Databáze

Jiří Zacpal

DEPARTMENT OF COMPUTER SCIENCE
PALACKÝ UNIVERSITY, OLOMOUC

KMI/YUDIT Úvod do informačních technologií

Osnova

- Základní pojmy
- Databázové modely
- Relační algebra
- Jazyk SQL
- Normalizace
- Transakce

Literatura

1. Connolly T., Begg C.: *Database Systems. A Practical Approach to Design, Implementation and Management, 3rd edition.* Addison Wesley, 2002. ISBN 0-201-70857-4
2. Pokorný J.: *Databázové systémy a jejich použití v informačních systémech.* Academia, 1992. ISBN 80-200-0177-8
3. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom: *Database Systems:The Complete Book*
4. Jiří Hronek: Databázové systémy (http://phoenix.inf.upol.cz/esf/ucebni/databa.pdf)

Databáze

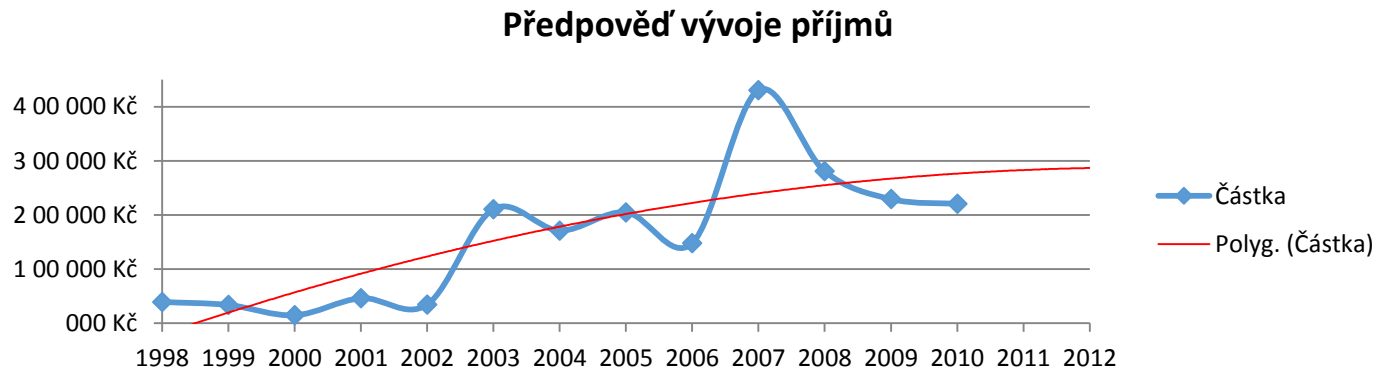
Data vs Informace

- Data
  - údaje získané pozorováním, nebo měřením
  - představují fakta, text, obraz, zvuk, video, nejčastěji v kontextu sledovaného procesu nebo situace
  - nezávislá na uživateli, většinou odráží současný stav reality
- Informace
  - informací se data a vztahy mezi nimi stávají vhodnou interpretací pro uživatele vytvořením struktur, které odhalují uspořádání, vzory, tendence a trendy
  - strukturovaná, organizovaná, shrnutá a interpretovaná data, silně závislé na tom, kdo je požaduje, tedy závislé na uživateli (individuální hloubka znalostí, zkušeností...)

Příklad

- Data
  - informace o faktuře:
    - číslo faktury,
    - datum vystavení,
    - částka
- Informace
  - vývoj příjmů v letech

Cislo_faktury	Datum_vystaveni	Datum_splatnosti	Cislo_odberatele	Castka
Faktura 01-05	22.2.2005	8.3.2005	11	2 400,00
Faktura 01-06	30.1.2006	13.2.2006	11	1 200,00
Faktura 01-07	15.1.2007	29.1.2007	1255	3 000,00
Faktura 01-08	29.1.2008	12.2.2008	13	36 000,00
Faktura 01-09	28.1.2009	11.2.2009	11	1 200,00
Faktura 01-10	28.1.2010	11.2.2010	11	1 200,00



Data vs Informace

- Klíčové body
  - **data** představují stavební kameny pro **informace**
  - **informace** získáváme zpracováním dat
  - **informace** se používají pro odhalení významu **dat**
  - přesnost, relevantnost a aktuálnost **informací** jsou důležité pro dobré **rozhodování**
  - dobré **rozhodování** je klíčové pro život firmy

Vývoj „databází“

- Manuální kartotéka
- Systém souborů
- Databázový systém

Kartotéka

- ukládání dat pomocí karet
- nevýhody:
  - složité vyhledávání
    - Potřebuji najít všechny odběratele, kteří jsou z Olomouce.
  - obtížná změna dat
    - Změnil se manažer pro firmy z Olomouckého kraje.
  - nadbytečnost dat
    - 100 firem z Olomouce, 100x PSČ.
  - složité získávání informací
    - Potřeboval bych všechny firmy, u nichž činil letos obrat více jak 100000 Kč.

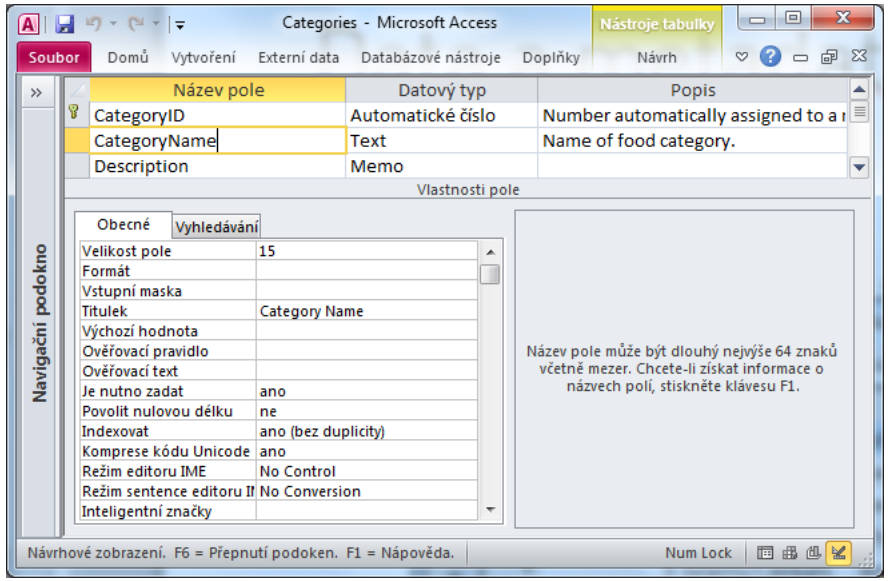
Systém souborů

- pro každou oblast samostatný soubor:
  - zaměstnanci
  - klienti
  - faktury
  - ...
- jednodušší organizace dat a vyhledávání
- problémy:
  - nekonzistence = různé verze dat v různých souborech
  - datové anomálie
    - modifikace – změna je potřeba udělat u všech výskytu dat (i v různých souborech)
    - vkládání – pokud vkládáme např. novou fakturu, musíme k ní přiřadit klienta, který může být nový nebo již založený v souboru klientů = možnost způsobení nekonzistence
    - mazání – při smazání klienta mohou zůstat faktury bez klienta

Databázový systém (DBS)

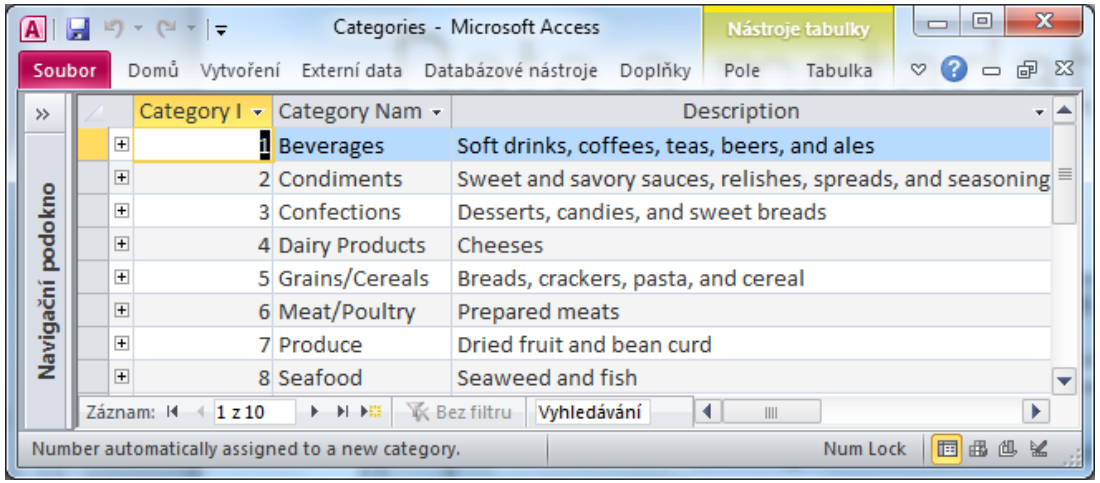
- zkráceně databáze
- tvoří ho:
  - data – syrová fakta z oblasti zájmu uživatele
  - metadata – popisují data a vztahy mezi daty v databázi
  - **programové vybavení** – Systém řízení báze dat (SŘDB) = kolekce programů, která:
    - řídí databázovou strukturu
    - kontroluje přístup k datům uloženým v databázi
    - umožňuje přístup více uživatelům do databáze
    - pomáhá k efektivnější správě dat
  - technické prostředky (hardware)
  - uživatelé

Data a metadata



metadata

data



Funkce SŘBD

- správa metadat
- správa dat
- transformace a prezentace dat
- správa zabezpečení
- víceuživatelský přístup
- zálohování a obnova
- správa integrity dat
- komunikační rozhraní s uživatelem

Uživatelé

- **administrátor databáze, správce dat**
  - centrální kontrola
  - definice paměťové struktury a přístupových metod
  - přidělování práv
  - …
- **aplikační programátor**
  - vytváří aplikační programy
- **znalý uživatel**
  - využívají DBS ad hoc prostřednictvím databázového jazyka
- **naivní uživatel**
  - využívají DBS jen prostřednictvím aplikací

Funkce databáze

- **správa dat**
- základní operace s daty:
  - vkládání nových položek
  - zrušení nepotřebných položek
  - oprava, aktualizace vybraných položek
  - vyhledávání informací v datech

Instance databáze

- zachycuje aktuální stav
- data uložená ke konkrétnímu časovému okamžiku
- základní operace s daty:
  - vkládání nových položek
  - zrušení nepotřebných položek
  - oprava, aktualizace vybraných položek
  - vyhledávání informací v datech

Typy databází

- podle počtu uživatelů:
  - jednouživatelská
  - víceuživatelská
    - databáze pro pracovní skupiny
    - podniková
- podle lokace:
  - centralizovaná
  - distribuovaná
- podle způsobu použití:
  - transakční nebo produkční – pro denní použití
  - datový sklad (warehouse) – pro získávání informací

Vlastnosti dat

- **perzistentní** - trvale uložená, přetrvávají mezi operacemi
- **sdílená** - přístupná více uživatelům
- **integrovaná** - sjednocené informační systémy bez redundance
- **zabezpečená** - přístup a operace s daty mohou být omezeny systémem práv
- **konzistentní** - tytéž údaje na různých místech mají stejnou hodnotu
- **zachovávají integritu** - data odrážejí aktuální realitu
- **spolehlivá** - dají se rekonstruovat po chybě
- **rozsáhlá** - nestačí vnitřní paměť, používají se sofistikované algoritmy pro implementaci operací, …

Datový model

Datový model

- Způsob organizace dat v databázi určuje **datový model**
- datový model je první krok při tvorbě databáze

Význam datových modelů

- datový model je **komunikační nástroj** mezi návrhářem, programátorem a koncovým uživatelem
- dobrý datový model je **základem** pro dobrou databázi
- datový model zachycuje **rozdílné pohledy** na data (informace): ředitel x úředník

Základní prvky datových modelů

- **entity** – objekty reálného světa (osoby, místa, věci, ...)
- **atributy** – vlastnosti entit (jméno, barva, ...)
- **vztah** - popisuje vztah mezi dvěma a více entitami
  - 1:m – př.: „MALÍŘ maluje OBRAZ“ (malíř může namalovat mnoho obrazů, ale obraz má jen jednoho malíře)
  - M:N nebo N:M – př.: „STUDENT má zapsán PŘEDMĚT“ (student si může zapsat více předmětů a jeden předmět si může zapsat více studentů)
  - 1:1 – př.: „ZAMĚSTNANEC vede PRODEJNU“ (zaměstnanec může vést jen jednu prodejnu a prodejna má jen jednoho vedoucího)

Obchodní pravidla

- přesný a jednoznačný popis pravidel a procedur, který jsou zavedeny v prostředí, které chceme modelovat
- příklady:
  - Zákazník může mít na účtu mnoho plateb.
  - Každý účet patří jen jednomu zákazníkovi.
  - Dělník nemůže pracovat více než 10 hodin denně.
  - Student se nemůže zapsat na dva předměty, který probíhají současně.
  - Student se nesmí zapsat na předmět, jestliže nesplnil prerekvizici.

Druhy datových modelů

Druhy datových modelů

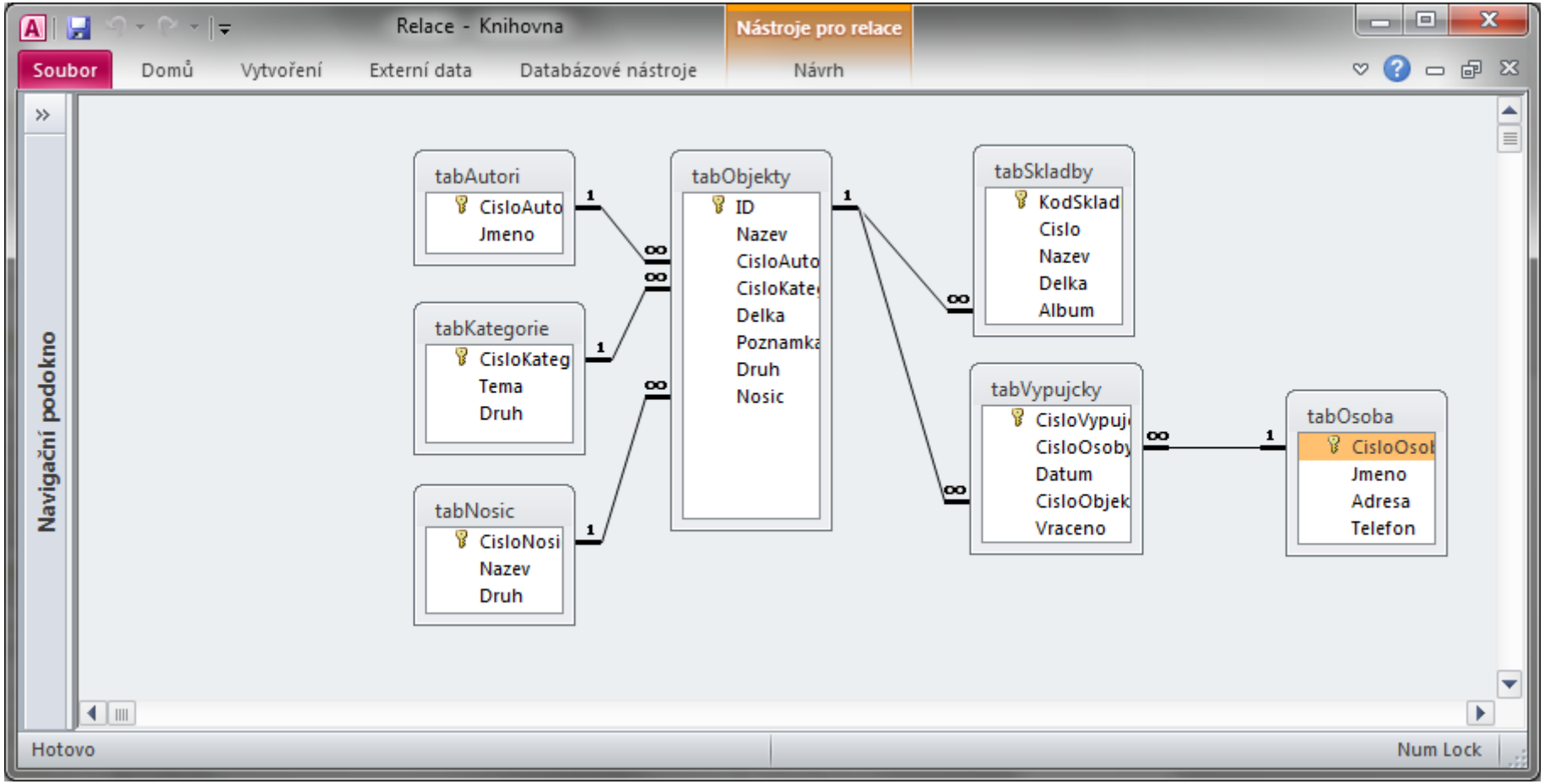
- V současnosti je stále nejrozšířenější model **relační**
  - data jsou strukturována ve formě tabulek
- Dřívější modely byly bližší fyzické (implementační) úrovni
  - kolekce záznamů s vazbami ve formě ukazatelů, podle topologie struktur se dělí na
    - **Hierarchický model** (systém IMS od IBM)
    - **Síťový model** (konference CODASYL, systém IDS od General Electric)
- Postrelační datové modely
  - Objektově-relační (standardy SQL99, SQL2003)
  - Objektový
  - XML (navazuje na hierarchický model)

Relační model

- **Relace** je tabulka se sloupci a řádky
- **Atribut** je pojmenovaný sloupec
- **Doména** je množina přípustných atomických hodnot pro jeden nebo několik atributů
- Vazby mezi entitami jsou reprezentovány opět relacemi. Formálně se s nimi pracuje stejně jako s entitními relacemi.
- **stupeň relace** = počet jejích atributů

Relační model

- relační schéma

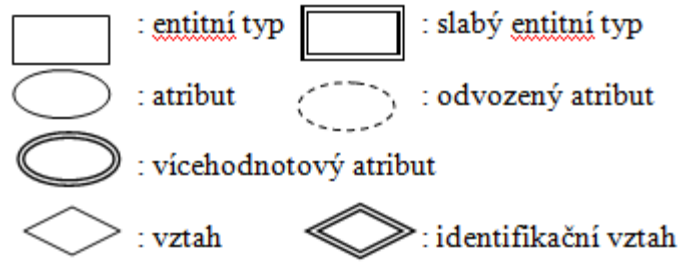


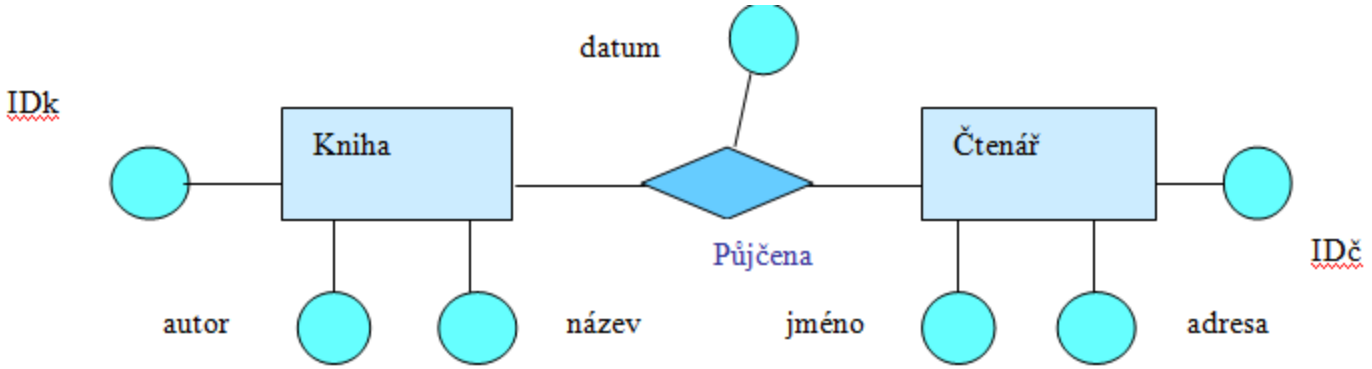
Relační model

- výhody:
  - jednoduchá změna struktury
  - jednoduchá implementace – tabulky
  - dotazovací schopnosti – SQL
- nevýhody
  - může vést k izolaci informací – relace nemusí být ve vztahu
  - usnadňuje implementaci špatného návrhu

E-R model

- **ER model** chápe realitu, případně její sledovanou část, jako množinu objektů (entity) a vztahů mezi nimi (relationship).
- Uznávaný standard, Chenova notace:
- Konstrukty E-R
  - Entitní množiny (entity)
  - Vztahy(relace)
  - Integritní omezení
    - identifikátory
    - Kardinalita a parcialita (členství ve) vztahu



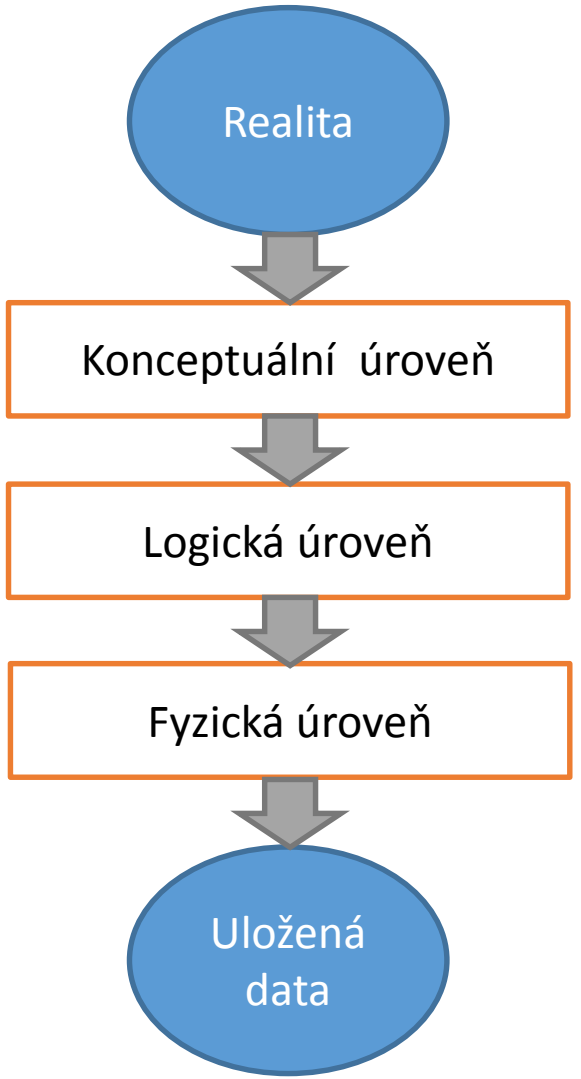


E-R model

- výhody:
  - výjimečná koncepční jednoduchost
  - vizuální prezentace
  - efektivní komunikační nástroj
  - integrace s relačním modelem
- nevýhody
  - špatná reprezentace omezení – pouze u vztahů
  - nemá jazyk pro manipulaci dat
  - omezená reprezentace vztahů – pouze mezi entitami

Abstrakce pohledu na data

- **Konceptuální** úroveň – zabývá se modelováním reality (ER model)
- **Logická** úroveň – vztahuje se ke konkrétnímu datovému modelu (relační model)
- **Fyzická** úroveň – řeší fyzické uložení dat



Principy konceptuálních modelů

1. oddělení konceptuální a interní úrovně
2. orientace na objekty, entity ne na záznamy a soubory
3. bohatší koncept, v relačním modelu jsou relace využívány na „všechno“, reprezentují entity, vícehodnotové atributy, asociace, agregace, dědičnost, …
4. možnost využít úrovně abstrakce v komplexních objektech k zakrytí detailů, možnost modelovat přímo aplikační objekty.
5. funkcionální podstata vztahů (atribut nebo funkce je jediným konstruktem)
6. ISA hierarchie (práce s nadtypy a podtypy)
7. hierarchický mechanismus (objekty lze konstruovat z jiných objektů, formou agregace, seskupováním do množin, tříd)

Klíče

Funkční závislost

- funkční závislost je definována mezi dvěma podmnožinami **atributů** v rámci jedné relace
- nechť X, Y jsou podmnožiny množiny jmen atributů $\{A_1,A_2,...,A_n\}$ relace R.
- X určuje Y (tj. Y je funkčně závislý na X), jestliže všechny záznamy (řádky) v tabulce, které mají stejnou hodnotu atributu X má stejnou hodnotu atributu Y.
- píšeme **$X \rightarrow Y$**
- je-li $Y \subset X$ říkáme, že závislost $X \rightarrow Y$ je triviální
- příklad:

  osCislo $\rightarrow$prijmeni (naopak to neplatí)

  oborIDno $\rightarrow$zkratkaOboru

Klíče

- **Klíč** – jeden či více atributů
- **Superklíč** – klíč, který jednoznačně identifikuje každou entitu (záznam), tj. atributy klíče, určují ostatní atributy
  - příklad:

    osCislo

    osCislo, prijmeni
- **Kandidátní klíč** – superklíč bez redundance (nadbytečných atributů)

  osCislo – je kandidátní klíč

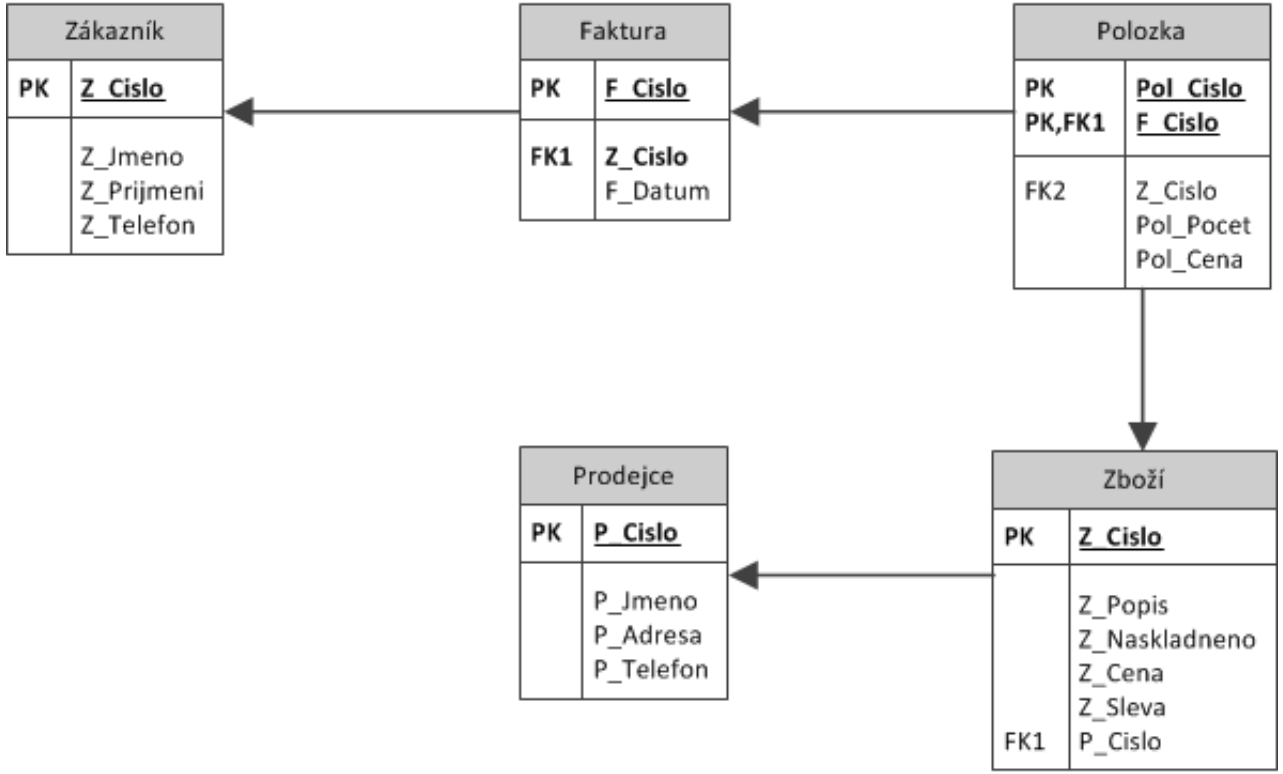
  osCislo, prijmeni – není

  jmeno, prijmeni – může, ale nemusí být
- **Primární klíč** – jeden z kandidátních klíčů
  - musí být unikátní a nesmí obsahovat hodnotu NULL

  osCislo

Cizí klíč

- atribut nebo atributy, které odpovídají hodnotám primárního klíče v odpovídající tabulce
- používají se k vyjádření vztahu mezi dvěma tabulkami



Relační model – algebraický pohled

Relační model – algebraický pohled

- **Schéma relace R** je výraz **R**(A_1 : D1, A2 : D2, … An : Dn), Ai ≠ Aj pro i ≠ j,
  - **R** je jméno schématu
  - A = {A1, A2,..., An} je konečná množina jmen atributů,
  - f(Ai) = Di je zobrazení přiřazující každému jménu atributu A_i neprázdnou množinu, kterou nazýváme **doménou atributu** D_i
- tělo relace je tvořeno množinou n-tic (a_1, a_2,..., a_n), což je konečná podmnožina kartézského součinu domén D_i příslušejících jednotlivým atributům A_i -

$$\mathbf{R} \subset \mathbf{D_1 \times D_2 \times \ldots \times D_n},$$

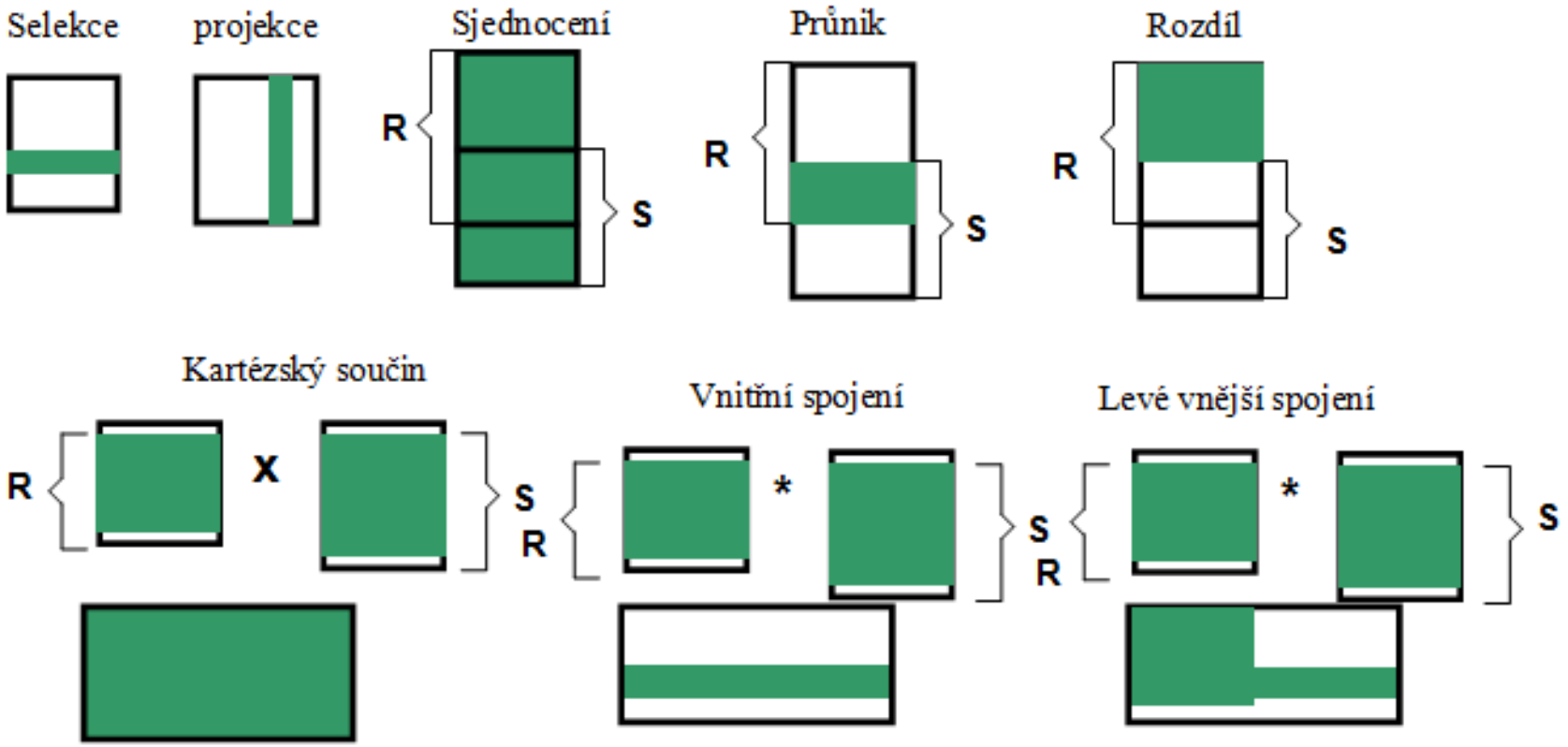
Schéma relační databáze

- **Schéma relační databáze** je dvojice (**R**, **I**), kde **R** je konečná množina relačních schémat $\{R_1(A_1), R_2(A_2), \ldots, R_m(A_m)\}$ a
- **I** je množina integritních omezení
  - klíče
  - **doménové IO**, testuje hodnoty vkládané do databáze podle oboru hodnot - domén atributů v tabulce, při dotazování testuje smysluplnost operací porovnání.
  - přípustnost nedefinované hodnoty **NULL** atributu
  - definice **implicitní hodnoty** atributu

Relační algebra

- RA definována jako dvojice (R,O)
  - nosičem R je množina relací
  - O je množina operací
- Kategorie operací
  - Množinové
    - Sjednocení, průnik, rozdíl – relace musí mít ekvivalentní záhlaví
      - Ekvivalencí záhlaví rozumíme stejný počet atributů relací a existence vzájemně jednoznačného přiřazení atributů z jedné a druhé relace, omezené na dvojice s odpovídajícími doménami
  - Kartézský součin
  - Speciální
    - Projekce
    - Selekce
    - Spojení

Relační algebra



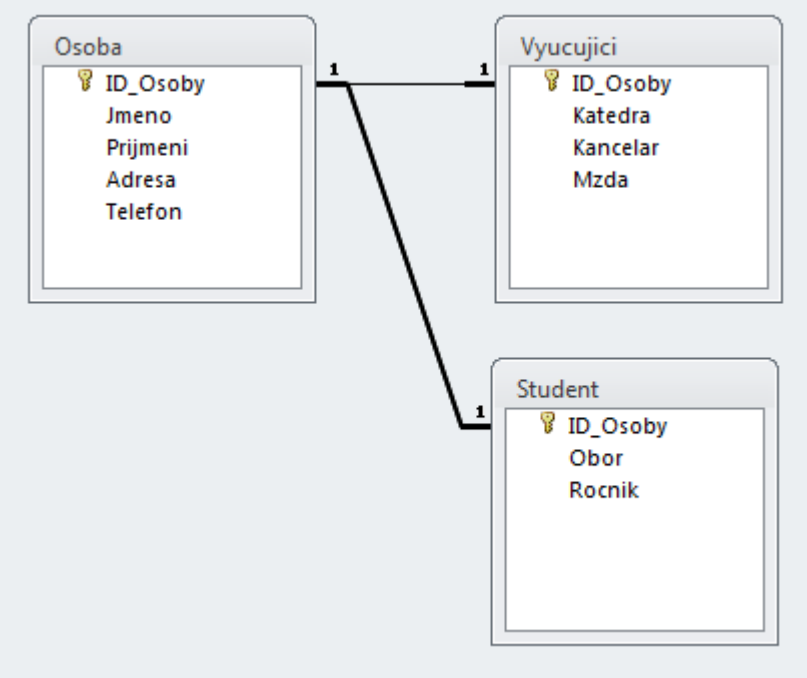
Vztahy

Vztahy v relačních databázích

- druhy vztahů:
  - 1:M – nejčastější
  - M:N – neleze přímo vytvořit, je potřeba převod na 1:M
  - 1:1 – používáno velmi zřídka

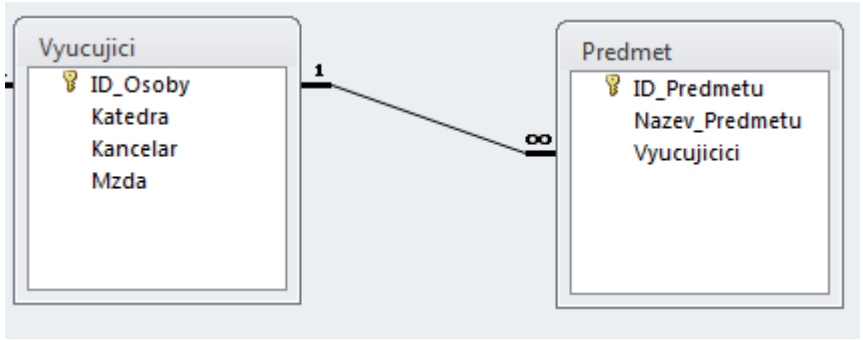
Vztah 1:1

- používá se, když je potřeba ukládat rozsáhlé záznamy
- příklad:



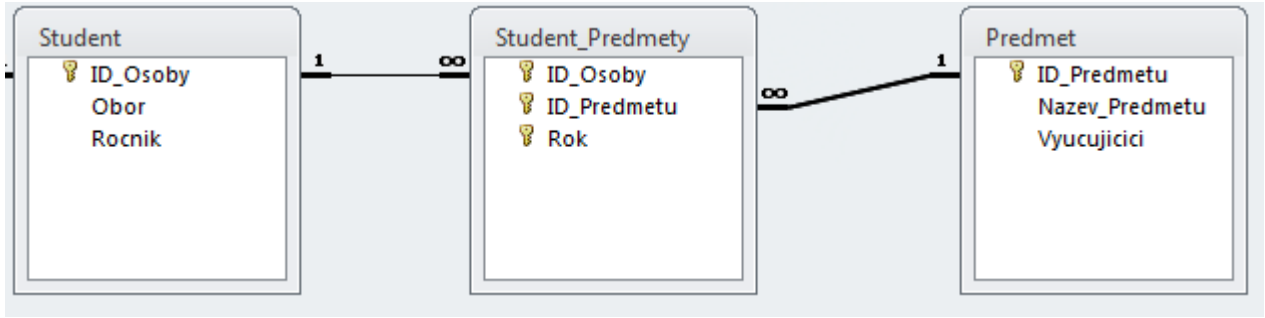
Vztah 1:M

- nejčastější použití
- vazba – primární klíč – cizí klíč
- příklad:



Vztah M:N

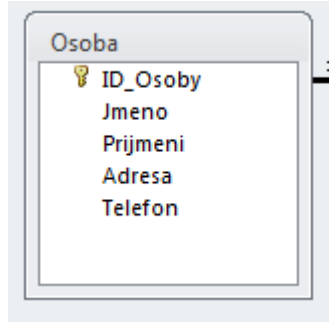
- nelze přímo vytvořit
- nutno rozložit pomocí propojovací tabulky na dva vztahy 1:M
- příklad:
  - vztah Student – Predmet je M:N



Indexy

Indexy

- používají se při hledání záznamů
- index se skládá z:
  - klíče indexu – atribut či více atributů
  - ukazatel na záznam v tabulce
- při prohledávání záznamů se pak nemusí procházet celé záznamy, ale jen indexy
- př: index - Prijmeni, Jmeno



Vlastnosti indexů

- unikátní
  - primární klíč je automaticky index
- zakázání vložení hodnoty NULL
- určení uspořádání záznamů v indexu
  - vzestupně, sestupně

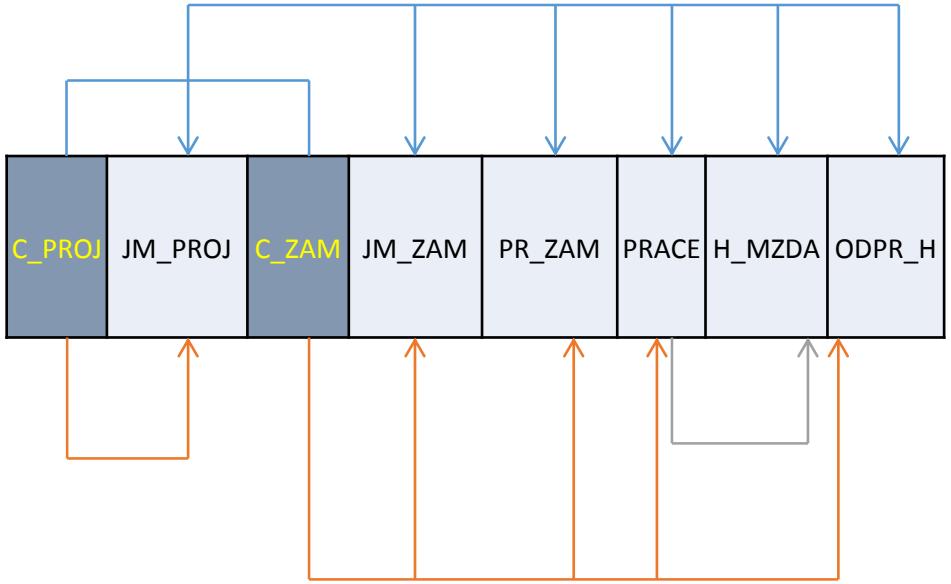
Normalizace relačních schémat

Normalizace relačních schémat

úloha	Pracovník_ID	pracovník_jméno	pracovník_příjmení	místnost_číslo	místnost_plocha
A1	5	Petr	Novák	3	87
A3	5	Petr	Novák	3	87
A9	5	Petr	Novák	3	87
B6	8	Jan	Holý	3	87
A3	8	Jan	Holý	3	87
C9	8	Jan	Holý	3	87
A1	8	Jan	Holý	3	87
…		…	…	…	…

- **Nevhodný návrh** relace signalizuje výskyt opakujících se položek v datech, ale také pozorujeme následující potíže :
  - **redundance**, pro každého pracovníka se opakují hodnoty o místnosti, …
  - nebezpečí vzniku **nekonzistence** při modifikacích jako důsledek redundance (v řádku změníme číslo a nezměníme plochu)
  - **anomálie při vkládání záznamů** : nemůžeme vložit úlohu bez pracovníka, který ji řeší, neboť by nebyly obsazeny klíčové atributy,
  - **anomálie při vypouštění záznamů** : přestanou-li řešit úlohy všichni pracovníci z jedné místnosti, ztratíme informaci i o její ploše.
- Problém vyřešíme **dekompozicí** relace za pomoci funkčních závislostí.

Příklad



1. normální forma

- Relace je 1. NF, jestliže:
  - všechny atributy jsou atomické, tj. dále již nedělitelné.
- Převod:
  1. Nahraďte každý skupinový atribut atomickými atributy.
  2. Určete primární klíč.

2. normální forma

- Relace je v 2. NF, jestliže:
  - je v 1. NF,
  - neobsahuje částečné závislosti (atribut, který není primárním klíčem závislý na celém primárním klíči).
  - jinak: žádný neklíčový atribut není závislý na podmnožině žádného klíče R.
- Převod:
  1. Určete všechny částečné závislosti.
  2. Pro každou částečnou závislost vytvořte zvláštní relaci.

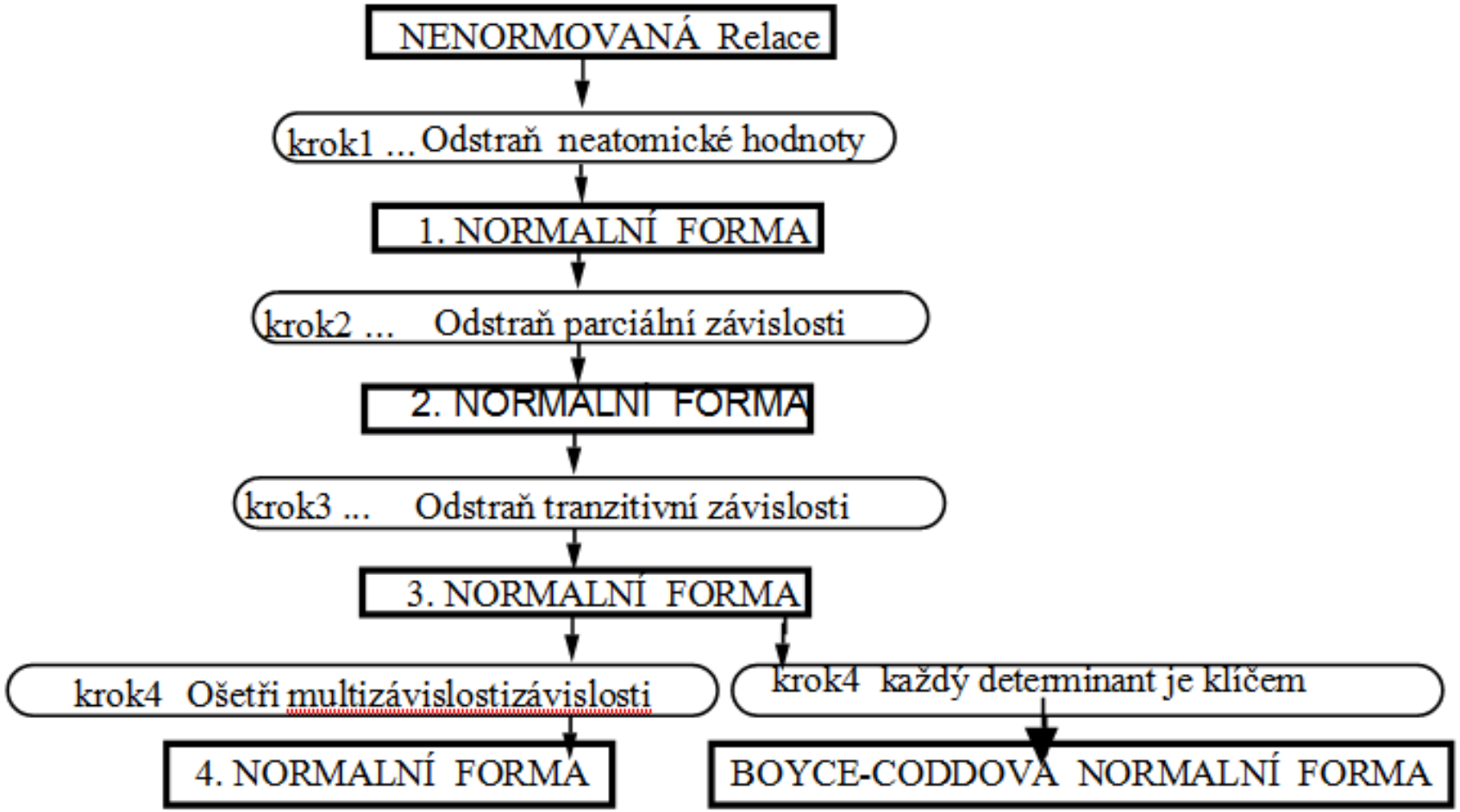
3. normální forma

- Relace je v 3. NF, jestliže:
  - je v 2. NF,
  - neobsahuje transitivní závislosti (nejsou zde závislosti mezi neklíčovými atributy).
- Převod:
  1. Určete všechny tranzitivní závislosti.
  2. Pro každou tranzitivní závislost vytvořte zvláštní relaci.

Boyce-Coddova normální forma

- Relace je v BCNF, jestliže:
  - je v 3. NF,
  - vždy, když $X \rightarrow Y$ a $Y \notin X$, pak X obsahuje klíč schématu R

Postup normalizace



Databázový jazyk SQL

Databázový jazyk

- Základní požadavky:
  - musí umožňovat vytvořit databázi a strukturu tabulek
  - musí umět vkládat, rušit a upravovat data
  - musí umět z dat získávat informace
  - musí být standardizovaný -> použitelný v různých databázových systémech

Databázové jazyky - SQL

- je jazyk pro definic dat (DDL)
  - příkazy pro vytvoření tabulky, indexu a pohledu
  - příkazy pro definic přístupových práv
- je jazyk pro manipulaci s daty (DML)
  - příkazy pro vkládání, úpravu a mazání dat
- je relativně jednoduchý
  - obsahuje méně než 100 příkazů
  - je to neprocedurální jazyk = obsahuje pouze příkazy, „**co**“ je třeba udělat a ne „**jak**“ to udělat
- je standardizován
  - V roce 1986 definován standard ANSI (American National Standard Institute), standard ISO – SQL/86, v roce 1989 přidán integritní dodatek – SQL/89
  - Další standardy (SQL3, SQL1999) evolučně směřují k relačně-objektovým databázím a různým rozšířením.

Datové typy

- číselné
  - celá
    - INTEGER
    - SMALLINT
  - desetinná
    - NUMBER (L,D) – maximální délka
    - DECIMAL (L,D) – minimální délka
- řetězce
  - CHAR (L) – do 255 znaků, pevná délka
  - VARCHAR(L) – proměnlivá délka, maximálně L znaků
- datum
  - DATE

Vytvoření struktury tabulky

- Základní syntaxe:

CREATE TABLE *jmeno* (

sloupec1 typ [omezení],

…

sloupec n typ [omezení],

PRIMARY KEY(sloupec 1,...,sloupec n),

FOREIGN KEY(sloupec 1, ..., sloupec n) REFERENCES *jmeno*,

CONSTRAINT omezení1);

Příklad

```
CREATE TABLE Zakaznik(
        Z_Cislo      INTEGER      NOT NULL    UNIQUE,
        Z_Jmeno      VARCHAR(15) NOT NULL,
        Z_Prijmeni VARCHAR (20),
        Z_Telefon VARCHAR (9),
        PRIMARY KEY(Z_Cislo),
        CONSTRAINT Om1 UNIQUE(Z_Jmeno,Z_Prijmeni));

CREATE TABLE Faktura(
        F_Cislo      INTEGER      NOT NULL    UNIQUE,
        Z_Cislo      INTEGER NOT NULL,
        F_Datum DATE,
        PRIMARY KEY(F_Cislo),
        FOREIGN KEY(Z_Cislo)REFERENCES Zakaznik);

CREATE TABLE Polozka(
        Pol_Cislo    INTEGER      NOT NULL    UNIQUE,
        F_Cislo      INTEGER NOT NULL,
        Pol_Pocet INTEGER DEFAULT 0,
        Pol_Cena INTEGER,
        Z_Cislo INTEGER,
        PRIMARY KEY(Pol_Cislo),
        FOREIGN KEY(F_Cislo)REFERENCES Faktura,
        FOREIGN KEY(Z_Cislo)REFERENCES Zbozi);
```

Další manipulace s tabulkou

- změna struktury – ALTER TABLE
- zrušení tabulky – DROP TABLE

SQL indexy

- Vytvoření indexu:

CREATE [UNIQUE] INDEX *jmeno* ON *jmenotabulky*(*sloupec1, ..., sloupec n*)

- složený index
  - index složený více než z jednoho sloupce
  - používá se hlavně k předcházení duplicit v záznamech

```
CREATE UNIQUE INDEX Testy
ON Test(C_Zam,Test,Datum)
```

C_Zam	T_Cislo	Test	Datum	Skore
110	1	WEA	15.5.2012	93
110	2	WEA	12.5.2012	87
111	1	HAZ	14.12.2012	91
111	2	WEA	18.2.2012	95
111	3	WEA	18.2.2012	95
112	1	CHEM	17.8.2012	91

Příkazy pro manipulaci s daty

- Základní příkazy:
  - Vkládání dat - INSERT,
  - Zrušení dat – DELETE,
  - Změna dat – UPDATE,
  - Dotazování - SELECT

Uložení změn do tabulky

- záznamy se neukládají ihned na disk -> při pádu aplikace může dojít ke ztrátě údajů
- pro okamžité uložení změn slouží příkaz COMMIT
- pro zrušení změn příkaz ROLLBACK
- před oběma příkazy musí být příkaz BEGIN TRANSACTION
- u MS Access tyto příkazy nejsou a záznamy se ukládají ihned po vykonání příkazů SQL

Příkaz SELECT

- slouží k výběru záznamů
- základní syntaxe:

SELECT *atributy* FROM *jmeno* [WHERE where_fráze]

Shlukování dat

- používá se klauzule GROUP BY
- kombinuje agregaci s výběrem
- základní syntaxe:

SELECT *atributy* FROM *jmeno*

[WHERE where_fráze]

[GROUP BY *sloupce*]

[HAVING *podmínky*]

[ORDER BY sloupce[ASC | DESC]]

Pohledy

- virtuální relace (tabulka), odvozená z bázových tabulek
- účel
  - odstínění informací z bázových tabulek + oprávnění uživatelů
  - realizují pro snadné použití složitější dotazy
- definovaný příkazem SELECT

CREATE VIEW pohled_jmeno AS

SELECT …

- zrušení pohledu

DROP VIEW pohled_jmeno

- Pohledy s klausulemi např. DISTINCT, GROUP BY, HAVING a spojení umožňují jen čtení

Spojení tabulek

- spojení tabulek pomocí stejných atributů je jednou z nejdůležitějších vlastností relačních databází
- možná spojení
  - kartézský součin
  - přirozené spojení – vybere pouze záznamy se stejnou hodnotou atributů (primární klíč, cizí klíč)

Vnější spojení

- zobrazuje i záznamy z jedné tabulky, ke které neexistuje záznam v druhé tabulce
- druhy
  - levé
  - pravé
  - úplné

Transakce

Transakce

- je posloupnost akcí nebo specifikace operací (jako jsou čtení a zápis dat, výpočty), které se **buď provedou všechny**, nebo **se neprovedou vůbec**
- Z hlediska aplikace je to logická i programová jednotka práce, odpovídající nějakému procesu v realitě.

Transakce

- při čtení potřebných dat musí být databáze **konzistentní**
- během operací transakce je dočasně i v nekonzistentním stavu, ale nakonec při potvrzení transakce musí být databáze opět konzistentní

Koncept transakce

- Po výpadku se databáze uvede do stavu na počátku transakce.
- Aby systém mohl splnit tyto úkoly, udržuje vlastními prostředky historii změn dat v přiměřeném časovém intervalu v žurnálu(log file).
- Pro zachování integrity a konzistence dat musí transakce splňovat vlastnosti **ACID**.

ACID

- **Atomicita (Atomicity)** – operace transakce musí úspěšně proběhnout všechny(je-li transakce potvrzena), nebo se neprovedou vůbec.
- **Konzistence (Consistency)** – izolované provádění transakce chrání konzistenci databáze. Transakce převádí databázi z jednoho konzistentního stavu do jiného.
- **Izolovanost (Isolation)** – dílčí efekty jedné transakce nejsou viditelné jiným, transakce mohou pracovat souběžně, ale každá transakce musí být odstíněna od výsledků operací ostatních neukončených nebo následujících transakcí. Databáze prochází takovými stavy, jako by souběžné transakce probíhaly sériově za sebou s vhodným uspořádáním
- **Trvalost(Durability)** – výsledky potvrzené transakce jsou perzistentně uloženy

Podrobnější informace

1. Connolly T., Begg C.: *Database Systems. A Practical Approach to Design, Implementation and Management, 3rd edition.* Addison Wesley, 2002. ISBN 0-201-70857-4
2. Pokorný J.: *Databázové systémy a jejich použití v informačních systémech.* Academia, 1992. ISBN 80-200-0177-8
3. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom: *Database Systems:The Complete Book*
4. Jiří Hronek: Databázové systémy (http://phoenix.inf.upol.cz/esf/ucebni/databa.pdf)

Zkouška

- Otázky přehledově z probíraných kapitol (hlavně z prezentací).
- Termíny?