14. OBJEKTOVĚ-RELAČNÍ DATABÁZE

Relační databáze

- stojí na matematickém základu – pojem relace
- základem je relační model dat
- relace (R, R*) – R je schéma relace a obsahuje pojmenované atributy včetně domény každého atributu – doména – obor hodnot které může atribut nabývat a rozsah těchto hodnot (omezení)
  - R* je tělo relace obsahující n-tice a představující hodnoty atributů, které jsou spolu v relaci
  - $R = \{A1:D1, A2:D2, ..., An:Dn\}$ → atribut pojmenovaný nad svou dimenzí
  - $R^* \subseteq D1 \times D2 \times ... \times Dn$
- relace představují tabulky a atributy jsou sloupce a n-tice jsou řádky
- relační databáze obsahuje kolekci tabulek/relací
- relace mají kandidátní klíče – nejmenší hodnoty atributů které jsou v rámci dané relace/tabulky unikátní
- z kandidátních klíčů (jednoho nebo více) se vybere a stanoví primární klíč relace kde každá n-tice (= řádek) je touto hodnotou jednoznačně identifikovaná
- vztahy relací jsou dány cizími klíči které se vážou na hodnoty primárních klíčů jiných relací × nejsou zde přímé reference :(ani ukazatelé
- vztah M:N nutno řešit vazební tabulkou :(
- velmi omezená množina datových typů sloupců :(
- jednoduchá navigace pomocí kurzoru :)
- jsou ale rozšířené a podporované napříč výrobci DB systémů :)
- standardizováno – SQL standard – rozšířeno třeba v SQL:1999 nebo SQL:2003 + se přidal typ XML Type
- dotazovací jazyk je standardizovaný SQL :) → viz tabulky a objektové... EXT a IND :)
- + integritní omezení (↳ udržují sémantiku DB), triggery (→ on change, ...), uložené procedury, pohledy, materializované pohledy...

– tabulky by měly být normalizované a splňovat alespoň 3NF

Objektově relační mapování

- základem je klasická relační databáze se vším co k ní patří
- na aplikační úrovni je provedeno automatické mapování / bindování do objektů
- struktura objektu/třídy odpovídá struktuře relační tabulky a tak je to mezi sebou aplikačně mapováno jak pro ukládání a změny tak pro získávání objektů a relačních DB
- třeba ORM Eloquent od Laravelu

Objektové databáze

- ačkoliv byla snaha o standardizaci tak se tak nikdy nestalo a proto se příliš nerozšířily :(
- modelem je objektový model na objektovém základě
- vznikly v době rozvoje OO a OO paradigmat s potřebou persistentně uchovávat objekty někde v databázi
- začala se dělat objektová analýza a modelování a poté i programování a vznikla tak nutnost ukládat data jako objekty
- oproti relačním databázím jsou o mnoho složitější a jak nemají standard a nerozšířily se tolik, tak je moc výrobců DB systémů nepodporuje :(
- po objektové struktuře se dá navigovat přímo pomocí OID :)
- OID je Object ID a je to jednoznačný identifikátor každého objektu
- vztahy jsou dány napřímo pomocí referencí :)
- M:N se tedy také řeší napřímo :)
- atributy mohou být mnoha datových typů včetně ADT (abstraktní datový typ) a také jiných objektů :)
  ↳ i složité datové typy
  ↳ včetně zapouzdření operací nad nimi a polymorfismu atd...
- nemohly nahradit relační databáze ale doplnit je - žít vedle nich
- nemají standardizovaný dotazovací jazyk :(většinou se s nimi pracuje pomocí klasických OO programovacích jazyků s nějakou nadstavbou - JAVA atd..!

Objektově - relační databáze

- snaha spojit výhody objektových a relačních databází a konečně to nějak standardizovat
- vychází z klasických relačních databází které jsou rozšířeny o OO prvky a vlastnosti
- standardizace od SQL:1999
- přidány ADT zapouzdřující jak data tak operace
- používá se obdoba OID pro identifikaci persistentních objektů
- lze tedy pomocí OID definovat i jiné typy vztahů mezi tabulkami - řeší M:N
- navigace pomocí kurzoru i reference
- data jsou v relačních tabulkách ale mohou mít bohatší a rozsáhlejší strukturu a přičemž tabulky nemusí být normalizované - porušují NF

SQL:1999

- triggery
- nové datové typy: CLOB, BLOB
- savepoint - pro částečný návrat transakce
- predikát SIMILAR - jako LIKE ale i s regexama

! - hlavní říct že přinesl UDT a že to je jako třída s atributy a metodami (jsou zapouzdřené) a že pak tohle UDT může být sloupec relační tabulky (přístup přes tečkovou notaci) nebo celá typová tabulka - odkazuje se referencemi REF a řádky pak tabulky jsou identifikovány OID

Objektové rysy

- zavádí ADT a konkrétní UDT - uživatelsky definovaný datový typ

UDT - něco jako třída v OO
- má dědičnost a atributy a základní metody

```
CREATE TYPE zamestnanec_t UNDER osoba_t
AS
(
 os_cislo   INTEGER,
 plat       REAL
)
INSTANTIABLE  - kdyby nebyl tak je možné jej jen dědit a toho zděděného dělat instance
NOT FINAL  - lze dědit
REF (os_cislo)  - čím se na to odkazuje
INSTANCE METHOD zvys_plat (castka REAL) RETURNS REAL
```

tohle od SQL:2003

⇒ je to prostě jako třída s atributy a metodami

- <u>sloupec tabulky</u> nyní může být datového typu zamestnanec_t

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

- na atributy se pak dostaneme tečkovou notací

```
... FROM zamestnanci_fakulty
WHERE zam.plat > 20000;
```

```
CREATE TABLE zamestnanci
             OF zamestnanec_t
```

- nebo může být <u>typová tabulka</u> kde každý řádek je prostě ten UDT a je identifikovaný pomocí OID

REF - reference na řádky typové tabulky   vedouci REF (zamestnanec_t)
- v klasické tabulce máme sloupec kde se odkazuje na nějaký záznam v typové tabulce

SQL:2003

- zavádí pole (ARRAY) a multimnožiny (MULTISET) s jejich operacemi SET, UNION, INTERSECT, ...
  ↳ jsou to oboje typy kolekcí

  UNION ∪, INTERSECT ∩

- UDT může být FINAL nebo NOT FINAL
  - umožní nebo zakáže dědění

[- XML Type podpora]