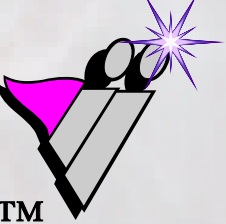


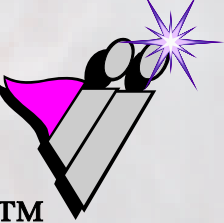
KAKO (NE) KORISTITI OBJEKTNO-RELACIJSKE MOGUĆNOSTI ORACLE DBMS-a

Zlatko Sirotić, dipl.ing.
Istra informatički inženjering d.o.o.
Pula



Teme

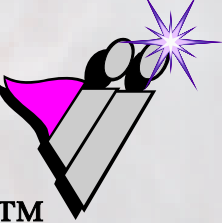
- ❖ Danas skoro svi RDBMS (relacijski DBMS) proizvodi imaju prefiks "O", tj. predstavljaju se kao **ORDBMS (objektno-relacijski DBMS)** sustavi, podržavajući (barem neke) OR mogućnosti definirane u SQL:1999 i SQL:2003 standardu.
- ❖ U radu se prikazuje **Dateov (C.J.Date) pogled** na relacijske i objektno-relacijske sustave.
- ❖ S tim u vezi prikazuju se neke objektne (ili relacijske, ili objektno-relacijske) **mogućnosti baze Oracle 10g/11g.**



Uvod

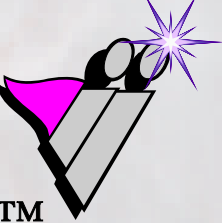


- ❖ **Relacijski model podataka** (matematičku teoriju) kreirao je **E.F.Codd** i opisao 1970. u radu "A Relational Model of Data for Large Shared Data Banks".
- ❖ Na temelju relacijskog modela IBM je 1974. napravio **prvi "laboratorijski" RDBMS - System R.**
- ❖ U tom projektu stvoren je i jezik **SEQUEL** (Structured English Query Language), kasnije skraćeno na **SQL.**
- ❖ **Prvi komercijalni RDBMS** pod imenom Oracle ponudila je 1979. Kompanija Relational Software Inc. (današnja Oracle Corporation).



OOPL jezici, OODBMS sustavi, ORDBMS sustavi

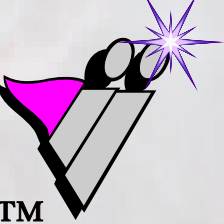
- ❖ **Prvi OOPL jezik Simula 67** nastao je 1967.
- dakle, prije pojave relacijskog modela.
Nagli razvoj OOPL jezika bio je 1980.-1985.
- ❖ Na krilima uspjeha OOPL-a, sredinom 80-tih pojavili su se **OODBMS** (objektno-orijentirani DBMS) sustavi, uglavnom radi potrebe perzistencije objekata.
- ❖ Sredinom 90-tih pojavili su se **ORDBMS** (objektno-relacijski DBMS) sustavi, npr. Informix, IBM DB2, Oracle (verzija 8.0, 1998.).
- ❖ ORDBMS sustavi su proširenje RDBMS sustava nekim objektnim mogućnostima.



Glavno pitanje – kako koristiti ORDBMS?



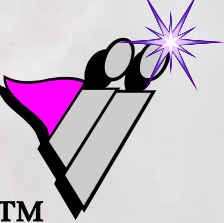
- ❖ David Maier, stručnjak na području objektnih baza (jedan od kreatora teksta "The Object-Oriented Database System Manifesto"), rekao je 2002.:
"I don't see any end application users using the new object-relational features. "
- ❖ Možda je to rekao zato što su ORDBMS sustavi konkurencija ODBMS sustavima (kojima se on bavi)?
- ❖ I Oracle potpredsjednik Tom Kyte u svojim knjigama (prva pokriva Oracle 8.0/8i, druga 9i/10g) kaže da on **ne koristi OR mogućnosti, osim unutar PL/SQL-a i za kreiranje OR view-ova nad relac. tablicama.**



Što o ORDBMS sustavima kaže C.J.Date

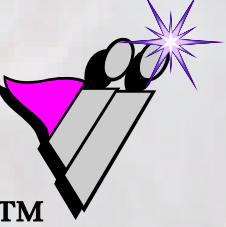


- ❖ **C.J.Date**, dugogodišnji suradnik E.J.Codda, poznati znanstvenik na području relacijskog modela i RDBMS sustava napisao je 2005.: “...**a proper object / relational system is just a relational system with proper type support** - which just means it's a proper relational system, no more and no less.”.
- ❖ Fundamentalno (i obimno, oko 1000 stranica) djelo “**An introduction to Database Systems**” (8.izdanje 2004, ukupno prodano oko 750 000 primjeraka) detaljno opisuje Dateov pogled na objektne i objektno-relacijske baze.



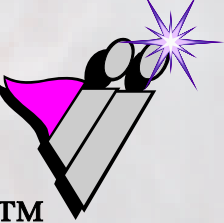
Što C.J.Date kaže za SQL

- ❖ **SQL \neq relacijski model**
- ❖ **SQL je (relativno) loš relacijski jezik**, nastao stihijski
- ❖ **dobar relacijski jezik (i RDBMS općenito):**
 - ne bi smio podržavati **NULL vrijednosti**
 - ne bi smio podržavati **duple retke** u tablici
 - mora bolje podržati ažuriranje **pogleda** (view-ova)
 - mora bolje podržati **integritetna ograničenja**
 - mora omogućiti bolju **optimizaciju**
 - mora biti (više) **simetričan...**
- ❖ **Date je svjestan da se SQL ne može tek tako izbaciti**, ali predlaže da se napravi bolji relacijski jezik, a **SQL bi se mogao bazirati na tom (boljem) jeziku.**



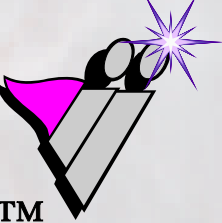
Tipovi podataka

- ❖ Tip podataka je imenovani **skup vrijednosti** koje zadovoljavaju određena ograničenja nad tipom (**type constraint**).
- ❖ Svakom tipu podataka pridruženi su **operatori** (read-only i update) za operiranje nad **vrijednostima** i **varijablama** toga tipa.
- ❖ Tipovi podataka mogu biti **sistemske** ili **korisnički-definirani**.
- ❖ Tipovi podataka mogu biti i **skalarni** ili **neskalarni** - skalarni tip nema korisniku vidljivih komponenti.



Tipovi podataka; Dateov relacijski jezik Tutorial D : SQL

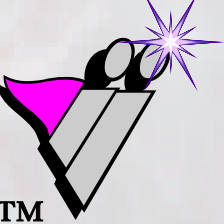
- ❖ Zbog lakšeg objašnjavanja relacijskog modela, Date je razvio vlastiti relacijski jezik **Tutorial D**.
- ❖ Tutorial D omogućava definiranje mogućih fizičkih reprezentacija tipa podataka (**possible physical representation**) i omogućava definiranje ograničenja nad tipom podataka (**type constraint**).
- ❖ SQL ne podržava niti jedno, niti drugo. Zato fizička reprezentacija u SQL-u nije korisniku sakrivena.
- ❖ Npr., tip podataka POINT je u Tutorial D skalarni tip, a u SQL-u je neskalarni tip.



Relacija (zapravo, relacijska varijabla) i tip podataka nisu jedno te isto

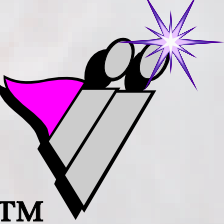


- ❖ **Relacija** je vrijednost. Pravo ime za (relacijsku) tablicu bilo bi **relacijska varijabla** (relvar).
- ❖ Date kaže da relacijski sustav mora podržavati i relacije i tipove podataka (domene):
"Tipovi podataka su skupovi stvari o kojima možemo govoriti; relacije su (istinite) tvrdnje o tim stvarima."
- ❖ Date:
 - **relacijska varijabla \neq tip podataka**
 - **klasa = tip podataka = domena**
(B.Meyer kaže: klasa je i tip podataka i modul)
 - **klasa \neq relacijska varijabla**



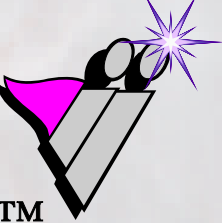
Relacijski model podataka

- ❖ Izvorno, relacijski model opisan je sa tri komponente
 - komponenta za definiranje **strukture** podataka
 - komponenta za **manipulaciju** nad podacima
 - komponenta **integritetnih ograničenja**
(prilično zanemarena u današnjim RDBMS sustavima)
- ❖ Dateova definicija – pet dijelova:
 - kolekcija sistemskih skalarnih tipova i obavezna **podrška za korisnički-definirane tipove podataka**
 - generator relacijskih tipova
 - definiranje relacijskih varijabli tih generiranih rel.tipova
 - operator za pridruživanje rel.vrijednosti rel.varijablama
 - kolekcija generičkih relacijskih operatora



Normalizacija podataka i relacijski model

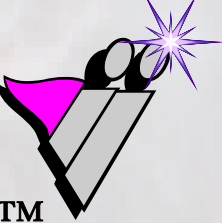
- ❖ Pravila za normalizaciju podataka predstavljaju **teoriju dizajna baze podataka.**
- ❖ Teorija dizajna baze podataka nije isto što i relacijski model, niti je ona proširenje relacijskog modela, već **teorija napravljena nad relacijskim modelom.**
- ❖ Suprotno nekadašnjem mišljenju, **sve su relacije (odnosno relacijske varijable) u 1.normalnoj formi.**
- ❖ Relacijska varijabla može imati attribute čija je vrijednost (druga) relacija – **RVA atributi (relational-valued attributes).**
No, najčešće to nije dobar dizajn baze podataka.



Date: najvažniji dodatak i dvije grube greške (great blunders) objektno-relacijskih DBMS sustava

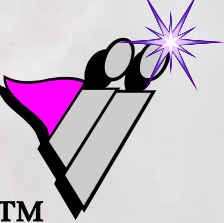


- ❖ **Najvažniji doprinos objektnog pristupa RDBMS-u: korisnički-definirani tipovi podataka (klase).**
- ❖ **Prva gruba greška ORDBMS sustava: izjednačavanje klase i relacijske varijable.**
Iz nje proizlaze mnoge loše posljedice, pa i ta da SQL standard i neki produkti uvode koncept **nadtablice i podtablice** (kao podklase nadtablice).
- ❖ **Druga gruba greška ORDBMS sustava: uvođenje pokazivača (ili referenci - u SQL standardu je to **REF** tip podataka) u relacijski model podataka.**
Njih je Codd (tvorac relacijskog modela) kritizirao u predrelacijskim, tj. hijerarhijskim i mrežnim bazama podataka.



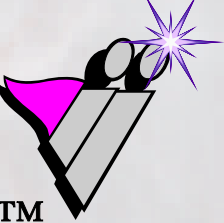
Oracle tipovi podataka

- ❖ **Sistemske skalarni tipovi podataka:** NUMBER, CHAR, VARCHAR2, DATE, CLOB, ali nema BOOLEAN (uveden u SQL:1999 standardu, po Dateu temeljan tip podataka).
- ❖ **Sistemske neskalarni tipovi:** npr. XMLTYPE; specijalni tipovi podataka ANYDATA, ANYDATASET, ANYTYPE koji su samoopisujući i podržavaju refleksivnost (kroz odgovarajući API).
- ❖ **Pokazivači(reference)** – REF tipovi podataka
- ❖ **Kolekcije** – zapravo su **generatori tipova podataka**
 - VARRAY (varijabilni niz)
 - NESTED TABLE (drugdje poznat kao MULTISSET)



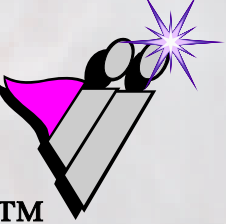
Oracle object types

- ❖ **Object types** su korisnički-definirani tipovi podataka; na neki način to je ono što OOPL jezici zovu **klasa**.
- ❖ Sastoje se od **atributa i metoda** (procedura i funkcija).
- ❖ Metode mogu biti one koje pripadaju objektu (**member**) ili cijeloj klasi (**static**) ili **konstruktori**. Posebne member metode su metode za uspoređivanje (**MAP**).
- ❖ Može se napraviti **podtip (podklasa)** koja može imati nove attribute/metode ili se postojeća metoda može **nadjačati** (ako nije FINAL), podržan je i **polimorfizam**.
- ❖ Za sada nema privatnih ili zaštićenih (private / protected) atributa/metoda – **svi su javni (public)**.



Oracle objektno-relacijske tablice

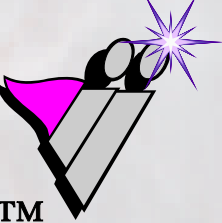
- ❖ Imenom **objektno-relacijske tablice** (object-relational tables) možemo nazvati tablice kod kojih su **stupci temeljeni na objektnom tipu (ili tipu-kolekciji)**, ali ne redak kao cjelina.
- ❖ Date smatra da su **takve vrste tablica sasvim prihvatljive i da su to zapravo relacijske tablice.**
- ❖ U nastavku kreiramo tip podataka `ADRESA_T` i koristimo ga za definiciju stupca `ADRESA` u objektno-relacijskoj tablici `OSOBA`:



Oracle objektno-relacijske tablice - kreiranje tipa i tablice tog tipa

```
CREATE TYPE adresa_t AS OBJECT (  
    grad VARCHAR2 (20) ,  
    ulica VARCHAR2 (30)  
)  
/
```

```
CREATE TABLE osoba (  
    ime VARCHAR2 (40) PRIMARY KEY,  
    adresa adresa_t  
)  
/
```



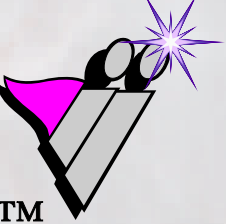
Oracle objektno-relacijske tablice - eksterna i interna struktura

❖ DESCRIBE OSOBA pokazat će samo **dva stupca**:

Name	Type
-----	-----
IME	VARCHAR2 (40)
ADRESA	ADRESA_T

❖ Međutim, tablica OSOBA **interno ima 4 stupca**:

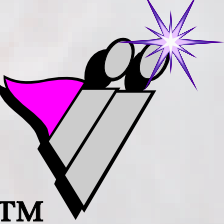
- vidljivi skalarni stupac IME
- **dva nevidljiva stupca** koja predstavljaju komponente stupca-objekta ADRESA
- vidljivi stupac ADRESA koji je dugačak samo 1 byte, a Oracle ga koristi za interne potrebe, kao oznaku tipa



Oracle objektno-relacijske tablice - nasljeđivanje, nadjačavanje i polimorfizam



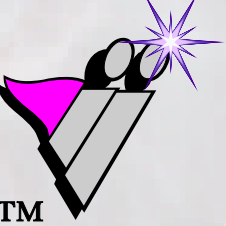
```
CREATE TYPE adresa_t AS OBJECT (  
    grad VARCHAR2(20),  
    ulica VARCHAR2(30),  
    MEMBER FUNCTION adresa_f RETURN VARCHAR2  
) NOT FINAL  
/  
CREATE TYPE BODY adresa_t AS  
    MEMBER FUNCTION adresa_f RETURN VARCHAR2 IS  
BEGIN  
    RETURN 'Generalna adresa: ' ||  
        grad || ' ' || ulica; ...
```



Oracle objektno-relacijske tablice - nasljeđivanje, nadjačavanje i polimorfizam



```
CREATE TYPE kucna_adr_t UNDER adresa_t (  
    OVERRIDING MEMBER FUNCTION adresa_f  
    RETURN VARCHAR2  
)  
/  
CREATE TYPE BODY kucna_adr_t AS  
    OVERRIDING MEMBER FUNCTION adresa_f  
    RETURN VARCHAR2 IS  
BEGIN  
    RETURN 'Kućna adresa: `  
        || grad || ' ' || ulica; ...
```

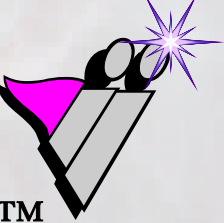


Oracle objektno-relacijske tablice - nasljeđivanje, nadjačavanje i polimorfizam



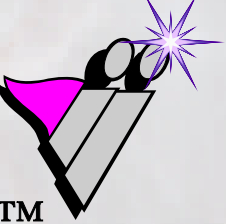
```
INSERT INTO osoba VALUES
  ('Ana', adresa_t ('G1', 'U1'));
INSERT INTO osoba VALUES
  ('Pero', kucna_adresa_t ('G2', 'U2'));

SELECT ime, o.adresa.adresa_f()
  FROM osoba o; -- obavezan alias tablice
IME      O.ADRESA.ADRESA_F()
-----
Ana      Generalna adresa:G1 U1
Pero     Kućna adresa:G2 U2
```



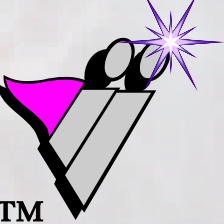
Oracle objektne tablice

- ❖ Oracle **objektne tablice** ili **tablice objekata** (object tables) su one tablice kod kojih **cijeli redak reprezentira jedan objekt.**
- ❖ To je onaj slučaj koji **Date naziva prvom grubom greškom.**
- ❖ Kreirat ćemo objektni tip ODJEL_T i na temelju njega objektnu tablicu ODJEL.
- ❖ Definirat ćemo objektni tip RADNIK_T koji ima referencu (**druga gruba greška**) na objektni tip ODJEL_T, pa kreirati objektnu tablicu RADNIK.
- ❖ Nakon toga ćemo raditi INSERT i SELECT:



Oracle objektne tablice – tip ODJEL_T i tablica ODJEL

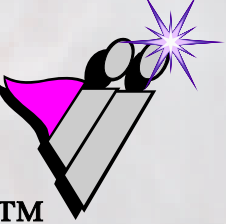
```
CREATE TYPE odjel_t AS OBJECT (  
    sifra VARCHAR(10),  
    naziv VARCHAR2(20)  
)  
/  
CREATE TABLE odjel OF odjel_t  
    (PRIMARY KEY (sifra))  
    OBJECT IDENTIFIER IS PRIMARY KEY  
-- Oracle bi inače automatski kreirao  
-- nevidljivi object identifier (OID)  
/
```



Oracle objektne tablice - tip RADNIK_T (sa REF na ODJEL_T) i tablica RADNIK



```
CREATE TYPE radnik_t AS OBJECT (  
  sifra VARCHAR(10),  
  naziv VARCHAR2(20),  
  odjel_oid REF odjel_t SCOPE IS odjel  
)  
/  
CREATE TABLE radnik OF radnik_t  
  (PRIMARY KEY (sifra))  
  OBJECT IDENTIFIER IS PRIMARY KEY  
/
```

Oracle objektne tablice - INSERT sa referencom i SELECT bez JOIN

O13
HrOUG

(implicitno dereferenciranje)

```
INSERT INTO radnik VALUES
```

```
  ('101', 'Ana',
```

```
    (SELECT REF (o)
```

```
      FROM odjel o -- obavezan alias
```

```
      WHERE sifra = '10'
```

```
    )
```

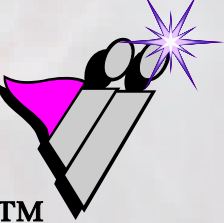
```
  );
```

```
SELECT sifra, naziv,
```

```
  -- implicitno dereferenciranje
```

```
  r.odjel_oid.naziv
```

```
FROM radnik r; -- ne treba JOIN
```

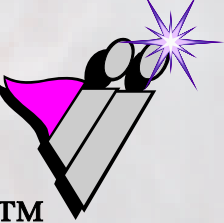


Oracle objektne tablice - DANGLING REFS

- ❖ **SCOPE FOR ne osigurava referencijalni integritet**, jer možemo izbrisati odjel u kojem radi neki radnik.
- ❖ Tako dobijemo reference koje pokazuju na nepostojeći objekt tj. **viseće reference (dangling REFS)**.
- ❖ Ako želimo izbjeći viseće reference, tj. osigurati referencijalni integritet (a u pravilu želimo), moramo tablici RADNIK kreirati **"dobri stari" vanjski ključ** na tablicu ODJEL:

```
ALTER TABLE radnik  
  ADD CONSTRAINT radnik_odjel_fk  
  FOREIGN KEY (odjel_oid) REFERENCES odjel
```

/

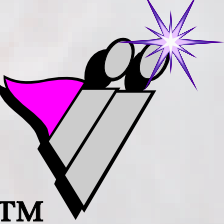


Oracle objektne tablice - PROGRAMER_T, podklasa od RADNIK_T



- ❖ Pretpostavimo (radi jednostavnijih SQL naredbi) da prethodno definirani tip RADNIK_T i na temelju njega kreirana tablica RADNIK nemaju REF atribut / stupac.
- ❖ Napravimo tip PROGRAMER_T kao podtip tipa RADNIK_T:

```
CREATE TYPE programer_t UNDER radnik_t  
  (programerski_staz NUMBER)  
/
```



Oracle objektne tablice - unos 3 podatka u tablicu sa 2 stupca !?

- ❖ Propustimo sada sljedeće INSERT naredbe:

```
INSERT INTO radnik VALUES ('101', 'Ana')
```

```
/
```

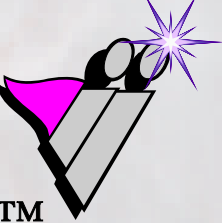
```
-- 3 podatka, a tablica ima 2 stupca!?
```

```
INSERT INTO radnik VALUES
```

```
  (programer_t ('102', 'Pero', 10))
```

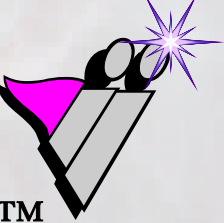
```
/
```

- ❖ **Kako to da uspijeva i druga naredba, koja unosi programera u tablicu RADNIK?**



Oracle objektne tablice - kako uspijeva unos programera u tablicu RADNIK

- ❖ Razlog je taj što tablica RADNIK ima i nevidljivi stupac PROGRAMERSKI_STAZ.
- ❖ Čim smo napravili podtip PROGRAMER_T, **Oracle je tablici RADNIK automatski dodao stupce koji odgovaraju dodatnim atributima podtipa** (u ovom slučaju imamo samo jedan dodatni atribut - PROGRAMERSKI_STAZ).
- ❖ Oracle bazi ne trebaju nadtablica i podtablica, jer se ista tablica može gledati istovremeno i kao **(implicitna) nadtablica i (implicitna) podtablica.**
- ❖ **No, bolje je koristiti relacijske tablice i poglede.**



Zaključak

- ❖ Ako gledamo SQL:1999 i SQL:2003 standarde i stanje na tržištu, jasno je da ORDBMS-i postoje.
- ❖ No, neke OO osobine ORDBMS-a zapravo pripadaju relacijskom modelu i "običnim" RDBMS-ima. Prije svega, **podrška za korisnički-definirane tipove podataka**.
- ❖ Mogućnost da korisnički definirane tipove podataka koristimo i za **definiranje "objektnih tablica"** Date naziva **"prva gruba greška"**.
- ❖ **Uvođenje pokazivača** (ili referenci, REFs) u relacijske baze Date naziva **"druga gruba greška"**.
- ❖ Takve "čiste objektne osobine" trebalo bi u relacijskoj bazi (u pravilu) izbjegavati.