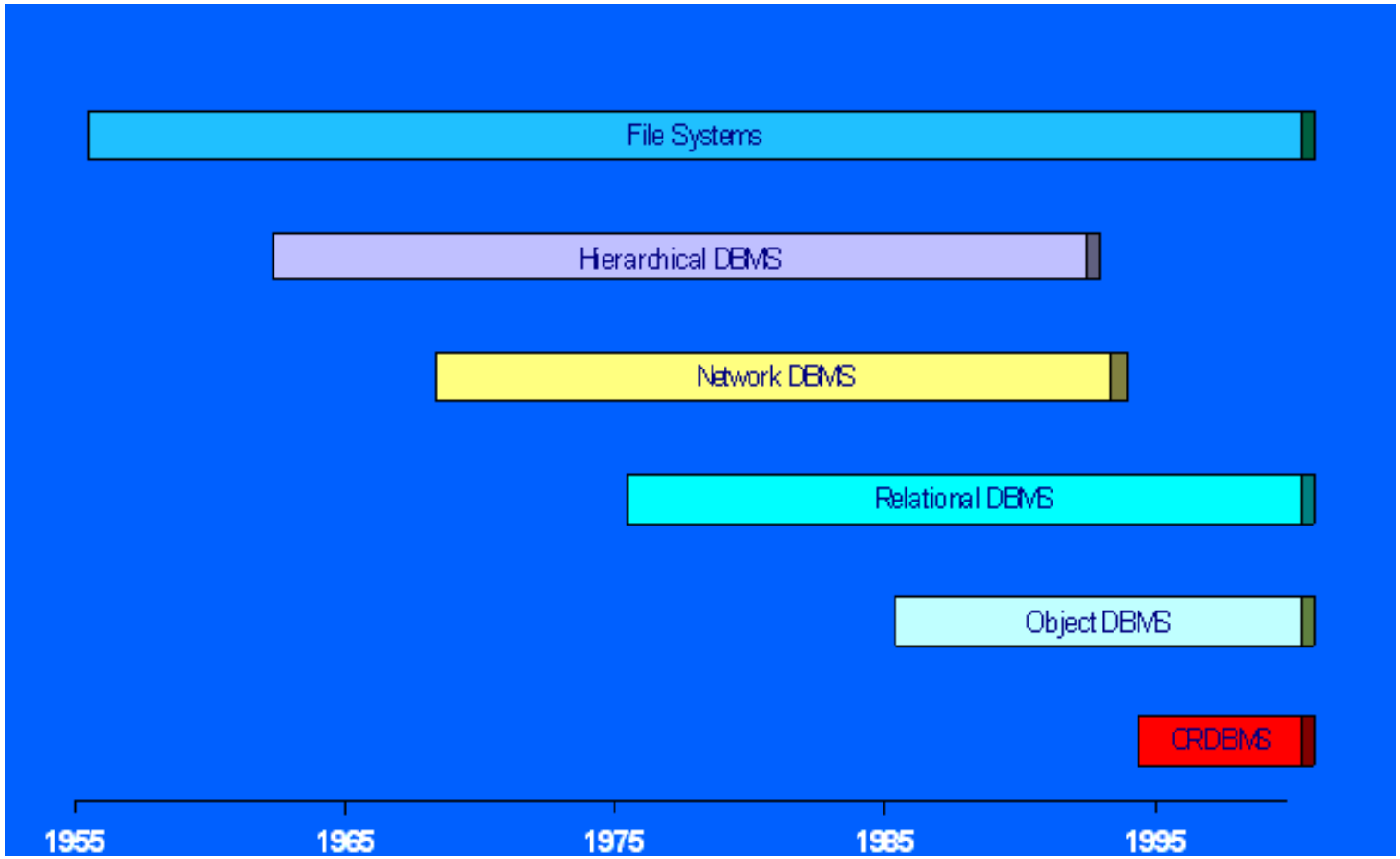
2. Objektově-relační databáze

2.1. Úvod

- **polovina 80.let – dosažení jasné převahy relačních databází nad konkurenčními (síťové, hierarchické). Faktory:**
  - **Jednoduchý datový model**
  - **Formální (matematická) podpora datového modelu a postupu návrhu**
  - **Jednoduché a dostatečně intuitivní modelovací techniky pro návrh (ER diagram)**
  - **Podpora nejvýznamnějších výrobců databázových produktů**
  - **Standardizace – jazyk SQL**
- **konec 80. let – rostoucí vliv paradigmatu objektové orientace (OO) a možnost práce se složitě strukturovanými daty. Faktory:**
  - **Prosazování OO v oblasti programování, návrhu a postupně i analýzy**
  - **Požadavky některých aplikačních oblastí překračující možnosti relačního modelu**

- **důsledek:**
  - **objektové databáze, tj. implementace objektového modelu dat (perzistentních)**
  - **rozšiřování schopností relačních SŘBD nad rámec relačního modelu dat směrem k objektové orientaci**

  - **objektově-relační mapování (Hibernate, Java Data Objects (JDO))**



2.2. Porovnání relačních, objektových a objektově-relačních databází

➢ Relační databáze a RSŘBD

- formálně zavedl Codd, 1970
- standardizace (SQL) od 1986

Model dat:

- relační
- kolekce tabulek
- velmi omezená množina datových typů hodnot (skalární/atomické)
- vztahy mezi řádky tabulek vyjádřeny pomocí cizích a kandidátních klíčů, ne explicitně ukazateli
- pro vztahy M:M je nutná vazební tabulka

Dotazovací jazyk:

- SQL
- neprocedurální (deklarativní)

Výpočetní model:

- založen na hodnotách ve sloupcích tabulek
- žádné reference či ukazatelé
- jednoduchá navigace po tabulce pomocí kurzoru

➢ Objektové databáze a OSŘBD

- umožňují modelování a vytváření perzistentních dat jako objektů
- OSŘBD musí splňovat kritéria SŘBD a musí to být OO systém (viz Manifest OODBS 1995 (není standard))
- neexistuje všeobecně přijatý standard obdobný SQL
- objektové databáze nenahrazují, nýbrž doplňují relační (přesněji dnes zpravidla objektově-relační)

Model dat:

- objektový, neexistuje standard (ODMG-93 byl pokusem)

- **třídy, atributy, operace, jednoznačný OID pro každý perzistentní objekt,**
- **podpora abstraktních datových typů (ADT), zapouzdření a polymorfismu**
- **atributy objektů mohou být jiné objekty → složité typy**
- **vztahy objektů vytvářené pomocí referencí prostřednictvím OID**
- **vztahy M:M lze vytvářet přímo**

Dotazovací jazyk:

- **snaha o vytvoření standardu skupinou ODMG (jazyk OQL)**
- **většinou běžné OO programovací jazyky**

Výpočetní model:

- **významná role OID**
- **navigace po objektové struktuře pomocí referencí prostřednictvím OID**

➢ Objektově-relační databáze a ORSŘBD

- cílem je spojit výhody relačního a objektového modelu
- objektově-relační rysy ve standardu SQL-1999
- podpora objektových rozšíření u významných dodavatelů SŘBD

Model dat:

- snaha o obohacení tabulek o objektovou orientaci
- z hlediska datové struktury jde o obecnější (vnořené) relace (nested relational model)
- perzistentní data jsou stále v tabulkách, ale hodnoty mohou mít bohatší strukturu definovanou jako ADT
- ADT zapouzdřuje data a operace
- je zavedena obdoba OID, která umožňuje vytvářet nový typ vazeb mezi tabulkami

Dotazovací jazyk:

- SQL-1999

Výpočetní model:

- navigace po tabulkách pomocí kurzoru i pomocí referencí

2.3. Vývoj standardu SQL

- 1975 - Sequel v System R
- 1986 - standard ANSI, 1986 - standard ISO- SQL/86, dominantní úloha dialektu SQL firmy IBM (DB2)
- 1989 – integritní dodatek (Integrity Addendum) - SQL/89,
- 1992 – SQL/92, tři úrovně souladu (Entry/Intermediate/Full)
- 1996 – dodatek týkající se uložených modulů (PSM/96)
- 1999 – SQL1999 – objektově-relační rysy
- 2002 – podpora multimédií a dolování v datech - SQL/MM – části: Framework, Full Text, Spatial, Still image, Data mining (1999 – 2002)
- 2003 – SQL2003 – podpora OLAP, podpora XML (SQL/XML)
- 2006 – rozšířená podpora XML v SQL/XML (XQuery)
- 2008 – dokončení SQL/XML a dalších částí (~4000 s.).
- 2011 – podpora temporálních dat

2.3.1. SQL: 1999 a pozdější (jen některé vybrané rysy)

➢ Nové relační rysy

- nové datové typy
  - LOB (Large Objects) – BLOB (Binary LOB), CLOB (Character LOB)
  - BOOLEAN – TRUE, FALSE, UNKNOWN
  - ARRAY – uspořádané pole (původně omezené délky bez zanoření, od 2003 omezení odstraněna)

```
dny_v_tydnu VARCHAR(10) ARRAY(7)
```

  - MULTISET (2003) s predikáty IS MEMBER OF, IS A SET, IS A MULTISET OF a množinovými, agregačními a některými dalšími operacemi

```
resitele VARCHAR(20) MULTISET
```

  - ROW – možnost definování složeného sloupce

```
CREATE TABLE Osoba (
   id INTEGER,
   jmeno ROW (
```

```
        krestni VARCHAR(20),
        prijmeni VARCHAR(30),
    ),
...
)

SELECT O.jmeno.krestni
FROM Osoba O
```

- nové predikáty
  - SIMILAR – obdoba LIKE, ale s regulárními výrazy

```
WHERE verze SIMILAR
    'SQL-(86|89|92|1999|2003)|SQL(1|2|3)'
```

- další vylepšení
  - více pohledů je aktualizovatelných
  - rekurzivní dotaz
  - SAVEPOINT – pro částečný návrat transakce
- nové typy DB objektů - databázové triggery, sekvence, …

➢ Objektové rysy (1999)

- strukturované uživatelem definované typy (UDT)
  - obdoba třídy:
    - atributy - zabudovaných typů, kolekce, UDT
    - metody - pro přístup k atributům systémem generované funkce pro čtení a modifikaci, metody mohou být polymorfní
    - porovnání hodnot UDT na základě uživatelem definovaných funkcí
    - jednoduchá dědičnost

```
CREATE TYPE zamestnanec_t
  UNDER t_osoba
AS (os_cislo    INTEGER,
    plat        REAL )
INSTANTIABLE
NOT FINAL
REF (os_cislo)
INSTANCE METHOD
  Zvys_plat(abs_proc BOOLEAN,castka REAL)
     RETURNS REAL
```

– typ sloupce tabulky může být UDT

```
CREATE TABLE Zamestnanci_fakulty (
   id    INTEGER PRIMARY KEY,
   zam   zamestnanec_t,
   ustav CHAR(3) FOREIGN KEY REFERENCES Ustav)
```

– hodnota v tomto případě nemá jednoznačnou identitu (obdobu OID)

– odkaz na atributy přes funkce nebo tečkovou notací

```
WHERE plat(zam) > 20000
WHERE zam.plat > 20000
```

- typované tabulky
  - tabulky s řádky nesoucími hodnoty strukturovaného UDT (odpovídají objektům)

    ```
    CREATE TABLE Zamestnanci OF zamestnanec_t
    ```

    – každý řádek tabulky má jednoznačnou identitu (hodnota typu REF)
- hodnoty REF
  - hodnoty identifikující řádek v typované tabulce

```
CREATE TABLE Ustav (
...zkratka CHAR(3) PRIMARY KEY
   vedouci REF(zamestnanec_t),
...)
```

REF hodnoty mají vždy zadaný typ, který odkazují, mohou mít zadané jméno typované tabulky - rozsah (scope)

REF hodnoty umožňují zpřístupnit atributy hodnoty strukturovaného typu, kterou identifikují

```
SELECT vedouci -> jmeno
FROM Ustav
WHERE zkratka = 'UIFS'
```

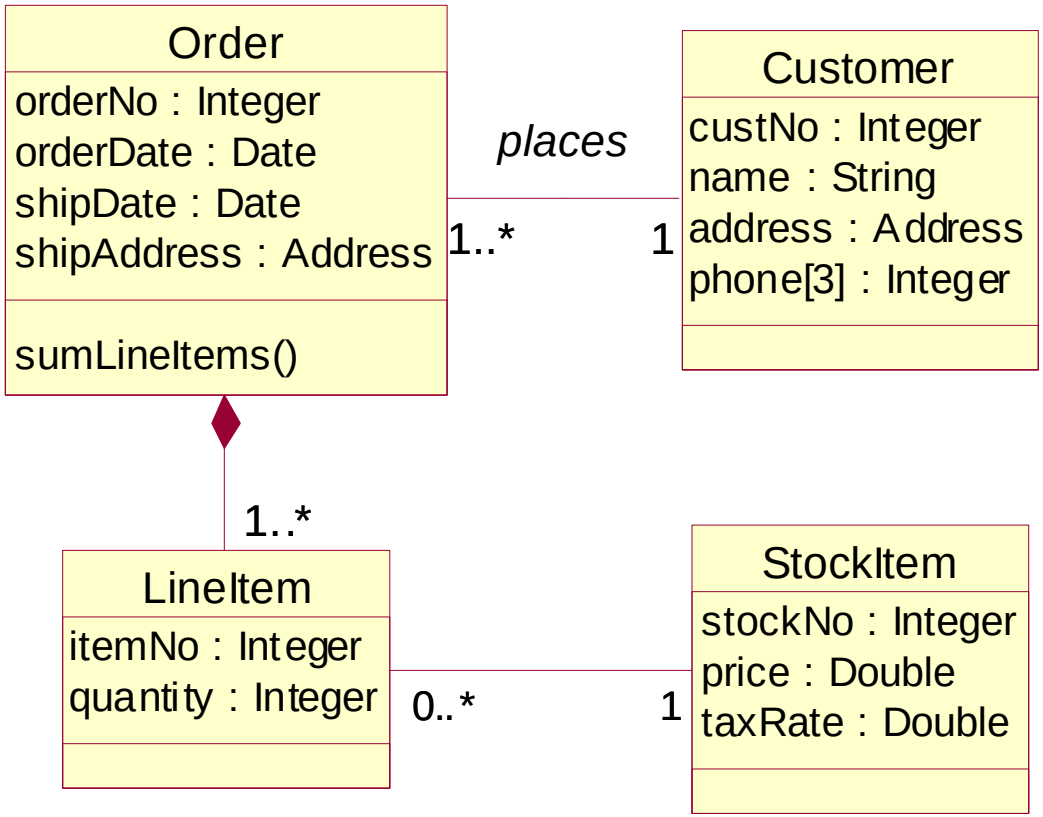
2.4. Objektové rysy databázového serveru Oracle

– objektová rozšíření od verze Oracle8 (1997)

– dva typy UDT:

- objektové typy
- kolekce – pole, vnořená tabulka

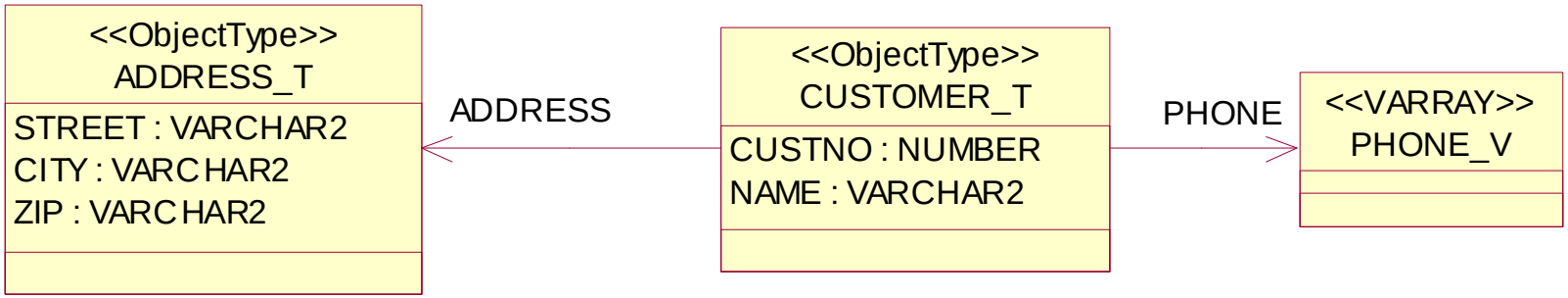
Př)



➢ Objektový typ

- jméno, atributy, metody (členské, statické, porovnání)

Př) Využijeme profil pro objektově-relační modelování v Rational Rose

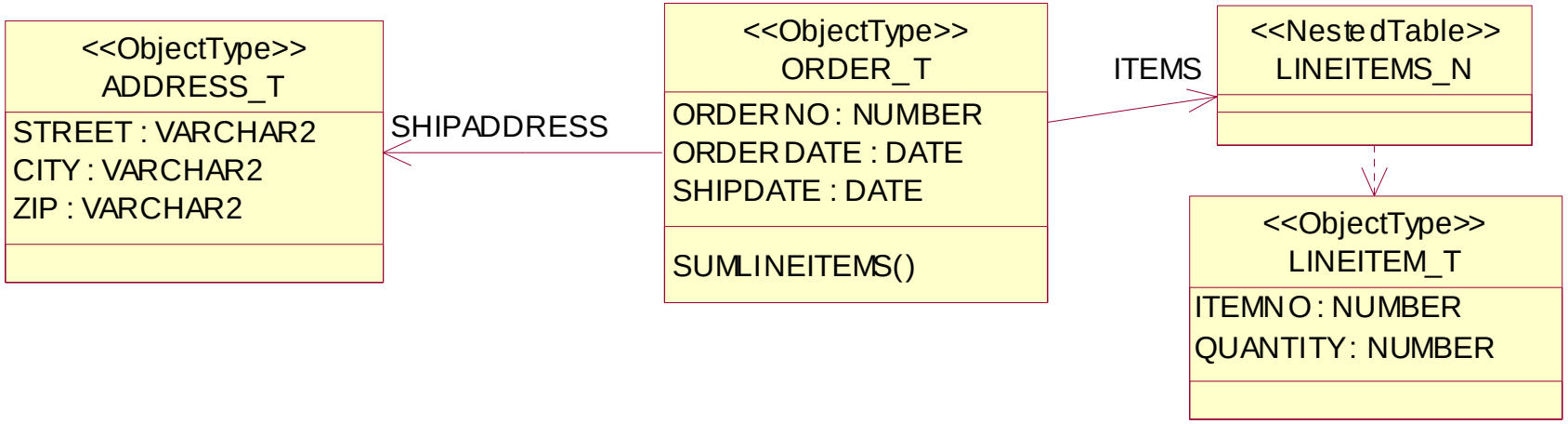



```
CREATE OR REPLACE TYPE ADDRESS_T AS OBJECT (
     STREET VARCHAR2(50),
     CITY VARCHAR2(50),
     ZIP NUMBER);

CREATE OR REPLACE TYPE PHONE_V AS VARRAY(3) OF NUMBER;

CREATE OR REPLACE TYPE CUSTOMER_T AS OBJECT (
     CUSTNO NUMBER,
     NAME VARCHAR2(50),
     ADDRESS ADDRESS_T,
     PHONE PHONE_V);
```

➢ Vnořená tabulka

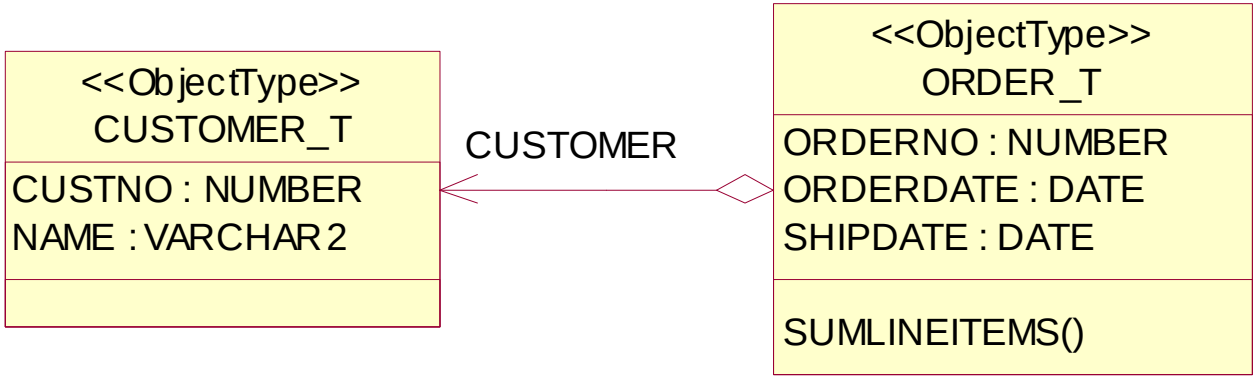



```
CREATE OR REPLACE TYPE LINEITEM_T AS OBJECT (
     ITEMNO NUMBER,
     QUANTITY NUMBER);

CREATE TYPE LINEITEMS_N AS TABLE OF LINEITEM_T;

CREATE OR REPLACE TYPE ORDER_T AS OBJECT (
     ORDERNO NUMBER,
     ORDERDATE DATE,
     SHIPDATE DATE,
     SHIPADDRESS ADDRESS_T,
     ITEMS LINEITEMS_N,
     MEMBER FUNCTION SUMLINEITEMS RETURN NUMBER);
```

➢ Reference REF




```
CREATE OR REPLACE TYPE ORDER_T AS OBJECT (
     ORDERNO NUMBER,
     ORDERDATE DATE,
     SHIPDATE DATE,
     CUSTOMER REF CUSTOMER_T,
     MEMBER FUNCTION SUMLINEITEMS RETURN NUMBER);
```

➢ Tabulka se sloupcovým objektem

- **vytvoření**

```
--Předpokládejme, že vytvoříme tabulku CUSTOMER ne jako
--objektovou
CREATE TABLE CUSTOMER (
      CUSTNO NUMBER PRIMARY KEY,
      NAME VARCHAR2(40),
      ADDRESS ADDRESS_T,
      PHONE PHONE_V);
```

CUSTOMER

CUSTNO	NAME	STREET	TOWN	ZIP	PHONE(1)	PHONE(2)	PHONE(3)

- **vložení řádku – použití konstruktoru sloupcového objektu**

```
INSERT INTO CUSTOMER
  VALUES (1, 'Jan Novák', ADDRESS_T('Božetěchova 2',
'Brno', 61266), PHONE_V(4205442450, 4206023344));
```

- **dotazování**

```
SELECT *
FROM CUSTOMER;
```

```
SELECT ADDRESS
FROM CUSTOMER
WHERE CUSTNO=1;
```

- modifikace

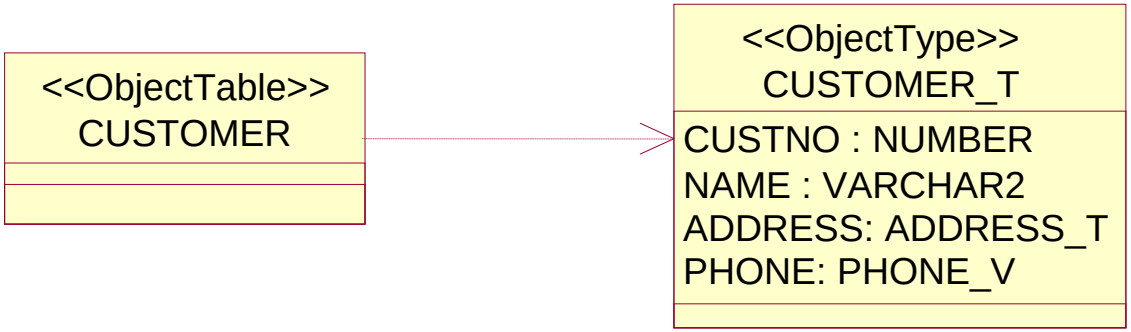
```
UPDATE CUSTOMER c
SET c.ADDRESS.STREET='Božetěchova 1'
WHERE CUSTNO=1;

UPDATE CUSTOMER
SET PHONE = PHONE_V(4205442450, 4206023345)
WHERE CUSTNO=1;
```

- rušení

```
DELETE FROM CUSTOMER
WHERE CUSTNO=1;
```

➢ Objektová tabulka



- **vytvoření**

```
--Předpokládejme, že vytvoříme tabulku CUSTOMER jako
--objektovou
CREATE TABLE CUSTOMER OF CUSTOMER_T;
```

CUSTOMER

OID	CUSTNO	NAME	STREET	TOWN	ZIP	PHONE(1)	PHONE(2)	PHONE(3)

- **vložení řádku – použití konstruktoru nebo jako u relační tabulky**

```
INSERT INTO CUSTOMER
  VALUES (CUSTOMER_T(1, 'Jan Novák',
    ADDRESS_T('Božetěchova 2', 'Brno', 61266),
    PHONE_V(4205442450, 4206023344)));
```

```
INSERT INTO CUSTOMER
  VALUES (2, 'Jiří Novák', ADDRESS_T('Božetěchova 1',
   'Brno', 61266), PHONE_V(4205442444, 4206023341));
```

- dotazování

```
SELECT *
FROM CUSTOMER;

SELECT *
FROM CUSTOMER c
WHERE c.ADDRESS.STREET='Božetěchova 2';
```

- modifikace – jako u relační tabulky se sloupcovým objektem

```
UPDATE CUSTOMER c
SET c.ADDRESS.STREET='Božetěchova 1'
WHERE CUSTNO=1;
```

- rušení – jako u relační tabulky

```
DELETE FROM CUSTOMER
WHERE CUSTNO=1;
```

- **Dědičnost, polymorfismus, (NOT) FINAL, (NOT) INSTANTIABLE, ...**

```
CREATE TYPE PERSON_T AS OBJECT (
   NAME VARCHAR2(100),
   SSN NUMBER)
NOT FINAL;

CREATE TYPE EMPLOYEE_T UNDER PERSON_T (
   DEPARTMENT_ID NUMBER,
   SALARY NUMBER)
NOT FINAL;

CREATE TYPE PART_TIME_EMP_T UNDER EMPLOYEE_T (
     NUM_HRS NUMBER);
```

- **Práce se vnořenými tabulkami v SQL**
  - **vytvoření**

```
CREATE TABLE ORDER_OBJ_N OF ORDER_T
NESTED TABLE
   ITEMS
STORE AS
   NESTED_ITEMS;
```

ORDERNO	ORDERDATE	SHIPDATE	SHIPADDRESS	ITEMS	CUSTOMER

DATA1	DATA2	DATA3	DATA4	NT_DATA
...	...	...	...	A
...	...	...	...	B
...	...	...	...	C
...	...	...	...	D
...	...	...	...	E

Storage Table

NESTED_TABLE_ID	Values
B	B21
B	B22
C	C33
A	A11
E	E51
B	B25
E	E52
A	A12
E	E54
B	B23
C	C32
A	A13
D	D41
B	B24
E	E53

- vložení řádku

```
-- pro vnořenou tabulku konstruktor
-- pro referenci použití funkce REF
INSERT INTO ORDER_OBJ_N VALUES (
1, SYSDATE, SYSDATE+10,
ADDRESS_T('Cejl 10','Brno',60200),
LINEITEMS_N(
   LINEITEM_T(1,1),
   LINEITEM_T(2,1),
   LINEITEM_T(3,2)),
(SELECT REF(c)
FROM CUSTOMER c
WHERE c.name='Jiří Novák'));
```

- dotazování

```
-- dotaz - dereference pomocí funkce DEREF, pokud mají
být
-- ve výsledku hodnoty odkazovaného objektu
SELECT ORDERNO, ORDERDATE, SHIPDATE, ITEMS, DEREF
(o.CUSTOMER)
FROM ORDER_OBJ_N o;
```

```
-- relační pohled na tabulku s kolekcemi (operace
-- „unnest”) – konstrukce TABLE (+) je obdoba vnějšího
-- spojení, zobrazí obsah kolekce i je-li prázdná
SELECT o.ORDERNO, o.ORDERDATE, o.SHIPDATE, i.*
FROM ORDER_OBJ_N o, TABLE(o.ITEMS) (+) i;
```

- modifikace vnořené tabulky – použití konstrukce TABLE

```
-- modifikace vnořené tabulky - vložení řádku
INSERT INTO TABLE (
   SELECT ITEMS FROM ORDER_OBJ_N WHERE ORDERNO=1)
VALUES (
   LINEITEM_T(4, 5));

-- modifikace vnořené tabulky - modifikace řádku
UPDATE TABLE (
   SELECT ITEMS FROM ORDER_OBJ_N WHERE ORDERNO=1) i
SET
   VALUE(i) = LINEITEM_T(4, 6)
WHERE
   i.ITEMNO = 4;
```

- rušení

```
-- rušení
```

```
DELETE FROM ORDER_OBJ_N
WHERE ORDERNO = 1;
```

- vytvoření těla metody – v PL/SQL nebo jiném programovacím jazyce, včetně Javy. Těla metod v PL/SQL a Javě se ukládají do databáze.

```
CREATE OR REPLACE TYPE BODY ORDER_T AS
MEMBER FUNCTION SUMLINEITEMS RETURN NUMBER IS
s INTEGER;
BEGIN
   SELECT SUM(QUANTITY)
   INTO s
   FROM TABLE(ITEMS);
   RETURN s;
END SUMLINEITEMS;
END;

-- dotaz s využitím metody
SELECT o.ORDERNO "Číslo objednávky", o.SUMLINEITEMS()
"Pocet kusů"
FROM ORDER_OBJ_N o
ORDER BY ORDERNO;
```

2.5. Využití objektových rozšíření SQL

- **Standardizované balíky rozšiřující SQL**

Př) SQL Multimedia and Application Packages (SQL/MM)

část 3 Spatial

```
CREATE TYPE ST_Geometry
AS (
ST_PrivateDimension SMALLINT DEFAULT -1,
ST_PrivateCoordinateDimension SMALLINT DEFAULT 2,
ST_PrivateIs3D SMALLINT DEFAULT 0,
ST_PrivateIsMeasured SMALLINT DEFAULT 0
)
NOT INSTANTIABLE
NOT FINAL
METHOD ST_Dimension()…,
… )
```

Podtypy: ST_Point, ST_Curve, ST_Polygon,
ST_LineString, …

➢ Aplikace pracující se „složitě“ strukturovanými daty

Př) Podpora pro práci s prostorovými daty v Oracle

```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES SDO_ORDINATE_ARRAY);

CREATE TYPE sdo_point_type AS OBJECT (
X NUMBER,
Y NUMBER,
Z NUMBER);
CREATE TYPE sdo_elem_info_array AS VARRAY (1048576) of
NUMBER;
CREATE TYPE sdo_ordinate_array AS VARRAY (1048576) of
NUMBER;
```

2.6. Podpora objektů v programátorských prostředích

- **SQL**
  - viz předchozí
- **PL/SQL**
  - Lze používat na většině míst PL/SQL, kde mohou být zabudované typy (analogicky k uloženým procedurám a funkcím)
- **Oracle Call Interface (OCI)**
  - knihovna funkcí pro přístup k relačním i objektově-relačním datům z C (a základ rozhraní pro další jazyky), správa vyrovnávací paměti objektů (VPO - object cache). Přístup:
    - asociativní – vstupní/výstupní proměnné SQL příkazů
    - navigační – navigace po struktuře objektů ve VPO (dereference a umístění objektu do VPO, načtení vzájemně vázaných objektů do VPO jedním voláním, uložení změn z VPO na server, ...)
- **Pro*C/C++**
  - umožňuje používat objektové typy v programech psaných v C a C++, mapování do typů C pomocí Oracle Type Translator

➢ Oracle C++ Call Interface (OCCI)

- C++ API umožňující rysy C++ pro přístup k databázi. Přístup:
  - asociativní
  - navigační – transparentní přístup k objektům v databázi, materializace ve vyrovnávací paměti aplikace.

➢ Oracle Objects for OLE (OO4O)

- přístup k objektům a kolekcím na serveru prostřednictvím OLE objektů z prostředí podporujících protokol COM.

➢ Java:

- JDBC
  - slabě typované mapování použitím java.sql.Struct nebo oracle.sql.STRUCT
  - silně typované uživatelské třídy, které musí implementovat rozhraní java.sql.SQLData (přenositelnost JDBC 2.0) nebo oracle.sql.ORAData

- **JPublisher**
  - **generuje javovské třídy pro objektové typy (implementací rozhraní java.sql.SQLData nebo oracle.sql.ORAData nebo slabě typovaného přístupu)**
- **Oracle SQLJ**
  - **opět možnost slabého nebo silného typování a použití nástroje JPublisher; typová kontrola při kompilaci SQLJ.**
- **Specializovaná rozhraní**
  - **např. Oracle Java Spatial API**
    - **balík *oracle.spatial.geometry* – podpora pro práci s SDO_GEOMETRY**
  - **analogicky Oracle Java Multimedia API**
    - **balík *oracle.ord.im* – podpora pro práci s typy reprezentujícími zvuk, obrázky, video atd.**

Literatura

1. Silberschatz, A., Korth H.F, Sudarshan, S.:Database System Concepts. Fourth Edition. McGRAW-HILL. 2001, str. 335 – 360.
2. Eisenberg, A., Melton, J.: SQL:1999, formerly known as SQL3. SIGMOD Record, Vol. 28, No. 1, March 1999. str.131 – 138.
3. SQL standards. Dostupné na http://www.wiscorp.com/SQLStandards.html (září 2008).
4. Price, J.: Oracle Database 10g SQL. Oracle Press McGra-Hill/Osborne. 2004. str. 367 – 426.
5. Dokumentace Oracle 11g Release 2. Dostupné na http://www.oracle.com/pls/db112/homepage (září 2013) - zejména Application Developer's Guide – Object-Relational Features, JDBC Developer's Guide and Reference.
6. http://www.odbms.org