

DBS – Transformace konceptuálního schématu na relační

Michal Valenta

Katedra softwarového inženýrství FIT
České vysoké učení technické v Praze
©Michal Valenta, 2013

BI-DBS, ZS 2014/15

<https://edux.fit.cvut.cz/courses/BI-DBS/>



CREATE TABLE

- CREATE TABLE

```
CREATE TABLE tabulka (  
    sloupec datovy_typ [io_sloupce [, io_sloupce...]],  
    ...  
    [io_tabulky [, io_tabulky ...]] );
```

```
CREATE TABLE VYPUJCKY (  
    c_kopie CHAR (3) NOT NULL,  
    c_zak    CHARACTER (6) NOT NULL,  
    cena     DECIMAL(5,2),  
    rod_c    CHARACTER (10) NOT NULL,  
    datum_v  DATE);
```

ALTER TABLE, DROP TABLE

- ALTER TABLE

ADD sloupec, DROP sloupec, ALTER sloupec,
ADD CONSTRAINT io, DROP CONSTRAINT io

```
ALTER TABLE KINA ADD pocet_mist INTEGER;
```

- DROP TABLE

DROP TABLE tabulka [CASCADE CONSTRAINTS]

```
DROP TABLE KINA CASCADE CONSTRAINTS;
```

Integritní omezení v SQL

- IO sloupce:
 - ▶ NOT NULL
 - ▶ DEFAULT
 - ▶ UNIQUE
 - ▶ PRIMARY KEY
 - ▶ REFERENCES
 - ▶ CHECK
- IO tabulky - stejné jako IO sloupce
(NOT NULL je speciálním případem CHECK)
složené IO vždy na úrovni tabulky
- pojmenování IO
není syntakticky nutné, ale vřele doporučované

Příklad DDL s IO

```
DROP TABLE KINA CASCADE CONSTRAINTS;
CREATE TABLE KINA ...
...
CREATE TABLE PŘESTAVENÍ
    (NAZEV_K Char_Varying(20) NOT NULL,
    JMENO_F Char_Varying(20) NOT NULL,
    DATUM date NOT NULL,
    CONSTRAINT PREDSTAVENI_PK
        PRIMARY KEY (NAZEV_K, JMENO_F)
    CONSTRAINT PREDSTVENI_KINA_FK
        FOREIGN KEY (NAZEV_K) REFERENCES KINA,
    CONSTRAINT PREDSTAVENI_FILMY_FK
        FOREIGN KEY (JMENO_F) REFERENCES FILMY);
```

Referenční integrita a kaskádní DELETE / UPDATE

Referenční integrita (cizí klíč) – v SQL 4 možné způsoby reakce:

```
[ CONSTRAINT constraint_name ]  
  FOREIGN KEY ( column_name [, ... ] )  
  REFERENCES reftable [ ( refcolumn [, ... ] ) ]  
  [ ON DELETE action ] [ ON UPDATE action ]  
action ::= [NO ACTION | RESTRICT |  
           CASCADE | SET NULL | SET DEFAULT]
```

```
CREATE TABLE order_items (  
  product_no integer REFERENCES products  
                ON DELETE RESTRICT,  
  order_id integer REFERENCES orders  
                ON DELETE CASCADE,  
  quantity integer,  
  PRIMARY KEY (product_no, order_id));
```

Poznámka: implementace tohoto rysu nebývá kompletní.

Typy dat v SQL

- numerické
- textové
- rozsáhlé znakové řetězce (CLOB)
- rozsáhlé bitové řetězce (BLOB)
- datum a čas
- interval

poznámka

NULL je prvkem každého datového typu

tříhodnotová logika: TRUE, FALSE, UNKNOWN

Konverze : implicitní, explicitní (pomocí funkce CAST)

Typy dat v SQL

přesné numerické typy

INTEGER, SMALLINT, NUMERIC, DECIMAL, NUMER

- DECIMAL(p,q)

p ... přesnost

q ... měřítko

aproximativní numerické typy

FLOAT

REAL

DOUBLE PRECISION

Typy dat v SQL

znakové řetězce

CHARACTER(n) (délka n, zprava mezery)

CHARACTER VARYING(n)

aproximativní numerické typy

DATE

TIMESTAMP

INTERVAL

- algoritmus převodu konceptuálního schématu na relační bývá součástí modelovacích nástrojů (Oracle Data Modeller, Enterprise Architekt, ...)
- pomocí nastavení je obvykle možné chování vestavěných generátorů výrazně upravit
- převod některých konstrukcí konceptuálního modelu (například ISA hierarchie) má několik možných variant, z nichž žádná není úplně přesná; optimální varianta pro konkrétní situaci závisí na dalších okolnostech (často používané dotazy, způsoby uložení dat, počet atributů v podtypech a nadtypu,...)
- v některých případech (povinnost nedeterminantu ve vztahu) nemáme na úrovni relačního modelu dostatečně efektivní mechanismus, který by kontrolu zajistil, proto se na kontrolu některých IO rezignuje

Entity

- název entity → název relace
- atributy entity → atributy relace
- domény atributů entity se “namapují” na domény relačních atributů
- povinnost atributů entity → NOT NULL na relační úrovni
- atributy identifikátoru entity → PRIMARY KEY
- alternativní klíče → UNIQUE
- u slabých entit je třeba do klíče přibrat identifikátory identifikačních vlastníků

Vztahy

- jediná možnost provázání dat ze dvou relací je **referenční integrita** (FOREIGN KEY)
- podle kardinality a parciality je třeba použít vztahové tabulky a IO NOT NULL a UNIQUE

Převod silné entity

ENTITA

- # * identifikátor
- * poviny_atribut
- nepoviny_atribut

Relační zápis (zjednodušené a používané i pro testy)

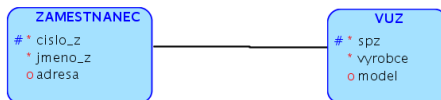
entita(identifikátor, povinny_atribut, nepovinny_atribut)

SQL

```
CREATE TABLE entita (  
    identifikator integer,  
    povinny_atribut varchar(20) NOT NULL,  
    nepovinny_atribut varchar(40),  
    CONSTRAINT entita_pk PRIMARY KEY (identifikator) );
```

Poznámka: domény atributů se zadávají již na konceptuální úrovni, ale v diagramech se obvykle nezobrazují.

Vztah 1:1, obě entity povinná účast



Relační zápis

zamestnanec_vuz(cislo_z, jmeno_z, adresa, spz, vyrobce, model)

SQL

```
CREATE TABLE zamestnanec_vuz (  
  cislo_z integer CONSTRAINT zamestnanec_vuz_pk PRIMARY KEY,  
  jmeno_z varchar(20) NOT NULL,  
  adresa varchar(40),  
  spz varchar(20) NOT NULL CONSTRAINT zamestnanec_vuz_uk UNIQUE,  
  vyrobce varchar(40) NOT NULL,  
  model varchar(40));
```

Poznámka: vzhledem k povinnosti členství na obou stranách jedna tabulka.

Vztah 1:1, 1 entita povinná účast



Relační schéma

zamestnanec(cislo_z, jmeno_z, adresa)

vuz(spz, vyrobce, model, cislo_z)

$\text{vuz}[\text{cislo_z}] \subseteq \text{zamestnanec}[\text{cislo_z}]$

SQL:

```
CREATE TABLE zamestnanec (  
  cislo_z integer CONSTRAINT zamestnanec_pk PRIMARY KEY,  
  jmeno_z varchar(20)  
  adresa varchar(40));
```

```
CREATE TABLE vuz (  
  spz varchar(20) CONSTRAINT vuz_pk PRIMARY KEY,  
  vyrobce(40) NOT NULL,  
  model varchar(40),  
  cislo_z integer NOT NULL UNIQUE,  
  CONSTRAINT zamestnanec_vuz_fk  
  FOREIGN KEY (cislo_z) REFERENCES zamestnanec(cislo_z));
```

Vztah 1:1, nepovinná účast



1. možnost

Jako v předchozím případě. Jediný rozdíl je ten, že atribut **cislo_z** v tabulce **vuz** bude **nepovinný**.

2. možnost

zamestnanec(cislo_z, jmeno_z, adresa)

vuz(spz, vyrobce, model)

zamestnanec_vuz (cislo_z, spz)

$\text{zamestnanec_vuz}[\text{spz}] \subseteq \text{vuz}[\text{spz}]$

$\text{zamestnanec_vuz}[\text{cislo_z}] \subseteq \text{zamestnanec}[\text{cislo_z}]$

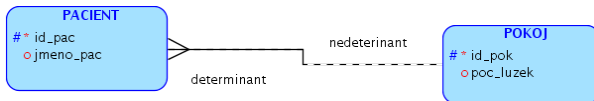
spz v relaci **zamestnanec_vuz** je **NOT NULL** a **UNIQUE**

SQL:

```
CREATE TABLE vuz (...); CREATE TABLE zamestnanec (...);  
CREATE TABLE zamestnanec_vuz (  
  cislo_z integer PRIMARY KEY REFERENCES zamestnanec (cislo_z),  
  spz varchar(20) NOT NULL UNIQUE REFERENCES vuz (spz));
```

Poznámka: v notaci relačního modelu nejsou obvykle vyznačena integritní omezení NOT NULL a UNIQUE. Zde jsou však pro správnou transformaci obě tato integritní omezení nezbytná!

Vztah 1:N, povinná účast determinantu



Relační zápis

pacient(id_pac, jmeno_pac, id_pok)
pokoj(id_pok, poc_luzek)

$\text{pacient}[\text{id_pok}] \subseteq \text{pokoj}[\text{id_pok}]$
 id_pok v relaci **pacient** NOT NULL

SQL

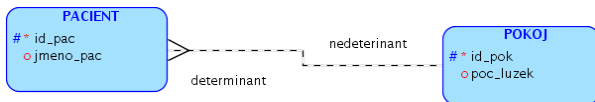
```
CREATE TABLE pacient (  
  id_pac integer CONSTRAINT pacient_pk PRIMARY KEY,  
  jmeno_pac varchar(20) NOT NULL,  
  id_pokoj integer NOT NULL);
```

```
CREATE TABLE pokoj (  
  id_pok integer CONSTRAINT pokoj_pk PRIMARY KEY,  
  poc_luzek integer NOT NULL);
```

```
ALTER TABLE pacient ADD CONSTRAINT pacient_pokoj_fk  
  FOREIGN KEY (id_pok) REFERENCES pokoj(id_pok));
```

Poznámka: informaci o parcialitě nedeterminantu ztrácíme.

Vztah 1:N, nepovinná účast determinantu



1. možnost

Jako v předchozím případě. Rozdíl je ten, že atribut **id_pok** v tabulce **pacient** bude **nepovinný**.

2. možnost

pacient(id_pac, jmeno_pac)

pokoj(id_pok, poc_luzek)

umisteni(id_pac, id_pok)

$\text{umisteni}[\text{id_pac}] \subseteq \text{pacient}[\text{id_pac}]$

$\text{umisteni}[\text{id_pok}] \subseteq \text{pokoj}[\text{id_pok}]$

id_pok v relaci **umisteni** **NOT NULL**

SQL:

```
CREATE TABLE pacient (...); CREATE TABLE pokoj (...);
```

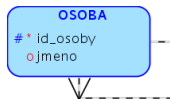
```
CREATE TABLE umisteni (
```

```
id_pac integer PRIMARY KEY REFERENCES pacient (id_pac),
```

```
id_pokoj integer NOT NULL REFERENCES pokoj (id_pok));
```

Poznámka: informaci o parcialitě nedeterminantu opět ztrácíme.

Rekurzivní vztah



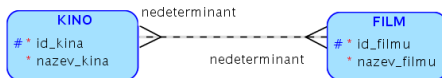
Relační zápis

osoba(id_osoby, jmeno, manager_id)
osoba[manager_id] \subseteq osoba[id_osoby]

SQL

```
CREATE TABLE osoba (  
  id_osoby integer PRIMARY KEY,  
  jmeno varchar(30),  
  manager_id integer);  
  
ALTER TABLE osoba  
  ADD CONSTRAINT osoba_manager_fk  
    FOREIGN KEY (manager_id) REFERENCES osoba (id_osoby));
```

Vztah M:N



Relační zápis

kino(id_kina, nazev_kina)

film(id_filmu, nazev_filmu)

predstaveni(id_kina, id_filmu)

$\text{predstaveni}[\text{id_kina}] \subseteq \text{kino}[\text{id_kina}]$

$\text{predstaveni}[\text{id_filmu}] \subseteq \text{film}[\text{id_filmu}]$

M:N vždy pomocí vztahové tabulky.

SQL

```
CREATE TABLE kino (...); CREATE TABLE film (...);
```

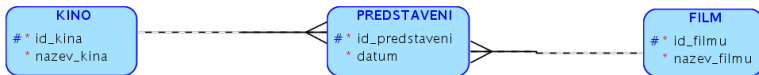
```
CREATE TABLE predstaveni (  
  id_kina integer REFERENCES kino (id_kina),  
  id_filmu integer REFERENCES film (id_filmu),
```

```
  CONSTRAINT predstaveni_pk PRIMARY KEY (id_kina,id_filmu));
```

Poznámka 1: informaci o parcialitě nedeterminantů opět ztrácíme.

Poznámka 2: jeden film v jenom kině nejvýše jednou? Zřejmě spíše dekompozice, viz dále.

Vztah M:N – dekompozice kvůli identifikaci vztahu, silná entita



Relační zápis

kino(id_kina, nazev_kina)

film(id_filmu, nazev_filmu)

predstaveni(id_predstaveni, datum, id_kina, id_filmu)

$\text{predstaveni}[\text{id_kina}] \subseteq \text{kino}[\text{id_kina}]$

$\text{predstaveni}[\text{id_filmu}] \subseteq \text{film}[\text{id_filmu}]$

SQL

CREATE TABLE kino (...); CREATE TABLE film (...);

CREATE TABLE predstaveni (

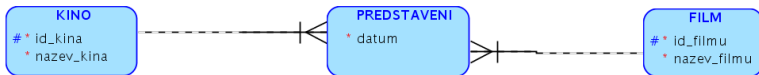
id_predstaveni integer **PRIMARY KEY**,

datum date NOT NULL,

id_kina integer NOT NULL REFERENCES kino (id_kina),

id_filmu integer NOT NULL REFERENCES film (id_filmu));

Vztah M:N – dekompozice kvůli identifikaci vztahu, slabá entita



Relační zápis

kino(id_kina, nazev_kina)

film(id_filmu, nazev_filmu)

predstaveni(datum, id_kina, id_filmu)

$\text{predstaveni[id_kina]} \subseteq \text{kino[id_kina]}$
 $\text{predstaveni[id_filmu]} \subseteq \text{film[id_filmu]}$

SQL

```
CREATE TABLE kino (...); CREATE TABLE film (...);
```

```
CREATE TABLE predstaveni (
```

```
datum date,
```

```
id_kina integer NOT NULL REFERENCES kino (id_kina),
```

```
id_filmu integer NOT NULL REFERENCES film (id_filmu),
```

```
CONSTRAINT PRIMARY KEY predstaveni_pk (datum, id_kina, id_filmu));
```

Slabá entita, identifikační závislost



Relační zápis

blok(id_bloku, nazev_bloku)
pokoj(cislo_pokoje, id_bloku)
 $\text{pokoj}[\text{id_bloku}] \subseteq \text{blok}[\text{id_bloku}]$

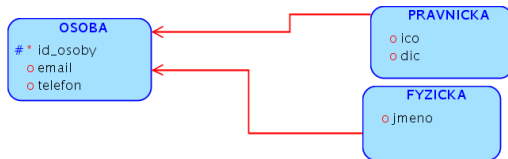
osoba(id_osoby, jmeno_osoby)
profil(id_osoby, fotka)
 $\text{profil}[\text{id_osoby}] \subseteq \text{osoba}[\text{id_osoby}]$

SQL

```
CREATE TABLE blok (...);
CREATE TABLE pokoj (
  cislo_pokoje integer id_bloku integer
    REFERENCES blok (id_bloku),
  PRIMARY KEY (id_bloku, cislo_pokoje));
```

```
CREATE TABLE osoba (...);
CREATE TABLE profil (
  fotka blob
  id_osoby integer
    REFERENCES osoba (id_osoby),
  PRIMARY KEY (id_osoby));
```

ISA hierarchie



varianta 1

osoba(id_osoby, email, telefon, **jmeno**, **ico**, **dic**)

je vhodné zavést rozlišovací atribut (pokud všechny atr. podtypu umožňují NULL, je to nezbytné)

varianta 2

osoba(id_osoby, email, telefon)

fyzicka(id_osoby, jmeno)

pravnicka(id_osoby, ico, dic)

$\text{fyzicka}[\text{id_osoby}] \subseteq \text{osoba}[\text{id_osoby}]$

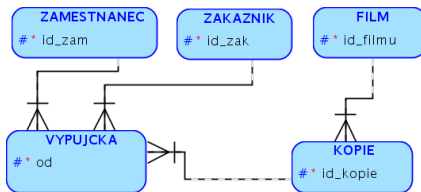
$\text{pravnicka}[\text{id_osoby}] \subseteq \text{osoba}[\text{id_osoby}]$

varianta 3

fyzicka(id_osoby, email, telefon, jmeno)

pravnicka(id_osoby, email, telefon ico, dic)

Migrace identifikátoru entity, složený cizí klíč



Relační zápis

zakaznik(id_zak)

zamestnanec(id_zam)

film(id_filmu)

kopie(id_filmu, id_kopie)

vypujcka(od, id_zak, id_zam, id_filmu, id_kopie)

$\text{kopie}[\text{id_filmu}] \subseteq \text{film}[\text{id_filmu}]$

$\text{vypujcka}[\text{id_zak}] \subseteq \text{zakaznik}[\text{id_zak}]$

$\text{vypujcka}[\text{id_zam}] \subseteq \text{zakaznik}[\text{id_zam}]$

$\text{vypujcka}[\text{id_filmu}, \text{id_kopie}] \subseteq \text{kopie}[\text{id_filmu}, \text{id_kopie}]$

Cyklus a jména rolí při implementaci



Relační zápis

zamestnanec(id_zam)

zvire(id_zvir, krmi_id_zam, sponzoruje_id_zam)

$zvire[krmi_id_zam] \subseteq zamestnanec[id_zam]$

$zvire[sponzoruje_id_zam] \subseteq zamestnanec[id_zam]$

Poznámka: není možné, aby se dva atributy v jedné relaci jmenovaly stejně.