


Objektově relační databáze

Jaroslav Pokorný

MFF UK, Praha

pokorny@ksi.mff.cuni.cz



Obsah

1. Úvod
2. Relační, OO a OR databáze
3. Od jazyka Sequel k SQL:1999
4. Objektově relační model v SQL:1999
5. OR rysy dostupných SŘBD
6. Modelování databází v OR přístupu
7. Závěr

Proč OR db technologie

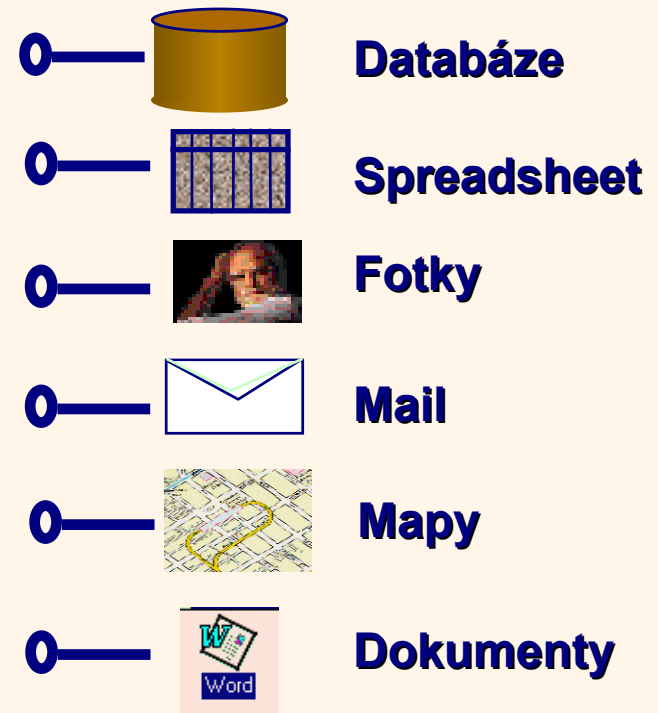
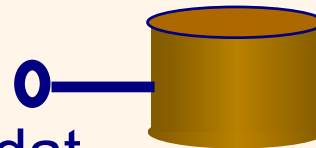
Požadavky nových aplikací:

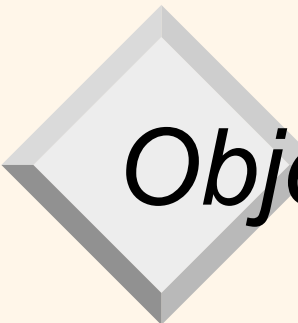
- nové typy objektů a funkcí
- OO analýza a návrh vs. relační db

"Relační databáze je podobná garáži, která vás nutí rozmontovat vaše auto a uložit díly do malých zásuvek..."

Cíl: integrace a správa dat
v jednom systému

SŘBD





Objektově relační modelování

- Rozšíření relačního modelu o objekty a konstrukty pro manipulaci nových datových typů; hierarchie objektů;
- atributy n-tic jsou složitých typů (včetně hnížděných relací);
- zachovány jsou relační základy včetně deklarativního přístupu k datům;
- kompatibilita s existujícími relačními jazyky (tvoří podmnožinu).

Hnízděná vs. normalizovaná relace

časopis	titul	autoři	klíčová_slova	datum		
				den	měsíc	rok
CW	OLAP	{Kusý, Klas}	{hvězda, dimenze}	23	duben	1998
SN	Databáze	{Novák, Fic}	{RDM, schéma}	15	květen	1998

časopis	titul	autor	klíčové_slovo	den	měsíc	rok
CW	OLAP	Kusý	hvězda	23	duben	1998
CW	OLAP	Kusý	dimenze	23	duben	1998
CW	OLAP	Klas	hvězda	23	duben	1998
CW	OLAP	Klas	dimenze	23	duben	1998
SN	Databáze	Novák	RDM	15	květen	1998
SN	Databáze	Novák	schéma	15	květen	1998
SN	Databáze	Fic	RDM	15	květen	1998
SN	Databáze	Fic	schéma	15	květen	1998

Normalizace do 4NF

časopis	titul
CW	OLAP
SN	Databáze

titul	autor
OLAP	Kusý
OLAP	Klas
Databáze	Novák
Databáze	Fic

titul	klíčové slovo
OLAP	hvězda
OLAP	dimenze
Databáze	schéma
Databáze	RDM

titul	den	měsíc	rok
OLAP	23	duben	1998
Databáze	15	květen	1998

Nevýhody 4NF:

- moc spojení v dotazech

Nevýhody pouhé 1NF

- ztráta vztahu 1 řádek = 1 objekt

Historie

1997 - komercializace OR přístupu

OR databáze ale existovaly již od r.1993:

- Montage (později Illustra)

tvůrci: Morgenthaler a Stonebraker
(zakladatel SŘBD Ingres) na bázi
univerzitního prototypu POSTGRES.

- UniSQL/X, tvůrce Won Kim.

Terminologie: rozšiřitelné databázové systémy

Rozšiřitelnost, uživatelsky definované typy a funkce

Požadavek: manipulace BLOB (v RSŘBD atomický)

Rozšiřitelnost: možnost přidávání nových datových typů + programů (funkce) „zabalенých“ do speciálního modulu

⇒ UDT (*uživatelsky definované typy*)
UDF (*uživatelsky definované funkce*)

Problém: zapojení do relačního SŘBD (včetně SQL !)

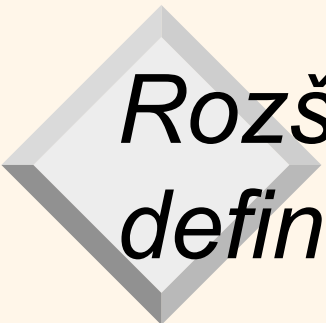
DB/2: relační extendery

Informix: DataBlades

ORACLE: cartridges

Sybase: Component Integration Layer.

univerzální servery



Rozšiřitelnost, uživatelsky definované typy a funkce

Př.: DB/2

obrazový, video, audio, textový, mapy, časové řady
VideoCharger (audio a video objekty v reálném
čase), XML

Př.: Informix v r. 1999 více než 25 DataBlades:
analýza dat pomocí fuzzy logiky, čištění (pro datové
sklady), zpracování dokumentů v ruském jazyce

Př.: Oracle

interMedia - cartridge pro prostorové objekty, časové
řady, vyhledávání ve vizuálních informacích
(obrázcích).

Fenomén SQL

**M. Stonebraker:
SQL je mezigalaktický
databázový žargon**

1986: standardizace ANSI

SQL86

1987: přijetí ISO

– dialekt SQL IBM ("průnik existujících implementací")

1989: rozšíření možností IO

SQL89

1992: nové datové typy, nové typy spojení, obecnější IO,
lepší strukturalizace

SQL92

1998 - Java (SQLJ).

1999: objektové rysy, rekurze, trigger

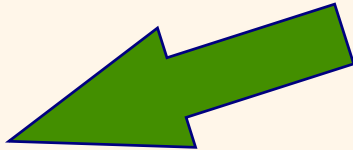
SQL:1999

≥ 2000: - kontejner pro budoucí standardy (temporální
SQL, XML, ...)

SQL4

SQL:1999

Pět částí:

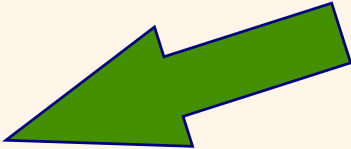
- SQL/Framework
- SQL/Foundations 
- SQL/CLI (Call Level Interface*)
- SQL/PSM (Persistent Store Modules **)
- SQL/Bindings***


* alternativa k volání SQL z aplikačních programů

** procedurální jazyk pro psaní transakcí (viz Microsoft ODBC)

*** dynamický, vnořený, přímý SQL

SQL:1999

- podpora objektů 
- uložené procedury
- triggery
- rekurzivní dotazy
- rozšíření pro OLAP (např. operátor CUBE)
- procedurální konstrukty
- výrazy za ORDER BY
- savepoints
- update prostřednictvím sjednocení a spojení



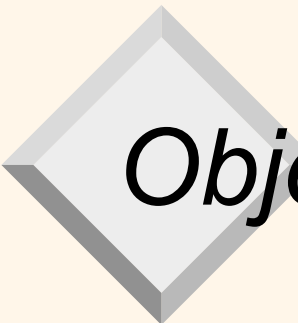
Co nového po r. 1999

- číslo 8 neodpovídá žádné části.
- část 9 – SQL/MED (Management of External Data)
- část 10 - SQL/OLB (Object Language Bindings)

2002 - dokončeno pět částí SQL/MM

- Základy (Framework),
- Úplné texty (Full Text),
- Prostorové objekty (Spatial),
- General Purpose Facilities (materiál společný ostatním částem, např. datové typy).
- Nepohyblivé obrazy (Still Image).

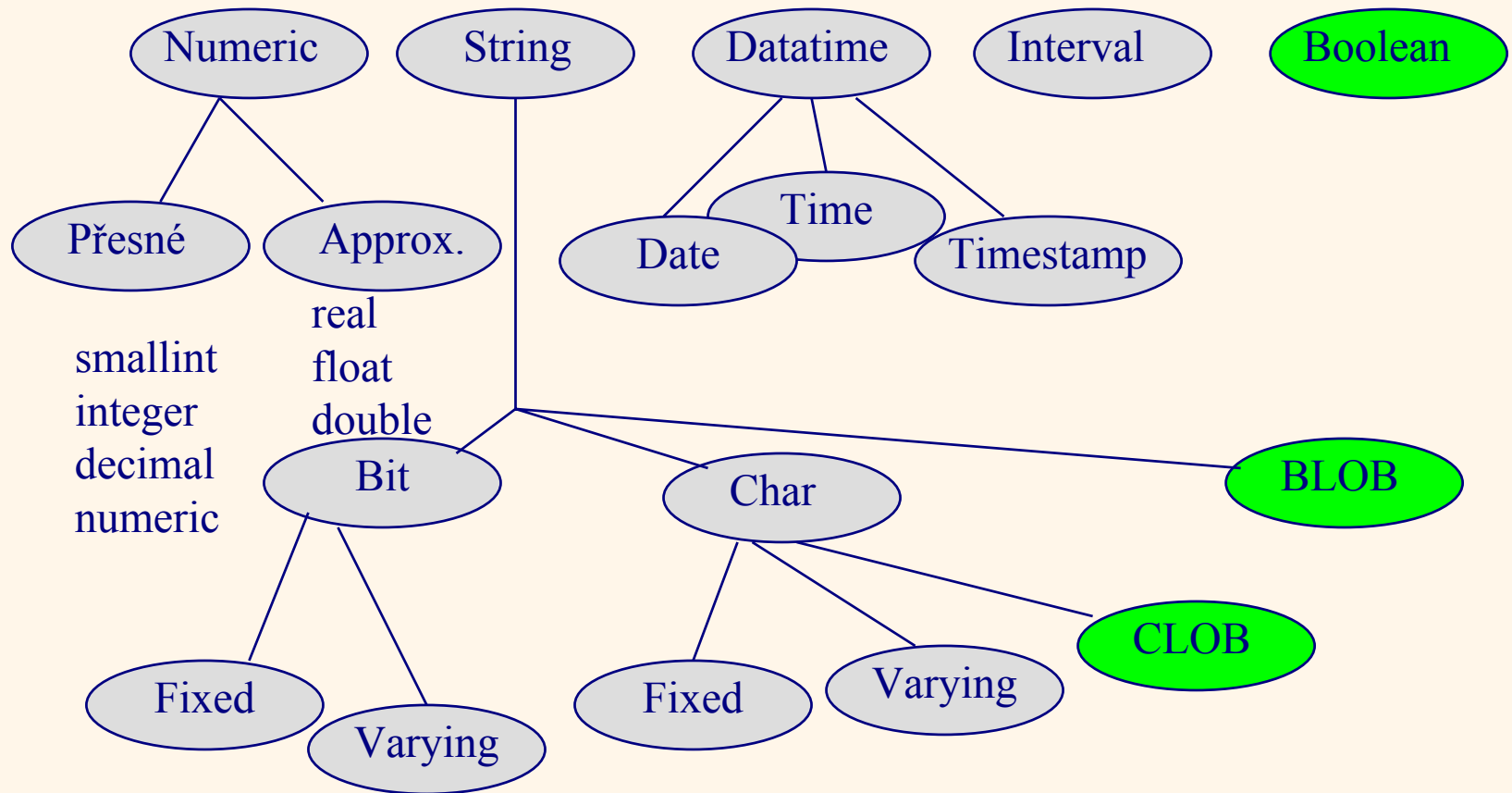
Zpoždění: část 7 – SQL/Temporal, část 11 - SQL/Schemata a část 14 SQL/XML.



Objektově relační model v SQL

- SQL3 pro podporu objektů používá:
 - *uživatелеm definované typy (ADT, pojmenované typy řádků a odlišující typy),*
 - *konstruktory typů pro typy řádků a typy odkazů,*
 - *konstruktory typů pro typy kolekcí (množiny, seznamy a multimnožiny),*
 - *uživatелеm definované funkce (UDF) a procedury (UDP),*
 - *velké objekty (Large Objects neboli LOB).*
- Standard SQL:1999 - podmnožina celkové koncepce

Předdefinované typy v SQL:1999





Nové typy v SQL:1999

Konstruované atomické typy:

- reference

Konstruované kompozitní typy:

- array /* podtyp collection */
- row

Pz.: v SQL4 je více typů kolekcí (v implementacích rovněž)

Pz.: k typům existují nové funkce (BIT_LENGTH, POSITION, SUBSTRING, ...)

Typ pole

```
CREATE TABLE zpravy(  
ID          INTEGER  
autoři      VARCHAR(15) ARRAY[20]  
titul       VARCHAR(100)  
abstrakt    FULLTEXT
```

- přístup ke složkám poziční, např. autoři[3],
- odhnízdění (UNNEST),

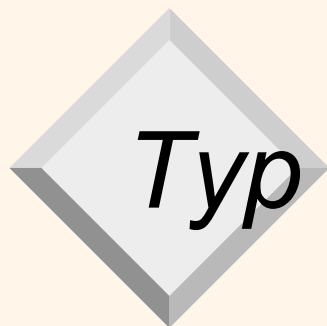
```
SELECT z.ID, a.jméno  
FROM zpravy AS z, UNNEST(z.autoři) as a(jméno)
```



Uživatелеm definované typy

UDT:

- *odlišující typy* (jsou zatím budované pouze na předdefinovaných typech)
- *strukturované typy* (mohou být definované s více atributy, které jsou předdefinovaných typů, typu ARRAY, nebo dalšího strukturovaného typu)
 - ◆ ADT
 - ◆ pojmenované typy řádků
- chování je realizováno pomocí funkcí, procedur a metod
- UDT mohou být organizovány do hierarchií s děděním

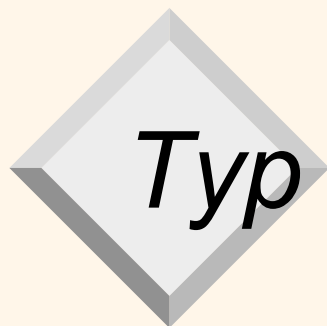


Typ řádku - nepojmenovaný

```
CREATE TABLE osoby (  
    jméno VARCHAR(20),  
    adresa ROW(ulice CHAR(30),  
               č_domu CHAR(6),  
               město CHAR(20),  
               PSČ CHAR(5)),  
    datum_narození DATE);
```

```
INSERT INTO osoby  
VALUES('J.Pokorný', ('Svojetická', '2401/2', Praha 10,  
                    10000), 1948-04-23);
```

```
SELECT o.adresa.město  
FROM osoby o
```

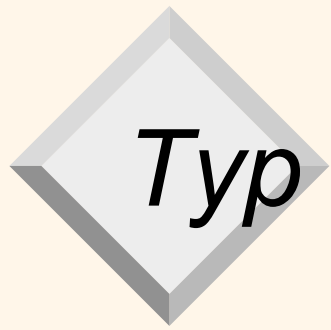


Typ řádku – pojmenovaný

- na rozdíl od ADT není zapouzdřený.

```
CREATE ROW TYPE účet_t (  
    č_účtu    INT,  
    klient    REF(zákazník_t),  
    typ       CHAR(1),  
    otevřen   DATE,  
    úrok      DOUBLE PRECISION,  
    zůstatek  DOUBLE PRECISION,  
);
```

```
CREATE TABLE účty OF účet_t  
    (PRIMARY KEY č_účtu );
```



Typ řádku – pojmenovaný ADT

- datová struktura (+ metody)
- vhodné pro modelování entit a jejich chování

Př.: osoba, student, oddělení, ...

```
CREATE TYPE zaměstnanec_t(  
  č_zam      INTEGER  
  jméno      VARCHAR(20));
```

jako typ sloupce

film	role	zam
Evita	sluha	(23, Kepka)
...

jako typ řádku

id	č_zam	jméno
23712	23	Kepka
...



Procedury a funkce

programy vyvolatelné v SQL: *procedury a funkce*

- procedury mají parametry typu IN, OUT, INOUT
- funkce mají parametry jen typu IN, vracejí hodnotu

konstrukce programů:

v PSM

- hlava i tělo v SQL (buď 1 SQL příkaz nebo BEGIN...END)
- hlava v SQL, tělo externě definované

volání programů:

- procedura: CALL jméno_procedury(p1,p2,...,pn)
- funkce: funkcionálně f(x,y)

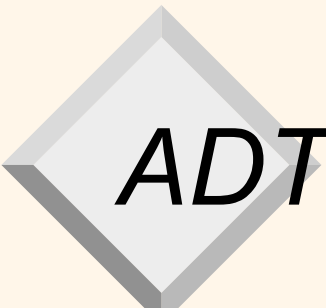
v ADT přibudou *metody*



- SQL/PSM umožňuje definovat procedury a funkce
- SQL:1999 přidává metody

Rozdíly mezi metodami a funkcemi:

- ◆ metody jsou vždy svázány s typem, funkce nikoliv,
- ◆ daný datový typ je vždy typem prvního (nedeklarovaného) argumentu metody,
- ◆ metody jsou uloženy vždy ve stejném schématu, ve kterém je uložen typ, ke kterému mají nejbližší. Funkce nejsou omezeny na specifické schéma.
- ◆ funkce i metody mohou být polymorfické, liší se v mechanismu volby konkrétní metody v run time,
- ◆ signatura a tělo metody jsou specifikovány odděleně,
- ◆ volání metody (tečková notace + argumenty v závorkách).



ADT - plány v SQL3

```
CREATE TYPE zaměstnanec_t
(PUBLIC
    č_zam          INTEGER
    jméno          VARCHAR(20),
    adresa         adresa_t,
    vedoucí       zaměstnanec_t,
    datum_nástupu DATE,
PRIVATE
    základní_plat  DECIMAL(7,2),
    příspěvek      DECIMAL(7,2),
PUBLIC
    FUNCTION odpr_léta(p zaměstnanec_t) RETURNS INTEGER
        <zdrojový kód pro výpočet počtu odpracovaných let>,
PUBLIC
    FUNCTION mzda(p zaměstnanec_t) RETURNS DECIMAL
        < zdrojový kód pro výpočet mzdy> );
```

ADT - skutečnost v SQL:1999

```
CREATE TYPE zaměstnanec_t AS(  
  č_zam          INTEGER  
  jméno          CHAR(20),  
  adresa         adresa_t,  
  vedoucí        zaměstnanec_t,  
  datum_nástupu  DATE,  
  základní_plat  DECIMAL(7,2),  
  příplatek      DECIMAL(7,2))  
NOT FINAL  
REF č_zam  
METHOD odpr_léta() RETURNS INTEGER  
METHOD mzda() RETURNS DECIMAL);
```

```
CREATE METHOD odpr_léta  
FOR zaměstnanec_t  
BEGIN ... END;
```

```
CREATE METHOD mzda  
FOR zaměstnanec_t  
BEGIN ... END;
```

Pz.: NOT FINAL ... může mít další podtyp

REF umožňuje chápat data (řádky) v tabulkách daného typu jako objekty

Podtypy

```
CREATE TYPE osoba_t AS(  
    jméno          VARCHAR(20),  
    adresa        adresa_t,  
NOT FINAL
```

```
CREATE TYPE zaměstnanec_t UNDER osoba_t(  
    č_zam          INTEGER  
    vedoucí        zaměstnanec_t,          /*zaměstnanec_t je  
    datum_nástupu  DATE,                  podtypem osoba_t */  
    základní_plat  DECIMAL(7,2),  
    příspěvek      DECIMAL(7,2))  
NOT FINAL  
REF č_zam  
METHOD odpr_léta() RETURNS INTEGER  
METHOD mzda() RETURNS DECIMAL);
```

Podtypy

```
CREATE TYPE zákazník_t UNDER osoba_t AS(  
    firma          VARCHAR(40);
```

```
CREATE TYPE dělník_t UNDER zaměstnanec_t ...
```

podtypy

- strukturované typy mohou být podtypem dalšího UDT
- UDT dědí strukturu (atributy a chování (metody) ze svých nadtypů
 - povoleno je jednoduché dědění (vícenásobné odloženo do SQL4)
 - vztahy mezi typy neimplikují žádné logické vztahy mezi daty tabulek těchto typů
- v SQL:1999 strukturované typy musí být NOT FINAL a odlišující typy musí být FINAL (v SQL4 to bude obecnější)
- *substituovatelnost*: na místě daného typu může být hodnota podtypu

Reference a dereference

Jak REF?

- ◆ generované uživatelem (REF USING <předdefinovaný typ>
- ◆ generované systémem (REF IS SYSTEM GENERATED) - implicitně
- ◆ odvozené (REF(<seznam atributů>)

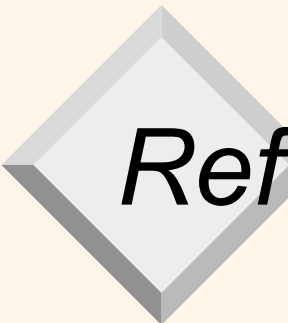
```
CREATE TYPE účet_t (  
  č_úctu      INT,  
  klient      REF(zákazník_t),  
  typ         CHAR(1),  
  otevřen     DATE,  
  úrok        DOUBLE PRECISION,  
  zůstatek    DOUBLE PRECISION,  
)  
FINAL REF IS SYSTEM GENERATED;  
CREATE TABLE účty OF účet_t  
REF IS OID SYSTEM GENERATED
```

reference

tabulka účty má zvláštní
atribut podobný oid v OO

tzv. *samoodkazující
sloupec*

tabulka



Reference a dereference

Dereference lze dělat pouze tehdy, je-li definováno umístění objektů typu REF (v SQL:1999 je to jedna tabulka)

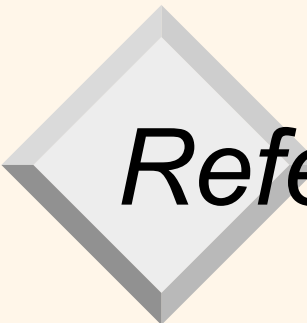
```
CREATE TABLE zákazníci OF zákazník_t;  
CREATE TABLE účty OF účet_t  
  (PRIMARY KEY č_účtu,  
  klient WITH OPTIONS SCOPE zákazníci  
  );
```

alokace

Pz.: připomíná referenční integritu

```
SELECT u.klient -> jméno  
FROM účty u  
WHERE u.klient->adresa.město = „Brno“ AND u.zůstatek > 100000;
```

**dereference,
cesta**



Reference a dereference

- odkaz na metodu:

```
SELECT u.klient -> jméno
```

```
FROM účty u
```

```
WHERE u.klient->mzda() > 10000;
```

- funkce Deref (např. ORACLE) místo ->

```
SELECT u.otvřen, Deref(u.klient)
```

```
FROM účty u;
```



Reference a dereference

- funkce REF

```
INSERT INTO účty
```

```
  SELECT 25, REF(z), 'A', '2002-10-20', 0, 0
```

```
  FROM zákazníci z
```

```
  WHERE jméno = 'Pokorný Jaroslav';
```

```
SELECT číslo_u, klient
```

```
FROM účty u
```

```
WHERE u.číslo_u = 25;
```

číslo_u	klient
25	123456

Podtabulky

- aparát odrážející vztahy mezi typy + logické vztahy mezi daty více tabulek


```
CREATE TABLE osoba  
  (jméno CHAR(20),  
   pohlaví CHAR(1),  
   věk INTEGER);
```

*výběr osob zahrnuje
zaměstnance
i zákazníky*

```
CREATE TABLE zaměstnanec UNDER osoba  
  (mzda FLOAT);
```


```
CREATE TABLE zákazník UNDER osoba  
  (č_úctu INTEGER);
```

- dědí sloupce, IO, triggery, ... dané nadtabulky



Problémy s OR SQL

- tabulky jsou jsou jediné pojmenované entity
- objekty v řádcích bez zapouzdření
- typ REF aplikovatelný pouze na objekty dané řádkem
- ADT je prvním krokem k OO
 - umožnit perzistenci \Rightarrow objekt musí být v tabulce
 - individuální instanci nelze přiřadit jméno
 - nelze použít dotazy na všechny instance ADT



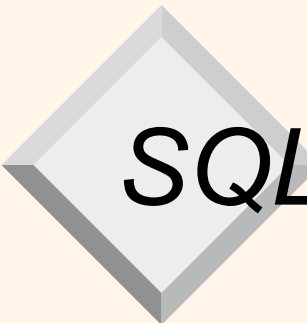
Problémy s OR SQL

Operační problémy:

- ORSŘBD podporují množinově orientovaný a nikoliv navigační přístup k objektům

Př.: dereferenční operace `GetObject(Ref)` musí být realizována pomocí `SELECTu`

- výsledky dotazů daných `SELECTem` ztrácejí strukturu a musí být znovu “přestavěny” do složitých objektových struktur OO programovacích jazyků



SQL3 v komerčních produktech

- Oracle 8TM : místo ADT -- *typ objektů*

notace: CREATE TYPE ... AS OBJECT(...);

kolekce:

- VARRAY (uspořádaný seznam dané délky)
- NESTED TABLE (*hnízděná tabulka* - neuspořádaná neohraničená kolekce prvků)

Př.: CREATE TYPE Kde_všude

AS VARRAY(4) OF Adresa

SQL3 v komerčních produktech

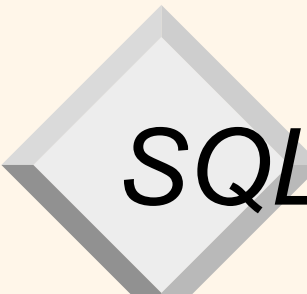
```
CREATE TYPE účet_t AS OBJECT
```

```
(č_úctu    INT,  
typ       CHAR(1),  
otevřen   DATE,  
úrok     DOUBLE PRECISION,  
zůstatek DOUBLE PRECISION  
);
```

```
CREATE TYPE účty_t AS TABLE OF účet_t
```

```
CREATE TABLE Zaměstnanci (
```

```
    ...  
    finance účty_t  
    ...)
```



SQL3 v komerčních produktech

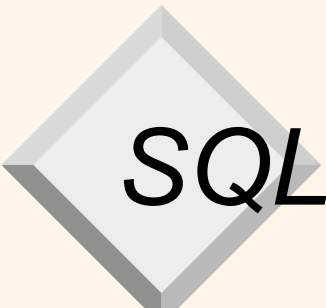
NESTED TABLE finance STORE AS T_ÚČTY

Dále:

- MEMBER FUNCTION nebo MEMBER PROCEDURE v příkazu CREATE TYPE.
- kód odpovídajícího programu je obsažen v odděleném příkazu CREATE TYPE BODY

D. Najdi zaměstnance, kteří mají účet s termínovaným vkladem.

```
SELECT * FROM Zaměstnanci AS Z, Z.finance AS FIN  
WHERE 'termínovaný vklad' IN (SELECT VP.typ  
FROM FIN);
```



SQL3 v komerčních produktech

V Informix Universal Server a DB/2 jsou k dispozici kolekce LIST, SET a MULTISSET

Další ORSŘBD:

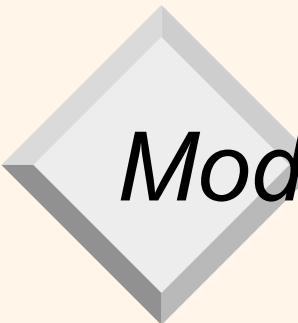
PostgreSQL (kategorie Open Source Software)

- množství vestavěných složených typů
- nekompatibilita s SQL:1999

SQL3 v komerčních produktech

```
CREATE TABLE zaměstnanci
(id INTEGER PRIMARY KEY,
jméno VARCHAR(30),
adresa ROW(      ulice CHAR(30),
                  číslo_d CHAR(6),
                  město CHAR(20),
                  PSČ CHAR(5)),
prémie MULTISSET(MONEY NOT NULL)
projekty SET (INTEGER),
děti LIST(osoba),
koníčky SET (VARCHAR(20))
```

kolekce



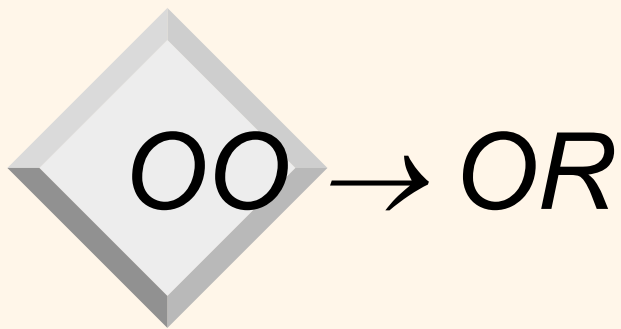
Modelování databází v OR přístupu

Příklady zobrazení :

- OO → OR
- prostorové objekty → OR
- XML → OR

Poznámka: hníždění by mělo vést u NF² relací k tzv. PNF (partitioned normal form) relacím.

Intuitivně: každá zahrnutá má klíč tvořený jednoduchými atributy.

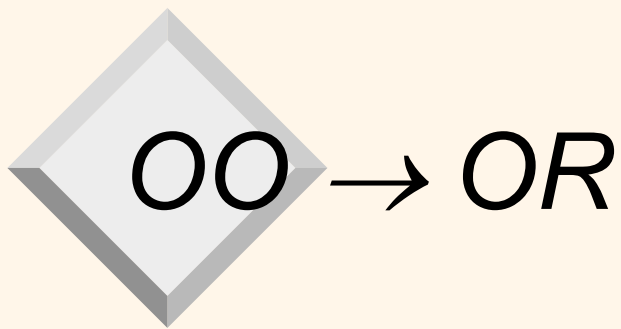


Problém a výhoda návrhu:

- ve „velkém“: schází modularita (výjimka: package v Oracle)
- v „malém“: odstranění 1NF vede k přirozenějšímu návrhu databáze

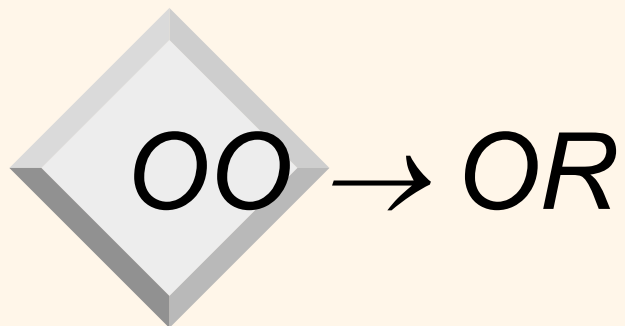
Dvě fáze návrhu:

- logický návrh nezávislý na ORSŘBD (zřejmě podle SQL:1999),
- specifický návrh uvažující konkrétní databázový produkt, ladění či optimalizaci.

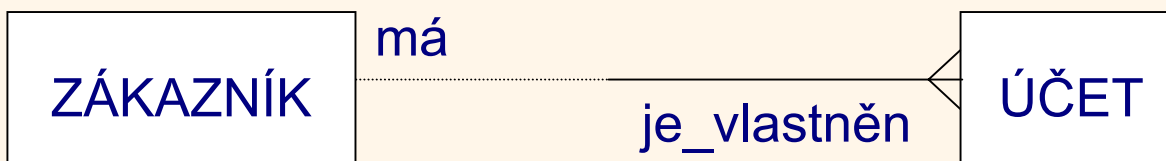


obecně: problém *objektově relačního zobrazení*

- UDT \approx třída v OO
- neinstanciovatelné UDT \approx abstraktní třída v OO
podle SQL:1999 je UDT asociován pouze s tabulkou
typovaná tabulka je extenzí třídy
- jednoduché atributy: přímo
- vícehodnotové atributy: kolekce (zde ARRAY, v ORACLE VARRAY)
- složené atributy: row nebo ADT
- vypočítané atributy: s triggerem, metodami



- vztahy 1:N



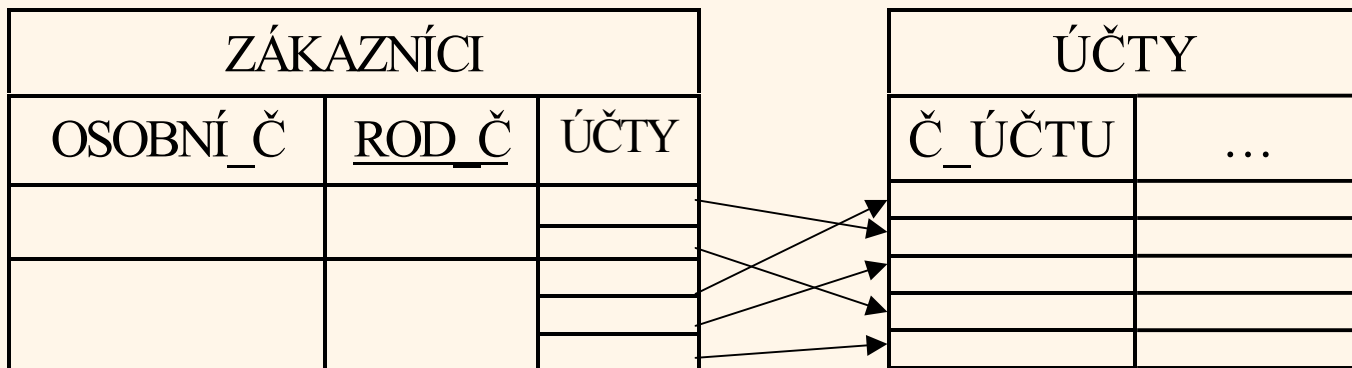
ZÁKAZNÍCI		
<u>OSOBNÍ_Č</u>	<u>ROD_Č</u>	...

ÚČTY		
<u>ČÍSLO</u>		<u>RODČ</u>

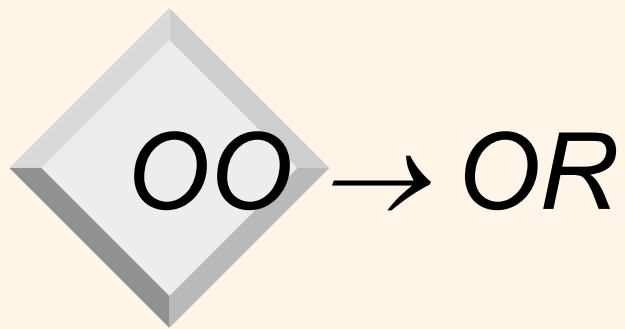
Řešení podle SQL92

OO → OR

ZÁKAZNÍCI		
OSOBNÍ_Č	<u>ROD_Č</u>	ÚČTY



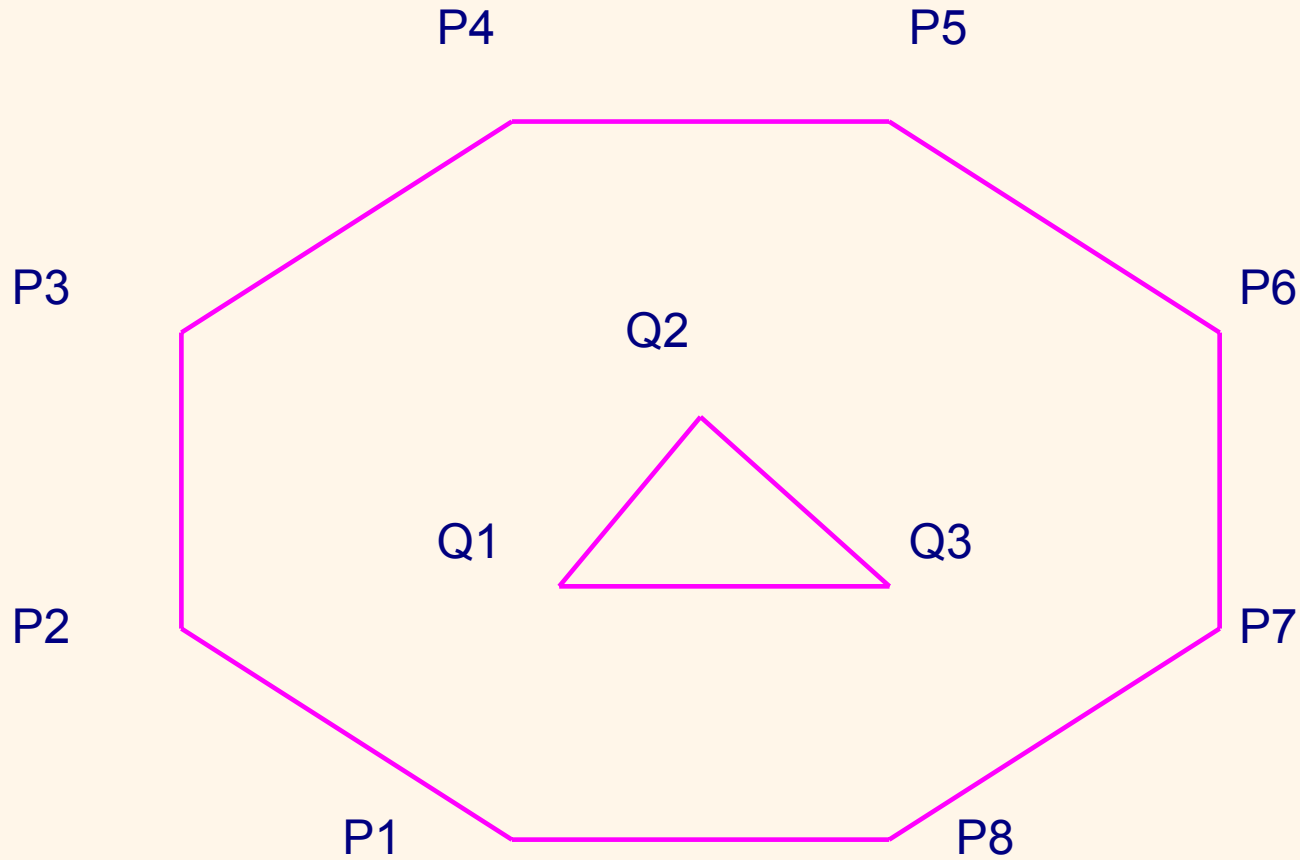
Řešení podle SQL:1999



- vztahy M:A:
pomocí ARRAY/ARRAY
- vztahy typu 1:1
pomocí REF/REF
- hierarchie tříd, tabulek
přímá podpora
- agregace a kompozice (případy celek-část)
pomocí ARRAY (v Oracle i pomocí
NESTED TABLE)



prostorové objekty → *OR*



prostorové objekty → RMD

REGION1 LAYER
ORDCNT
4

REGION1 DIM				
DIMNUM	LB	UB	TOLERANCE	DIMNAME
1	0	100	0,4	X osa
2	0	100	0,4	Y osa

REGION1 GEOM							
GID	ESEQ	ETYPE	SEQ	X1	Y1	X2	Y2
1234	0	3	0	P1(X)	P1(Y)	P2(X)	P2(Y)
1234	0	3	1	P2(X)	P2(Y)	P3(X)	P3(Y)
1234	0	3	2	P3(X)	P3(Y)	P4(X)	P4(Y)
1234	0	3	3	P4(X)	P4(Y)	P5(X)	P5(Y)
1234	0	3	4	P5(X)	P5(Y)	P6(X)	P6(Y)
1234	0	3	5	P6(X)	P6(Y)	P7(X)	P7(Y)
1234	0	3	6	P7(X)	P7(Y)	P8(X)	P8(Y)
1234	0	3	7	P8(X)	P8(Y)	P1(X)	P1(Y)
1234	1	3	0	Q1(X)	Q1(Y)	Q2(X)	Q2(Y)
1234	1	3	1	Q2(X)	Q2(Y)	Q3(X)	Q3(Y)
1234	1	3	2	Q3(X)	Q3(Y)	Q1(X)	Q1(Y)

prostorové objekty → OR

např. jako
NESTED TABLE

REGION1 GEOM							
GID	ESEQ	ETYPE	SEQ	X1	Y1	X2	Y2
1234	0	3	0	P1(X)	P1(Y)	P2(X)	P2(Y)
1234	0	3	1	P2(X)	P2(Y)	P3(X)	P3(Y)
1234	0	3	2	P3(X)	P3(Y)	P4(X)	P4(Y)
1234	0	3	3	P4(X)	P4(Y)	P5(X)	P5(Y)
1234	0	3	4	P5(X)	P5(Y)	P6(X)	P6(Y)
1234	0	3	5	P6(X)	P6(Y)	P7(X)	P7(Y)
1234	0	3	6	P7(X)	P7(Y)	P8(X)	P8(Y)
1234	0	3	7	P8(X)	P8(Y)	P1(X)	P1(Y)
1234	1	3	0	Q1(X)	Q1(Y)	Q2(X)	Q2(Y)
1234	1	3	1	Q2(X)	Q2(Y)	Q3(X)	Q3(Y)
1234	1	3	2	Q3(X)	Q3(Y)	Q1(X)	Q1(Y)

XML → OR

- jednoduché elementy bez atributů: jednoduché relační atributy
- posloupnost jednoduchých elementů: n-tice databázových atributů
- elementy s atributy: n-tice databázových atributů
- regulární výrazy:
 - $e?$ databázový atribut s NULL
 - $e+$ hnížděná relace s NOT NULL
 - e^* hnížděná relace
 - $e_1 | e_2$ samostatné relace + nadrelace (hierarchie)

Problémy:

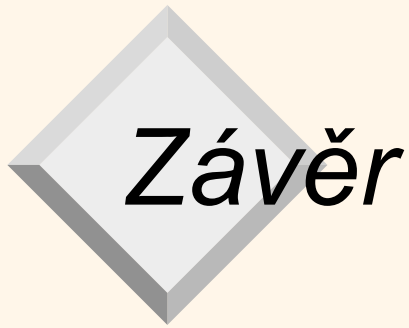
- klíče relací! Nejsou-li přirozené, musí se určit umělé
- rekurzivní struktury

Závěr

Whitemarsh Information Systems Corp.: článek

"Great News, the Relational Model is Dead"

- spíše provokující reakce na SQL:1999
- SQL:1999 navazuje na relační model
- patrné výhody:
 - ◆ vyšší výkon ORSŘBD
 - ◆ pružnost díky rozšiřitelnosti
 - ◆ větší role metadat



Závěr

Cíle OR technologie z hlediska praxe:

- obdržet maximum z rozsáhlých investic do relační technologie (data, nabyté zkušenosti),
- využít výhody v pružnosti, produktivitě a provozních přínosů OO modelování,
- integrovat databázové služby do systémů výroby a dalších aplikací.

Závěr

- současné implementace ORSŘBD jsou zatím v počátcích
 - paralela:
 - výtky OOSŘBD - málo databázové
 - výtky ORSŘBD - málo objektové
- chybí vývojové prostředky, nové metodologie
- největší problém a současně výhoda: univerzálnost

Závěr

Chris Date: článek

"Great News, the Relational Model is Very Much Alive!"

- kritika objektového rozšíření, ukazatelů (může být jen na fyzické úrovni)
- uzavírá: "Za sto let, až všechno "hyper" týkající se ukazatelů, objektů a hyperkostek (atd. atd.) bude nadlouho zapomenuto, věřím, že SŘBD budou stále založeny na Coddově relačním modelu dat."