relacija.

Relacija R je skup preslikavanja imena atributa u odgovarajuće domene primenom odgovarajuće funkcije. Drugim rečima, relacija R je skup n-torki $(A_1:d_1, A_2:d_2, . . . , A_n:d_n)$ tako da je $d_1 \in D_1$, $d_2 \in D_2$, . . . , $d_n \in D_n$.

Svaki element n-torke se sastoji od atributa i vrednosti za taj atribut.

UOBIČAJENI ZAPIS RELACIJE U OBLIKU TABELE

U relacionom modelu, relacija se uobičajeno opisuje preko tabele

U relacionom modelu, relacija se uobičajeno opisuje preko tabele. Kada relaciju zapisujemo kao tabelu, imena atributa označavamo kao zaglavlja kolona a torke tabele zapisujemo u formi $(d_1, d_2, . . . , d_n)$,gde je svaka vrednost uzeta iz odgovarajućeg domena.

Na ovaj način, u relacionom modelu, **relaciju tretiramo kao podskup Dekartovog proizvoda domena atributa. Tabela je jednostavno fizička prezentacija takve relacije.**

***Na primer**: relacija KADAR je na slici 1. predstavljena kao tabela. U ovom primeru, relacija KADAR ima atribute #radnika, prezime, ime, starost od kojih svaki ima odgovarajući domen. Relacija KADAR je podskup Dekartovog proizvoda domena ili bilo koji skup torki od četiri elementa tako da je prvi element iz domena #radnika, drugi iz domena prezime, treći iz domena ime itd. Torke ove relacije su:*

1. *{(#radnika: 234, prezime: MARKOVIĆ, ime: MARKO, starost: 25)}*
2. *{(#radnika: 258, prezime: PETROVIĆ, ime: IVAN, starost: 34)}*
3. *{(#radnika : 108, prezime: JANKOVIĆ, ime: JANKO, starost: 56)}*

Sve relacije su tabele, ali sve tabele nisu relacije . Karakteristike relacija koje čine osnovu relacionog modela su:

- **Redovi sadrže podatke o entitetu (svaki red predstavlja jednu torku relacije)**
- **Kolone sadrže podatke o atributima entiteta**
- **Svi podaci u kolonama su iste vrste**
- **Svaka kolona ima jedinstveno ime**
- **Ćelije kolona sadrže samo po jednu vrednost**
- **Redosled kolona nije bitan**
- **Redosled redova nije bitan**
- **Dva reda ne mogu biti identična**

#RADNIKA	PREZIME	IME	STAROST
234	Marković	Marko	25
258	Petrović	Petar	34
108	Janković	Janko	56

Slika 2.1 Relacija KADAR zapisana u obliku tabele. Izvor: [Autor]

ŠEMA RELACIJE

Uobičajeno je da se šema relacije obeležava navođenjem njenog naziva, skupa atributa i evenatualno skup ograničenja.

Šema relacije se sastoji od:

1. **naziva šeme relacije,**
2. **skupa atributa relacije i**
3. **skupa ograničenja integriteta relacije**

Šema relacije opisuje semantiku relacije na prirodnom jeziku. Skup ograničenja integriteta opisuje odnose između elemenata domena atributa relacije. Praktično, ova komponenta ograničava koje će se torke pojaviti u instanci ove relacije.

Uobičajeno je da se šema relacije obeležava tako što se navede njen naziv, a nakon toga skup atributa (s tim da se uz svaki atribut može navesti domen i eventualno ograničenje ključa), i evenatualno skup ograničenja.

Na primer:

1. *činjenicu da je R šema relacije definisana nad atributima A_1, A_2,..., A_n, označavamo sa R(A_1, A_2,..., A_n).*
2. *Relacija sa imenom R i listom atributa A_1, A_2,..., A_n, primarnim ključem A_1: R($\underline{A_1}$, A_2,..., A_n)*
3. *Relacija sa imenom R i listom atributa A_1, A_2,..., A_n, listom domena atributa D1, D2,..., Dn i ključem A_1*
4. *R($\underline{A_1}$:D_1, A_2:D_2,..., A_n:D_n)*

SVOJSTVA RELACIJA

Mogu se definisati dva svojstva: (1) Poredak n-torki relacije može biti proizvoljan (2) U relaciji, bilo koje dve n-torke nisu jednake odakle proizilazi definicija ključa relacije

Relacija se karakteriše sledećim svojstvima:

1. **Poredak n-torki relacije (redova tabele) može biti proizvoljan u bazi podataka. Ovo svojstvo ne zahteva specifikaciju ograničenja**
2. U relaciji, bilo koje dve n-torke nisu jednake, odnosno bilo koja dva reda tabele nisu jednaka. Iz ovog ograničenja proizilazi definicija ključa relacije.

Ključ relacije u relacionom modelu podataka je onaj podskup atributa čije vrednosti jedinstveno identifikuju n-torke relacije. Moguće je da relacija ima više različitih ključeva. Svi ključevi relacije čine kandidate za ključ. Ako neki skup atributa u posmatranoj relaciji nije ključ, ali je ključ u nekoj drugoj relaciji u modelu, onda se on naziva strani ključ.

Da bi neki skup atributa relacije bio kandidat za ključ, on mora zadovoljiti nezavisno od vremena, dva uslova:

1. Uslov jedinstvenosti: Vrednost ključa svake n-torke relacije jedinstveno određuje n-torku, tj. **ne postoje dva reda u tabeli ta kva da imaju sve iste vrednosti svih atributa koji čine ključ.**
2. Uslov neredudantnosti: Ne postoji ni jedan atribut kao deo ključa koji se može izostaviti iz ključa a da se pri tome uslov jedinstvenosti ne gubi, tj. **ključ je unija minimalnog broja atributa**.

Od niza kandidata za ključ bira se jedan, koji se zove primarni ključ. Iz definicije primarnog ključa proizilazi ograničenje integriteta entiteta.

Poglavlje 3

Ograničenja integriteta relacionog modela

ŠTA SU OGRANIČENJA INTEGRITETA?

Tu spadaju: integritet domena, integritet entiteta i referncijalni integritet

Ograničenja integriteta relacionog modela omogućuju besprekoran prelazak relacionog modela iz jednog stanja u drugo. To su:

1. Integritet domena. Određuje da svi atributi (kolone) u relaciji moraju biti deklarisani za određeni domen. Domen je skup vrednosti istog tipa pa se mogu tretirati kao bazne vrednosti od kojih se izvlače stvarne vrednosti koje se pojavljuju u kolonama tabele.
2. Integritet entiteta. Neka je atribut A deo primarnog ključa relacije R. Tada atribut A ne sme poprimiti null vrednost. To znači da ne postoji n-torka u relaciji R takva da je vrednost atributa A te n-torke jednaka null vrednosti.
3. Referencijalni integritet. Ograničenje referencijalnog integriteta je specificirano između dve tabele i koristi se za održavanje konzistentnosti između redova te dve tabele

3.1 Integritet domena i integritet entiteta

INTEGRITET DOMENA

Znači definisanje važećeg skupa vrednosti za atribut

Za svaki atribut treba definisati:

- tip podataka,
- dužinu ili veličinu
- da li je dozvoljena nulta vrednost
- da li je vrednost jedinstvena ili ne

Takođe, može se definisati podrazumevana vrednost, opseg (vrednosti između) i / ili

specifične vrednosti za atribut. Ove definicije obezbeđuju da određeni atribut ima pravu i ispravnu vrednost u bazi podataka.

Primer:

```
CREATE TABLE Zaposleni (
    ID INT PRIMARY KEY,
    Ime VARCHAR(50) NOT NULL,
    Prezime VARCHAR(50) NOT NULL,
    Plata DECIMAL(10, 2) NOT NULL,
    Pogodnost CHAR(1),
    CONSTRAINT CHK_Pogodnost CHECK (Pogodnost IN ('D', 'N'))
);
```

Prethodni primer koristi ključne reči NULL i NOT NULL navedene u istoj liniji u kojoj se navode imena kolona, nakon vrste podataka. NOT NULL je ograničenje kojim se onemogućava ulazak vrednosti NULL u kolonu; drugim rečima, potrebno je obavezno uneti podatke u NOT NULL kolonu za svaki red u tabeli. NULL je obično podrazumevana vrednost ako NOT NULL nije navedena, što dozvoljava unošenje NULL vrednosti u kolonu.

CHK ograničenja se mogu koristiti da bi se proverila valjanost podataka unesenih u određene kolone tabele. Provera ograničenja se koriste da bi se obezbedile izmene u bazi podataka, koje se mogu unositi u kolone, bilo u samu bazu podataka ili kroz aplikaciju.

ŠTA JE INTEGRITET ENTITETA?

Ograničenjem integriteta entiteta se uvodi ograničenje da primarni ključevi ne mogu biti null

Ograničenjem integriteta entiteta se uvodi ograničenje da vrednosti atributa koji ulaze u primarne ključeve ne mogu imati null vrednosti tj. da za njih mora postojati odgovarajuća vrednost.

To je zato što se vrednost primarnog ključa koristi za identifikaciju pojedinačnih n-torki u tabeli. **Ako bi postajala null vrednosti za primarne ključeve, to bi značilo da ne možemo identifikovati te redove.**

Ostale vrednosti atributa mogu imati null vrednosti.

Primer:

```
CREATE TABLE Studenti (
    ID INT PRIMARY KEY,
    Ime VARCHAR(50) NOT NULL,
    Prezime VARCHAR(50) NOT NULL,
    Email VARCHAR(100),
    GodinaRodjenja INT,
    CONSTRAINT CHK_GodinaRodjenja CHECK (GodinaRodjenja >= 1900 AND GodinaRodjenja
<= YEAR(GETDATE()))
);
```

U ovom primeru tabela "Studenti" ima kolone "ID" (celobrojni primarni ključ), "Ime" (tekstualni tip koji ne sme biti NULL), "Prezime" (tekstualni tip koji ne sme biti NULL), "Email" (tekstualni tip koji može biti NULL) i "GodinaRodjenja" (celobrojni tip).

Dodali smo ograničenje (CHECK constraint) na kolonu "GodinaRodjenja" koje definiše da vrednost mora biti veća od ili jednaka 1900. Takođe, koristimo funkciju YEAR(GETDATE()) da bismo dobili trenutnu godinu i proverili da li je vrednost "GodinaRodjenja" manja ili jednaka trenutnoj godini.

Ograničenje integriteta entiteta pomaže u održavanju konzistentnosti podataka tako da osigurava da samo validni podaci budu uneti u odgovarajuće kolone. U ovom primeru, ograničenje proverava da li godina rođenja padne unutar očekivanih granica, sprečavajući unos netačnih ili nerealnih vrednosti.

3.2 Referencijalni integritet

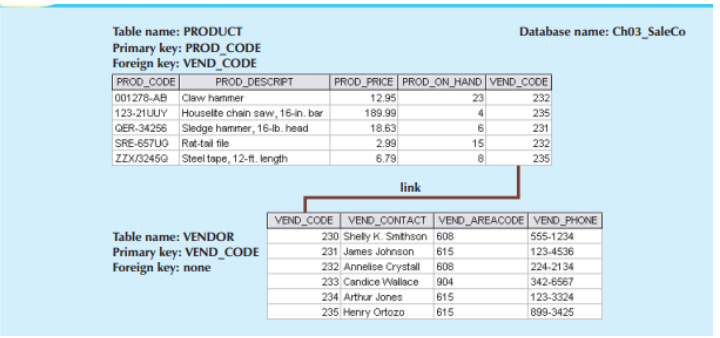
ŠTA JE REFERENCIJALNI INTEGRITET?

Specificira odnos između dve tabele i koristi se za održavanje konzistentnosti između redova tih tabela

Referencijalni integritet specificira odnos između dve tabele i koristi se za održavanje konzistentnosti između redova tih tabela.
Neka postoji relacija S sa prostim primarnim ključem (od jednog atributa) A definisanim nad domenom D. Neka postoji relacija R sa atributom A definisanim nad domenom D. Tada vrednost atributa A bilo koje n-torke u relaciji R može biti ili null vrednost ili vrednost "k", pri čemu vrednost "k" postoji u domenu D i postoji n-torka u relaciji S takva da je vrednost ključnog atributa A te relacije jednaka vrednosti "k". Relacije R i S ne moraju biti različite. Atribut A relacije R, koji postoji kao primarni ključ u relaciji S, naziva se spoljni ključ ili strani ključ .

Pravila koja se moraju poštovati prilikom referencijalnog integriteta su :

1. **Ne možete da izbrišete zapis iz primarne tabele (relacije S) ako za nju postoje redovi u povezanoj tabeli (relacija R).**
2. **Ne možete promeniti vrednost primarnog ključa u primarnoj tabeli (relaciji S) ako za tu torku postoje redovi u povezanoj relaciji R.**
3. **Ne možete uneti vrednost u polje spoljnog ključa povezane tabele koje ne postoji u primarnom ključu primarne tabele.**
4. **Međutim, u spoljni ključ možete uneti vrednost Null, što bi značilo da su zapisi u primarnoj i povezanoj tabeli nepovezani.**



Slika 3.1.1 Primer jednostavne relacione baze podataka. Izvor: [1]

Važno je napomenuti da vrednost VEND_CODE u jednoj tabeli može da ukazuje na odgovarajuću vrednost u drugoj tabeli. Na primer, vrednost VEND_CODE 235 u tabeli PRODUCT ukazuje na dobavljača Henry Ortozo u tabeli VENDOR. Iz toga možemo zaključiti da proizvod "Houselite motor na lanac, dužina 16 inča" isporučuje Henry Ortozo i da se može kontaktirati pozivom na broj 615-899-3425. Ista veza može se uspostaviti i za proizvod "Čelična traka, dužina 12 stopa" u tabeli PRODUCT.

OGRANIČENJE INTEGRITETA STRANIH KLJUČEVA

Postoje dva ograničenja: (1) Kaskadno ažuriranje primarnog i sekundarnog ključa (2) Kaskadno brisanje primarnog i sekundarnog ključa

Postoje dva ograničenja integriteta stranog ključa:

1. **Kaskadno ažuriranje primarnog i sekundarnog ključa**
2. **Kaskadno brisanje primarnog i sekundarnog ključa**

Kaskadno ažuriranje primarnog i sekundarnog ključa

Svaki put kada u primarnoj tabeli promenite primarni ključ reda, vrednosti stranih ključeva se ažuriraju u odgovarajućim redovima u povezanoj tabeli. Ovo ograničenje prevazilazi pravilo 2 u ograničenjima referencijalnog integriteta.

```
CREATE TABLE Autobusi (
    AutobusID INT PRIMARY KEY,
    Model VARCHAR(50),
    GodinaProizvodnje INT
);

CREATE TABLE Rute (
    RutaID INT PRIMARY KEY,
    Naziv VARCHAR(50),
    AutobusID INT,
    FOREIGN KEY (AutobusID) REFERENCES Autobusi(AutobusID) ON UPDATE CASCADE
);
```

Na primer, ako promenimo vrednost primarnog ključa "AutobusID" za autobus sa ID-jem 5 na 10, sve rute koje su povezane sa tim autobusom će automatski ažurirati svoje vrednosti spoljnog ključa "AutobusID" iz 5 u 10.

Osim toga, ako ažuriramo primarni ključ "RutaID" u tabeli "Rute", vrednost spoljnog ključa "RutaID" u drugim povezanim tabelama takođe će biti ažurirana kaskadno, ukoliko je definisano odgovarajuće pravilo kaskadnog ažuriranja.

Kaskadno brisanje primarnog i sekundarnog ključa

Kad god izbrišete red u primarnoj tabeli, odgovarajući redovi se automatski brišu u odgovarajućoj povezanoj tabeli. Ovo ograničenje prevazilazi pravilo 1 u ograničenjima referentnog integriteta.

```
CREATE TABLE Odeljenja (
    OdeljenjeID INT PRIMARY KEY,
    Naziv VARCHAR(50)
);

CREATE TABLE Zaposleni (
    ZaposleniID INT PRIMARY KEY,
    Ime VARCHAR(50),
    Prezime VARCHAR(50),
    OdeljenjeID INT,
    FOREIGN KEY (OdeljenjeID) REFERENCES Odeljenja(OdeljenjeID) ON DELETE CASCADE
);
```

Poglavlje 4

Primeri:Svojstva i integritet relacionog modela

PRIMERI: RELACIJE, ATRIBUTI, N-TORKE

Elementi su opisani kroz sledeće primere:

Relacioni model pruža jedinstven način za predstavljanje podataka kao dvodimenzionalnih tabela koje se zovu relacije. Kao primer na slici 4.1. je data relacija **Knjige**. Kolone relacije se nazivaju atributi. Na slici 4.1. atributi relacije **Knjige** su: *knjiga_id , naslov, autor,* i *godina_izdanja.* Nazivi atributa se pojavljuju na vrhu kolona. Ime relacije i skup atributa relacije se naziva šema relacije ili relacija, a prikazuje se kao ime relacije praćeno listom atributa koji se stavljaju u zagradi. Šema za relaciju **Knjige** je:

Knjige(knjiga_id , naslov, autor, godina_izdanja)

U relacionom modelu, baza podataka se sastoji od više relacija. Skup šema tih relacija baze podataka se naziva relaciona šema baze podataka. Redovi relacija se nazivaju n-torke ili torke. Torka ima po jednu komponentu za svaki atribut relacije.

Na primer, prva od tri torke na slici 4.1. ima četiri komponente: 1, Ubiti pticu rugalicu, Harper Li, i 1960 za atribute: knjiga_id , naslov, autor, i godina_izdanja, respektivno.

.

Relacioni model zahteva da svaka komponenta svake torke bude atomična, što znači da mora imati elementarni tip kao što su integer ili string. Vrednost komponente ne može imati strukturu sloga, liste, polja itd.

Takođe se pretpostavlja da je svakom atributu dodeljen domen odnosno određen tip. Komponenta bilo koje torke relacije mora imati vrednost koja pripada tom domenu.

Na primer, torka relacije Knjige na slici 4.1. mora imati prvu komponentu integer, drugu i treću komponentu string, četvrtu komponentu integer.

Sada se šema za relaciju Knjige može prikazati kao:

Knjige (knjiga_id :integer, naslov:istring, autor:istring, godina_izdanja:integer)

knjiga_id	naslov	autor	godina_izdanja
1	Ubiti pticu rugalicu	Harper Li	1960
2	1984	Džordž Orvel	1949
3	Gordost i predrasude	Džejn Ostin	1813

Slika 4.1 Tabela relacije Knjige. Izvor: [Autor]

PRIMERI: KLJUČEVI

Ključ je kombinacija jedne ili više kolona koje se koriste za identifikaciju pojedinačnih redova u tabeli. Postoje kandidat ključ, primarni ključ, surogat ključ

Ključ je kombinacija jedne ili više kolona koje se koriste za identifikaciju pojedinačnih redova u tabeli. Ključevi koji sadrže dve ili više kolona nazivaju se složeni ključevi.

Kandidat ključ je determinanta koja određuje sve druge kolone u relaciji.

Na primer: Relacija PROIZVOD

PROIZVOD (SKU, SKU_OPIS; ODELJENJE; KUPAC)

ima dva kandidata za ključ: SKU i SKU_OPIS. Kupac je determinanta ali on nije kandidat za ključ jer ne određuje sve kolone u relaciji već samo ODELJENJE.

Kandidati ključevi jedinstveno identifikuju redove u relaciji. Za datu vrednost kandidat ključa može se naći jedan i samo jedan red u tabeli koji ima tu vrednost.

Npr. za datu vrednost SKU koja iznosi 100100 se može naći samo jedan red sa podacima o proizvodu.

Jedan od kandidata ključeva se može izabrati za primarni ključ. Tabela može imati samo jedan primarni ključ.

Primarni ključ može imati jednu kolonu ili može biti složen. Primarni ključ ne sme imati null vrednost. *Npr.*

PROIZVOD (SKU, SKU_OPIS; Odeljenje, Kupac)
Surogat ključ je veštačka kolona koja se dodaje tabeli da bi služila kao primarni ključ. DBMS dodeljuje surogat ključu jedinstvenu vrednost prilikom kreiranja reda tabele i ta vrednost se nikada ne menja. Surogat ključ se koristi kada je primarni ključ isuviše veliki ili kada se ne može definisati.
Npr. STANARINA (Ulica, Grad, Država, PoštanskiBroj, Iznos)

Primarni ključ ove tabele je skup atributa {Ulica, Grad, Država, PoštanskiBroj } . *Da bi se postigle dobre performanse, primarni ključ treba da bude kratak i numerički, što se ne može*

reći za ovako definisan primarni ključ. Struktura tabele u koju je dodat surogat ključ bi izgledala:

STANARINA (StanarinaID, Ulica, Grad, Država, PoštanskiBroj, Iznos)

PRIMERI: STRANI (SPOLJNI) KLJUČ

Strani ključ je kolona ili skup kolona koji su primarni ključ neke tabele koja nije tabela u kojoj se one pojavljuju

Strani ključ je kolona ili skup kolona koje su primarni ključ neke tabele koja nije tabela u kojoj se one pojavljuju. **To je ključ strane tabele u odnosu na onu u kojoj se ključ pojavljuje.** Na primer:

ODELJENJE (OdeljenjeIme, IznosBudžeta, MenadžerIme)

ZAPOSLEN (ZaposlenBroj, ZaposlenIme, *OdeljenjeIme*)

U ovom tekstu, strani ključ je označen iskošenim (italic) i podvučenim tekstom.

Strani ključevi predstavljaju relacije između redova tabela.

U ovom primeru, u stranom ključu OdeljenjeIme koji se nalazi u tabeli ZAPOSLEN se čuva relacija između zaposlenog i njegovog odeljenja. U mnogim slučajevima moramo biti sigurni da se vrednost stranog ključa podudara sa vrednošću primarnog ključa.

Na primer: vrednosti ODELJENJE.OdeljenjeIme se moraju podudarati sa vrednostima ZAPOSLEN.OdeljenjeIme.

Da bi se to obezbedilo treba kreirati ograničenje referencijalnog integriteta kojim se vrednosti stranih ključeva ograničavaju. U ovom slučaju, moramo kreirati ograničenje:

ZAPOSLEN.OdeljenjeIme mora postajati u ODELJENJE.OdeljenjeIme.

DEKLARISANJE KLJUČEVA PRILIKOM KREIRANJA RELACIJA

Postoje dva načina za deklarisanje ključeva...

Postoje dva načina da se atribut ili skup atributa proglasi ključem u okviru CREATE TABLE naredbi koje se koriste za kreiranje relacija:

1. **Možemo jedan od atributa proglasiti ključem kada taj atribut pripada šemi relacije**
2. **Možemo listi elemenata deklarisanih u šemi (koji su dosad bili atributi) dodati deklaraciju koja će reći da će određeni atribut ili skup atributa formirati ključ**

Ukoliko se ključ sastoji od više atributa, metoda koja se koristi je metoda definisana pod 2. Ukoliko je ključ samo jedan atribut, možemo koristiti bilo koji od navedenih načina.

Postoje dve deklaracije koje mogu da se koriste da označe da je neki atribut ključ:

- PRIMARY KEY, ili
- UNIQUE.

Efekat proglašenja skupa atributa S da bude ključ za relaciju R koristeći PRIMARY KEY ili UNIQUE je sledeći:

Dve torke u R ne mogu da se slažu po svim atributima iz skupa S, osim ako je neki od njih NULL

Svaki pokušaj da se doda ili izmeni torka koja ne poštuje ovo pravilo, može dovesti do toga da DBMS odbije zahtev koji bi izazvao narušavanje postavljenog integriteta. Ako se koristi PRIMARY KEY tada atribut iz S ne sme da bude NULL. Kao što je navedeno, svaki zahtev koji je pokušaj da se prekrši pravilo će biti odbijen od strane sistema. NULL vrednost je dozvoljena ukoliko je S UNIQUE.

PRIMERI: DEKLARISANJE KLJUČEVA

Primeri deklarisanja ključeva navedenim načinima

PRIMER 1: Primeri deklarisanja ključeva korišćenjem PRIMARY KEY i UNIQUE

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Email VARCHAR(100)
);
```

U ovom primeru, polje "StudentID" je deklarisano kao primarni ključ koristeći ključnu reč PRIMARY KEY. To znači da će vrednosti u polju "StudentID" biti jedinstvene i neće biti dozvoljenih duplikata. Ovo obezbeđuje da svaki student ima jedinstvenu identifikaciju.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    Email VARCHAR(100) UNIQUE
);
```

U ovom primeru, polje "Email" je deklarisano kao jedinstveni ključ koristeći ključnu reč UNIQUE. To znači da vrednosti u polju "Email" moraju biti jedinstvene i neće biti dozvoljenih duplikata. Svaki zaposleni mora imati jedinstvenu email adresu.

Primarni ključ definiše jedinstveni identifikator za red u tabeli, dok jedinstveni ključ (UNIQUE) garantuje jedinstvenost vrednosti u određenoj koloni, ali ne mora biti primarni ključ. Oba ključa imaju važnu ulogu u održavanju integriteta podataka i postavljanju restrikcija na vrednosti polja u tabeli.

DEKLARISANJE KLJUČA KOMBINACIJOM DVA ATRIBUTA

Ilustrativni primer

```
CREATE TABLE Orders (
    OrderID INT,
    CustomerID INT,
    OrderDate DATE,
    PRIMARY KEY (OrderID, CustomerID)
);
```

U ovom primeru, primarni ključ je deklarisan kao kombinacija dva atributa - "OrderID" i "CustomerID". Ova kombinacija obezbeđuje jedinstvenu identifikaciju za svaki red u tabeli "Orders". Vrednosti u kombinaciji ova dva atributa će biti jedinstvene, ali same vrednosti pojedinačnih atributa ne moraju biti jedinstvene.

Na primer, možemo imati više narudžbina (OrderID) od istog kupca (CustomerID), ali kombinacija OrderID i CustomerID će biti jedinstvena za svaku narudžbinu.

Deklarisanje ključa koristeći kombinaciju dva ili više atributa omogućava preciznije identifikovanje redova u tabeli i obezbeđuje jedinstvenost u okviru te kombinacije atributa.

PRIMER: ŠEMA BAZE PODATAKA SA TEMOM FILM

Opis baze podataka sa temom filma

Baza podataka sa temom filma omogućava čuvanje informacija o različitim filmovima, glumcima i njihovim ulogama. Baza sadrži podatke o naslovu filma, žanru, godini izdanja i oceni filma. Takođe, čuva informacije o glumcima, uključujući njihovo ime, prezime, datum rođenja i zemlju porekla. Odnos između filmova i glumaca se uspostavlja kroz povezivanje njihovih uloga, gde se definiše koja uloga pripada određenom filmu i određenom glumcu.

Baza podataka omogućava pregled informacija o filmovima, uključujući njihove karakteristike kao što su naslov, žanr, godina izdanja i ocena. Takođe, omogućava pregled informacija o glumcima, uključujući njihove biografske podatke poput imena, prezimena, datuma rođenja i zemlje porekla. Kroz tabelu uloga, moguće je videti koje uloge su povezane sa određenim filmom i glumcem.

Ova baza podataka olakšava organizaciju i upravljanje informacijama o filmovima, glumcima i njihovim ulogama, pružajući korisne informacije o filmovima i njihovim glumcima.

Tabela "Filmovi" (Movies):

film_id	naslov	zanr	godina_izdanja	ocena
1	Forrest Gump	Drama	1994	8.8
2	The Shawshank Redemption	Drama	1994	9.3
3	The Godfather	Crime	1972	9.2

Slika 4.2 Tabela Filmovi. Izvor: [Autor]

Tabela "Glumci" (Actors):

glumac_id	ime	prezime	datum_rodjenja	zemlja
1	Tom	Hanks	9. jul 1956	SAD
2	Morgan	Freeman	1. jun 1937	SAD
3	Marlon	Brando	3. april 1924	SAD

Slika 4.3 Tabela Glumci. Izvor: [Autor]

Tabela "Uloge" (Roles):

uloga_id	film_id	glumac_id	uloga
1	1	1	Forrest Gump
2	2	2	Ellis Boyd "Red" Redding
3	3	3	Don Vito Corleone

Slika 4.4 Tabela Uloge. Izvor: [Autor]

DOPUNA PRIMERA: ŠEMA BAZE PODATAKA SA TEMOM FILM

Opis tabela Filmovi, Glumci i Uloge

U ovom primeru imamo tri tabele: "Filmovi" (Movies), "Glumci" (Actors) i "Uloge" (Roles). Tabela "Filmovi" čuva informacije o filmovima sa kolonama kao što su film_id, naslov, žanr, godina izdanja i ocena. Tabela "Glumci" čuva informacije o glumcima sa kolonama kao što su glumac_id, ime, prezime, datum rođenja i zemlja. Tabela "Uloge" povezuje filmove i glumce sa kolonama uloga_id, film_id, glumac_id i uloga.

Ovaj primer prikazuje odnose između filmova, glumaca i njihovih uloga. Tabele su međusobno povezane putem spoljnih ključeva kako bi se omogućila veza između podataka.

VIDEO

Relational Data Model : Relations, Degree, Arity, Domain, Constraints, Referential Integrity

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 5

Relaciona algebra

OPERACIJE RELACIONE ALGEBRE

Postoji četiri klase operacija

Operacije tradicionalne relacione algebre se mogu razvrstati u četiri klase:

1. Podrazumevani set operacija - unija i razlika
2. Operacije za izveštavanje koje se baziraju na uklanjaju delova relacija kod kreiranja rezultata: presek eliminiše sve što nije zajedničko skupovima, selekcija eliminiše neke redove koji ne ispunjavaju tražene uslove kod izveštavanja, i projekcija eliminiše neke kolone.
3. Operacije koje kombinuju zapise dve relacije, uključujući Dekartov proizvod i razne vrste join operacija, koje selektivno uparuju zapise iz dve relacije.
4. Operacije preimenovanja koje ne utiču na zapise iz relacija, ali menjaju šemu relacije tako što menjaju imena atributa ili ime samog odnosa.

PODRAZUMEVANI SET OPERACIJA

Operacije unija i razlike nad relacijama

Dve najčešće operacije iz podrazumevanog seta operacija nad skupovima su **unija i razlika**. Ako posmatramo proizvoljne skupove R i S, pomenute operacije mogu biti definisane na sledeći način:

- R U S - unija skupova R i S predstavlja skup elemenata koji pripadaju skupu R, skupu S ili i skupu R i skupu S. Ovakav element se u skupu unije pojavljuje samo jednom čak i u situaciji kada pripada i skupu S i skupu R pre unije.
- R - S - razlika skupova R i S predstavlja skup elemenata koji pripadaju skupu R, a ne pripadaju skupu S. Imajte na umu da R - S nije ekvivalent razlici S - R jer se u tom slučaju u skupu nalaze elementi koji pripadaju skupu S, ali ne pripadaju skupu R.

OPERACIJE AŽURIRANJA REALIZOVANE KORIŠĆENJEM PODRAZUMEVANOG SETA OPERACIJA

Osnovne operacije ažuriranja treba da omoguće: UPIS n-torki u relaciju, BRISANJE n-torki iz relacije, MENJANJE n-torki u relaciji

Kod operacija ažuriranja se eksplicitno zahteva da su sve relacije koje ulaze u operaciju definisane nad istim domenima. Ovo pravilo treba još čvršće zahtevati, da su odgovarajući atributi relacija iste semantike. Očigledno je da domen "skup svih mogućih datuma" u različitim relacijama može imati različit smisao.

Osnovne operacije ažuriranja treba da omoguće:

- UPIS n-torki u relaciju
- BRISANJE n-torki iz relacije
- MENJANJE n-torki u relaciji

UNIJA je operacija nad dve relacije koja omogućava dodavanje različitih n-torki jedne relacije drugoj relaciji, te omogućava upis u relaciju. Oznaka za operator je "U".

RAZLIKA je operacija nad dve relacije koja omogućava izbacivanje istih n-torki u obe relacije. Ova operacija nije komutativna, te je poredak operanada važan. Rezultat su n-torke iz prve navedene relacije. Ovaj operator omogućava brisanje n-torki iz relacije. Oznaka za operator je "-"

Da bi skup operatora ažuriranja bio potpun, potreban je operator za menjanje sadržaja n-torki relacije. **Za menjanje (ažuriranje) nije uveden poseban operator, već se to izvodi uzastopnom primenom operatora razlike (pri čemu se uklanja n-torka čiji sadržaj želimo izmeniti) pa zatim operatora unije (pri čemu se dodaje n-torka sa željenim sadržajem).**

PRIMER: PODRAZUMEVANI SET OPERACIJA

Primer operacije unja relacione algebre nad skupovima podataka

Pretpostavimo da imamo dve tabele "Studenti" (Students) i "Radnici" (Workers) sa sledećim podacima:

Tabela "Studenti":

student_id	ime
1	Marko
2	Ana
3	Petar

Slika 5.1 Relacija Studenti. Izvor: [Autor]

Tabela "Radnici":

worker_id	ime
1	Jovan
2	Milica
3	Lazar
4	Marija

Slika 5.2 Relacija Radnici. Izvor: [Autor]

Unija ove dve tabele će rezultirati kombinovanim skupom svih zapisa iz obe tabele, pri čemu će se ukloniti duplikati. Evo kako izgleda unija:

```
SELECT student_id, ime FROM Studenti
UNION
SELECT worker_id, ime FROM Radnici;
```

Rezultat unije će biti:

student_id	ime
1	Marko
2	Ana
3	Petar
4	Marija
5	Jovan
6	Milica
7	Lazar

Slika 5.3 Unija relacija Studenti i Radnici. Izvor: [Autor]

Unija kombinuje podatke iz obe tabele "Studenti" i "Radnici" i prikazuje jedinstvene zapise. U ovom primeru, unija daje rezultat koji sadrži sve studente i radnike, bez duplikata, uključujući njihove jedinstvene identifikatore (student_id i worker_id) i imena.

OPERACIJA RAZLIKE RELACIJA

Ilustrativni primer operacije razlike relacija

Pretpostavimo da imamo dve tabele "TimA" (TeamA) i "TimB" (TeamB) sa sledećim podacima:

Tabela "TimA":

id	ime
1	Marko
2	Ana
3	Jovan

Slika 5.4 Relacija TimA. Izvor: [Autor]

Tabela "TimB":

id	ime
2	Ana
3	Jovan
4	Milica
5	Lazar

Slika 5.5 Relacija TimB. Izvor: [Autor]

Razlika između ovih tabela će rezultirati skupom svih zapisa iz prve tabele koji se ne pojavljuju u drugoj tabeli. Evo kako izgleda razlika:

```
SELECT * FROM TimA
EXCEPT
SELECT * FROM TimB;
```

Rezultat razlike će biti:

id	ime
1	Marko

Slika 5.6 Razlika relacija TimA i TimB. Izvor: [Autor]

U ovom primeru, razlika između tabela "TimA" i "TimB" prikazuje samo jedan red koji postoji u tabeli "TimA", ali ne postoji u tabeli "TimB". U ovom slučaju, rezultat prikazuje samo Marka, jer je on jedini član tima koji se ne pojavljuje u drugom timu.

Razlika (eng. difference) omogućava poređenje dve tabele i pronalaženje zapisa koji se pojavljuju samo u jednoj tabeli, a ne u drugoj.

OPERACIJE ZA IZVEŠTAVANJE - SELEKCIJA

SELEKCIJA je operacija nad jednom relacijom koja izdvajanje skupa n-torki jedne relacije u novu. Izdvajaju se samo one n-torke koje zadovoljavaju uslov uz operator selekcije.

Osnovne operacije izveštavanja treba da omoguće prikaz sadržaja dela relacionog modela (jedne ili više relacija) uz mogućnost restrikcije samo nekih atributa ili samo nekih n-torki. Osnovne dve operacije za izveštavanje iz jedne relacije su PROJEKCIJA i SELEKCIJA.

Operacija SELECT se koristi za izbor podskupa torki iz relacije koji zadovoljavaju uslov selekcije. Može se smatrati da je SELECT operacija filter koji čuva samo one torke koje zadovoljavaju kvalifikovani uslov. Alternativno, operaciju SELECT možemo razumeti kao operaciju kojom ograničavamo torke u relaciji na samo one koje zadovoljavaju uslov. Operacija SELECT se takođe može vizualizovati kao horizontalno deljenje relacije u dva skupa torki - one torke koje zadovoljavaju uslov i koje se biraju, i one torke koje ne zadovoljavaju uslov i odbacuju se.

Na primer: Neka je data relacija R_1 (A,B,C) dat uslov B>15 tada je relacija R_2 (A,B,C) rezultat relacije R_1 po uslovu B>15 ako je:

$R_2 (A,B,C) = R_1[B>15] (A,B,C)$

Generalno, operacija SELECT se označava kao:
SELECT [uslov selekcije]R

gde je uslov selekcije Boolean izraz (uslov) specificiran nad atributima relacije R. Relacija koja je rezultat SELECT operacije ima iste atribute kao R. Boolean izraz specificiran u uslovu selekcije je sastavljen od više klauzula oblika:
<ime atributa> <op poređenja> <konstantna vrednost>
ili <ima atributa> <op poređenja> <ime atributa>
gde je <ime atributa> ime atributa R, <op poređenja> je obično jedan od operatora { , , ≤, >, ≥, ≠}, i <konstantna vrednost> je konstantna vrednost iz domena atributa. Klauzule se mogu povezati standardnim Boolean operatorima AND, OR i NOT, formirajući opšte uslove selekcije.
Operator SELECT je unaran; to jest, primenjuje se na jednu relaciju.

OPERACIJE ZA IZVEŠTAVANJE - PROJEKCIJA

PROJEKCIJA je operacija nad jednom relacijom koja omogućava izdvajanje skupa atributa jedne relacije u novu relaciju, a pri tom eliminiše duple n-torke.

Ako razmišljamo o relaciji kao tabeli, SELECT operacija bira neke od redova iz tabele dok odbacuje druge redove. Operacija PROJECT, s druge strane, **bira određene kolone iz tabele i odbacuje ostale kolone.** Ako nas zanimaju samo određeni atributi relacije,

operaciju PROJECT koristimo da bismo projektovali relaciju samo na te atribute. Zbog toga se rezultat operacije PROJECT može vizualizirati kao vertikalna particija relacije u dve relacije: jedna ima potrebne kolone (atribute) i sadrži rezultat operacije, a druga sadrži odbačene kolone.

Na primer: Ako je dana relacija $R_1(A,B,C,D,E)$ tada je relacija $R_2(A,D)$ rezultat projekcije relacije R_1 po atributima A i D ako je:

R_2 (A,D) = PROJEKCIJA (A,D) od R_1 (A,B,C,D,E) ili

R_2 (A,D) = R_1 [A,D] (A,B,C,D,E)

Opšti oblik operacije PROJECT je:
PROJEKCIJA [lista-atributa](R)

gde je [lista atributa] željena podlista atributa svih atributa relacije R. Napomenimo da je R u najjednostavnijem slučaju samo ime relacije baze podataka. Rezultat operacije PROJECT ima samo atribute navedene u [lista atributa] u istom redosledu kao što se pojavljuju na listi.
Ako lista atributa uključuje samo nonkey atribute R, onda će se verovatno pojaviti duple torke. Operacija PROJECT uklanja duple torke, tako da je rezultat operacije PROJECT skup različitih torki, i stoga valjana relacija. To je poznato kao eliminacija duplikata.

OPERACIJE ZA IZVEŠTAVANJE - DEKARTOV PROIZVOD, PRESEK

DEKARTOV PROIZVOD omogućava nastanak nove relacije sa svim atributima obe relacije; PRESEK sadrži samo one n-torke koje se istovremeno nalaze u obe relacije

DEKARTOV PROIZVOD ili CARTESIAN PROIZVOD **je operator nad dve relacije koji omogućava nastanak nove relacije sa svim atributima obe relacije tako što spaja svaku n-torku jedne relacije sa svakom n-torkom druge relacije .** Oznaka za operator je "X".

Kako se primenom ove operacije mogu dobiti rezultati koji nemaju nikakvog smisla, možemo selektovati (SELECT) samo srodne torke dve relacije specificirajući nakon Dekartovog proizvoda odgovarajući uslov selekcije. Pošto je ova sekvenca CARTESIAN PROIZVODA praćena SELECT-om, često se koristi za kombinovanje povezanih torki iz dve relacije, kreirana je specijalna operacija, nazvana JOIN, koja ovu sekvencu specificira kao jednu operaciju.

PRESEK **je operator koji za rezultat daje relaciju koja sadrži samo one n-torke koje se istovremeno nalaze u obe relacije**. Oznaka za operator je "∩"

Ako definišemo proizvoljne skupove R i S, R ∩ S - presek skupova R i S čine elementi koji pripadaju skupovima R i S, tj. elementi koji su zajednički za oba skupa.

Simbol (Ime)	Primer upotrebe
σ	$\sigma_{salary>=85000}(instructor)$
Selekcija	Vrati redove ulazne relacije koji zadovoljavaju predikat.
Π	$\Pi_{ID,salary}(instructor)$
Projekcija	Izlistaj određene atribute iz svih redova ulazne relacije. Ukloni duplikate torki iz rezultujućeg izlaza.
⋈	$instructor \bowtie department$
Natural join	Izlistaj parove redova iz dve ulazne relacije koji imaju iste vrednosti za sve atribute koji imaju isto ime.
×	$instructor \times department$
Dekartov proizvod	Izlistaj sve parove redova iz dve ulazne relacije (bez obzira da li imaju iste vrednosti za zajedničke atribute).
∪	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
Unija	Izlistaj uniju (spoj) torki iz dve ulazne relacije.

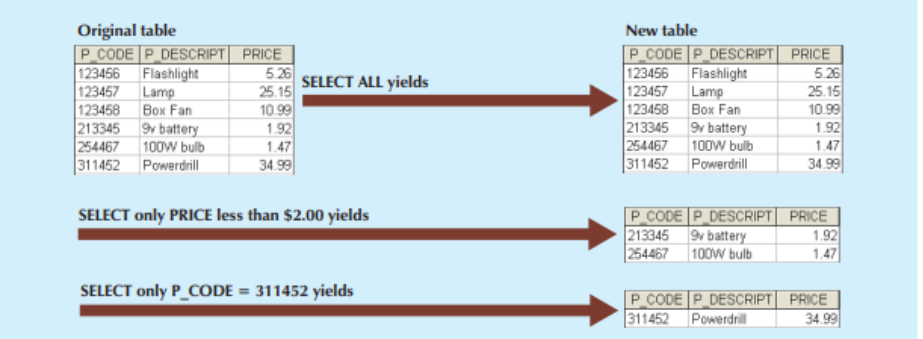
Slika 5.7 Operatori relacione algebre. Izvor: [Autor]

PRIMER: SELEKCIJA

Operator selekcije, primenjen na relaciju R proizvodi novu relaciju kao podskup torki skupa R koje odgovaraju postavljenom uslovu C

Operator selekcije, primenjen na relaciju R proizvodi novu relaciju kao podskup torki skupa R. Torke u rezultujućoj relaciji su takve da zadovoljavaju neke uslove C nad atributima relacije R. Šema rezultujuće relacije je ista kao i šema relacije R i po konvenciji se prikazuje u istom redosledu atributa kao što je prikazano u relaciji nad kojom se vrši selekcija. Uslov C koji mora biti zadovoljen se može definisati kao uslov IF u programskim jezicima poput Jave ili C-a. Jedina razlika u uslovima je u tome što u uslovu C mogu postojati samo konstante ili atributi koji su sadržani u relaciji R. Postupak selekcije se odvija tako što se uslov C proverava na svakoj torci *t* relacije R i za atribut za koji je definisan uslov. Ako je uslov C tačan za atribut koji se traži u uslovu, tada možemo reći da je torka *t* ispunila uslov C i postaje deo rezultata selekcije. U drugom slučaju *t* se izostavlja iz rezultata selekcije.

Drugim rečima, SELECT vraća horizontalni podskup tabele. Efekat SELECT-a je prikazan na Slici 5.8.



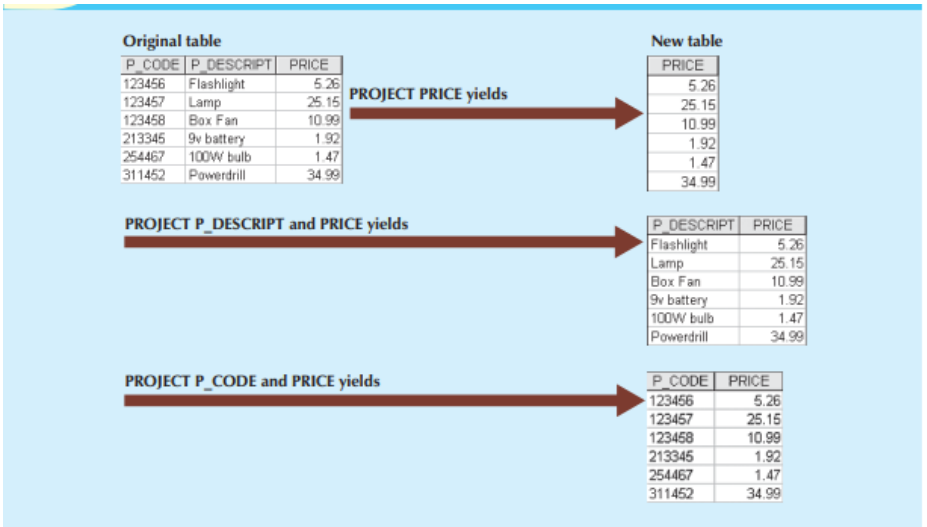
Slika 5.8 Operator SELECT. Izvor: [1]

PRIMER: PROJEKCIJA

Projekcija obezbeđuje da se iz relacije R stvori nova relacija koja ima neku od kolona koja je sadržana u R

Projekcija obezbeđuje da se iz relacije R stvori nova relacija koja ima neku od kolona koja je sadržana u R.

Drugim rečima, PROJECT (PROJEKCIJA) daje sve vrednosti za odabrane atribute. Odnosno, PROJECT daje vertikalni podskup tabele. Efekat PROJECT-a je prikazan na Slici 5.9.



Slika 5.9 Operator PROJECT. Izvor: [1]

OPERACIJE KOJE KOMBINUJU ZAPISE DVE RELACIJE - JOIN

Pored DEKARTOVOG PROIZVODA, za izveštavanje iz više relacija čluži JOIN

Operacija JOIN, se koristi za kombinovanje povezanih torki iz dve relacije u pojedinačne "duže" torke. Ova operacija je veoma važna za bilo koju relacionu bazu podataka sa više od jedne relacije, jer nam omogućava da obradimo uspostavljene relacije (eng. relationships) koje postoje među relacijama (eng. relation).

JOIN (spajanje) omogućava kombinovanje informacija iz dve ili više tabela. JOIN je prava snaga iza relacijske baze podataka, omogućavajući korišćenje nezavisnih tabela koje su povezane zajedničkim atributima.

Operacija JOIN može biti specificirana kao operacija CARTESIAN PRODUCT koju prati SELECT operacija. Međutim, JOIN je vrlo važna operacija jer se vrlo često koristi prilikom specificiranja upita za bazu podataka.

Opšti oblik JOIN operacije za dve relacije R (A1, A2, ..., An) i S (B1, B2, ..., Bm) je:

R JOIN [uslov spajanja] S
Rezultat JOIN-a je relacija Q sa n + m atributa Q(A1, A2, ..., An, B1, B2, ..., Bm) u datom redosledu; Q ima jednu torku za svaku kombinaciju torki - jednu iz R i jednu iz S - kad god kombinacija zadovoljava uslov spajanja.

Tabele "CUSTOMER" i "AGENT" prikazane na Slici 5.10 će biti korišćene da ilustruju nekoliko tipova spajanja (JOIN).

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Slika 5.10 Tabele CUSTOMER i AGENT. Izvor: [1]

OPERACIJE KOJE KOMBINUJU ZAPISE DVE RELACIJE - THETA JOIN

THETA JOIN je operacija JOIN sa opštim uslovom spajanja

THETA JOIN je operacija JOIN sa opštim uslovom spajanja oblika A_i θ (teta) B_j, A_i je atribut iz R, B_j je atribut iz S, a θ je jedan od operatora poređenja { =, ≠, <, <=, >=, > } .

Rezultat THETA JOIN-a je nova relacija koja ima sve atribute prethodnih relacija, a n-torke nastaju konkateniranjem n-torki relacija.

Na primer: ako je urađen THETA JOIN relacija R_1 (A,B,C) i R_2 (D,E,F) dobiće se relacija R_3 (A,B,C,D,E,F)

Najčešća upotreba JOIN-a uključuje samo uslove spajanja koji podrazumevaju poređenje na jednakost. Takav JOIN, gde je jedini operater poređenja koji se koristi =, naziva se EQUIJOIN.
Prirodno spajanje (eng .naturaljoin) povezuje tabele tako što selektuje samo one redove koji imaju zajedničke vrednosti u njihovim zajedničkim atributima. Prirodno spajanje rezultat je trofaznog procesa:

a. Prvo, kreira se PROIZVOD (PRODUCT) tabela na osnovu tabele, čime se dobijaju rezultati prikazani na Slici 5.10.

b. Drugo, izvodi se SELEKCIJA (SELECT) na rezultatima koraka a. kako bi se dobili samo redovi za koje su vrednosti AGENT_CODE jednake. Ovo zajedničke kolone nazivamo kolonama za spajanje (eng. joincolumns). Korak b. daje rezultate prikazane na Slici 5.11.

c. Izvodi se PROJEKCIJA (PROJECT) nad rezultatima koraka b. kako bi se dobila samo jedna kopija svakog atributa i time eliminišu duplirane kolone. Korak c. daje izlaz prikazan na Slici 5.12.

Konačni rezultat prirodnog spajanja daje tabelu koja ne uključuje nespojene parove i pruža samo kopije podudaranja.

KORACI ZA NATURAL JOIN

Ilustracija primene operatora NATURAL JOIN

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

Slika 5.11 Natural join, korak a: PRODUCT. Izvor: [1]

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

Slika 5.12 Natural join, korak b: SELECT. Izvor: [1]

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

Slika 5.13 Natural join, korak c: PROJECT. Izvor: [1]

KOMBINOVANJE OPERACIJA PRILIKOM FORMIRANJA UPITA

Prikaz načina na koji se operacije mogu kombinovati prilikom postavljanja upita

Relaciona algebra, kao i sve algebre dozvoljava formiranje izraza proivoljne složenosti primenom operacija na rezultate izvršenja drugih operacija. Izraz relacione algebre se može

konstruisati primenom operacija na podizraz korišćenjem zagrada koje ukazuju na način na koji su operacije grupisane. Takođe je moguće izraze predstaviti kao stabla izraza, što je lakše za razumevanje.

Pretpostavimo da imamo dve tabele: "Customers" (Kupci) i "Orders" (Porudžbine), koje izgledaju ovako:

CustomerID	Name	City
1	Mark	New York
2	Sarah	Los Angeles
3	John	Chicago

Slika 5.14 Tabela Customers. Izvor: [Autor]

OrderID	CustomerID	Product	Quantity
101	1	Laptop	2
102	2	Smartphone	1
103	1	Tablet	3
104	3	Camera	1

Slika 5.15 Tabela Orders. Izvor: [Autor]

Sada želimo da napravimo upit koji će nam vratiti imena i gradove kupaca koji su iz New Yorka i koji su naručili više od 1 proizvoda. Upit koristi sledeće operatore relacione algebre:
SELECT (SELEKCIJA): Odabira redove koji ispunjavaju određeni uslov. U ovom slučaju, uslovi su "City = New York" i "Quantity > 1".
PROJECT (PROJEKCIJA): Odabira samo određene kolone koje su nam potrebne. U ovom slučaju, to su "Name" i "City" kolone.

Relaciona algebra upita:

```
PROJECT(Name, City)(SELECT(City = 'New York' AND Quantity > 1)(Customers JOIN
Orders))
```

Rezultat ovog upita će biti:

Name	City
Mark	New York

Slika 5.16 Tabela kao rezultat upita Izvor: [Autor]

Ovaj upit koristi JOIN operaciju kako bi spojio tabele "Customers" i "Orders" na osnovu zajedničkog atributa "CustomerID". Zatim koristi SELECT operaciju za filtriranje redova sa odgovarajućim uslovima i na kraju PROJECT operaciju da prikaže samo imena i gradove tih kupaca.

IZRAZ RELACIONE ALGEBRE KAO STABLO IZRAZA

Dat je prikaz stabla izraza

Da bismo prikazali stablo izraza za dati upit, moramo prvo razumeti redosled izvršavanja operatora. U relacionoj algebri, obično se najpre primenjuju operacije selekcije, zatim spajanje (join) i na kraju projekcija. Takođe, bitno je uočiti da se neki operatori mogu izvršavati paralelno.

Stablo izraza za dati upit bi izgledalo ovako:

Slika 5.17 Stablo izraza za primer upita. Izvor: [Autor]

Oznake u stablu:
π predstavlja operaciju projekcije.
σ predstavlja operaciju selekcije sa uslovima City=NY i Quantity>1.
⋈ predstavlja operaciju prirodnog spajanja (natural join) između tabela Customers i Orders na osnovu zajedničkog atributa CustomerID.
Ovo stablo prikazuje kako se redom izvršavaju navedene operacije. Prvo se vrši selekcija, zatim spajanje, a na kraju projekcija, kako bi se dobio krajnji rezultat upita.

U linearnoj notaciji, stablo izraza se prikazuje kao niz operatora i operanada u redosledu izvršavanja. Svaki operator se pridružuje operandu koji sledi. Evo kako bi izgledao isti upit u linearnoj notaciji:

π(Name, City) (σ(City = ’New York’ ∧ Quantity > 1) (Customers ⋈ Orders)).

VIDEO

Relational Algebra - Basic Operators

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 6

Pokazna vežba

NAČIN ORGANIZACIJE POKAZNIH VEŽBI

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata.

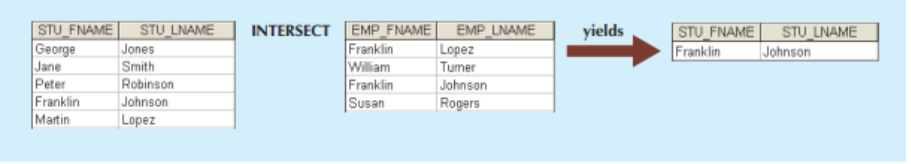
1. U uvodnom delu pokaznih vežbi se daje pokazni primer koji studentima treba da pomogne u samostalnom rešavanju zadataka.
2. Zadatke koji su zadati za samostalni rad, student samostalno rešava uz pomoć asistenta.

6.1 Pokazni primer

PRIMERI OPERATORA PRESEK, DEKARTOV PROIZVOD - (5 MIN.)

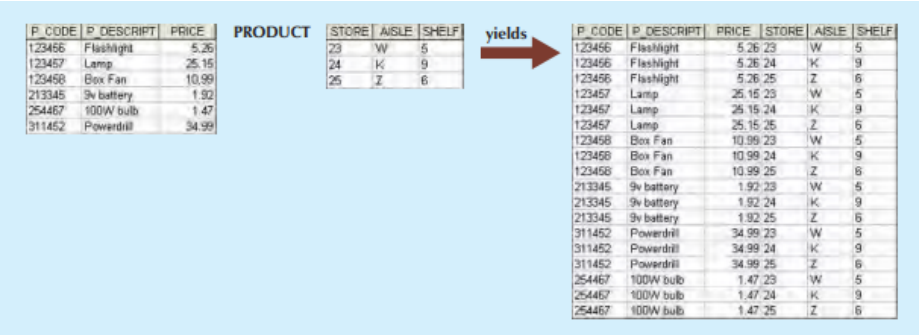
Presek dve relacije je nova relacija u kojoj se nalaze sve torke 1. relacije, koje se nalaze u 2. relaciji. Kod Dekartovog proizvoda se svaka torka prve relacije spaja sa svakom iz druge

Što se tiče relacionog preseka (∩) sličan je slučaj. Presek iz dve polazne relacije formira novu koja sadrži sve torke prve relacije a koje se nalaze u drugoj relaciji. Ta relacija je moguća samo između unijski kompatibilnih relacija. To je izvedena operacija i može se izvesti iz r ∩ s=r-(r-s).



Slika 6.1.1 Primer relacionog preseka. Izvor: [1]

Relacioni Dekartov proizvod (X) - iz dve polazne relacije formira se nova sa torkama dobijenim tako što se svaka torka prve relacije spaja sa svakom iz druge. Takva relacija sadrži sve atribute polaznih relacija. Označavanje za puni naziv atributa se može koristiti relacija.atribut. Nakon određivanja Dekartovog proizvoda moguće je izvršiti restrikciju po vrstama i/ili kolonama kako bi se dobile samo određene informacije.



Slika 6.1.2 Primer Dekartovog proizvoda Izvor: [1]

PRIMERI PROJEKCIJE - (4 MIN.)

Primer projekcije nad relacijom Zaposleni

Sadržaj relacije "Employees" (Zaposleni) je prikazan na slici 7.3.

EmployeeID	First_Name	Last_Name	Department	Salary
101	John	Smith	HR	50000
102	Mary	Johnson	Sales	60000
103	Robert	Williams	Marketing	55000
104	Sarah	Davis	HR	52000

Slika 6.1.3 Relacija Zaposleni. Izvor: [Autor]

Sada ćemo primeniti projekciju na atribut "Last_Name" kako bismo dobili samo prezimena zaposlenih:

Projekcija po atributu "Last_Name":
π(Last_Name)(Employees)

Ovde smo koristili operator π (projekcija) kako bismo izvukli samo kolonu "Last_Name" iz relacije "Employees". Na ovaj način dobijamo novu tabelu koja sadrži samo prezimena svih zaposlenih, slika 7.4.

Slika 6.1.4 Rezultat operacije projekcije. Izvor: [Autor]

PRIMERI SELEKCIJE - (5 MIN.)

Primer selekcije nad relacijom Zaposleni

Operacijom selekcije se biraju sve n-torke relacije koje zadovoljavaju iskaz naveden u oznaci operacije.

Upotrebljen logički iskaz ima za argumente poređenja vrednosti dva atributa zadate relacije (nad istim domenima), ili poređenje jednog atributa zadate relacije sa konstantom koja pripada odgovarajućem domenu. Operacije logičkih iskaza su negacija, ILI i I.

Teta selekcija (Θ selekcija): je relacijska operacija iz skupa {<, >, =, ≤,≥, ≠}

Neka su A_i, A_j atributi relacije R sa zajedničkim domenom nad kojim je definisana operacija Θ i neka je c konstanta iz istog domena

Operacija Θ selekcije se označava kao $R[A_i \Theta A_j]$ odnosno $R [A_i \Theta c]$:

$R[A_i \Theta A_j] = \{x|x\ R\ \hat{}\ x[A_i] \Theta x[A_j]\}$

Primer: Sadržaj relacije Zaposleni prikazan je na slici 3. Pretpostavimo da želimo izdvojiti zaposlene iz odeljenja "HR". Koristićemo operator σ (selekcija) kako bismo odabrali redove koji ispunjavaju ovaj uslov.

Selekcija za odeljenje "HR":

σ(Department = ’HR’)(Employees)

EmployeeID	First_Name	Last_Name	Department	Salary
101	John	Smith	HR	50000
104	Sarah	Davis	HR	52000

Slika 6.1.5 Rezultat operacije selekcije. Izvor: [Autor]

U ovom primeru, koristili smo operator σ (selekcija) sa uslovom "Department = ’HR’" kako bismo izdvojili samo redove gde je vrednost atributa "Department" jednaka "HR". Na ovaj način dobijamo novu tabelu koja sadrži samo zaposlene iz odeljenja "HR".

POKAZNI PRIMER BR. 1 - (4 MIN.)

Primer kreiranja i modifikacija šeme relacione baze podataka

Šema relacione baze podataka se sastoji iz četiri relacije: Product, PC, Laptop i Printer čiji su sadržaji dati u sledećoj sekciji.
Product(model,maker, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)

Relacija Product opisuje različite vrste proizvoda i sadrži atribute: *model, maker i type*. Relacije *PC, Laptop i Printer* predstavljaju različite proizvode. Radi jednostavnosti, iako to nije realno, pretpostavimo da je *model* jedinstven kod svih proizvoda i da predstavlja primarni ključ relacije *Product, PC, Laptop i Printer*. Relacija *PC* korišćenjem atributa *speed, ram, hd, price* opisuje brzinu procesora PC-a u GHz, količinu memorije (u MB), veličine hard diska (u GB), kao i cenu. Relacija *Laptop* se sastoji od sličnih atributa uz dodatak atributa *screen* koji se odnosi na veličinu ekrana (u inčima). Relacija *Printer* sadrži atribute *color* koji označava indikatoru da li štampač u boji ili ne, *type* koji se odnosi na tip štampača (laser ili ink-jet) i *price*.

Napisati sledeće deklaracije:

1. Napišite šeme relacija Product, PC, Laptop i Printer
2. Pronađite proizvođače koji prave laptopove sa najmanje 100GB hard diska.
3. Koji PC modeli imaju brzinu najmanje 3.00.
4. Pronađite broj modela i cene za sve proizvode tipa B
5. Pronađite brojeve modela svih laserskih štampača u boji.

POKAZNI PRIMER BR. 1 - SADRŽAJI RELACIJA - (4 MIN.)

Na slikama su dati primeri sadržaja relacija koji se može naći u opisanoj bazi podataka

maker	model	type
ABC	1001	Laptop
XYZ	2002	Smartphone
DEF	3003	Tablet
GHI	4004	Camera

Slika 6.1.6 Tabela "Product". Izvor: [Autor]

model	speed	ram	hd	price
1001	2.4	8	500	600
1002	3.0	16	1000	900
1003	2.8	4	250	400
1004	2.2	4	500	350

Slika 6.1.7 Tabela "PC". Izvor: [Autor]

model	speed	ram	hd	screen	price
2001	2.2	8	500	15.6	700
2002	2.8	16	1000	14.0	1000
2003	2.5	4	256	13.3	500
2004	2.0	4	500	15.6	600

Slika 6.1.8 Tabela "Laptop". Izvor: [Autor]

model	color	type	price
3001	Yes	Inkjet	150
3002	No	Laser	300
3003	Yes	Inkjet	120
3004	No	Laser	250

Slika 6.1.9 Tabela "Printer". Izvor: [Autor]

POKAZNI PRIMER BR. 1 - REŠENJE - (3 MIN.)

U nastavku su date šeme za relacije iz zadatka 1

Šema za relaciju Product:

Product(
maker: string,
model: integer,
type: string
)

Šema za relaciju PC:
PC(
model: integer,
speed: integer,
ram: integer,
hd: integer,
price: integer
)

Šema za relaciju LAPTOP:
LAPTOP(
model: integer,
speed: integer,
ram: integer,
hd: integer,
screen:integer,
price: intege)

Šema za relaciju PRINTER:

PRINTER(
model: integer,
color: string,
type: string,
price: integer
)

POKAZNI PRIMER BR. 1 - NASTAVAK REŠENJA - (5 MIN.)

U nastavku su dati odgovori na pitanja koja se odnose na relacionu algebru

2. Pronađite proizvođače koji prave laptopove sa najmanje 100GB hard diska.

$\pi_{maker}(Product \times \sigma_{hd \geq 100}(Laptop))$

3. Koji PC modeli imaju brzinu najmanje 3.00.

$\pi_{model}(\sigma_{model \geq 3.00}(PC))$

4. Pronađite broj modela i cene za sve proizvode koje je napravio proizvodjač B (bilo kog tipa - PC, LAPTOP, PRINTER)

$\pi_{model,price}(\sigma_{maker=B(Product)}) \times (\pi_{model,price}(PC) \cup \pi_{model,price}(Laptop) \cup \pi_{model,price}(Printer))$

5. Pronađite brojeve modela svih laserskih štampača u boji.

$\pi_{model}(\sigma_{type=laser \wedge color=true}(Printer)$

PRIMERI OPERATORA UNIJA - (5 MIN.)

Unija na osnovu dve polazne relacije formira nova koja sadrži sve torke iz obe relacije.

Operacija relacione algebre unija (U) je specifična u odnosu na tu operaciju u matematici. Relaciona unija znači da se iz dve polazne relacije formira nova koja sadrži sve torke iz obe relacije. Ova relacija nije moguća između bilo koje dve relacije i mora biti zadovoljeno sledeće:

1. Šeme relacija moraju imati isti broj atributa
2. Atributi šema relacija redom odgovaraju po značenu i tipu (ne mora po nazivu).

To se naziva unijska kompatibilnost. Kada se uniraju vrste? Samo kada imaju kompatibilne kolone.

Pretpostavimo da imamo dve tabele, "Employees" (Zaposleni) i "Contractors" (Saradnici), koje sadrže informacije o zaposlenima i spoljnim saradnicima u jednoj kompaniji.

ID	Name	Department
1	John	HR
2	Mary	Sales
3	Robert	Marketing

Slika 6.1.10 Tabela Zaposleni. Izvor: [Autor]

ID	Name	Project
101	David	ProjectA
102	Sarah	ProjectB
103	Michael	ProjectC

Slika 6.1.11 Tabela Saradnici. Izvor: [Autor]

```
SELECT ID, Name, Department FROM Employees
UNION
SELECT ID, Name, Project AS Department FROM Contractors;
```

```
SELECT ID, Name, Department FROM Employees
UNION
SELECT ID, Name, Project AS Department FROM Contractors;
```

REZULTAT PRIMENE OPERATORA UNIJE - (3 MIN.)

Dat je rezlutat upita

Rezultat unije će biti:

ID	Name	Department
1	John	HR
2	Mary	Sales
3	Robert	Marketing
101	David	ProjectA
102	Sarah	ProjectB
103	Michael	ProjectC

Slika 6.1.12 Rezultat unije. Izvor:[Autor]

Primetite da su svi redovi iz obe tabele kombinovani u rezultujućoj tabeli, pri čemu su očuvani jedinstveni redovi.

PRIMERI OPERATORA RAZLIKA - (5 MIN.)

Razlika na osnovu dve relacije formira novu koja sadrži torke prve relacije koje se ne nalaze u drugoj.

Relaciona razlika (-) iz dve polazne relacije formira novu koja sadrži sve torke prve relacije koje se ne nalaze u drugoj. Ova operacija je moguća samo između unijski kompatibilnih relacija.

Pretpostavimo da imamo dve tabele, "Employees" (Zaposleni) i "FormerEmployees" (BivšiZaposleni), koje sadrže informacije o trenutnim i bivšim zaposlenima u jednoj kompaniji.

ID	Name	Department
1	John	HR
2	Mary	Sales
3	Robert	Marketing
4	Laura	Finance

Slika 6.1.13 Tabela Zaposleni. Izvor: [Autor]

ID	Name	Department
2	Mary	Sales
4	Laura	Finance
5	Alex	IT

Slika 6.1.14 Tabela BivšiZaposleni. Izvor: [Autor]

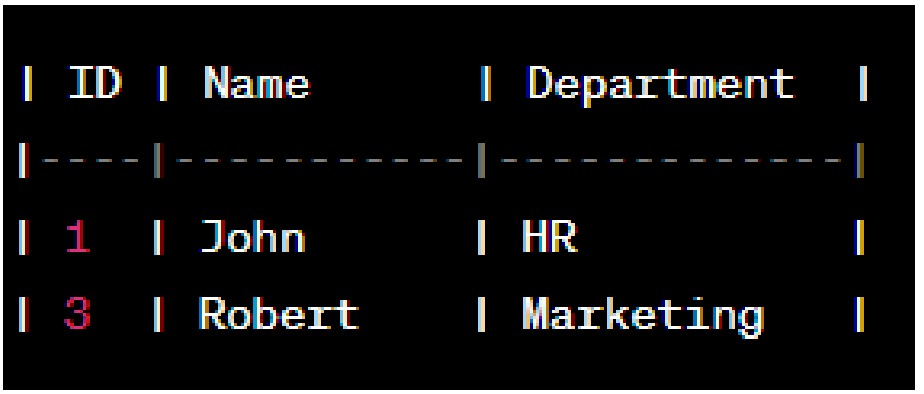
Relaciona razlika (DIFFERENCE) upit:

```
SELECT ID, Name, Department FROM Employees
EXCEPT
SELECT ID, Name, Department FROM FormerEmployees;
```

REZULTAT PRIMENE OPERATORA RAZLIKE - (2 MIN.)

Dat je rezultat upita

Rezultat razlike će biti:



ID	Name	Department
1	John	HR
3	Robert	Marketing

Slika 6.1.15 Rezultat upita. Izvor: [Autor]

U ovom primeru, izvršili smo razliku između tabela "Employees" i "FormerEmployees" koristeći operator EXCEPT (takođe poznat i kao MINUS ili DIFFERENCE). Rezultat će sadržati sve redove koji se nalaze samo u tabeli "Employees", a nisu prisutni u tabeli "FormerEmployees".

6.2 Pokazna vežba - samostalni rad

ZADATAK ZA SAMOSTALNI RAD

Primeri relacione algebre - Potrebno vreme za rešavanje svakog zadatka je 7,5 minuta.

Na slici 7.1. je data šema relacione baze, dok je na slikama 7.2, 7.3 i 7.4 dat prikaz torki relacija Classes, Battles i Outcome.

```
Classes(class, type, country, numGuns, bore, displacement)
Ships(name, class, launched)
Battles(name, date)
Outcomes(ship, battle, result)
```

Slika 6.2.1 Primer relacija relacione baze podataka

class	*type*	*country*	*numGuns*	*bore*	*displacement*
Bismarck	bb	Germany	8	15	42000
Iowa	bb	USA	9	16	46000
Kongo	bc	Japan	8	14	32000
North Carolina	bb	USA	9	16	37000
Renown	bc	Gt. Britain	6	15	32000
Revenge	bb	Gt. Britain	8	15	29000
Tennessee	bb	USA	12	14	32000
Yamato	bb	Japan	9	18	65000

Slika 6.2.2 Relacija Classes [Izvor: NM IT350-2020/2021.]

name	*date*
Denmark Strait	5/24-27/41
Guadalcanal	11/15/42
North Cape	12/26/43
Surigao Strait	10/25/44

Slika 6.2.3 Relacija Battles [Izvor: NM IT350-2020/2021.]

ship	*battle*	*result*
Arizona	Pearl Harbor	sunk
Bismarck	Denmark Strait	sunk
California	Surigao Strait	ok
Duke of York	North Cape	ok
Fuso	Surigao Strait	sunk
Hood	Denmark Strait	sunk
King George V	Denmark Strait	ok
Kirishima	Guadalcanal	sunk
Prince of Wales	Denmark Strait	damaged
Rodney	Denmark Strait	ok
Scharnhorst	North Cape	sunk
South Dakota	Guadalcanal	damaged
Tennessee	Surigao Strait	ok
Washington	Guadalcanal	ok
West Virginia	Surigao Strait	ok
Yamashiro	Surigao Strait	sunk

Slika 6.2.4 Relacija Outcome [Izvor: NM IT350-2020/2021.]

ZADATAK ZA SAMOSTALNI RAD - NASTAVAK

Primeri relacione algebre - nastavak

name	*class*	*launched*
California	Tennessee	1921
Haruna	Kongo	1915
Hiei	Kongo	1914
Iowa	Iowa	1943
Kirishima	Kongo	1915
Kongo	Kongo	1913
Missouri	Iowa	1944
Musashi	Yamato	1942
New Jersey	Iowa	1943
North Carolina	North Carolina	1941
Ramillies	Revenge	1917
Renown	Renown	1916
Repulse	Renown	1916
Resolution	Revenge	1916
Revenge	Revenge	1916
Royal Oak	Revenge	1916
Royal Sovereign	Revenge	1916
Tennessee	Tennessee	1920
Washington	North Carolina	1941
Wisconsin	Iowa	1944
Yamato	Yamato	1941

Slika 6.2.5 Relacija Ships [Izvor: NM IT350-2020/2021.]

Na osnovu prethodno definisanih relacija, potrebno je:

1. Pronaći sve flote koje su pripadale Velikoj Britaniji
2. Pronaći imena brodova i bitke u kojima su potopljeni
3. Pronaći brodove pokrenute pre 1921 godine
4. Pronaći brodove koji su potopljeni u borbi *Denmark Strait*
5. Pronaći ime i broj topova angažovanih u borbi za *Guadalcanal.*
6. Pronaći sve flote koje su imale više od 1000 oružja

Potrebno vreme za rešavanje svakog zadatka je 7,5 minuta.

Poglavlje 7

Domaći zadatak

UPUTSTVO ZA SLANJE DOMAĆEG ZADATKA

Domaći zadatak treba poslati korišenjem sledećg uputstva

Na osnovu zadatka za pokazne vežbe i zadataka za samostalni rad, napisati deklaracije (izraze sa odgovarajućim operatorima relacione algebre) kojima se mogu pronaći traženi podaci.

Svaki student radi dva zadatka sa spiska zadataka, po jedan iz svake grupe. Ostali zadaci mogu da posluže za vežbanje i pripremu ispita, ali ih ne šaljete profesoru ili asistentima na ocenjivanje.
Zadatke koje treba da uradite za domaći zadatak određujete po sledećoj formuli:
Broj indeksa % 15 + 1 (Npr., 2378 % 15 + 1 = 9 – Student radi 9. zadatak iz obe grupe).

Prilikom slanja domaćih zadatka, neophodno je da ispunite sledeće:
Subject mail-a mora biti IT250-DZbr (u slučaju kada šaljete domaći za ovu nedelju to je IT250-DZ02)
U prilogu mail-a treba da se nalazi arhiviran projekat koji se ocenjuje imenovan na sledeći način: IT250-DZbr-BrojIndeksa-Ime Prezime. Na primer, IT250-DZ02-1234-VeljkoGrkovic
Telo mail-a treba da ima pozdravnu poruku
Arhivu sa zadatkom poslati na adresu predmetnog asistenta:
milica.vlajkovic@metropolitan.ac.rs (studenti u Beogradu i online studenti) ili
tamara.vukadinovic@metropolitan.ac.rs (studenti u Nišu).

Svi poslati mail-ovi koji ne ispunjavaju navedene uslove NEĆE biti pregledani. Za sva pitanja ili nedoumice u vezi zadatka, možete se obratiti asistentu

DOMAĆI ZADATAK 2

Rešiti zadatke prema postavljenom tekstu - vreme izrade svakog zadatka je 45 min.

Zadatak 1: Vreme izrade zadatka 45 min.

1. Svi brojeve modela crno-belih štampača
2. Proizvođače koji prodaju manje od tri različita modela PC-a
3. Modele PC-a koji imaju brzinu jednaku 50GHz

4. Proizvođače koji proizvode najviše dva različita računara (PC ili laptop) sa brzinama ne manjim od 2.50GHz
5. Modele crno-belih laserskih štampača čija je cena veća od 850
6. Sve laptopove čija je cena veća od 1000
7. Proizvođače koji imaju najmanje tri modela računara sa različitim brzinama
8. Proizvođače koji prodaju tačno tri različita modela PC-a
9. Svi brojeve modela kolor štampača
10. Proizvođače koji prodaju manje od dva različita modela PC-a
11. Proizvođače koji proizvode najmanje dva različita računara (PC ili laptop) sa brzinama ne manjim od 2.80GHz
12. Sve računare (PC ili laptop) čija je cena manja od 1100
13. Sve proizvođače štampača u boji
14. Modele laserskih štampača u boji čija je cena manja od 850
15. Sve računare (PC ili laptop) koji imaju brzinu veću od 2.6GHz i najmanje 8GB rama

Zadatak 2: Vreme izrade zadatka 45 min.

1. Sporazumi sklopljeni u Vašingtonu iz 1921. godine koji zabranjuje korišćenje brodova težih od 35000 tona. Pronaći brodove koji krše ovaj sporazum.
2. Sve zemlje koje poseduju borbene brodove (bb) i kruzere (bc)
3. Sve zemlje koje su posedovale borbene brodove (bb) pre 1921. godine
4. Sve zemlje koje poseduju borbene brodove (bb) ili borbene kruzere (bc)
5. Nazive brodova koje je posedovao Japan pre 1930. godine
6. Nazive borbenih brodove (bb) koje je posedovala Velika Britanija nakon 1930. godine
7. Nazive borbenih kruzera (bc) koje su posedovale Sjedinjene Američke Države nakon 1930. godine
8. Nazive brodova koji su potopljeni pre 1943. godine
9. Nazive brodova koji su oštećeni za vreme 1943. godine
10. Klase brodova koje imaju više od 8 topova
11. Klase brodova čiji topovi imaju kalibar (bore) najmanje 16“
12. Broj brodova potopljenih u borbi za Danski tesnac (Denmark strait)
13. Broj Japanskih brodova potopljenih u borbama oko Gvadalkanala
14. Broj brodova SAD-a koji iz borbe za Surigao tesnac (Surigao strait) ostali neštećeni
15. Broj brodova koji pripadaju klasama Revenge ili Renown

Poglavlje 8

Zaključak

ZAKLJUČAK

Šta smo naučili u ovoj lekciji?

Relacija u matematici se može definisati kao bilo koji podskup Dekartovog proizvoda dva skupa. Pojam i definicija relacije iz matematike su iskorišćeni za definiciju i razvoj relacionog modela (baze) podataka.

Najvažniji termini koji se koriste u relacionom modelu su: relacija, šema relacije, šema baze podataka, ključ (kandidat ključ, primarni ključ, surogat ključ, strani ključ) kao i ograničenja referencijalnog integriteta, koji su objašnjeni u ovoj lekciji.

Ključevi definišu funkcionalne zavisnosti, odnosno drugi atributi zavise od ključa i mogu se pronaći ako se zna vrednost ključa.
U lekciji se takođe govori u primeni operatora relacione algebre u pretraživanju relacija. Relacioni model podržava funkcije relacione algebre: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT i DIVIDE.

LITERATURA

Za pisanje ove lekcije je korišćena sledeća literatura:

1. http://corpgov.crew.ee/Materjalid/Database%20Systems%20-%20Design,%20Implementation,%20and%20Management%20(9th%20
2. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, DATABASE SYSTEMS The Department of Computer Science, Stanford University, 2009 by Pearson Education Inc.