Objektové a objektově relační databázové stroje

J. Pokorný

M. Valenta

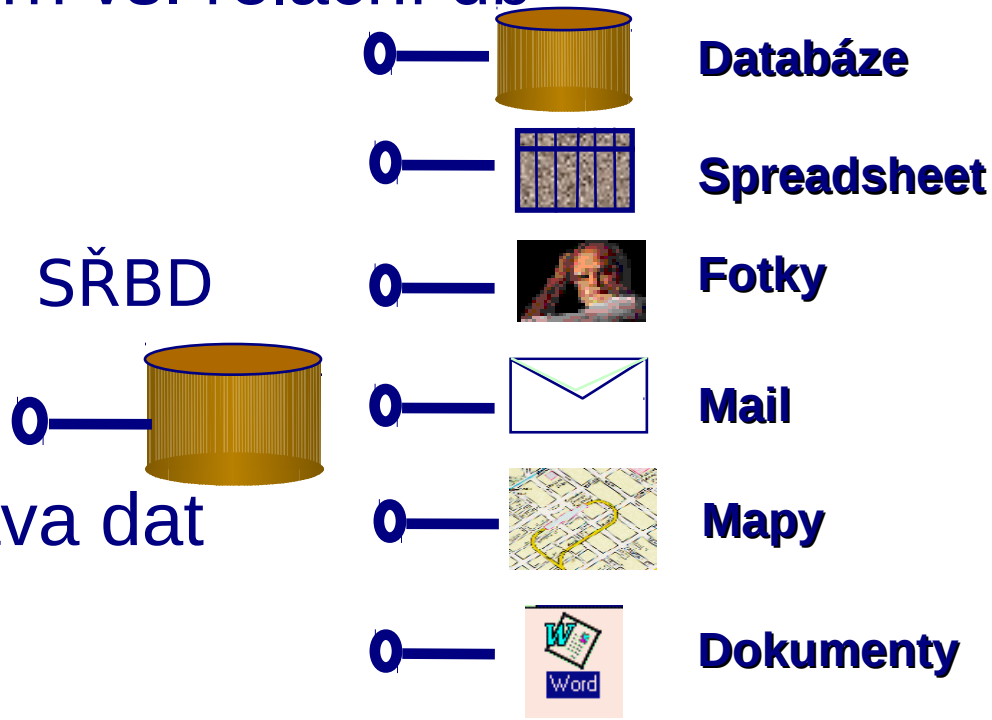
Obsah

Proč více db technologií

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..."



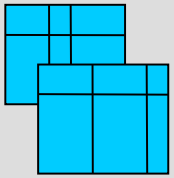
Cíl: integrace a správa dat v jednom systému

Objektově orientované databáze

objektový datový model:

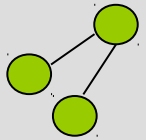
- je v souladu s viděním světa (entita ⇒ objekt)
- definice složitých objektů a jejich manipulace

RSŘBD
Výkonné OLTP
Dostupnost dat
Utajení
Prostředky pro správu dat
Standardní jazykové rozhraní
Řízení paměti
Souběžné zpracování dat
Integrita



Operace na složitých objektech
Rekurzivní struktury
Abstraktní datové typy
Rozhraní k OO jazyku
Složité transakce

OOSŘBD



Funkcionalita relačních a OO SŘBD

Objektově orientované databáze

1993: konsorcium vůdčích výrobců OOSŘBD ⇒ návrh standardu ODMG-93.

- nadmnožina obecnějšího modelu Common Object Model (COM) vytvořeného skupinou OMG. Převzat byl jeho definiční jazyk IDL.
- dotazovací část Object Query Language (OQL), která souvisí s koncepcí dotazovací části standardu SQL92.
- rozhraní k OO PJ C++, Smalltalk (k Java: nahrazeno Java Data Objects (JDO))

2001: byla skupina rozpuštěna (verze ODMG 3.0)

OOPJ + SŘBD = OOSŘBD

Základní koncepty ODMG-93

- *třída* (nebo *typ*), *instance* (nebo *objekt*), *atribut*, *metoda* a *integritní omezení*
  - třída - šablona pro instance (objekty), které mohou sdílet atributy a metody.
    - doména atributů: primitivní typ dat, abstraktní typ dat (ADT), nebo odkaz na třídu.
    - metoda je funkce (její implementace je skryta) aplikovatelná na instance třídy (výpočet založený na hodnotách atributů).
- *identifikátor objektu* (*OID*)
  - každý objekt má jednoznačný identifikátor, prostřednictvím kterého lze z databáze získat odpovídající objekt.

Základní koncepty ODMG-93

- *zapouzdření*
  - data jsou “zabalena” spolu s metodami. Jednotkou zapouzdření je objekt. Metody jsou platné pouze na příslušných objektech, se kterými jsou zapouzdřeny.
- *hierarchie tříd*, *dědění*
  - podtřída $\Rightarrow$ hierarchie
  - dědění je proces znamenající pro podtřídu osvojení všech atributů a metod z nadtřídy.
    - Preferuje se vícenásobné dědění ($\Rightarrow$ problémy např. řešení konfliktů stejných jmen zděděných atributů a metod).

Nástup OO db technologie

Zdroje: OO programování, OO analýza a návrh

- 2. pol. 80. let - OO databáze
  - koncem roku 1991 13 OOSŘBD (prototypy + komerční systémy)

- Př.: Gemstone, Objectivity/DB, ObjectStore, POET, VERSANT, Jasmine

  Princip: shora dolů (od aplikace k datům)

Nástup OR db technologie

Důvody:

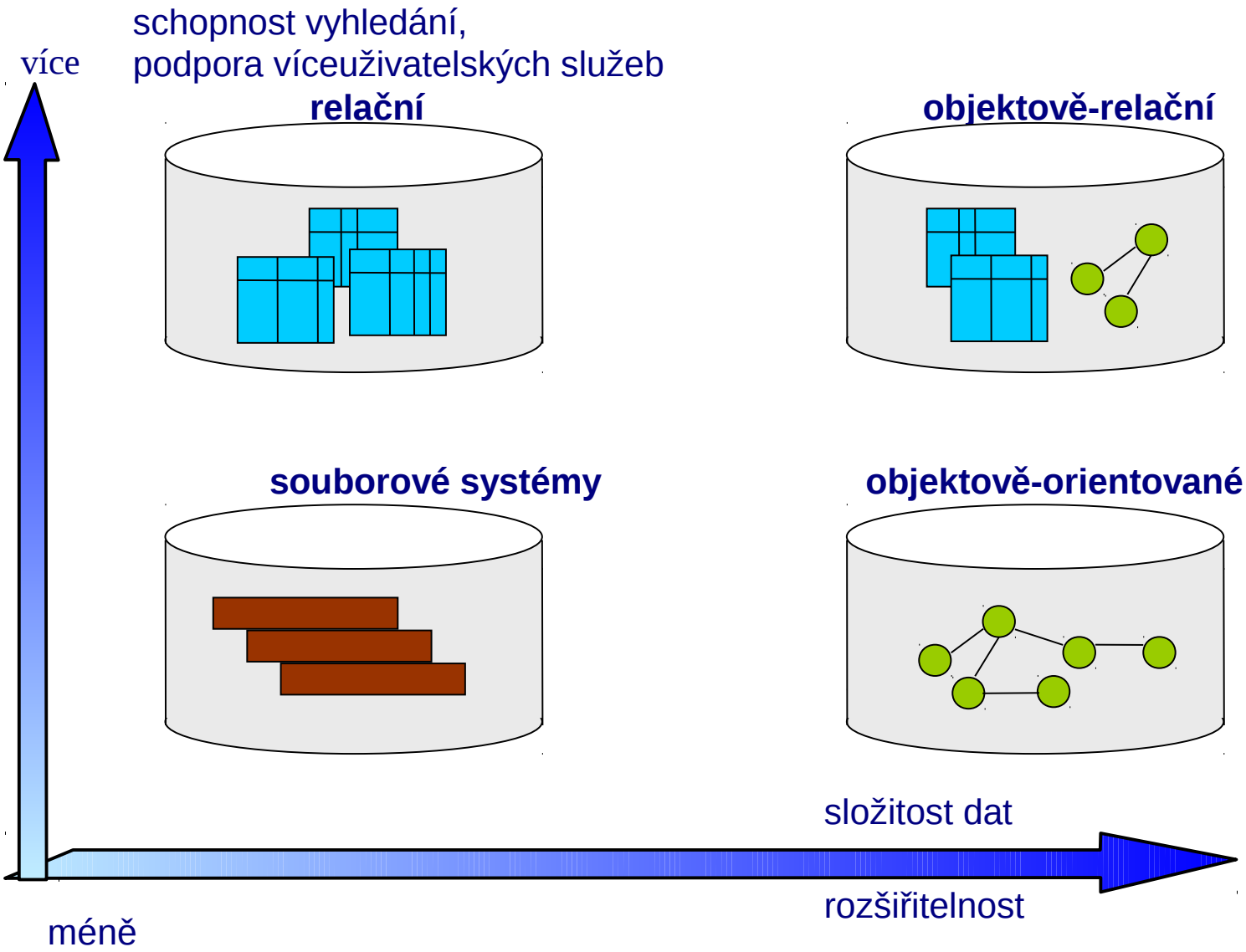
- obdržet maximum z rozsáhlých investic do relační technologie (data, nabyté zkušenosti),
- využít výhody v pružnosti, produktivitě a provozních přínosů OO modelování,
- integrovat databázové služby do systémů výroby a dalších aplikací.

Nástup OR db technologie

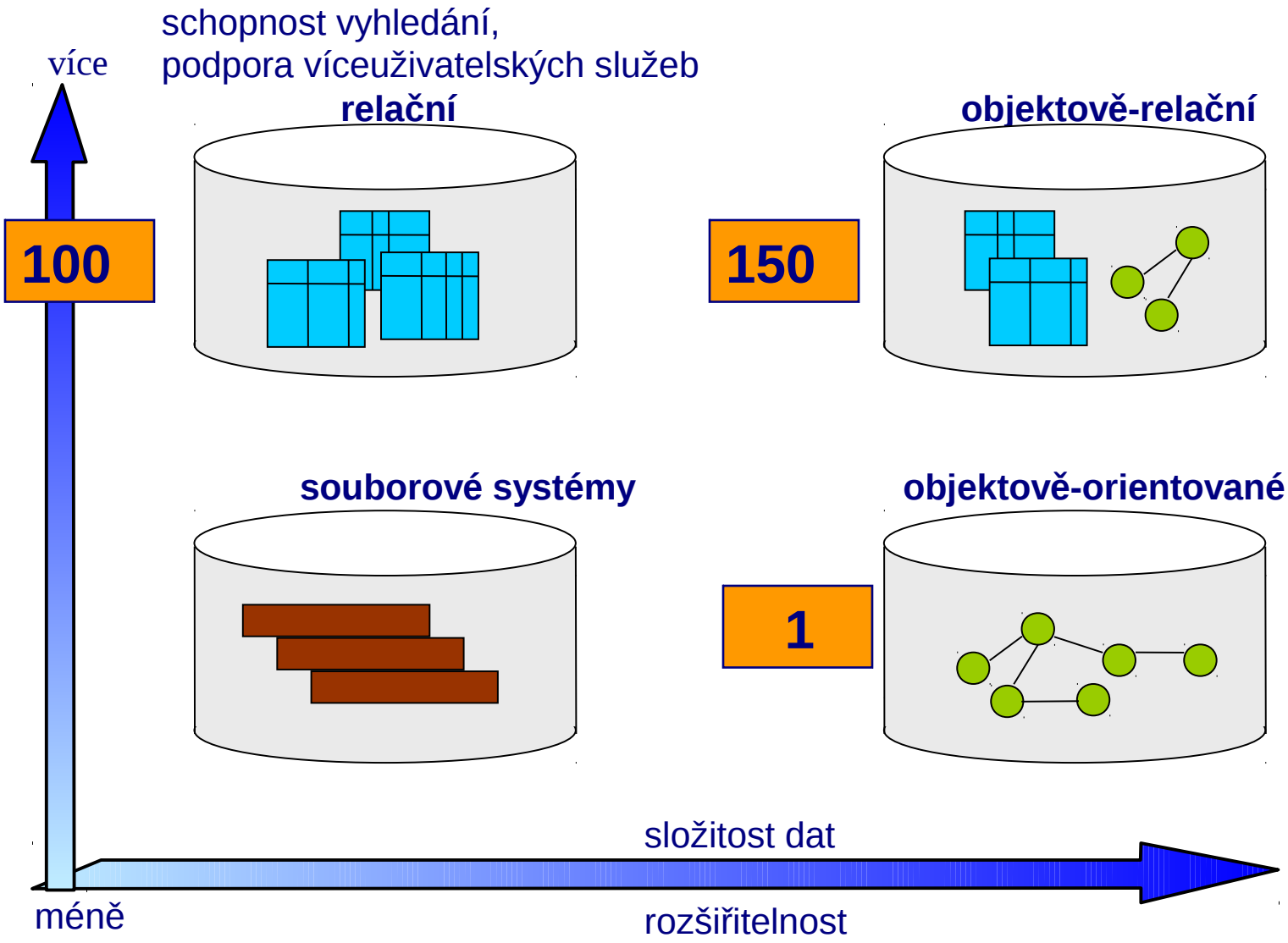
- 90. léta - OR přístup:
  - objektové obálky, objektové manažery
    - kombinace objektového a relačního přístupu
    - Př.: VisualAge C++ Data Access Builder, ObjectStore Gateway, Persistence, UniSQL/M, Gemstone/Gateway.
  - ORSŘBD
    - kombinace OO a relačních SŘBD
    - Př.: 1992: UniSQL/X, dále: HP - OpenODB (později Odapter)

      1993: Montage Systems (později Illustra) - komerční verze Postgres

      dále: DB/2, INFORMIX, ORACLE, Sybase Adaptive Server+Java, OSMOS, Unidata



Svět databázových technologií



Odhad relativní velikosti trhu v r. 2005 (Yuan, 1997)

Objektově relační databáze

Dva přístupy:

- *univerzální paměť*, kdy všechny druhy dat jsou řízeny SŘBD, jde o integraci (různými způsoby!)

  $\Rightarrow$ univerzální servery
- *univerzální přístup*, kdy všechna data jsou ve svých původních (autonomních) zdrojích

  Technika: middleware
  - brány (min. dva nezávislé servery)
  - zobrazení schémat, transformace dotazů
  - objektové obálky: Persistence Software, Ontologic, HP, Next, ... (problémy: výkon)
  - DB založené na Web

Objektově relační modelování

- Rozšíření relačního modelu o objekty a konstrukty pro manipulaci nových datových typů,
- atributy n-tic jsou složité typy, včetně hnízděných 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:

- spojení v dotazech

Nevýhody pouhé 1NF

- ztráta vztahu řádek = 1 objekt

Rozšiřitelnost, uživatelsky definované typy a funkce

Možnosti ADT:
black box
white box

Požadavek: manipulace BLOB (v RSŘBD atomický)

Rozšiřitelnost: možnost přidávání nových datových typů + programů (funkce) „zabalených“ 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 v r. 1999 má 19 extenderů:

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

Př.: Informix v r. 1999 více než 25 DataBlades:

- analýza dat pomocí fuzzy logiky
- čištění (pro datové sklady)
- zpracování dokumentů v ruském jazyce

Rozšiřitelnost, uživatelsky definované typy a funkce

Pokusy o standardizaci ISO: SQL/MM

(Př.: Full-Text - řeší ADT + odpovídající funkce)

Trend: UDT a UDF dělají třetí firmy

- INFORMIX uvádí 20 firem vyvíjejících DataBlades (více než 140)
- ORACLE (vyvíjí se více než 200 cartridges)

Implementace: technologie „plug in“ pomocí různých technik:

Př.: DataBlades - přímý přístup k databázovému jádru

ORACLE 7.3 - architektura více serverů a API

Příklad - textový extender

```
SELECT časopis, datum, titul
FROM ČLÁNKY
WHERE CONTAINS(text_článku, '("databáze" AND
                ("SQL" |"SQL92") AND NOT "dBASE")') = 1;
```

Další funkce: NO_OF_MATCHES (kolikrát se zadaný vzorek vyskytoval v textu), RANK (hodnota pořadí v odpovědi na základě nějaké míry).

```
SELECT časopis, titul
FROM ČLÁNKY
WHERE NO_OF_MATCHES (text_článku, 'databáze') > 10;
SELECT časopis, datum, RANK(text_článku, '("databáze" AND
                ("SQL" |"SQL92") )') AS relevantni
FROM ČLÁNKY
ORDER BY relevantni DESC;
```

„Opravdové“ ORSŘBD

Won Kim: „rozšiřitelnost je pouze druhotný, i když užitečný rys, jde o důsledek OO přístupu“

Stonebraker: „rozšiřitelnost typu “plug-in” (např. ORACLE) jako sice vhodnou pro konektivitu aplikace-aplikace, nicméně nemá nic do činění s databázovou “plug-in”. Jde o pouhý middleware, který nezakládá OR technologii“.

Požadavky:

- datový model s hlavními rysy ODMG-93
- odpovídající objektový jazyk vyšší úrovně

Řešení: objektové rozšíření SQL naplňuje (přibl.) tyto požadavky

Stav: standard SQL:1999 + zbytek v SQL4

SQL:1999

Pět části:

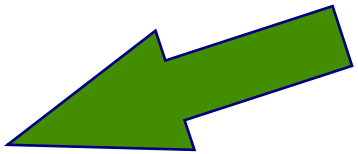
- SQL/Framework
- SQL/Foundations
- SQL/CLI (Call Level Interface*)
- SQL/PSM (Persistent Store Modules**)
- SQL/Bindings***

* alternativa k volání SQL z aplikačních programů

** procedurální jazyk pro psaní transakcí

*** dynamický, vnořený, přímý SQL

SQL:1999

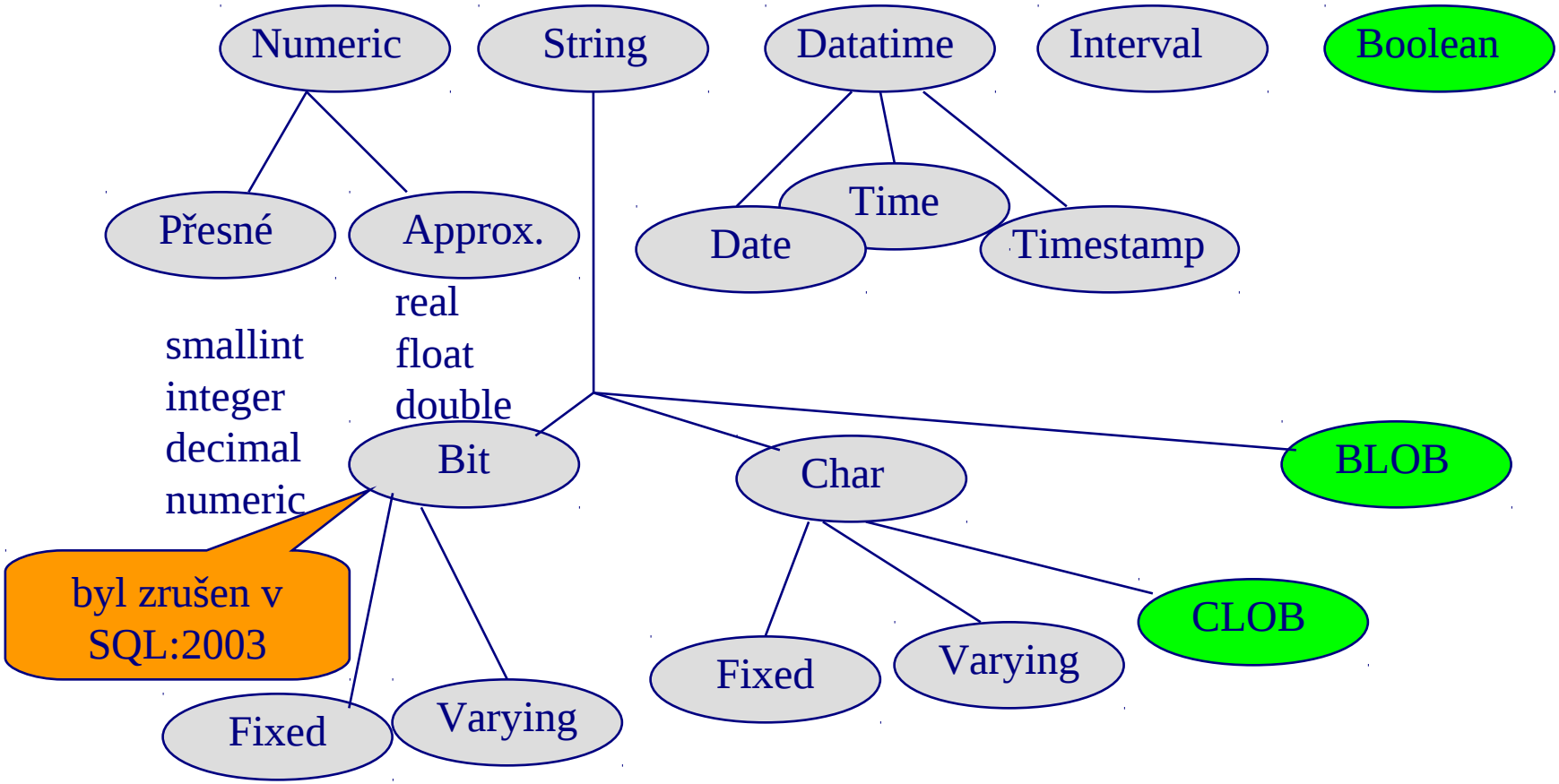


- podpora objektů
- uložené procedury
- triggery
- rekurzivní dotazy
- rozšíření pro OLAP
- procedurální konstrukty
- výrazy za ORDER BY
- savepoints
- update prostřednictvím sjednocení a spojení

Objekty: od SQL3 k SQL:1999

- SQL3 pro podporu objektů používá:
  - *uživatelem 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),
  - *uživatelem definované funkce* (UDF) a *procedury (UDP)*,
  - *velké objekty* (Large Objects neboli LOB).
- Standard SQL:1999 - podmnožina celkové koncepce

Předdefinované typy v SQL:1999



Typ Boolean

```
SELECT č_odd, EVERY(plat > 20000) AS bohatší,
    SOME(plat > 20000) AS chudší
FROM zam
GROUP BY č_odd;
```

Výsledek:

č_odd	bohatší	chudší
A35	FALSE	FALSE
J48	TRUE	TRUE
Z52	FALSE	TRUE

Další typy v SQL:1999

Konstruované atomické typy:

- reference

Konstruované kompozitní typy:

- array /* podtyp collection */
- multiset /* podtyp collection */
- row

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

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

Typ pole

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

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

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

Další typy v SQL:1999

UDT:

- *odlišující typy* (jsou zatím budované pouze na předdefinovaných typech)
- *strukturované typy* (mohou být definované s více atributy, které jsou předefinovaných 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

Odlišující typy

Princip: - přejmenování předdefinovaných typů + odlišné chování

```
CREATE TYPE TYP_MÍSTNOSTI
AS CHAR(10) FINAL;

CREATE TYPE METRY
AS INTEGER FINAL;

CREATE TYPE KV_METRY
AS INTEGER FINAL;

CREATE TABLE místnosti(
m_id          TYP_MÍSTNOSTI
m_délka       METRY
m_šířka       METRY
m_obvod       METRY
m_plocha      KV_METRY);
```

hlásí chybu

```
UPDATE místnosti
SET m_plocha =
m_délka
```

OK

```
UPDATE místnosti
SET m_šířka = m_délka
```

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 UDT přibudou *metody*

Procedury a funkce

Př.: DB2 UDB/OSF White Box ADT

```
CREATE TYPE bod AS (
      x DOUBLE,
      y DOUBLE,
      );

CREATE FUNCTION distance(p1 BOD, p2 BOD) RETURNS INTEGER
LANGUAGE SQL INLINE NOT VARIANT
RETURN sqrt((p2..y-p1..y)*(p2..y-p1..y) + (p2..x-p1..x)*(p2..x-p1..x));

SELECT Z.jméno
FROM zam Z, město M
WHERE M.název = ‘Ostrava’
            AND distance(Z.bydliště, M.střed) < 25;
```

ADT

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

Rozdíly metod a funkcí:

- 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 nejblíže. 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),
    příplatek             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                CHAR(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 */
    základní_plat        DECIMAL(7,2),
    příplatek            DECIMAL(7,2))
NOT FINAL
REF č_zam
METHOD odpr_léta() RETURNS INTEGER
METHOD mzda() RETURNS DECIMAL);
```

Podtypy

CREATE TYPE úředník_t UNDER zaměstnanec_t …

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)
- 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_účtu      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
   (PRIMARY KEY č_účtu );
```

reference

tabulka

tabulka účty má zvláštní atribut podobný oid

tzv. *samoodkazující sloupec*

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

dereference, cesta

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

Reference a dereference

odkaz na metodu:

```
SELECT u.klient -> jméno
FROM účty u
WHERE u.klient->mzda() > 10000;
```

vyřešení odkazu - hnízdění:

```
SELECT u.otevřen, DEREF(u.klient)
FROM účty u;
```

Podtabulky

- aparát nezávislý na aparátu typů

```
CREATE TABLE osoba
    (jméno CHAR(20),
    pohlaví CHAR(1),
    věk INTEGER);
CREATE TABLE zaměstnanec UNDER osoba
    (mzda FLOAT);
CREATE TABLE zákazník UNDER osoba
    (č_účtu INTEGER);
```

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

Za SQL:1999, 2003

```
CREATE TABLE zaměstnanci
   (id INTEGER PRIMARY KEY,
   jméno VARCHAR(30),
   adresa ROW(      uliceCHAR(30),
                    číslo_d CHAR(6),
                    město CHAR(20),
                    PSČ CHAR(5)),
   projekty SET (INTEGER),
   děti LIST(osoba),
   odměny MULTISET (MONEY)
```

kolekce

SQL4 v komerčních produktech

- INFORMIX: kolekce set, multiset, list (bez omezení délky)
- Oracle 8™: místo ADT -- *typ objektů*

  notace: CREATE TYPE … AS OBJECT(…);
  kolekce:
  - *Varray* (uspořádaný seznam dané délky)
  - *hnízděná tabulka* (neuspořádaná neohraničená kolekce prvků)

Př.: CREATE TYPE Kde_všude AS VARRAY(4) OF Adresa

SQL4 v komerčních produktech

```
CREATE TYPE Auta AS TABLE OF Auto_t
CREATE TABLE FIRMY (
    vozový_park Auta
    …)
NESTED TABLE vozový_park STORE AS t_vozy;
```

Pz.: lze zadat, kde mají být „podtabulky“ Auta uloženy

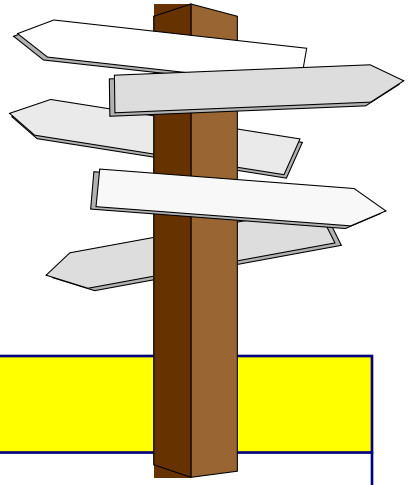
```
SELECT *
FROM FIRMY AS f, f.vozový_park AS vp
WHERE ‘Buick’ IN (SELECT vp.značka FROM vp);
```

```
SELECT REF(o) INTO reftoosoba
FROM osoby AS o
WHERE o.jméno = ‘Novák, J.’
```

Problémy s SQL

- tabulky 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, musí být objekt v tabulce
  - individuální instanci nelze přiřadit jméno
  - nelze použít dotazy na všechny instance ADT

ODMG 3.0 vs. SQL



rys	SQL4	ODMG 3.0
objektový model	obecný	klasický (vychází z COM od OMG)
JDD	SQL4	ODL (vychází z IDL od OMG)
dotazovací jazyk	SQL4	OQL (v dotazech vychází z SQL92)
JMD	SQL4	C++, Smalltalk, Java

Porovnání standardů SQL4 a ODMG 3.0

Závěr

- Konvergence SQL4 a ODMG 3.0
- současné implementace ORSŘBD se zatím v počátcích
  - paralela:
    - výtky OOSŘBD - málo databázové
    - výtky ORSŘBD - málo objektové
- chybí vývojové prostředky, nové metodologie
- největší problém a současně výhoda: univerzálnost