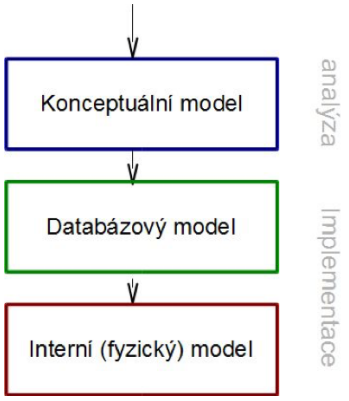
II. Softwarové inženýrství

Update: 5. května 2018

Obsah

1 Modelování databázových systémů, konceptuální modelování, datová analýza, funkční analýza; nástroje a modely.

1.1 Modelování databázových systémů



Databázový systém můžeme modelovat **třemi datovými modely**. Ve fázi analýzy se používá **konceptuální model**, který modeluje realitu na logickou úroveň databáze. Konceptuální model je výsledkem datové analýzy a je **nezávislý na konkrétní implementaci**.

V implementační fázi si pak pomáháme **databázovými modely**, kde modelujeme vazby a vztahy (realitu) na konkrétní tabulky (obecně SŘBD). Databázový model můžeme dále dělit na **relační** a **síťový** model. **Fyzickým uložením dat** na paměťové médium se zabývá **interní model**.

1.1.1 Základní pojmy

- **Entita** – objekt reálného světa.
- **Atribut** – vlastnost entity (možné hodnoty jsou označeny jako doména atributu).
- **Entitní typ** – množina entit se stejnými atributy.
- **Vztah** – vztah mezi **dvěma entitními** typy.
- **Kardinalita vztahu** – dělení vztahů podle počtu entit vstupujících do vztahu – 1:1, 1:N, M:N.

1.2 Datová analýza a konceptuální model

Datová analýza **zkoumá objekty reálného světa, jejich vlastnosti a vztahy**. Zabývá se strukturou obsahové části systému (**strukturou databaze**). Výsledkem datové analýzy je **konceptuální model**. V rámci datové analýzy zpracováváme zadání (specifikaci požadavků na IS):

- podtrhneme **podstatná jména** = identifikujeme **objekty**,
- podtrhneme **slovesa** = identifikujeme **vazby** mezi objekty,

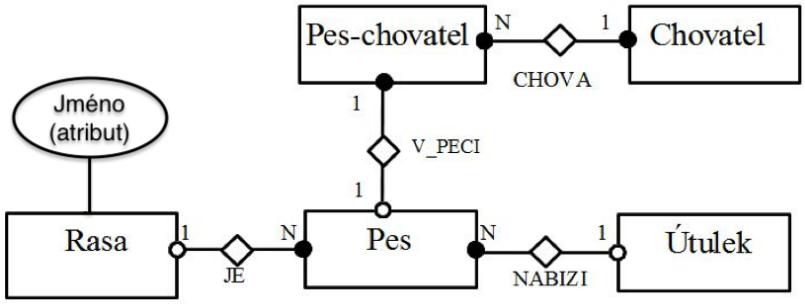
- najdeme **vlastnosti** a **stavy** nalezených objektu = identifikujeme **atributy**.

Z takto získaných informací sestavíme konceptuální model. **Konceptuální model** je jednoduchý **popis entit a jejich vzájemných vztahů**. Jedná se o jakýsi prvotní jednoduchý návrh námi vytvářené databáze. Je kladen důraz na zobrazeních všech entit, jejich vztahů a je **nezávislý** na SŘBD. Skládá z:

- **ER Diagram**, lineární zápis **entit**, lineární zápis **vztahů**, **datový slovník**, popis dalších IO (**integritních omezení**).

1.2.1 ER (Entity-Relationship) Diagram

Grafické znázornění **konceptuálního modelu** (objektů a vztahů mezi nimi). Může mít několik podob v závislosti na používaném prostředí a detailnosti s jakou jej potřebujeme vypracovat. **Atributy** můžou být v grafu znázorněny **ovály** spojenými s **objekty** (**obdélníky**), **vazba** 1:N může být znázorněna „hráběmi" místo N, či celý diagram se může podobat třídnímu diagramu s atributy vepsanými do objektu.



1.2.2 Lineární zápis entit a vztahů

Lineárním zápisem **popisujeme objekty**, jejich vlastnosti a vztahy **z pohledu implementačního**. Lineárním zápisem entit jsou v podstatě definovány **tabulky a jejich atributy** včetně **primárních** a ***cizích klíčů***.

- Příklad lineárního zápisu entity: `Pes (IDPes, jmeno, pohlavi, vek, CRasa, IDUtulek)`.
- Příklad lineárního zápisu vztahů: `NABIZI (Útulek, Pes) 1:N`.

1.2.3 Datový slovník

Podrobný rozpis jednotlivých atributů. Tabulka obsahuje typ atributů, velikost, integritní omezení, atd.

Integritní omezení obsahují další specifikace atributů, které nejsou dány typem a délkou. Nejčastěji se týkají formátu atributu (podmínka v jakém má být formátu) – např: login se skládá z třech čísel a třech písmen, nebo rodné číslo je složeno z data narození, apod.

Další integritní omezení – konceptuální schéma obsahuje také soupis dalších IO, které se týkají entit (tabulek) a vazeb mezi nimi. Může jít například o omezení vícenásobné vazby, vyjádření hierarchie mezi entitama, apod.

Pes	**Typ**	**Délka**	**Klíč**	**NOT NULL**	**IO**
IDPes	int	8	primarni	ano	pravidla pro tvar čipového čísla
jmeno	varchar	50			
vek	int	2			
CRasa	int	2	sekundarni		

Tabulka 1: Datový slovník pro tabulku Pes.

1.3 Funkční analýza

Zatímco datová analýza se zabývá strukturou obsahové části systému (strukturou databaze), **funkční analýza řeší funkce systému**. Funkční analýza tedy vyhodnocuje manipulaci s daty v systému. Skrze **DFD** (Data Flow Diagramy) **analyzuje toky dat**, **základní funkce systému a aktéry**, kteří se systémem pracují. Výstupem jsou pak **minispecifikace** – podrobné analýzy elementárních funkcí systému.

Cílem je popsat vytvářený systém jako „černou skříňku“, definovat její **vnější chování** a strukturalizovat **okolí systému**, které se systémem komunikuje. **Popsat všechny funkce, které se budou s daty provádět.**

Otázky na požadavky

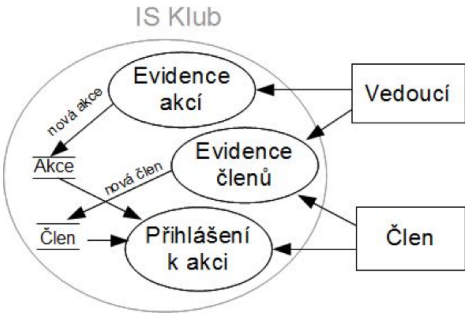
- **PROČ** nový systém.
- **ČEMU** má sloužit.
- **KDO** s ním pracuje - běžně, příležitostně, pravidelně zřídka.
- **VSTUPY** – objekty, atributy
- **VÝSTUPY** – výstupní sestavy, požadované informace
- **FUNKCE** – jaké výpočty, odvozování, výběry, třídění, ...
- **Vazby na OKOLÍ systému** – odkud data a kam.

Nefunkční požadavky

- Požadavky na **výsledný program**.
- **Vnější požadavky**: ostatní nefunkční implementační požadavky, použití **standardů**, **cenová** omezení, **časové** požadavky.

1.3.1 Diagram datových toků (DFD)

DFD je grafický nástroj pro **modelování funkcí a vztahů v systému**. Znázorňuje nejen procesy (funkce) a datové toky, ke kterým v systému dochází, ale definuje také hlavní aktéry a jejích omezení nad systémem. DFD diagram obsahuje tyto prvky: **aktér** (obdélník mimo systém), **proces** (kruh uvnitř systému), **datové toky** (šipky) a **paměť** (viz. obr. Akce a Člen).



DFD diagramy lze zakreslit v různých úrovních. Např. proces Evidence akcí na obrázku lze dále rozkreslit dalším DFD, obsahující procesy vytvoření a editace akce. DFD nejvyšší úrovně se nazývá **kontextový diagram**. Znázorňuje pouze práci aktérů se systémem jako celkem. Systém v kontextovém diagramu vystupuje jako černá skříňka a v diagramu tedy nejsou použity prvky procesu a paměti. **Hlavní znaky DFD:**

- Má několik úrovní podrobnosti.
- Definuje **hranici systému**.
- Definuje **všechny akce**, které mezi systémem a jeho okolím probíhají.

1.3.2 Minispecifikace = algoritmy elementárních funkcí

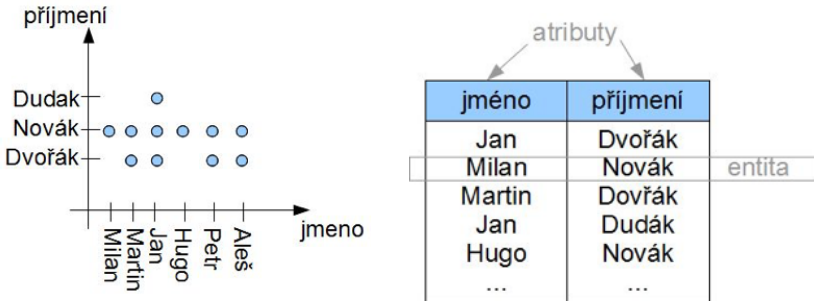
- Popisuje logiku každé z funkcí **poslední úrovně DFD**.
- Každému **elementárnímu (nerozložitelnému) procesu** z poslední úrovně DFD odpovídá **jedna minispecifikace**.
- Popisuje postup, jak jsou **vstupní data transformována na výstupní**.
- Popisuje, co **funkce znamená**, ne jak se to spočítá.
- Používá se **přirozený jazyk** s omezeným množstvím jasně definovaných pojmů, aby byla **srozumitelná** jak pro analytika, tak i uživatele a programátorovi.

```
IF všechny výrobky v objednávce jsou rezervovány,
THEN pošli objednávku k dalšímu zpracování oddělení prodeje.
OTHERWISE,
    FOR EVERY nezarezervovaný výrobek v objednávce DO:
        Zkus najít volný výrobek a rezervuj ho.
        IF výrobek není na skladě,
        THEN informuj správce.
```

2 Relační datový model, SQL; funkční závislosti, dekompozice a normální formy.

2.1 Relační datový model

Relační datový model představuje **způsob uchování dat v tabulkách**. Relační se mu říká proto, jelikož tabulka je definována přes Relaci.



Obrázek 1: Tabulka s dvěma atributy jako relace (vlevo), relace zobrazena tabulkou (vpravo).

Relace je tabulka definována jako **podmnožina kartézského součinu domén**. Relace na obrázku je tedy podmnožina kartézského součinu množin {Dudak, Novák, Dvořák} × {Milan, Martin, Jan, ..., Aleš}.

Na rozdíl od matematické relace se ta databázová **mění v čase** (přidáváním a odebíráním prvků relace). Kromě základních **množinových operací** se u databázové relace setkáme s operaci **selekce** – výběr řádků a **projekce** – výběr sloupců.

- **Doména** je **množina všech hodnot, kterých může daný atribut nabývat** (obor hodnot atributu). V praxi je doména dána **integritním omezením** (IO). Doména atributu Přijmení z obrázků je množina {Dudak, Novák, Dvořák}.
- **Atribut** je vlastnost entity (z pohledu tabulky jde o sloupec).
- **Relační schéma** můžeme chápat jako strukturu tabulky (atributy a domény). Relační schéma R je výraz tvaru R(A, f), kde R je jméno schématu, A = A1, A2,..., An je **konečná množina jmen atributů**, f je zobrazení přiřazující každému jménu atributu Ai neprázdnou množinu (obor hodnot atributu), kterou nazýváme **doménou atributu** Di, tedy **f(Ai) = Di**.

Příklad pro tabulku (relaci) Učitel

- **Atributy:** `ID, jméno, příjmení, funkce, kancelář`.
- **Domény:**
  - D1 – tři písmena z příjmení, tří cifry pořadového čísla,
  - D2 – kalendář jmen,

  - D3 – množina příjmení,
  - D4 – množina funkcí (asistent, vědec, učitel,...),
  - D5 – A101, A102, ... A160.

- **Relační schéma:** `Učitel (ID, jméno, příjmení, funkce, kancelář)`.
- **Relace:** `Učitel = {(nov001, lukas , novak , vědec, A135), (kom123, jan, komensky, učitel, A111), ...}`

2.1.1 Základní úlohy relačního modelu:

1. Návrh „správné“ **struktury databáze bez redundancí** – **funkční závislosti**, **normální formy**.
2. **Vyhledávání informací** z databáze – (dotazovací) **relační jazyky**.

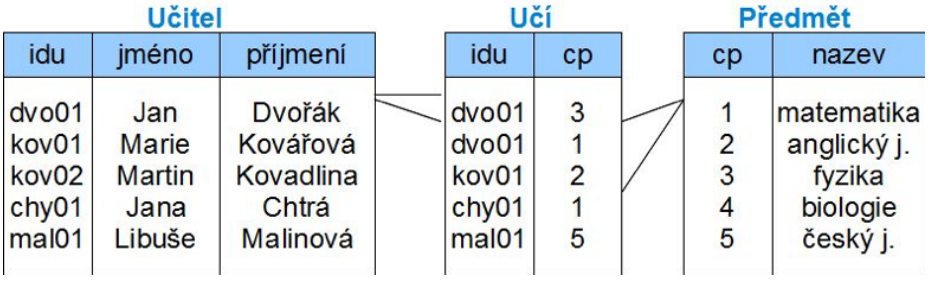
2.1.2 Vlastnosti relačního datového modelu

Z definice relace vyplývají tyto jejich tabulkové vlastnosti:

- **Homogenita** (stejnorodost) sloupců (prvky domény).
- Každý údaj (hodnota atributu ve sloupci) je **atomickou položkou**.
- Na **pořadí** řádků a sloupců **nezáleží** (jsou to množiny prvků/atributů).
- Každý řádek tabulky je **jednoznačně identifikovatelný** hodnotami jednoho nebo několika atributů (primárního klíče).

2.1.3 Vazby relačního modelu

Obecně se vazby v relačním modelu realizují pomocí další relace (tabulky). Jedná se o tzv. **vazební tabulku**. Ta obsahuje ty atributy relací (tabulek, které se vazby účastní), které jednoznačné identifikují jejich entity – primární klíče. Obsahuje-li tabulka atribut, který slouží jako primární klíč v jiné tabulce, pak obsahuje cizí klíč. Vazební tabulka tedy obsahuje cizí klíče. Příklad vazby M:N:



2.2 SQL (Structured Query Language)

SQL (Structured Query Language) **je relační jazyk založen na predikátovém kalkulu**. Na rozdíl od jazyků založených na relační algebře, kde se dotaz zadává algoritmem, tyto jazyky se soustředí na to **co se má hledat**, ne jak.

- Standardizovaný **strukturovaný dotazovací jazyk**, který je používán pro práci s daty v **relačních databázích**. (DQL - Data Query Language).
- Navržen IBM jako **dotazovací jazyk** (původní název Sequel).
- Základem je **n-ticový relační kalkul**.
- Standardy podporuje prakticky každá relační databáze, ale obvykle nejsou implementovány vždy **všechny požadavky normy**.
- Obsahuje i příkazy pro **vytvoření** a **modifikace** tabulek, pro **ukládání**, **modifikaci** a **rušení** dat v databázi a řadu dalších příkazů.
- **Příklad**: `CREATE TABLE Drazitel (jmeno CHAR(20), adresa CHAR(30), aukce NUMBER(4), zisk NUMBER(4)); INSERT INTO clovek VALUES( 'nj001', 'Jan', 'Novotný', '777111222'); SELECT telefon FROM clovek WHERE prijmeni = "Novotný";`

2.2.1 Příkazy SQL

- **Vytváření** a **modifikace** relačního schematu (tabulek, databází) - `CREATE`, `ALTER` (`MODIFY`, `ADD`, `DROP`), `DROP` = vytvoř, uprav, smaž.
- **Modifikací** dat - `INSERT`, `UPDATE`, `DELETE` = vlož, uprav, smaž.
- **Vyhledávání** v relacích - `SELECT`, `ORDER BY`, `GROUP BY`, `JOIN` = vyhledej, seřaď, shlukuj, spoj.
- **Transakce** - `COMMIT`, `ROLBACK` = úspěšně provedená transakce, save-point uvnitř transakce ke kterému se dá vrátit byla-li transakce přerušena.
- Další, pro podmínky, logické operatory,... (`WHERE, LIKE, BETWEEN, IN, IS NULL, DISTINCT/UNIQUE, JOIN, INNER JOIN, OUTER JOIN, EXISTS, HAVING, COUNT, VIEW, INDEX`,...).

2.3 Relační jazyky

Jazyky pro formulaci požadavků na výběr dat z relační databáze (dotazovací jazyky) se dělí do dvou skupin:

- **Jazyky založené na relační algebře**, kde jsou výběrové požadavky vyjádřeny jako posloupnost speciálních operací prováděných nad daty. Dotaz je tedy **zadán algoritmem**, jak vyhledat požadované informace.

- **Jazyky založené na predikátovém kalkulu**, které požadavky na výběr zadávají jako predikát charakterizující **vybranou relaci**. Je úlohou překladače jazyka nalézt odpovídající algoritmus. Tyto jazyky se dále dělí na
  - **n-ticové** relační kalkuly,
  - **doménové** relační kalkuly.

2.4 Relační algebra

Relační algebra je velmi silný **dotazovací jazyk** vysoké úrovně. Nepracuje s jednotlivými enticemi relací, ale **s celými relacemi**. Operátory relační algebry se aplikují na relace, výsledkem jsou opět relace. Protože relace jsou množiny, přirozenými prostředky pro manipulaci s relacemi budou množinové operace.

I když relační algebra v této podobě **není vždy implementována v jazycích SŘBD**, je její zvládnutí nutnou podmínkou pro správnost manipulací s relacemi. I složitější dotazy jazyka SQL, který je deskriptivním dotazovacím jazykem, mohou být bez zkušeností s relační algebrou problematické.

2.4.1 Základní operace relační algebry

Jsou dány relace **R** a **S**. **Množinové operace:**

- **Sjednocení** relací téhož stupně: $R \cup S = \{x | x \in R \vee x \in S\}$
- **Průnik** relací: $R \cap S = \{x | x \in R \wedge x \in S\}$
- **Rozdíl** relací: $R - S = \{x | x \in R \wedge x \notin S\}$
- **Kartézský součin** relace R stupně m a relace S stupně n: $RxS = \{rs | r \in R \wedge s \in S\}$, kde $rs = \{r1, ..., rm, s1, ...sn\}$

Další relační operace:

- **Projekce** (výběr sloupců) relace R, jedná se o unární operaci $\Pi_{\mathbf{X}}(R)$, kde X je množina názvů atributů.
- **Selekce** (výběr řádků) z relace R podle podmínky P. Selekce je unární relační operace $\sigma_{\varphi(\mathbf{X})}(R)$, kde R je relace, $\varphi(\mathbf{X})$ predikátová formule hovořící o jednotlivých prvcích a jejich příslušnosti do relací.
- **Spojení** relací R s atributy A a S s atributy B (join). Značí se $R \bowtie S$, výsledkem je množina všech kombinací prvků relace **R** a **S**. Takto definovaný join se nazývá Přirozené spojení (natural join). Exsitují i další (outer, inner, left, right ...).

2.5 N-ticový relační kalkul

- Dr. Codd definoval n-ticový relační kalkul pro RDM jazyk matematické logiky - predikátový počet je využit pro výběr informací z relační databáze.

- Název odvozen z oboru hodnot jeho proměnných - **relace je množina prvků = n-tic**.
- Je **základem pro jazyk typu SQL**.
- Syntaxe je **přizpůsobena** programovacímu jazyku: **matematické vyjádření** $\{x|F(x)\}$ nahradíme zápisem `x WHERE F(x)`
  - Kde x je proměnná pro hledané n-tice (struktura relace).
  - F(x) je **podmínka**, kterou má x splňovat (výběr prvků relace).

2.5.1 Definice

Výraz n-ticového relačního kalkulu je výraz tvaru `x WHERE F(x)`, kde x je jediná volná proměnná ve formuli F. Základní operace relační algebry se dají vyjádřit pomocí výrazů n-ticového relačního kalkulu, tedy n-ticový relační kalkul je relačně úplný.

Platí:

$R \cup S$	=> x WHERE R(x) OR S(x)
$R \cap S$	=> x WHERE R(x) AND S(x)
R - S	=> x WHERE R(x) AND NOT S(x)
R x S	=> x, y WHERE R(x) AND S(y)
$R[a_1,a_2,...,a_k]$	=> $x.a_1, x.a_2, ..., x.a_k$ WHERE R(x)
R(P)	=> x WHERE R(x) AND P
R[A*B]S	=> x, y WHERE R(x) AND S(y) AND x.A * y.B

2.6 Funkční závislost

Funkční závislost je v databázi **vztah mezi atributy** takový, že máme-li atribut Y je funkčně závislý na atributu X píšeme X → Y, pak se **nemůže stát**, aby **dva řádky mající stejnou** hodnotu atributu **X** měly **různou hodnotu Y**. Je-li Y, X říkáme, že závislost X → Y je **triviální**.

- FZ je definována **mezi dvěma podmnožinami atributů** v rámci jednoho schématu relace. Jde o vztah mezi atributy, nikoliv mezi entitami.
- FZ je definována na **základě všech možných aktuálních relací**, není tedy možné soudit na funkční závislost z vlastností jediné relace. Tak můžeme poznat jen neplatnost funkční závislosti.
- FZ jsou **tvrzení o reálném světě**, o významu atributů nebo **vztahů mezi entitami**, je nutné realitu brát v úvahu při návrhu schématu databáze.

Příklad: Atribut '*datum narození* ' je funkčně závislý na atributu '*rodné číslo*' (nemůže se stát, že u záznamů se stejnými rodnými čísly bude různé datum narození).

Pomocí funkčních závislostí můžeme **automaticky navrhnout schéma databáze** a předejít problémům jako je **redundance**, **nekonzistence databáze**, zablokování při vkládání záznamů, apod.

2.7 Armstrongovy axiomy

K určení **klíče schématu** a logických implikací množiny závislostí potřebujeme **nalézt uzávěr F+**, nebo určit, zda daná závislost X → Y je prvkem F+. K tomu existují pravidla zvaná Armstrongovy axiomy. Jsou **úplná** (dovolují odvodit z dané množiny závislostí F všechny závislosti patřící do F+) a **bezesporná** (dovolují z F odvodit pouze závislosti patřící do F+).

- **Reflexivita** – je-li Y ⊂ X ⊂ A, pak X → Y
- **Tranzitivita** – pokud je X → Y a Y → Z, pak X → Z
- **Pseudotranzitivita** – pokud je X → Y a WY → Z, pak XW → Z
- **Sjednocení** – pokud je X → Y a X → Z, pak X → YZ
- **Dekompozice** – pokud je X → YZ, pak X → Y a X → Z
- **Rozšíření** – pokud je X → Y a Z ⊂ A, pak XZ → YZ
- **Zúžení** – pokud je X → Y a Z ⊂ Y, pak X → Z

Závislost, která má na pravé straně pouze jeden atribut, nazýváme **elementární**.

2.7.1 Určení klíče pomoci funkčních závislostí

Ze zadání jsme určili atributy A = {učitel, jméno, příjmení, email, předmět, název, kredity, místnost, čas} a funkční závislosti F:

- učitel → jméno, příjmení, email
- předmět → název, kredity
- místnost, čas → učitel, předmět

Rozšíření:

- učitel, **místnost**, **čas** → jméno, příjmení, email, **místnost**, **čas**
- předmět → název, kredity
- místnost, čas → učitel, předmět

Dekompozice 1:

- učitel, **místnost**, **čas** → jméno, příjmení, email, místnost, čas, **učitel**, **předmět**
- předmět → název, kredity

Dekompozice 2:

- učitel, místnost, čas → jméno, příjmení, email, místnost, čas, učitel, předmět, **název**, **kredity**

Atributy **učitel**, **místnost**, **čas** je klíč schématu velké relace. V dalším kroku je třeba provést dekompozici a tuto velkou relaci rozbít na menší relace.

2.8 Dekompozice

Dekompozice relačního schématu je **rozklad relačního schématu na menší** relač. sch. (rozloží velkou tabulku na menší) aniž by došlo k narušení redundance databáze. Mezi základní vlastnosti dekompozice patří - **zachování informace** a **zachování funkčních závislostí**.

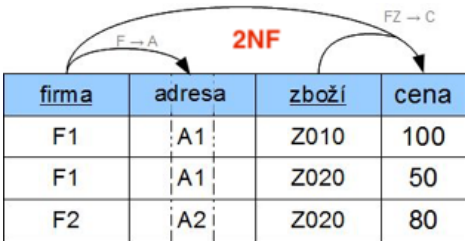
- **Algoritmus dekompozice (metoda shora dolů)** – na počátku máme celé relační schéma se všemi atributy, snažíme se od tohoto schématu odebírat funkční závislosti a tvořit schémata nová. **Exponenciální složitost**, **BCNF**.
- **Algoritmus syntézy (zdola nahoru)** – vytvoří pro každou funkční závislost novou relaci. Pak tyto malé relace spojuje do větších celků. **Menší složitost, 3NF**.

Binární dekompozice, kterou budeme dále řešit je rozklad jednoho relačního schématu na dvě. Obecná dekompozice vznikne postupnou aplikací binárních. Dekompozice relačního schématu R(A,f) je množina relačních RO={R1(A1, f2), R2(A2, f2), ...}, kde A = A1 ∪ A2 ∪ A3 ∪ ...

2.9 Normální formy

Normální formy relací (NF) prozrazují jak dobře je databáze navržena (čím vyšší NF tím lepší).

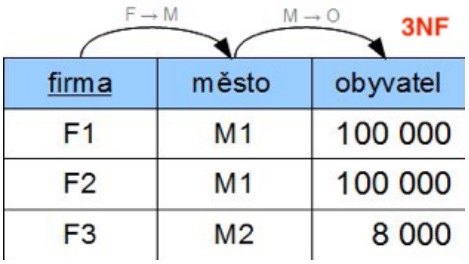
- **1 NF** – definuje tabulky, které obsahují **pouze atomické atributy**. Žádné složené atributy - např. v jednom atributu je Jméno i Příjmení.
- **2 NF** – je v 1NF + **každý sekundární atribut je úplně závislý na každém klíči schématu**. Neboli neexistuje závislost sekundárních na podklíči (pokud se klíč skládá z více atributů). Např.: když AB → CD, pak nesmí být B → C. Atribut adresa není závislý na všech klíčích FZ, ale pouze na F.



firma	adresa	zboží	cena
F1	A1	Z010	100
F1	A1	Z020	50
F2	A2	Z020	80

- **3 NF** – je 2NF + žádný sekundární atribut **není tranzitivně závislý** na žádném klíči schématu. Nesmí existovat závislosti mezi sekundárními atributy (Model auta ->

značka auta). Když AB → CD, pak nesmí C → D. **Příklad porušení 3NF** – atribut počet obyvatel je tranzitivně závislý (přes atr. město) na klíči.



- **BCNF** (Boyce-Coddova normální forma) – 3NF + je-li funkční závislost (X → Y) ∈ F+ a Y ∉ X, pak X obsahuje klíč schématu. **Musí být závislost sekundárních atributů na primárních nikoli naopak**. Když AB → CD, pak nesmí C → A

3 Transakce, zotavení, log, ACID, operace COMMIT a ROLLBACK; problémy souběhu, řízení souběhu: zamykání, úroveň izolace v SQL.

4 Procedurální rozšíření SQL, PL/SQL, T-SQL, triggery, funkce, procedury, kurzory, hromadné operace.

5 Základní fyzická implementace databázových systémů: tabulky a indexy; plán vykonávání dotazů.

6 Objektově-relační datový model a XML datový model: principy, dotazovací jazyky.

7 Datová vrstva informačního systému; existující API, rámce a implementace, bezpečnost; objektově-relační mapování.

8 Distribuované SŘBD, fragmentace a replikace.