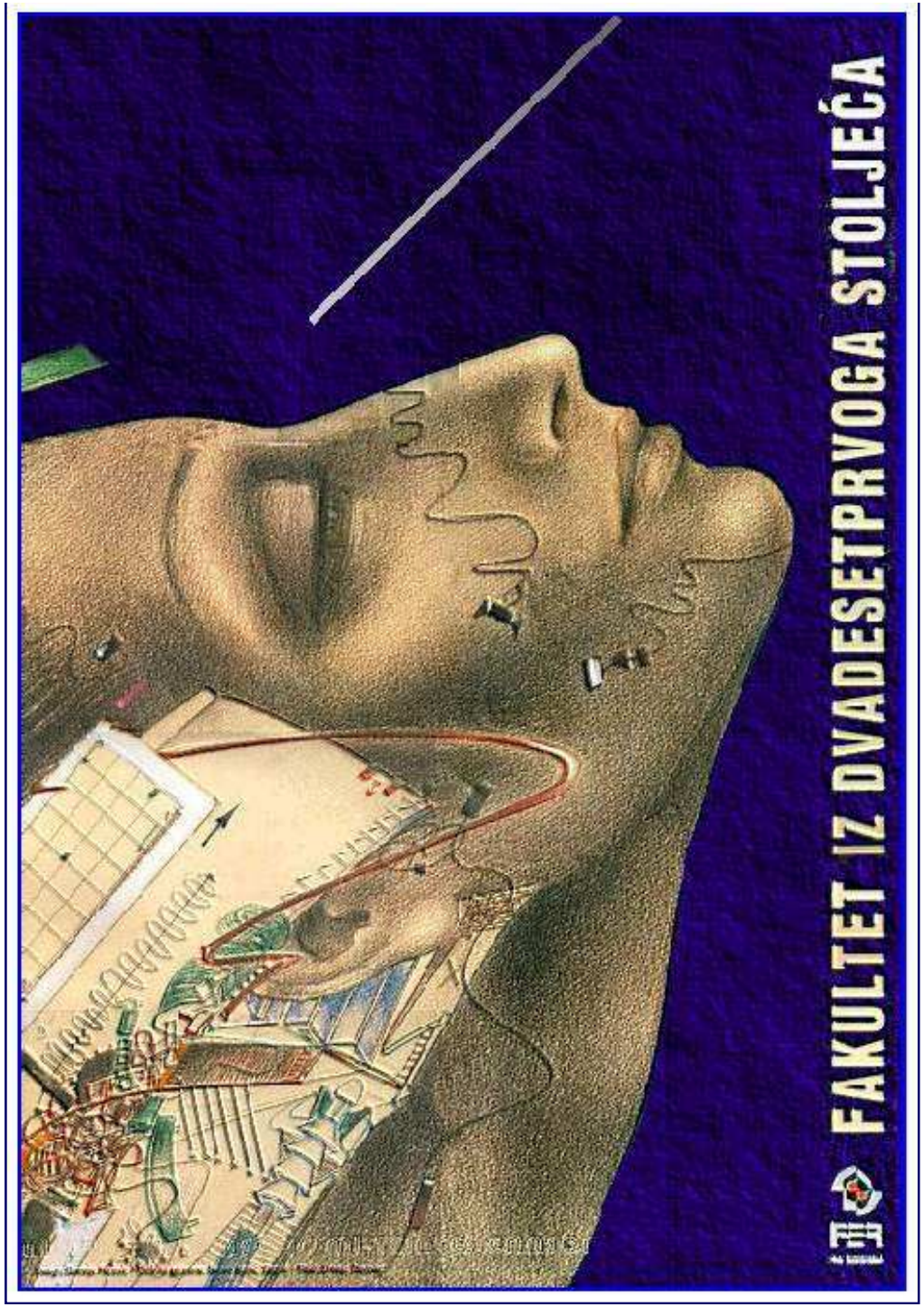
Napredni modeli i baze podataka

Predavanja

4.
Objektno-relacijske baze podataka (1/2)

Listopad 2008.



Pregled

- Evolucija sustava za upravljanje bazama podataka
- Ugnježdene relacije
- Objektno-relacijski model podataka
- Objektno-relacijske mogućnosti prema SQL standardu - 1.dio
  - Strukturirani tipovi podataka
  - Tipizirane tablice
  - Nasljeđivanje tipova i tablica
- Objektno-relacijska proširenja u IBM Informix sustavu - 1. dio
  - Imenovani i neimenovani ROW tip podatka
  - Nasljeđivanje tipova i tablica

Evolucija sustava za upravljanje bazama podataka

- relacijski sustav (*relational DBMS* - RDBMS)
  - podržava upite vrlo visokog nivoa
  - relacijski model je temelj komercijalnih relacijskih SUBP-ova
- objektno orijentirani sustav (*object DBMS* - ODBMS)
  - podržava 'zanimljive' podatkovne tipove (audio, video zapisi ...)
  - nisu doživjeli uspjeh jer nisu ponudili učinkovitost dobro pozicioniranih relacijskih SUBP-ova
- objektno-relacijski sustav (*object-relational DBMS* - ORDBMS) ili prošireni relacijski sustav (*enhanced relational systems*)
  - pokušaj spajanja najboljeg iz relacijskog i objektno-orijentiranog pristupa
  - objektno-relacijska proširenja relacijskih SUBP-ova uključila su većinu prednosti objektne orijentacije, zadržavajući relaciju kao temeljnu apstrakciju
  - prototip objektno-relacijskih sustava: Postgres (PostgreSQL)
  - primjer komercijalnih sustava: Oracle, IBM Informix

Ugnježdene relacije

- Primjer: informacijski sustav knjižnice
- Svaka je knjiga opisana:
  - naslovom,
  - skupom autora,
  - izdavačem,
  - skupom ključnih riječi
- Relacija *knjiga* ne zadovoljava 1NF

knjiga

naslov	autori	izdavac	kljucneRijeci
		(naziv,mjesto)	
Compilers	{Smith, Jones}	(McGraw-Hill, New York)	{parsing, analysis}
Networks	{Jones, Frick}	(Oxford, London)	{Internet, Web}

1NF verzija ugnježđene relacije

- Relacija *knjiga1NF:* 1NF verzija relacije *knjiga*

knjiga1NF

naslov	autor	nazivIzdavac	mjestoIzdavac	kljucnaRijec
Compilers	Smith	McGraw-Hill	New York	parsing
Compilers	Jones	McGraw-Hill	New York	parsing
Compilers	Smith	McGraw-Hill	New York	analysis
Compilers	Jones	McGraw-Hill	New York	analysis
Networks	Jones	Oxford	London	Internet
Networks	Frick	Oxford	London	Internet
Networks	Jones	Oxford	London	Web
Networks	Frick	Oxford	London	Web

Ugnježđena relacija nakon normalizacije

- dekompozicijom nastaju relacijske sheme:

KNJIGA={ naslov, imeIzdavac, mjestoIzdavac }

KNJIGAAUTOR={ naslov, autor }

KNJIGAKR={ naslov, kljucnaRijec }

K_{KNJIGA}={ naslov }

$K_{KNJIGAAUTOR}$={ naslov, osoba }

$K_{KNJIGAKR}$={ naslov, kljucnaRijec }

knjigaN(KNJIGA)

naslov	nazivIzdavac	mjestoIzdavac
Compilers	McGraw-Hill	New York
Networks	Oxford	London

knjigaAutor(KNJIGAAUTOR)

naslov	autor
Compilers	Smith
Compilers	Jones
Networks	Jones
Networks	Frick

knjigaKR(KNJIGAKR)

naslov	kljucnaRijec
Compilers	parsing
Compilers	analysis
Networks	Internet
Networks	Web

Problemi s normaliziranim relacijama

- nužno obaviti operaciju spajanja u upitima
- 1NF relacijski pogled (*knjiga1NF)* definiran spajanjem normaliziranih relacija:
  - eliminira spajanje u upitima
  - izgubljeno jedan-na-jedan podudaranje između n-torke i knjige
  - redundancija
- prikaz pomoću ugnježdene relacije je najprirodniji

Ugnježđene relacije

- Motivacija:
  - dozvoliti neatomarne domene (npr. skup cijelih brojeva, skup n-torki)
  - omogućiti intuitivno modeliranje za aplikacije s kompleksnim podacima

- Intuitivna definicija:
  - dozvoljene su relacije tamo gdje su bile dozvoljene samo atomarne vrijednosti - relacije unutar relacija
  - zadržana matematička osnova relacijskog modela
  - narušena prva normalna forma (1NF)

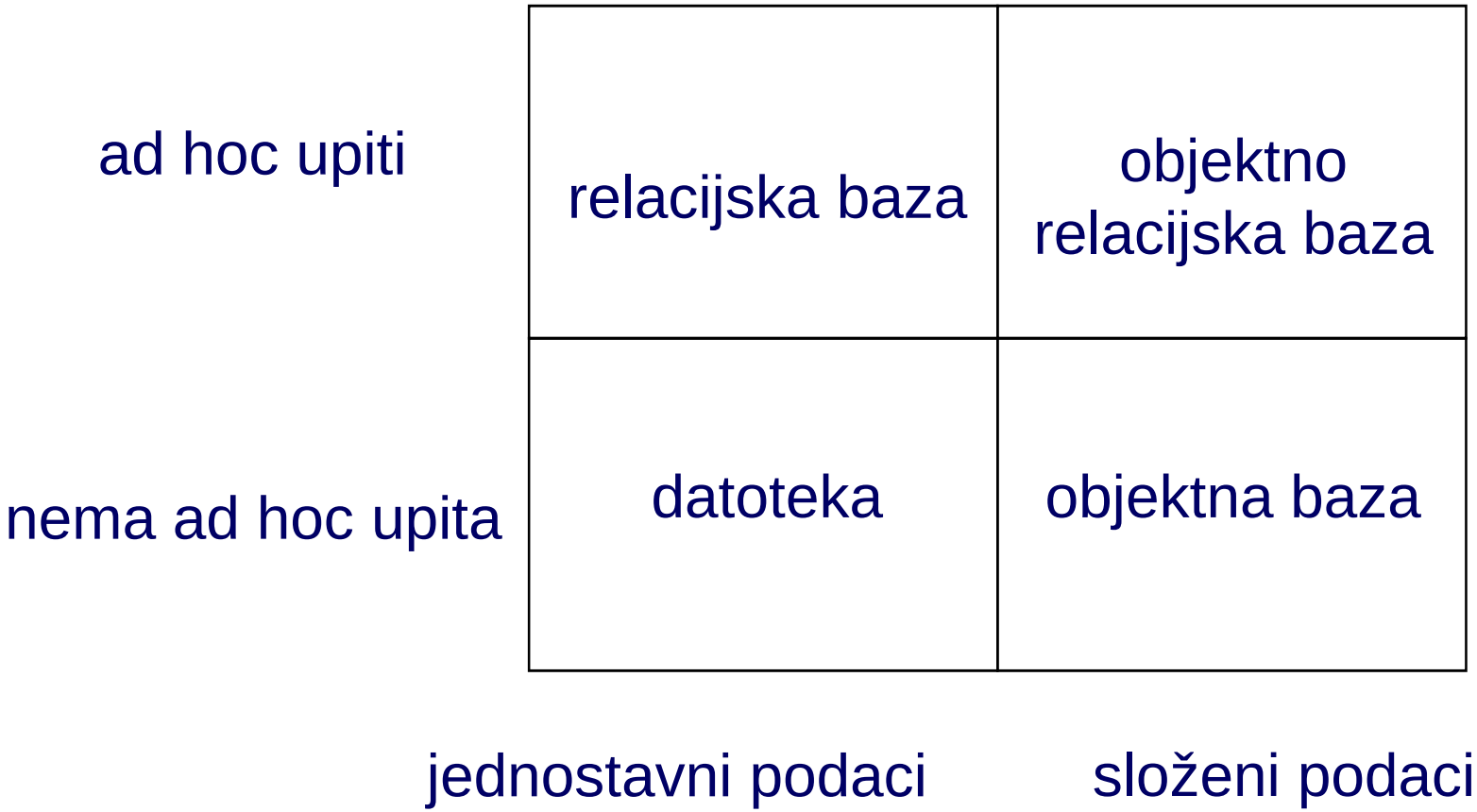
Objektno-relacijski model podataka (1)

- Temeljen je na relacijskom modelu podataka.
- Sačuvane su relacijske karakteristike, kao što je deklarativan pristup podacima.
- Proširuje relacijski model uvođenjem objektne orijentacije i konstrukcija koje omogućuju rukovanje s novim tipovima podataka.
- Dozvoljava da atributi u n-torkama imaju složene vrijednosti, uključujući i ugnježdene relacije.
- Znatno su proširene mogućnosti modeliranja podataka.
- Zadržana je kompatibilnost s postojećim relacijskim jezicima.
- Ne postoji jedinstveni model - modeli se razlikuju se po tome koliko objektnog proširenja uključuju.

Objektno-relacijski model podataka (2)

- Proširenja relacijskog modela:
  - Apstraktni tipovi podataka (objektni tipovi, strukturirani korisnički-definirani tipovi...)
  - Identifikatori objekta i reference
  - Metode za objektne tipove, ugnježđivanje
  - Korisnički definiran CAST
  - Objektne tablice
  - Nasljeđivanje tipova i tablica
  - Ugnježđene relacije (složeni atributi, kolekcije)

DBMS Matrix



M. Stonebreaker, D. Moore:
Object-relational DBMSs – the next great wave, Morgan Kaufmann, 1996

SQL

- objedinjuje funkcije jezika za definiciju podataka (DDL) i jezika za rukovanje podacima (DML)
- razvoj započeo 70-tih godina
  - IBM San José Research Laboratory (California, USA)

- *Structured Query Language* je standardni jezik relacijskih baza podataka (*database language*)
  - 1986. godine - SQL-86 ili SQL1 (prva verzija standarda)
  - 1992. godine - SQL-92 ili SQL2
  - 1999. godine - SQL:1999
  - 2003. godine - SQL:2003
- proizvođači komercijalnih sustava često ugrađuju i svoje nestandardne DDL i DML naredbe
  - programski kod postaje neprenosiv između različitih SQL sustava
  - otežava se usaglašavanje oko budućih standarda

SQL: objektno-relacijske mogućnosti

- većina objektno-relacijskih koncepata uključena u SQL:1999
- SQL:2003 uvodi dodatna proširenja
- reakcija na rastuću popularnost ODBMS i uključivanje objektno–relacijskih aspekata u komercijalne RDBMS
- posebna važnost: standard dozvoljava proširenja koja omogućavaju nove primjenu (multimedia, prostorni podaci, upravljanje dokumentima, polustrukturirani podaci i WWW sadržaji, ...)
- objektno-relacijski koncepti nisu potpuno implementirani niti u jednom SUBP
  - različiti SUBP-ovi koriste različite pristupe
- neke mogućnosti ugrađene u sve glavne komercijalne SUBP

SQL: tipovi podataka (1)

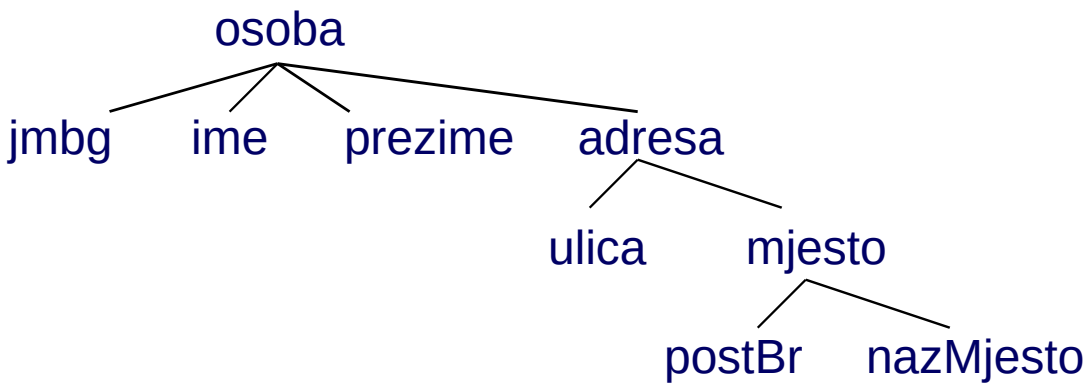
- SQL:1999 podjela tipova podataka
  - unaprijed definirani tipovi (*predefined types*)
    - integer, float, character, boolean, datetime, interval ...
  - izgrađeni tipovi (c*onstructed types*)
    - izgrađeni atomarni tipovi (*constructed atomic types*)
      - referenca (*reference*)
    - izgrađeni kompozitni tipovi (*constructed composite types*)
      - kolekcije (*collection*): polje (*array*), multiset
      - *row*
  - korisnički definirani tipovi (*user-defined types*)
    - *distinct type*
    - strukturirani tip (*structured type*)

SQL: tipovi podataka (2)

- Korišteni primjer: osobe na visokom učilištu
- relacija *osoba* nije u 1NF

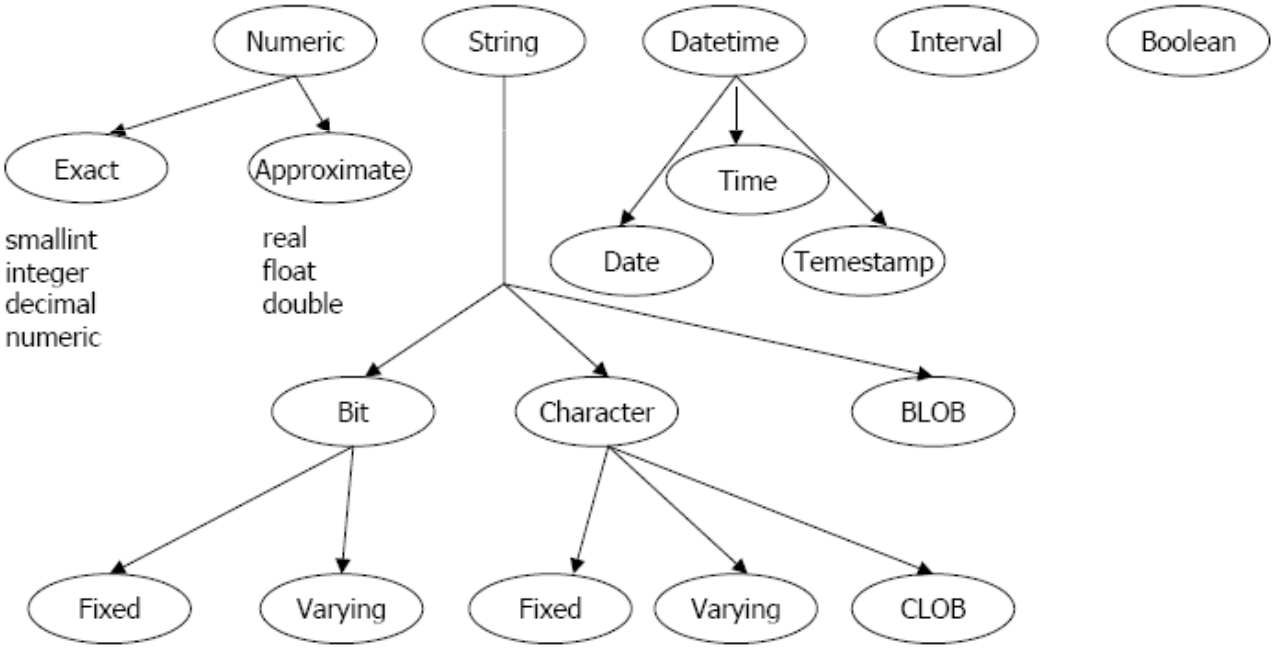
osoba

jmbg	ime	prezime	adresa		
			ulica	mjesto	
				postBr	nazMjesto
1102989560037	Hrvoje	Novak	Ilica 25	10000	Zagreb
0608989755015	Ana	Kolar	Marmontova 18	21000	Split



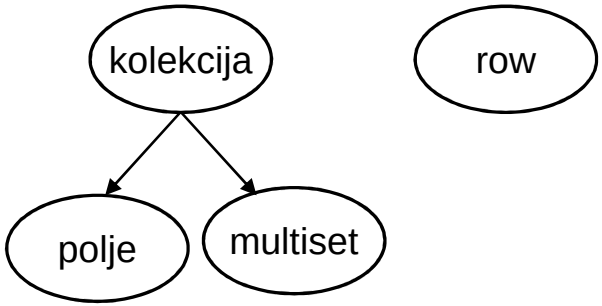
Unaprijed definirani tipovi

- naziv tipa definiran standardom
- atomaran tip - vrijednost nije izgrađena od vrijednosti drugih podatkovnih tipova



Izgrađeni tipovi

- naziv tipa definiran standardom
- dijele se na:
  - atomarne
    - referenca (*reference type - REF type*)
      - tip čija vrijednost pokazuje na lokaciju na kojoj je pohranjena vrijednost referenciranog tipa
      - mogu pokazivati samo na n-torke tablica temeljene na strukturiranom tipu (*tipizirane tablice*)
  - kompozitne
    - svaka vrijednost je složena od jedne ili više vrijednosti koje mogu pripadati različitim podatkovnim tipovima



- specificira se pomoću konstruktora tipa (ARRAY, REF, ROW)

ROW tip (1)

- izgrađeni kompozitni tip
- sekvenca od jednog ili više elemenata (*field*)
- element se definira parom (*ime_elementa*, *tip_podatka*)

```
ROW(postBr      CHAR(5),
    nazMjesto   VARCHAR(20)
    )
```

- može biti korišten za definiranje složenih atributa:

```
CREATE TABLE osoba (
        jmbg      CHAR(13),
        ime       VARCHAR(25),
        prezime   VARCHAR(25),
        adresa    ROW(ulica VARCHAR(50),
                      mjesto ROW (postBr      CHAR(5),
                                  nazMjesto   VARCHAR(40)
                                  )
                      )
);
```

ROW tip (2)

- vrijednost ROW tipa sadrži po jednu vrijednost za svaki element tog ROW tipa
- **v**rijednost elementa mora odgovarati definiranom podatkovnom tipu tog elementa
- za pridruživanje vrijednosti elementima koristi se ROW konstruktor
- u ROW konstruktoru može biti navedena lista vrijednosti, ali vrijednosti mogu biti i rezultat upita
  - npr. ROW tipu:

```
ROW(postBr      CHAR(5),
    nazMjesto   VARCHAR(20)
    )
```

može se pridružiti vrijednost ('10000','Zagreb') pomoću:

```
ROW('10000', 'Zagreb')
```

SQL: ROW tip (3)

- upisivanje zapisa u relaciju *osoba*:

```
CREATE TABLE osoba (
 jmbg    CHAR(13),
 ime     VARCHAR(25),
 prezime VARCHAR(25),
 adresa  ROW(ulica VARCHAR(50),
             mjesto ROW (postBr    CHAR(5),
                         nazMjesto VARCHAR(40)
                        )
                   )
);
```

```
INSERT INTO osoba VALUES(
         '1102989560037',
         'Hrvoje',
         'Novak',
          ROW('Ilica 25',
               ROW ('10000','Zagreb')
             )
);
```

- za pristup elementima koji su dio složenog atributa koristi se *dot* notacija:

```
SELECT o.adresa.mjesto.postBr
  FROM osoba o
 WHERE o.adresa.ulica = 'Ilica 25';
```

SQL: Korisnički definirani tipovi

- Korisnički definirani tipovi (*user-defined types* – UDT)
  - nazivaju ih i *apstraktnim tipovima podataka* (*abstract data types*)
  - u biti definicija klase, sa svojom strukturom i metodama
  - definirani i imenovani od strane korisnika

(1) **distinct tip**

- temeljen na unaprijed definiranom tipu koji se naziva izvorni tip (*source type*)
- nema nasljeđivanja tipova

(2) **strukturirani tip**

- iskazan kao lista atributa
- podržana hijerarhija tipova

SQL sintaksa za kreiranje korisnički definiranih tipova

```
CREATE TYPE <user-defined type body>

<user-defined type body> ::= <user-defined type name> [UNDER <supertype name>]
                             [AS <representation>] [[NOT] INSTANTIABLE]
                             [NOT] FINAL
                             [REF IS SYSTEM GENERATED | REF USING <predefined type> |
                                  REF FROM <attribute name> [{,<attribute name> }...]]
                             [<method specification list>]

<representation> ::= <predefined type> | <member list>

<member list> ::= (<attribute definition> [{,<attribute definition> }...])

<attribute definition> ::= <attribute name> {<data type> | <collection type>}
                           [<reference scope check>] [DEFAULT <default value>]

<data type> ::= <predefined type> | <reference type>

<collection type> ::= <data type> ARRAY [<unsigned integer>]   /* [] dio sintakse */

<method specification list> ::= <method specification> [{,<method specification> }... ]

<method specification> ::=
    <partial method specification> | <overriding method specification>

<overriding method specification> ::= OVERRIDING <partial method specification>

<partial method specification> ::= [CONSTRUCTOR] METHOD <method name>
                                    <SQL parameter declarations> RETURNS <data type>
```

Strukturirani tipovi (1)

- imenovani, korisnički-definiran podatkovni tip
- može imati proizvoljno složenu strukturu
- vrijednost strukturiranog tipa sastoji se od određenog broja vrijednosti atributa (*attribute values*)
- atribut strukturiranog tipa
  - "pohranjeni podatak"
  - za atribut se specificira ime i podatkovni tip

- pohranjeni podatak ⇒ stanje ⇒ atributi
- ponašanje ⇒ semantika ⇒ metode

Strukturirani tipovi (2)

- Definicija strukturiranog tipa *mjestoT*:

```
CREATE TYPE mjestoT AS (
        postBr     CHAR(5),
        nazMjesto  VARCHAR(40)
)
NOT FINAL;
```

  - ako je u definiciji tipa navedena ključna riječ **NOT FINAL**, moguće je kreirati podtip tog tipa
    - trenutno ograničenje: u definiciji strukturiranog tipa, mora se uvijek navesti **NOT FINAL**

Strukturirani tipovi: Načini korištenja (1)

- dva načina korištenja strukturiranih tipova:
  1. kao tip atributa relacije
     - za modeliranje vrsta činjenica o entitetima (npr: slika, audio, video, vremenski nizovi, točka, linija,…)
  2. kao tip n-torke relacije
     - tipovi i funkcije za n-torke relacije (npr: zaposlenici, odsjeci (zavodi), visoka učilišta, studenti, …)
     - za modeliranje veza između entiteta i ponašanje entiteta
- strukturirani tipovi se mogu koristiti:
  - u SQL naredbama, svugdje gdje se mogu koristiti ostali (unaprijed definirani) tipovi
    - tip atributa drugog strukturiranog tipa
    - tip parametra funkcija, metoda i procedura
    - tip SQL varijable
    - tip domena ili atributa u tablicama
  - za definiranje tablica i pogleda

Strukturirani tipovi: Načini korištenja (2)

- strukturirani tip se može navesti kao tip atributa u:
  - CREATE TYPE naredbi drugog strukturiranog tipa
  - CREATE TABLE naredbi
- kreiranje tablica sa složenim atributima

```
CREATE TYPE mjestoT AS (
        postBr     CHAR(5),
        nazMjesto  VARCHAR(40)
) NOT FINAL;
CREATE TYPE adresaT AS (
        ulica      VARCHAR(50),
        mjesto     mjestoT
) NOT FINAL;
CREATE TABLE osoba (
        jmbg       CHAR(5)
        ime        VARCHAR(25),
        prezime    VARCHAR(25),
        adresa     adresaT
);
```

Strukturirani tipovi: Metode

- SQL razlikuje tri tipa pohranjenih procedura:
  - Funkcija:
    - samo ulazni parametri (izlazni je vraćen kao "vrijednost" funkcije)
    - poziva se korištenjem funkcijske notacije: **`imeFunkcije(parametri)`**
    - više funkcija istog imena, isti broj parametara – razlikuju se prema tipu argumenata
    - nisu vezane uz strukturirani tip
  - Metoda:
    - specijalni slučaj funkcije
    - čvrsto vezana uz jedan strukturirani tip
    - prvi parametar je implicitan: tip parametra je strukturirani tip uz kojeg je metoda vezana
  - Procedura:
    - ulazni i izlazni parametri
    - pozivanje CALL naredbom
    - ako imaju isto ime, moraju imati različit broj parametara/argumenata
    - nisu vezane uz strukturirani tip

Strukturirani tipovi: Ugrađene metode (1)

- implementacija strukturiranog tipa sakrivena je od korisnika
  - s tipom se manipulira samo kroz metode koje su na njemu definirane.
  - aplikacije *svemu* pristupaju kroz funkcionalno sučelje
    - promjena implementacije ne utječe na aplikacije ukoliko sučelje ostane nepromijenjeno
  - vrijednosti atributa su učahurene (*encapsulated*) - može im se pristupiti pozivanjem *observer i mutator* funkcija

- za strukturirane tipove postoji tri tipa ugrađenih metoda:
  - *constructor* funkcija
  - *observer* funkcije
  - *mutator* funkcije

Strukturirani tipovi: Ugrađene metode (1)

- *constructor* funkcija
  - koristi se za kreiranje instanci tipa
  - istog imena kao tip
  - implicitno definirana u trenutku kreiranja tipa
    - funkcija bez argumenata
    - vraća pretpostavljene (*default*) ili NULL vrijednosti atributa
  - uvijek mora biti pozvana uz korištenje izraza *new*
- *observer* i *mutator* funkcije
  - jedna za svaki atribut strukturiranog tipa
  - istog imena kao atribut
  - *observer* - vraća vrijednosti atributa strukturiranog tipa
  - *mutator* - modificira vrijednosti atributa strukturiranog tipa
  - poziva se korištenjem dot notacije (**varijabla.imeFunkcije**)

Strukturirani tipovi: Ugrađene metode (3)

- Primjer:
  - sekvenca kôda iz SQL rutine koja koristi *construcor* funkciju i *mutator* funkcije za kreiranje instance tipa mjestoT

```
BEGIN
   DECLARE novoMjesto mjestoT;
   /* poziv constructor funkcije */
   SET novoMjesto = new mjestoT();
   /* poziv mutator funkcija */
   novoMjesto.postBr('10000');
   novoMjesto.nazMjesto('Zagreb');
   INSERT INTO mjesto VALUES(novoMjesto);
END
```

  - ovdje nova instanca tipa nije *objekt*
    - da bi bila objekt, mora biti korištena u kombinaciji s *tipiziranim tablicama*

Strukturirani tipovi: Ugrađene metode (4)

- Primjer:
  - korištenje *observer* funkcije za dohvat vrijednosti atributa strukturiranog tipa

```
CREATE TYPE mjestoT AS (
        postBr     CHAR(5),
        nazMjesto  VARCHAR(40)
) NOT FINAL;
CREATE TYPE adresaT AS (
        ulica      VARCHAR(20),
        mjesto     mjestoT
) NOT FINAL;
CREATE TABLE osoba (
        jmbg       CHAR(5)
        ime        VARCHAR(25),
        prezime    VARCHAR(25),
        adresa     adresaT
);
```

```
SELECT o.adresa.mjesto.postBr
  FROM osoba o
 WHERE o.adresa.ulica = 'Ilica 25';
```

Strukturirani tipovi: Korisnički-definirane metode (1)

- Korisnički-definirane metode nad strukturiranim tipovima
  - korisnici mogu definirati vlastite metode
  - metode se specificiraju kao dio definicije tipa

```
CREATE TYPE zaposlenikT (
        imeZaposlenik   VARCHAR(15),
        prezZaposlenik  VARCHAR(15),
        placa           INTEGER) NOT FINAL
METHOD placaUzPovisicu(postotak INTEGER) RETURNS INTEGER;
```

  - tijelo metode definira se odvojeno

```
CREATE METHOD placaUzPovisicu (postotak INTEGER) RETURNS INTEGER
       FOR zaposlenikT
  BEGIN
     RETURN (100 + postotak) * SELF.placa / 100;
  END
```

    - korištenjem ključne riječi SELF može se pristupiti vrijednosti implicitnog parametra u implementaciji metode (instanca tipa za kojeg je metoda definirana)

Strukturirani tipovi: Korisnički-definirane metode (2)

- Primjer korištenja korisnički-definirane metode
  - upit kojim se za svakog zaposlenika vraća prezime i iznos plaće uz navedeni postotak povišice na plaću:

```
{ CREATE TABLE zaposlenik OF zaposlenikT; }

SELECT prezZaposlenik
     , placaUzPovisicu (7)
  FROM zaposlenik
```

Strukturirani tipovi: Korisnički-definirane metode (3)

- Primjer: nadjačavanje *constructor* funkcije
  - nadjačana *constructor* funkcija koristi se za postavljanje vrijednosti atributa u trenutku kreiranja instance tipa
  - u definiciji tipa mora biti specificirana ključna riječ OVERRIDING

```
CREATE TYPE mjestoT AS (
        postBr     CHAR(5),
        nazMjesto  VARCHAR(40)) NOT FINAL
OVERRIDING constructor METHOD /* novi konstruktor s parametrima */
   mjestoT(postBr CHAR(5), nazMjesto VARCHAR(40)) RETURNS mjestoT;
```

```
CREATE METHOD mjestoT (pPostBr CHAR(5), pNazMjesto VARCHAR(40))
       RETURNS mjestoT FOR mjestoT
  BEGIN
     SET SELF.postBr = pPostBr;
     SET SELF.nazMjesto = pNazMjesto
     RETURN SELF;   /* modificirana instanca tipa mjestoT */
  END
```

```
...
DECLARE novoMjesto mjestoT;
SET novoMjesto = new mjestoT ('10000', 'Zagreb');
...
```

Strukturirani tipovi: Tipizirane tablice (1)

- ***tipizirana tablica*** (*typed table*) - tablica definirana na temelju strukturiranog tipa
  - atributi tipa postaju kolone tablice
  - dodatna kolona koja sadrži jedinstveni identifikator objekta, odnosno *referencu* (*self-referencing column)*
    - identifikator objekta je jedinstven samo unutar tipizirane tablice
- SQL sintaksa za kreiranje tipizirane tablice:

```
CREATE TABLE <table name> OF <user-defined type>
   [UNDER <supertable name>][ <table element list>]

<table element list> ::= (<table element> [{,<table element>}...])

<table element> ::= <table constraint>
   | <self-referencing column specification> | <column options>

<self-referencing column specification > ::=
   REF IS <self-referencing column name> <reference generation>

<reference generation> ::= SYSTEM GENERATED | USER GENERATED | DERIVED

<column options> ::= <column name> WITH OPTIONS <column option list>

<column option list> ::= [SCOPE <table name>[<reference scope check>]]
                [DEFAULT <default value>] [ <column constraint>... ]
```

Strukturirani tipovi: Tipizirane tablice (2)

- Primjer: kreiranje tablice *mjesto* na temelju tipa *mjestoT*

1.
```
CREATE TYPE mjestoT AS (
      postBr     CHAR(5),
      nazMjesto  VARCHAR(40))
   INSTANTIABLE
   NOT FINAL
   REF IS SYSTEM GENERATED;
```

  - mogućnost direktnog kreiranja instance tipa
    - INSTANTIABLE klauzula u definiciji tipa - za tip postoji *constructor* funkcija i korisnik može direktno kreirati instance tipa
      - nužno za tip koji se koristi zajedno s tipiziranom tablicom
    - NOT INSTANTIABLE klauzula u definiciji tipa - za tip ne postoji *constructor* funkcija
  - način generiranja vrijednosti reference na objekt (*object reference*)
    - definiran REF IS klauzulom u definiciji tipa
      - SYSTEM GENERATED - generiranje obavlja sustav
      - USING - kreiranje vrijednosti reference obavlja korisnik
      - FROM - korisnik specificira listu atributa iz strukturiranog tipa, koja će biti korištena za izvođenje jedinstvene reference na objekt

Strukturirani tipovi: Tipizirane tablice (3)

2.

```
CREATE TABLE mjesto OF mjestoT
(PRIMARY KEY (postBr),
 REF IS mjestoID SYSTEM GENERATED);
```

- OF klauzula - sadrži strukturirani tip nad kojim je definirana tipizirana tablica
- REF IS klauzula:
  - način generiranja vrijednosti reference - mora biti konzistentan s načinom generiranja strukturiranog tipa
    - SYSTEM GENERATED, USER GENERATED, DERIVED
  - dodjeljuje ime dodatnoj koloni
- Primjer: upisivanje zapisa u tablicu *mjesto*

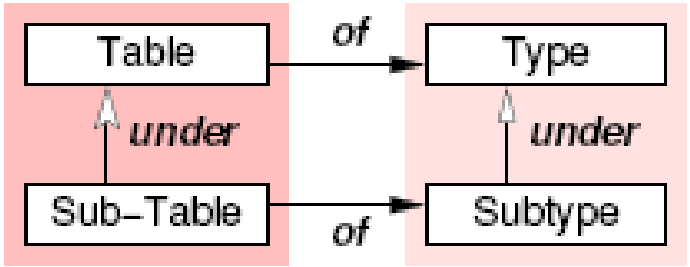
```
INSERT INTO mjesto VALUES ('10000', 'Zagreb');
```

mjesto

mjestoID	postBr	nazMjesto
1023456734	10000	Zagreb

Nasljeđivanje (*Inheritance*)

- nasljeđivanje tipova
  - podtip nasljeđuje atribute i metode nadređenog tipa
- nasljeđivanje tablica
  - moguće je samo za tipizirane tablice
  - odgovara E-R pojmu specijalizacije/generalizacije
  - omogućava više tipova istog objekta, dozvoljavajući istovremeno postojanje entiteta u više od jedne tablice



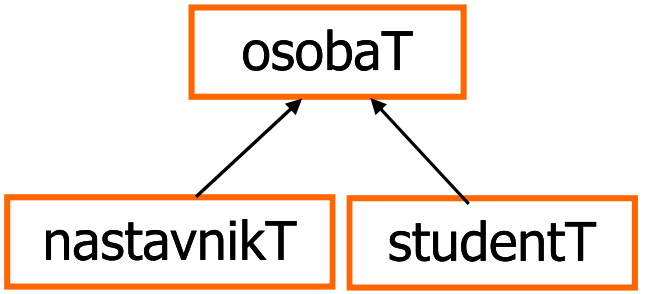
- nije dozvoljeno višestruko nasljeđivanje
  - svaki podtip/podtablica ima samo jednu temeljnu (korijensku) nadklasu/nadtablicu

Nasljeđivanje tipova (1)

- podtip nasljeđuje atribute i metode nadređenog tipa
- iz nekog tipa može biti kreiran drugi tip, ako u definiciji tipa navedena ključna riječ NOT FINAL
- podtip može redefinirati učinak metode ponovnim deklariranjem metode, korištenjem nadjačavanja metoda:
  - u definiciji podtipa, prilikom deklaracija metoda, umjesto **`METHOD`** koristi se **`OVERRIDING METHOD`**

Nasljeđivanje tipova (2)

- Primjer:
  - čuvaju se dodatne informacije o osobama koje su studenti i osobama koje su nastavnici
  - za definiranje tipova *studentT* i *nastavnikT* moguće je koristiti nasljeđivanje




```
CREATE TYPE osobaT AS (
        jmbg        CHAR(5)
        prezime     VARCHAR(25))
        INSTANTIABLE NOT FINAL
        REF IS SYSTEM GENERATED
METHOD starost(jmbg CHAR(5)) RETURNS INTEGER;

CREATE TYPE studentT UNDER osobaT AS (
        razinaStudij CHAR(20)
        zavod        VARCHAR(100))
        INSTANTIABLE NOT FINAL;

CREATE TYPE nastavnikT UNDER osobaT AS (
        placa       INTEGER
        zavod       VARCHAR(100))
        INSTANTIABLE NOT FINAL;
```

- *studentT* i n*astavnikT* su podtipovi tipa *osobaT*
- *osobaT* je nadređen tipovima *studentT* i nastavnik*T*
- *studentT* i n*astavnikT* od *osobaT* nasljeđuju:
  - atribute *jmbg* i *prezime*
  - metodu *starost*

Nasljeđivanje tablica (1)

- omogućeno je samo za tipizirane tablice
- tipovi podtablica moraju biti podtipovi tipa nadređene tablice
- svaki atribut koji postoji u nadređenoj tablici postoji i u podtablicama
- svaka n-torka iz podtablice implicitno postoji i u nadređenoj tablici
  - n-torka u podtablici odgovara n-torki u nadređenoj tablici, ako ima iste vrijednosti svih naslijeđenih atributa
  - korespondentne n-torke u podtablici i nadređenoj tablici predstavljaju isti entitet

Nasljeđivanje tablica (2)

- Primjer:
  - kreiranje tablica *student* i *nastavnik* kao podtablica tablice *osoba*

```
CREATE TABLE osoba OF osobaT
(PRIMARY KEY (jmbg),
 REF IS osobaID SYSTEM GENERATED;

CREATE TABLE student OF studentT
       UNDER osoba;

CREATE TABLE nastavnik OF nastavnikT
       UNDER osoba
```

- prethodno mora biti definirana hijerarhija tipova
- atributi *jmbg* i *prezime* postoje i u tablicama *student* i *nastavnik*
- svaka n-torka iz *student* i *nastavnik* implicitno postoji u tablici *osoba*

Pravila vezana uz hijerarhiju tipova i tablica (1)

- nije dozvoljeno višestruko nasljeđivanje
  - svaki podtip/podtablica ima samo jednu temeljnu (korijensku) nadklasu/nadtablicu
  - radne verzije SQL:1999 standarda omogućavale su višestruko nasljeđivanje, ali je konačnoj verziji standarda ispuštena
    - Primjer: za pohranu informacija o asistentima, koji istovremeno održavaju nastavu ali su i studenti, predlagan je oblik naredbe za definiciju tipa:

```
CREATE TYPE asistentT UNDER studentT, nastavnikT ...
```

- da bi se izbjegao konflikt zbog dva pojavljivanja atributa *zavod*, moguće je atribut preimenovati

```
CREATE TYPE asistentT
       UNDER studentT WITH (zavod AS zavodStudent)
           , nastavnikT WITH (zavod AS zavodNastavnik)...
```

Pravila vezana uz hijerarhiju tipova i tablica (2)

- primarni ključ definira se samo za temeljnu (korijensku) nadtablicu, a nasljeđuju ga sve njezine podtablice
- REF IS klauzula može biti definirana samo na razini temeljnog (korijenskog) nadtipa/nadtablice. Identifikator objekta nasljeđuju sve podtablice
- integritetska ograničenja koja se navode u definiciji tablice, smiju biti navedena samo uz atribute koje je u tablicu uveo strukturirani tip nad kojim je tablica definirana, a ne na naslijeđenim atributima
- hijerarhiju tipova moguće je definirati neovisno o hijerarhiji tablica
- NOT INSTANTIABLE klauzula može biti korištena u hijerarhiji tipova koji nisu vezani uz tipizirane tablice
- svi tipovi vezani uz hijerarhiju tipiziranih tablica moraju biti definirani kao INSTANTIABLE, kako bi bilo podržano korištenje INSERT naredbi na tipiziranim tablicama

Hijerarhija tablica: upisivanje n-torki

- u INSERT naredbi navode se vrijednosti i za naslijeđene atribute
- vrijednost identifikatora objekta se automatski generira prilikom izvođenja INSERT naredbe

```
INSERT INTO osoba VALUES ('111', 'Kolar');
INSERT INTO osoba VALUES ('222', 'Novak');
INSERT INTO student VALUES ('333', 'Horvat', 'preddiplomski', 'ZPR');
INSERT INTO student VALUES ('444', Novak', 'diplomski', 'ESA');
INSERT INTO nastavnik VALUES ('555', 'Jurak', 5000, 'ZPM');
```

- najspecifičnija tablica (*most-specific table*) n-torke
  - tablica u koju je n-torka direktno upisana
- najspecifičniji tip (*most-specific type*)
  - odgovara tipu tablice u koju je n-torka direktno upisana

Hijerarhija tablica: dohvat n-torki (1)

- upit postavljen nad nadređenom tablicom:
  - pristupa samo onim atributima koji postoje u nadređenoj tablici:
  - n-torke koje su rezultat upita mogu uključivati:
    - n-torke direktno upisane u nadređenu tablicu i n-torke upisane u njezine podtablice ili
    - samo n-torke direktno upisane u nadređenu tablicu
      - korištenje ključne riječi `ONLY` uz naziv nadređene tablice u FROM dijelu SELECT naredbe

Hijerarhija tablica: dohvat n-torki (2)

- u hijerarhiju tablica upisane su n-torke sljedećim naredbama:

```
INSERT INTO osoba VALUES ('111', 'Kolar');
INSERT INTO osoba VALUES ('222', 'Novak');
INSERT INTO student VALUES ('333', 'Horvat', 'preddiplomski', 'ZPR');
INSERT INTO student VALUES ('444', Novak', 'diplomski', 'ESA');
INSERT INTO nastavnik VALUES ('555', 'Jurak', 5000, 'ZPM');
```

- upiti koji vraćaju n-torke direktno upisane u nadređenu tablicu i n-torke upisane u njezine podtablice:

```
SELECT jmbg
     , prezime
  FROM osoba
```

jmbg	prezime
111	Kolar
222	Novak
333	Horvat
444	Novak
555	Jurak

```
SELECT jmbg
     , prezime
  FROM nastavnik
```

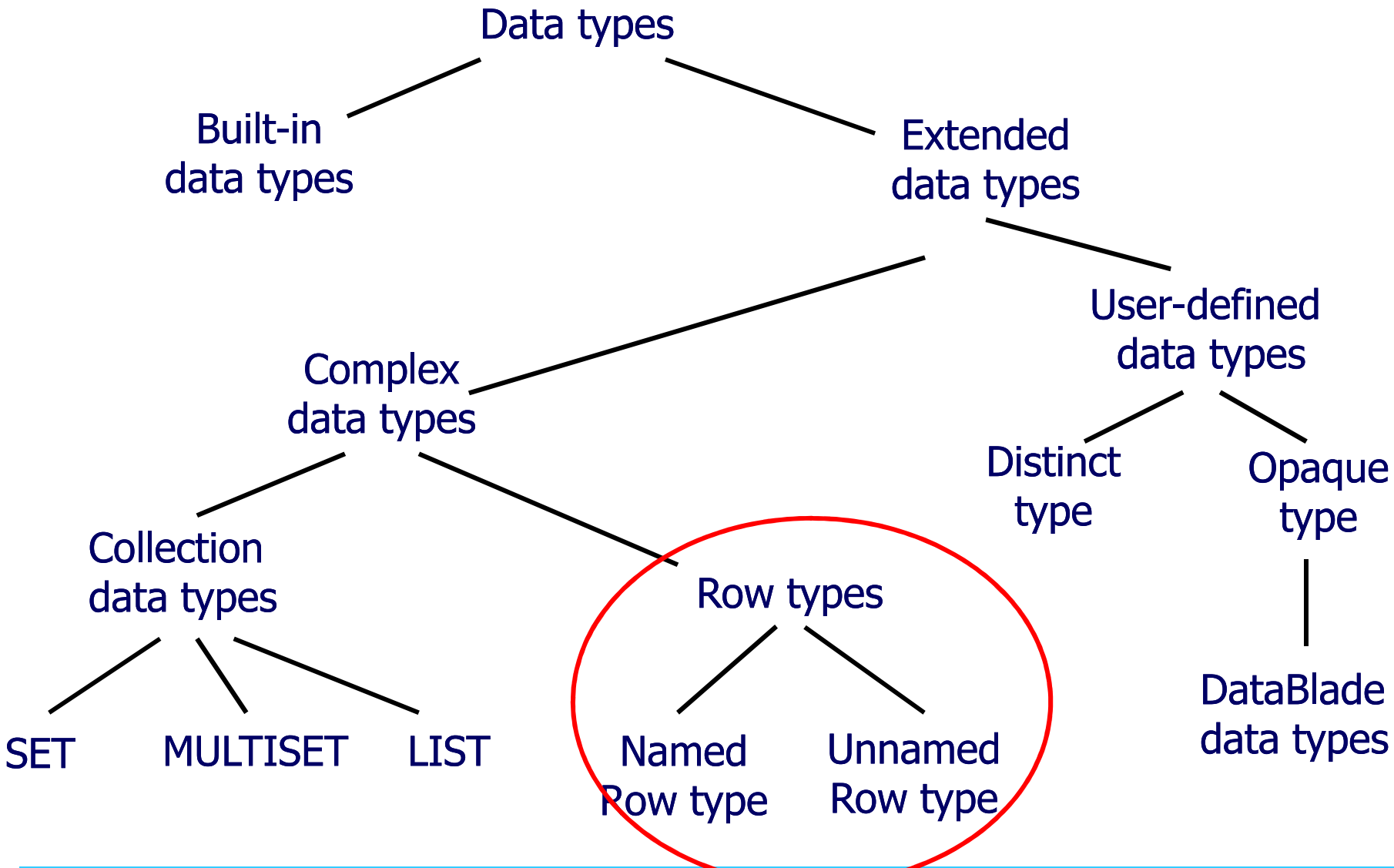
jmbg	prezime
555	Jurak

- upit koji vraća samo n-torke direktno upisane u nadređenu tablicu :

```
SELECT jmbg
     , prezime
  FROM ONLY(osoba)
```

jmbg	prezime
111	Kolar
222	Novak

IBM Informix implementacija



Složeni tip podatka (*Complex data type*)

- definira se na temelju drugih tipova podataka. Gradivi elementi mogu biti bilo koji tip podatka, uključujući i druge složene tipove podataka

ROW tip

- struktura koja se sastoji od atributa (*struct*). Pojedini atributi mogu biti bilo kojeg tipa

mjestoT

postBr	INTEGER
nazMjesto	CHAR(20)
slikaGlavnogTrga	jpegTip

osobaT

ime	CHAR(20)
prezime	CHAR(20)
mjestoRodjenja	mjestoT

Imenovani ROW tip (*Named row type*) (1)

- identificiran je svojim nazivom
- u bazi podataka ne smiju postojati dva takva tipa s istim imenom
- dva tipa ove vrste ne mogu biti jednaka niti u kojem slučaju (unatoč jednakim imenima atributa i jednakim tipovima)
- definicija je pohranjena u rječniku podataka u relaciji *sysxtdtypes*
- definiranje relacije na temelju ROW tipa, hijerarhijske veze tipova (a time i hijerarhije relacija) moguće je jedino uz korištenje imenovanih ROW tipova

Imenovani ROW tip (2)

- kreiranje:

```
CREATE ROW TYPE rowTypeName
       (fieldName fieldType [NOT NULL], ...)
       [UNDER superRowTypeName]
```

- uništavanje:

```
DROP ROW TYPE rowTypeName RESTRICT
```

Restrict: odbij ovu operaciju ako je na temelju ovog ROW tipa definiran neki drugi ROW tip **(**ili relacija**)**

- Primjer:

```
CREATE ROW TYPE mjestoT (postBr    INTEGER NOT NULL
                       , nazMjesto CHAR(50));
```

```
DROP ROW TYPE mjestoT RESTRICT;
```

Imenovani ROW tip (3)

- sljedeća dva ROW tipa ne smatraju se jednakim:

```
CREATE ROW TYPE mjestoStanT (
        postBr    INTEGER
      , nazMjesto CHAR(50)
);

CREATE ROW TYPE mjestoRodT (
        postBr    INTEGER
      , nazMjesto CHAR(50)
);
```

Imenovani ROW tip – korištenje (1)

- kreiranje relacije čije su n-torke imenovanog ROW tipa

```
CREATE TABLE mjesto OF TYPE mjestoT;
```

- kreiranje relacije čiji je atribut tipa imenovanog ROW tipa

```
CREATE TABLE osoba (
     jmbg         CHAR(13)
   , mjestoStan   mjestoT);
```

- moguće je "dublje ugnijezditi" ROW tipove (npr. ROW tip čiji je atribut ROW tipa čiji je atribut ROW tipa, itd.)

```
CREATE ROW TYPE osobaT (ime CHAR(20)
                      , mjestoStan mjestoT
                      , mjestoRod  mjestoT);
CREATE ROW TYPE poduzeceT (naziv CHAR(20), direktor osobaT);
```

```
DROP ROW TYPE mjestoT RESTRICT → ERROR
```

Imenovani ROW tip – korištenje (2)

- ime relacije ili *alias* ime relacije navedeno u SELECT listi → ROW objekt

```
CREATE ROW TYPE mjestoT (postBroj INTEGER
                       , nazMjesto CHAR(20));

CREATE TABLE mjesto OF TYPE mjestoT;
```

```
SELECT mjesto FROM mjesto
SELECT m FROM mjesto m
```

→ **objekti tipa mjestoT jer je relacija definirana pomoću imenovanog ROW tipa**

```
CREATE TABLE drzava (oznDrzava  CHAR(2)
                   , nazDrzava  CHAR(20));
```

```
SELECT drzava FROM drzava
SELECT d FROM drzava d
```

→ **objekti tipa unnamed ROW (CHAR(2), CHAR(20))**

"dot" notacija za pristup elementima ROW objekta

```
CREATE ROW TYPE tDrzava
    (oznDrzava CHAR(2), nazDrzava CHAR(20));
CREATE ROW TYPE mjestoT
    (postBroj INTEGER, nazMjesto CHAR(20), drzava tDrzava);
```

```
CREATE TABLE mjesto OF TYPE mjestoT;
CREATE TABLE drzava OF TYPE tDrzava;
```

```
SELECT mjesto.drzava FROM mjesto
SELECT mjesto.drzava.nazDrzava FROM mjesto
```

→ objekti tipa tDrzava
→ objekti tipa tCHAR(20)

```
INSERT INTO mjesto VALUES (
   10000
 , 'Zagreb'
 , (SELECT drzava FROM drzava WHERE oznDrzava = 'hr'))
```

→ treća vrijednost koja se upisuje u relaciju mjesto je objekt tipa tDrzava

Instanciranje imenovanog ROW objekta

- sintaksa za instanciranje objekta (*literal*) ROW tipa

```
ROW (literal, ...)
```

```
ROW ('hr', 'Hrvatska')
```

→ instancira se neimenovani ROW objekt

```
ROW ('hr', 'Hrvatska')::tDrzava
```

→ imenovani ROW objekt se dobije tek primjenom CAST operatora

- primjer:

```
CREATE ROW TYPE tDrzava (oznDrzava CHAR(2), nazDrzava CHAR(20));
CREATE ROW TYPE mjestoT
    (postBroj INTEGER, nazMjesto CHAR(20), drzava tDrzava);
```

```
CREATE TABLE mjesto OF TYPE mjestoT;
```

```
INSERT INTO mjesto VALUES (
   10000
 , 'Zagreb'
 , ROW ('hr', 'Hrvatska')::tDrzava
);
```

→ u relaciju mjesto se ne može za vrijednost atributa *drzava* ubaciti neimenovani ROW objekt (jer je tip atributa *drzava* imenovani ROW tip). Nužno je korištenje CAST operatora.

Neimenovani ROW tip (*Unnamed row type*)

- slično imenovanim ROW tipu
- sintaksa za deklariranje ROW tipa:

```
ROW (fieldName fieldType, ...)
```

- identificiran je samo svojom strukturom
- ne može se koristiti za definiranje relacija čije su n-torke imenovani tipovi
- dva neimenovana tipa koja imaju jednaku strukturu (pri tome je dovoljno da su korespondentni tipovi atributa jednaki, imena atributa nisu važna) smatraju se jednakim

```
{1.} ROW (postBroj INTEGER, nazMjesto CHAR(20))
{2.} ROW (mbrStud INTEGER, prezStud CHAR(20))
{3.} ROW (postBroj INTEGER, nazMjesto CHAR(30))
```

→ ROW tipovi pod rednim brojevima 1. i 2. se smatraju međusobno jednakim

Neimenovani ROW tip - kreiranje

- eksplicitno deklariranje neimenovanog ROW tipa

```
ROW (fieldName fieldType, ...)
```

```
ROW (postBroj INTEGER, nazMjesto CHAR(20))
```

- instanciranje objekta (literal) koji je neimenovanog ROW tipa

```
ROW (literal, ...)
```

- instanciranjem ROW objekta koji je neimenovanog ROW tipa implicitno se deklarira neimenovani tip

```
ROW(51000, 'Rijeka')
```

→ objektu je pridružen korespondentni neimenovani tip, u ovom slučaju se radi o ROW(INTEGER, VARCHAR)

- ukoliko je potrebno, moguće je upotrijebiti CAST operator

```
ROW(51000, 'Rijeka')::ROW (pbr SMALLINT, naziv CHAR(20))
```

Neimenovani ROW tip - korištenje

- definiranje atributa relacije koji je po tipu ROW

```
CREATE TABLE trokut (
  tocka1   ROW(x FLOAT, Y FLOAT)
, tocka2   ROW(x FLOAT, Y FLOAT)
, tocka3   ROW(x FLOAT, y FLOAT)
);
```

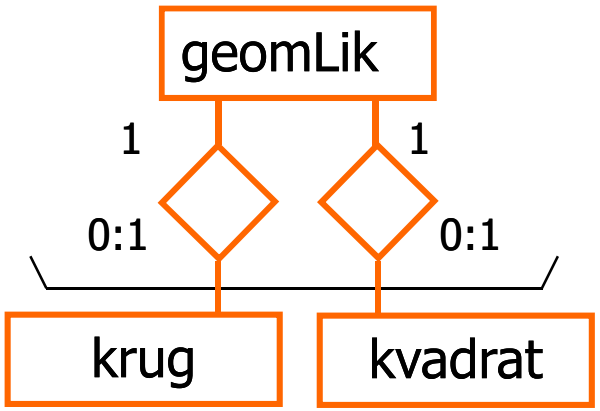
- moguće je i daljnje ugnježđivanje

```
CREATE TABLE osoba (
  jmbg    CHAR(13)
, adresa  ROW(ulica  CHAR(20)
            , mjesto ROW(postBr INTEGER
                       , nazMjesto CHAR(20))
            )
);
```

```
INSERT INTO osoba VALUES (
   '1234567890123'
  , ROW('Ilica 1'
      , ROW(10000, 'Zagreb'))
);
```

→ u ovoj INSERT naredbi nije nužno (ali bi bilo dopušteno) koristiti CAST operator

Hijerarhija relacija

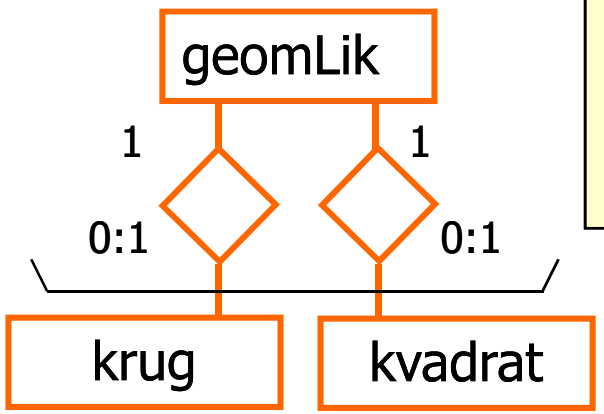



```
CREATE ROW TYPE tGeomLik (id SERIAL
                          , boja CHAR(20));
CREATE ROW TYPE tKrug (radijus FLOAT)
                  UNDER tGeomLik;
CREATE ROW TYPE tKvadrat (duljina FLOAT)
                  UNDER tGeomLik;
```

```
CREATE TABLE geomLik OF TYPE tGeomLik;
CREATE TABLE krug OF TYPE tKrug
                        UNDER geomLik;
CREATE TABLE kvadrat OF TYPE tKvadrat
                        UNDER geomLik;
```

```
INSERT INTO geomLik VALUES (0, 'žuto');
INSERT INTO krug VALUES(0, 'crveno', 2);
INSERT INTO krug VALUES(0, 'zeleno', 4);
INSERT INTO kvadrat VALUES(0, 'crveno', 5);
```

Hijerarhija relacija




```
INSERT INTO geomLik VALUES (0, 'žuto');
INSERT INTO krug VALUES(0, 'crveno', 2);
INSERT INTO krug VALUES(0, 'zeleno', 4);
INSERT INTO kvadrat VALUES(0, 'crveno', 5);
```

```
SELECT *
  FROM geomLik
```

id	boja
1	žuto
4	crveno
2	crveno
3	zeleno

```
SELECT *
  FROM ONLY(geomLik)
```

id	boja
1	žuto

```
SELECT *
  FROM krug
```

id	boja	radijus
2	crveno	2.0000000000
3	zeleno	4.0000000000

Polimorfizam

```
CREATE FUNCTION povrsina (krug tKrug) RETURNING FLOAT;
   RETURN krug.radijus * krug.radijus * 3.1415926;
END FUNCTION;
```

```
CREATE FUNCTION povrsina (kvadrat tKvadrat) RETURNING FLOAT;
   RETURN kvadrat.duljina * kvadrat.duljina;
END FUNCTION;
```

- funkcija *povrsina* se na razne načine evaluira za razne tipove objekata → polimorfizam

```
SELECT geomLik.id, geomLik.boja, povrsina(g)
  FROM geomLik g
```

ERROR: Routine (povrsina) cannot be resolved

```
SELECT id, boja, povrsina(krug) povrsina
  FROM krug
  UNION
SELECT id, boja, povrsina(kvadrat) povrsina
  FROM kvadrat;
```

id	boja	povrsina
2	crveno	12.56637040000
3	zeleno	50.26548160000
4	crveno	25.00000000000