10. Objektinės technologijos reliacinėse DBVS

- Reliacinės DBVS tiek Lietuvoje, tiek ir visame pasaulyje paplitę nepalyginamai plačiau negu kitų tipų DBVS.
- „Grėsmę viešpatavimui“ kelia tik objektinės, o pastaraisiais metais NewSQL ir NoSQL DBVS.
- RDBVS kūrėjai pripažįsta OT privalumus ir diegia jas, išplėsdami RDBVS ir SQL galimybes.
- Šiuolaikinės RDBVS vadinamos objektinėmis-reliacinėmis DBVS.

10.1. Objektinės duomenų bazės

- RDB grindžiamas griežtais matematiniais apibrėžimais.
- ODB griežto teorinio pagrindo neturi.
- ODB būdingi objektinės technologijos principai:
  - **Objektai**. Duomenys organizuojami objektais.
  - **Klasės.** RDB duomenų tipas keičiamas hierarchine klase.
  - **Paveldimumas**. Objektai paveldi protėvių savybes.
  - **Atributai**. Taip modeliuojamos objekto charakteristikos.
  - **Pranešimai ir metodai**. Objektai bendrauja pranešimais.
  - **Inkapsuliacija**. Vidinė objekto sandara paslėpta.
  - **Objektų identifikatoriai**. Objektai atskiriami pagal OID.

Pagrindinis ODB privalumas - dalykinei sričiai būdingos duomenų struktūros vaizdavimas objektais vieningas: DB-je ir taikomojoje programoje.

ODB trūkumai:

– OID - sąvoka yra artima nuorodos į duomenis sąvokai, kuri buvo būdinga iki–reliacinėms DB;

– ODB neturi griežto matematinio pagrindo – standartizavimo problemos.

Komercinių RDBVS kūrėjai, atsižvelgdami į OT privalumus ir siekdami išlaikyti turimas pozicijas rinkoje, įdiegė daug sąvokų, kurios anksčiau buvo būdingos tik ODBVS.

RDBVS, išsaugodamos visus reliacinio modelio privalumus, įgavo naujų bruožų (nauji SQL sakiniai), tapo objektinėmis-reliacinėmis DBVS (ORDBVS).

10.2. Objektinės-reliacinės duomenų bazės

Siekiant perimti geruosius ODB bruožus, RDB-se įdiegta:

- **Dideli duomenų objektai.**
- **Vartotojo apibrėžiami duomenų tipai**.
- **Struktūriniai duomenų tipai**.
- **Vartotojo apibrėžiamos funkcijos**.
- **Objektų identifikatoriai**.
- **Duomenų aktyvumas**.

Papildžius ORDB dideliais objektais ir kitomis objektinėmis priemones, pradėti kurti **reliaciniai plėtiniai** (angl. *relational extender*):

DB2 – „Relational Extenders“

Oracle – „Data Options“

Informix – „Data Blades“.

Reliaciniai plėtiniai, funkcionuojantys ORDBVS DB2:

- **Grafinis plėtinys**. Palaikomi formatai: GIF, JPEG, BMP, TIFF. Pradiniai peržiūrai gali pateikti nedidelį vaizdo fragmentą, atlikti kontekstinę vaizdų paiešką ir pan.
- **Video plėtinys**. Leidžia operuoti populiariausiais formatais: MPEG1, MPEG2, AVI, Quicktime; automatiškai segmentuoti įrašus pagal scenos pasikeitimus, surasti scenos atstovą ir pan.
- **Audio plėtinys**. Duomenų formatai: AIFF, MIDI, WAVE, MP3. Funkcijos leidžia sužinoti daugelį įrašų charakteristikų.

- **Erdvinis plėtinys**(angl. *spatial extender*) erdvinei informacijai apie geografinius objektus saugoti ir apdoroti.
- **Tekstų plėtinys** (angl. *text extender*), kuris:

– palaiko įvairiais tekstų procesoriais (Microsoft Word, Word Perfect, AmiPro) sukurtus tekstus;

– sukuria spec. tipo indeksus kontekstinei (pagal žodžius, jų dalis ir frazes) paieškai tekstuose;

– tekstai gali būti sudaryti įvairiomis kalbomis (anglų, vokiečių, prancūzų ir pan.);

– paieškose gali būti atsižvelgiama į žodžių sinonimus bei formas.

Užklausa, naudojanti tekstų plėtinio funkciją **CONTAINS**:

SELECT *Nr*, *Pavardė*
FROM *Vykdytojai*
WHERE CONTAINS(*CV*,
'"stažuotė" & ("Vokietija" | "Prancūzija") & NOT "JAV"'
) > 0

10.3. Naujų duomenų tipų apibrėžimas

Naudojant bazinius tipus, sakiniu **CREATE TYPE** galima apibrėžti naujus tipus (angl. *user-defined type*):

CREATE TYPE *USD* **AS DECIMAL**(15, 2) ;

CREATE TYPE *EUR* **AS DECIMAL**(15, 2) ;

Automatiškai sukuriama:

- funkcija naujo tipo reikšmės gavimui iš bazinio;
- funkcija bazinio tipo reikšmės gavimui iš naujojo;
- galimybė lyginti tarpusavyje apibrėžiamojo tipo reikšmes: =, <, > ir kt.

Raktiniai žodžiai **DISTINCT** ir **WITH COMPARISONS** paseno.

Vartotojo apibrėžti tipai vartojami kaip standartiniai:

ALTER TABLE *Projektai* **ADD** *Vertė_USD* *USD*
ALTER TABLE *Projektai* **ADD** *Vertė_EUR* *EUR*

Sukūrus naują tipą, automatiškai generuojama funkcija naujojo tipo reikšmėms sukurti,

INSERT INTO *Projektai*
(*Nr, Pavadinimas, Vertė_USD, Vertė_EUR*)
VALUES(4, 'Inventoriaus apskaita',
USD(10000.00), *EUR*(40000.00))

SELECT *Pavadinimas* **FROM** *Projektai*
WHERE *Vertė_USD* > *USD*(1000.00)

SELECT *Pavadinimas* **FROM** *Projektai*
WHERE *Vertė_EUR* > *EUR*(1000.00)

SELECT *Pavadinimas* **FROM** *Projektai*
WHERE *Vertė_USD* > 1000 – **neteisingas**

SELECT *Pavadinimas* **FROM** *Projektai*
WHERE *Vertė_USD* < *Vertė_EUR* – **neteisingas**

Lyginant, reikalingas tipų suvienodinimas:

SELECT *Pavadinimas* **FROM** *Projektai*
WHERE DECIMAL(*Vertė_USD*) <
DECIMAL(*Vertė_EUR*)

– sintaksė teisinga, tačiau logiškai ji neprasminga.

SELECT *Pavadinimas* **FROM** *Projektai*
WHERE DECIMAL(*Vertė_USD*) <
DECIMAL(*Vertė_EUR*) * 1.12.

Tačiau, jei kursas yra kintamas, tai pastarąją užklausą tenka parametrizuoti.

Paprasčiausias būdas parametrizuoti – naujos funkcijos.

Struktūriniai duomenų tipai

CREATE TYPE *Adresas* **AS**
(*Miestas* **VARCHAR**(63),
Gatvė **VARCHAR**(32),
Namas **SMALLINT**) ;

ALTER TABLE *Vykdytojai*
ADD *NamųAdresas Adresas*
CHECK ((*NamųAdresas*).*Namas* > 0) ;

UPDATE *Vykdytojai* **SET** *NamųAdresas* =
ROW('Kaunas', 'Naugarduko', 24)
WHERE *Nr* = 1;

SELECT *Pavardė*, *Adresas* **FROM** *Vykdytojai*
WHERE *Nr* = 1;

Nr	*Pavardė*	*Adresas*
1	Jonaitis	('Kaunas', 'Naugarduko', 24)

SELECT *Pavardė*, *Adresas*().*Gatvė*, *Adresas*().*Namas*
FROM *Vykdytojai* **WHERE** *Nr* = 1;

Nr	*Pavardė*	*Gatvė*	*Namas*
1	Jonaitis	Naugarduko	24

10.4. Naujų funkcijų apibrėžimas

Reiškinys *USD*(10) + *USD*(20) yra neteisingas, jei duomenų tipui *USD* neapibrėžta sudėties operacija.

Neapibrėžus funkcijų, reikšmes galima tik lyginti
USD(10) < *USD*(20)

CREATE FUNCTION "+" (*USD*, *USD*)
RETURNS *USD*
SOURCE "+" (**DECIMAL**(15, 2), **DECIMAL**(15, 2))

CREATE FUNCTION "/"(*USD*, **DECIMAL**(10, 5))
RETURNS *USD*
SOURCE "/"(**DECIMAL**(15, 2), **DECIMAL**(10, 5))

```
UPDATE Projektai
SET  Vertė_USD = Vertė_USD + USD(1000)
WHERE  Nr = 4
```

```
CREATE FUNCTION Sum_USD(USD)
RETURNS USD
SOURCE SUM(DECIMAL(15, 2))
```

```
SELECT Sum_USD(Vertė_USD)
FROM Projektai
```


Apibrėžiamas funkcijas galima realizuoti SQL reiškiniais:

```
CREATE FUNCTION USD_to_EUR(X USD)
RETURNS EUR
      LANGUAGE  SQL
      CONTAINS  SQL
      RETURN EUR(DECIMAL(X / 1.12))
```

CONTAINS SQL – reiškinyje nėra vykdoma jokia užklausa.

Alternatyva – **READS SQL DATA**.

Skaliarinė funkcija, nurodyto argumentu vykdytojo visiems projektams skiriamų valandoms apskaičiuoti:

```
CREATE FUNCTION Sum_Valandos(Nr  INTEGER)
    RETURNS INTEGER
    LANGUAGE SQL
    NOT DETERMINISTIC
    NO  EXTERNAL  ACTION
    READS SQL DATA
    RETURN (SELECT SUM(Valandos)
              FROM Vykdymas WHERE Vykdytojas = Nr)
```


NOT DETERMINISTIC – apskaičiuojant funkcijos reikšmę, tai pačiai argumento reikšmei gali būti gaunami skirtingi rezultatai;

NO EXTERNAL ACTION – nepasikeis jokio išorinio objekto būsena

```
SELECT Pavardė, Sum_Valandos(Nr) FROM Vykdytojai
```

SELECT frazėje esantys reiškiniai skaičiuojami kiekvienai eilutei.

Todėl, ši užklausa struktūriškai panaši į užklausą su koreliuota daline užklausa.


```
CREATE FUNCTION Inches2M(in  DOUBLE)
    RETURNS DOUBLE
    LANGUAGE SQL
    CONTAINS  SQL
    DETERMINISTIC
    NO  EXTERNAL  ACTION
    RETURN (in * 0.0254) ;
```

```
CREATE FUNCTION GetAddress(ad Address)
    RETURNS CHAR(100)
    LANGUAGE SQL CONTAINS  SQL
    DETERMINISTIC  NO  EXTERNAL  ACTION
    RETURN (ad().Miestas || ', ' || ad().Gatvė || ' g. ' ||
              CAST(ad().Namas AS CHAR(3)) ;
```


Galimas ir lankstesnis valiutų konvertavimas:

```
CREATE TABLE Valiutų_kursai
    (Kodas CHAR(3) NOT  NULL  RIMARY  KEY,
     Kursas DECIMAL(10, 4)) ;
```

```
INSERT  INTO  Valiutų_kursai
VALUES  ('USD', 1.12), ('CAD', 1.51);
```

Valiutų_kursai

Kodas	*Kursas*
USD	1.12
CAD	1.51


```
CREATE FUNCTION USD_to_EUR(Vertė  USD)
  RETURNS EUR
  LANGUAGE SQL
  NOT DETERMINISTIC
  NO  EXTERNAL  ACTION
  READS SQL DATA
  RETURN (SELECT Vertė/Kursas
            FROM Valiutų_kursai
            WHERE Kodas = 'USD')
```

Taip apibrėžtos funkcijos, pasikeitus kursui, nereikia perkurti. Pasikeitus kursui, tereikia atnaujinti lentelės *Valiutų_kursai* duomenis.

Daugelis DBVS leidžia apibrėžti **išorines** (angl. *external*) **funkcijas**.

Išorine vadinama funkcija, kuri:

- realizuota kuria bazine programavimo kalba,
- jos vykdomasis (mašininis) kodas yra DLL-e,
- duomenų bazėje yra saugoma tik funkcijos sąsajos apibrėžimas.

Tarkime, litų konvertavimo į valiutą funkcija, realizuota **C** kalba. Paruošus vykdomąjį modulį su funkcijos kodu, DB-ei pranešama apie funkcijos egzistavimą ir sąsają:

```
CREATE FUNCTION Litai_Valiuta(Litai)
RETURNS Valiuta
EXTERNAL NAME 'konvertavimas ! litai_valiuta'
  LANGUAGE C
  PARAMETER STYLE SQL
  DETERMINISTIC
  NO SQL
  NO EXTERNAL ACTION
```


JAVA kalba:

```
CREATE FUNCTION Litai_Valiuta(Litai)
RETURNS Valiuta
EXTERNAL NAME
    'Valiutos:Konvertavimas.litai2valiuta(
                        java.math.BigDecimal)'
  LANGUAGE JAVA
  PARAMETER STYLE JAVA
  DETERMINISTIC
  NO EXTERNAL ACTION
```


- **EXTERNAL NAME** nurodoma:
  - ✓ C: bibliotekos (DLL) vardas (*konvertavimas*) ir joje esančios funkcijos vardas(*litai_ valiuta*);
  - ✓ JAVA: JAR vardas (*Valiutos*), klasės vardas (*Konvertavimas*), jos metodo vardas (*litai2valiuta*) ir argumento klasės vardas (*java.math.BigDecimal*).
- **LANGUAGE** – programavimo kalba (**C** | **JAVA**) nuo kurios, pvz. gali priklausyti parametrų perdavimo tvarka;
- **PARAMETER STYLE** – argumentų atitikimo stilius;
- **NO SQL** – funkcijos kūne nėra SQL sakinių (**CONTAINS SQL**).


```
SELECT Pavadinimas,
       Vertė_ Valiuta + EUR_Valiuta(Vertė_Litais)
       AS "Bendra vertė valiuta"
FROM Projektai
```

DBVS, pasiruošdama vykdyti užklausą su kreipiniu į išorinę funkciją, sintaksinę analizę atlieka pasitelkusi tik sakiniu **CREATE FUNCTION** apibrėžtą sąsają.