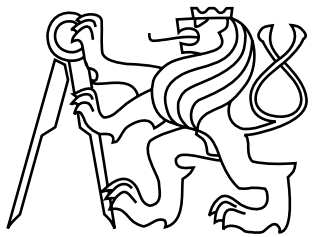
DBS – Transformace konceptuálního modelu na relační

Michal Valenta

Katedra softwarového inženýrství FIT
České vysoké učení technické v Praze

BI-DBS, ZS 2010/11

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



Kontext

- algoritmus převodu konceptuálního modelu 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 parametrech, které jsou mimo konceptuální modelování (často používané dotazy, způsoby uložení dat...)
- v některých případech (povinnost nedeterminantu ve vztahu) nemáme na úrovni relačního modelu dostatečně efektivní mechanismus, který by kontrolu zajistil

Princip

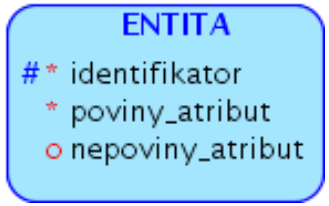
Entity

- název entity $\rightarrow$ název relace
- atributy entity $\rightarrow$ atributy relace
- domény atributů entity se “namapují“ na domény relačních atributů
- povinnost atributů entity $\rightarrow$ NOT NULL na relační úrovni
- atributy identifikátoru entity $\rightarrow$ PRIMARY KEY
- alternativní klíče $\rightarrow$ 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



Relační zápis (zjednodušené a používané pro písemky)

entita(identifikator, 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 účást

Relační zápis

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

SQL

```
CREATE TABLE zamestnanec_vuz (
cislo_z integer CONSTRAINT zamestnantec_vuz_pk PRIMARY KEY,
jneno_z varchar(20) NOT NULL,
adresa varchar(40),
spz varchar(20) NOT NULL CONSTRAINT zamestnantec_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 účást

Relační model

zamestnanec(cislo_z, jmeno_z, adresa)
vuz(spz, vyrobce, model, cislo_z)

vuz[cislo_z] $\subseteq$ zamestnanec[cislo_z]

SQL:

```
CREATE TABLE zamestnanec (
cislo_z integer CONSTRAINT zamestnantec_pk PRIMARY KEY,
jneno_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 zamestnantec(cislo_z));
```

Vztah 1:1, nepovinná účást

1. možnost

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

2. možnost

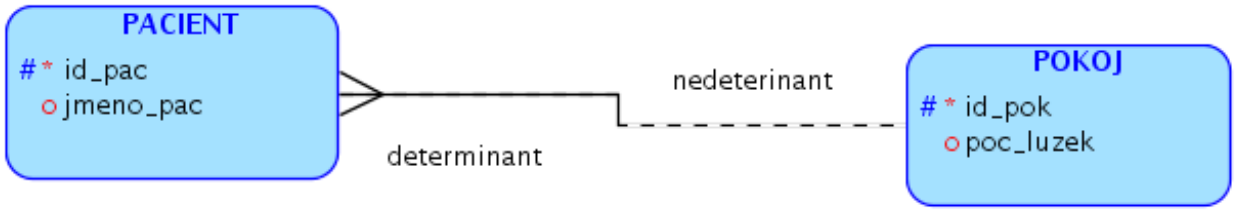
zamestnanec(cislo_z, jmeno_z, adresa)
vuz(spz, vyrobce, model)
zamestnanec_vuz (cislo_z, spz)

zamestnanec_vuz[spz] $\subseteq$ vuz[spz]
zamestnanec_vuz[cislo_z] $\subseteq$ zamestnanec[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)

pacient[id_pok] $\subseteq$ pokoj[id_pok]
id_pok v relaci pacient NOT NULL

SQL

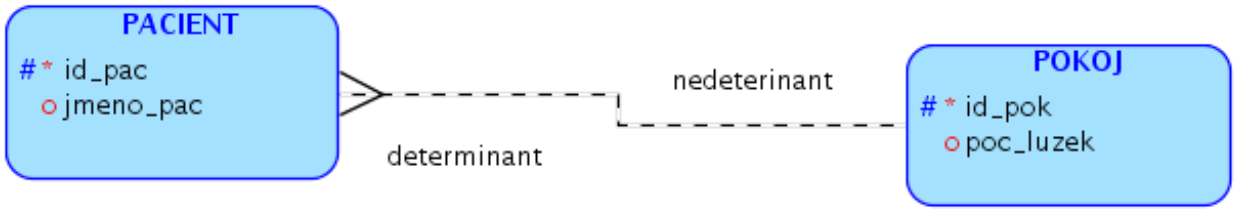
```
CREATE TABLE pacient (
id_pac integer CONSTRAINT pacient_pk PRIMARY KEY,
jmeno_pac varchar(20) NOT NULL,
id_pokoj integer NOT NULL);

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 pokoj bude nepovinný.

2. možnost

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

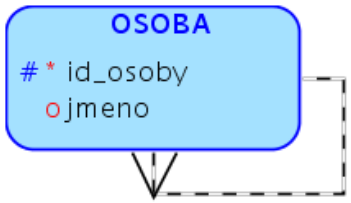
umisteni[id_pac] $\subseteq$ pacient[id_pac]
umisteni[id_pok] $\subseteq$ pokoj[id_pok]
id_pok v relaci umisteni NOT NULL

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

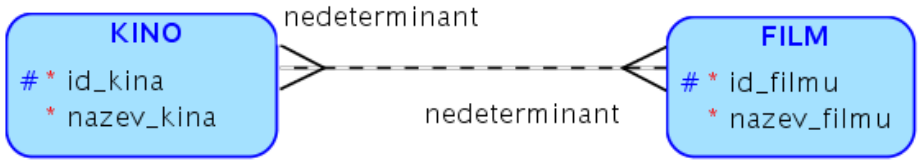
osoba(id_osoby, jmeno, manager_id)
osoba[manager_id] $\subseteq$ osoba[id_osoby

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)

predstaveni[id_kina] $\subseteq$ kino[id_kina]
predstaveni[id_filmu] $\subseteq$ film[id_filmu]
M:N vždy pomocí vztahové tabulky.

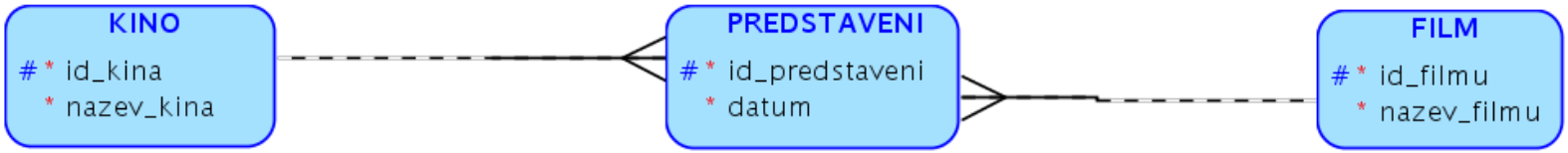
SQL

```
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

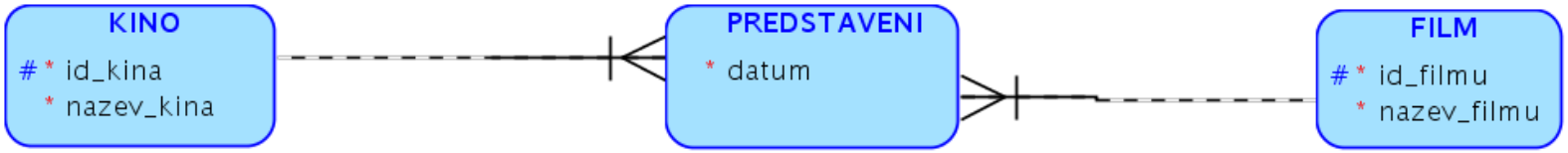
kino(id_kina, nazev_kina)
film(id_filmu, nazev_filmu)
predstaveni(id_predstaveni, datum, id_kina, id_filmu)

predstaveni[id_kina] $\subseteq$ kino[id_kina]
predstaveni[id_filmu] $\subseteq$ film[id_filmu]

SQL

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

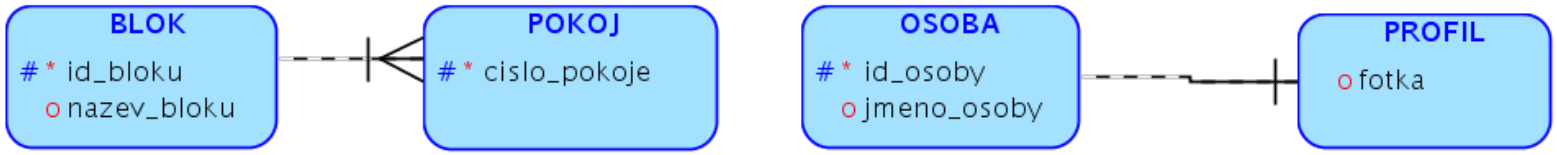
kino(id_kina, nazev_kina)
film(id_filmu, nazev_filmu)
predstaveni(datum, id_kina, id_filmu)

predstaveni[id_kina] $\subseteq$ kino[id_kina]
predstaveni[id_filmu] $\subseteq$ 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)
pokoj[id_bloku] $\subseteq$ blok[id_bloku]

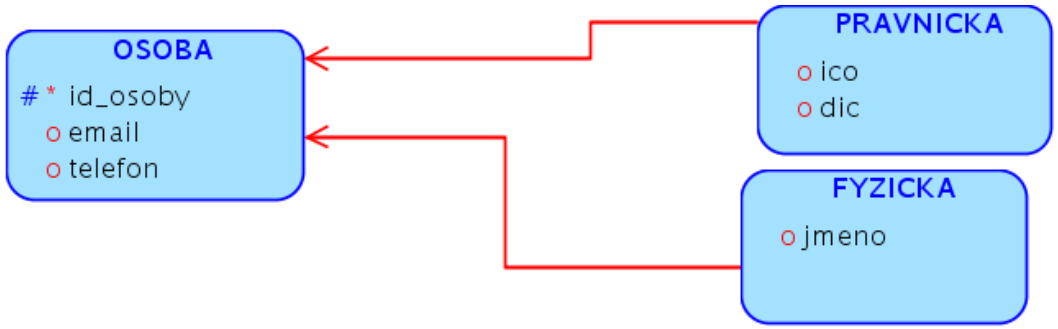
osoba(id_osoby, jmeno_osoby)
profil(id_osoby, fotka)
profil[id_osoby] $\subseteq$ osoba[id_osoby]

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

varianta 2

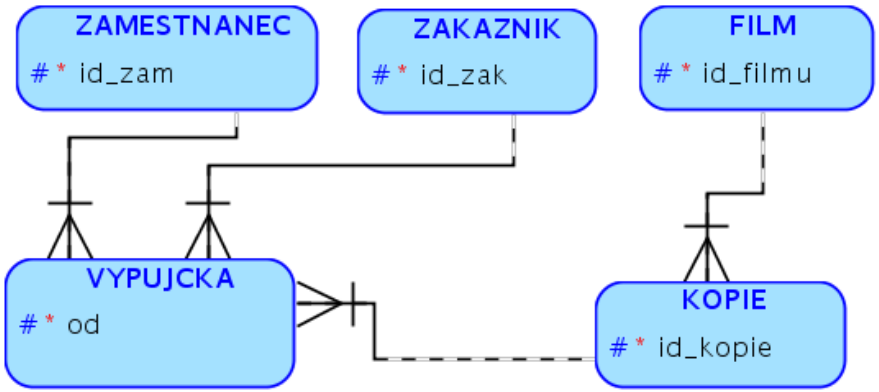
osoba(id_osoby, email, telefon)
fyzicka(id_osoby, jmeno)
pravnicka(id_osoby, ico, dic)

fyzicka[id_osoby] $\subseteq$ osoba[id_osoby]
pravnicka[id_osoby] $\subseteq$ osoba[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)
kopie[id_filmu] $\subseteq$ film[id_filmu]
vypujcka[id_zak] $\subseteq$ zakaznik[id_zak]
vypujcka[id_zam] $\subseteq$ zakaznik[id_zam]
vypujcka[id_filmu, id_kopie] $\subseteq$ kopie[id_filmu, 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 jmenovali stejně.