Základní terminologie

Databáze (DB)

- „balíček dat“ + další
- Logicky seřazené kolekce datových instancí

Database management system (DBMS)

- Systém pro správu databází
- Oracle, MS SQL Server, Sybase, Informix, Progress

Databázový (informační) systém

- Všechno, aby se databáze mohla provozovat (HW, knowhow…)

Proč DB?

1. Ukládání dat
2. Sdílení dat, znovu použitelnost
3. Obecné rozhraní a jazyky
4. Bezpečnost informací
5. Administrace a údržba

Historie

- Databázové modely a systémy
    - Hierarchické (stromová struktura) databáze
    - Síťové (i n-n, na rozdíl od hierarchického) databáze
    - Relační databáze (tabulky)
    - Objektové databáze
    - XML databáze
    - NoSQL databáze
- Nyní – relační, zpracovávání big data

Proč tolik systémů?

 Různé architektury, požadavky, kontexty, formy dat

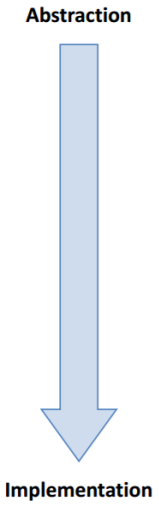
Konceptuální (databázové) modelování

Jakým způsobem navrhnout databázi

- Modelování reality
- Zorganizovat získané informace

Stakeholders = všichni lidé pohybující se kolem našeho systému

Vrstvy databázového modelování



Konceptuální vrstva

- Nejabstraktnější
- Účelem je zachytit, jak vypadá reálný problém, který má systém řešit (kus reálného světa)
- Modely – ER (entitně-relační model), UML

Logická vrstva

- Určuje, jak jsou komponenty zastoupeny v logických strukturách, které jsou strojově interpretovatelné
- Oprostí nás od detailů
- Zobrazí konceptuální představu do databázových struktur
- Sety, relace, funkce, grafy, stromy… (tradiční matematické struktury)
- Tabulky, objekty, kolekce

Fyzická vrstva

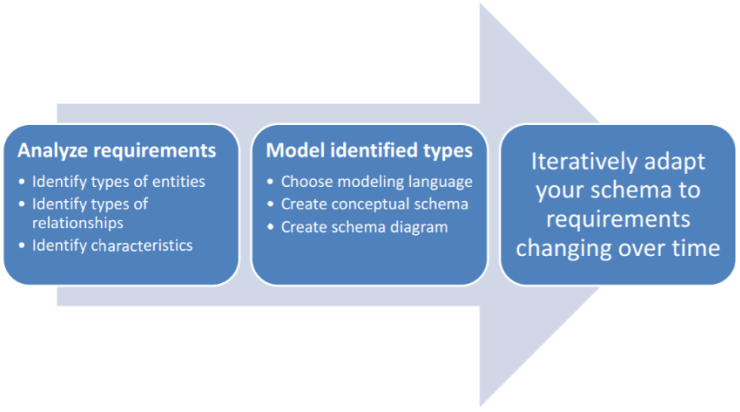
- Definuje, jak je logická vrstva implementována na konkrétní technické prostředí

Konceptuální modelování

- Proces vytváření konceptuálního schématu

Terminologie

- Model = modelovací jazyk
  - Set konstrukcí pro vyjádření něčeho
  - UML model = {class, attribute, association}
  - Relační model = {relational schema, attribute}
- Schéma = vyjádření pomocí jazyka
  - Instance modelu
- Diagram = vizualizace schématu



První krok – *analýza požadavků*

- Začít s požadavky od více stakeholderů
- Identifikovat entity
  - Viz příklad z přednášky v prezentaci
- Identifikovat vztahy mezi entitami
- Charakteristiky entit

Druhý krok – *vytvoření schématu*

- Schéma – a vytvořit z něj diagram
- Vybrat jazyk
  - Použitelné modely – UML, ER, OCL, ORM, OWL, DL
    - ER
      - Není standardizován, mnoho notací, způsobů zápisu
      - Více orientován na design dat
    - UML
      - Častěji používaný
      - Standardizováno – OMG (Object Management Group)
      - Design kódu

- Přenést krok 1 do konkrétního jazyka

Typy entit

Entitní typ – třída

- Entitní typ může být zdroj pro více hierarchií
- Může mít maximálně jednu generalizaci
- Silné – má alespoň jednu identifikaci
- Slabé – nemá identifikaci

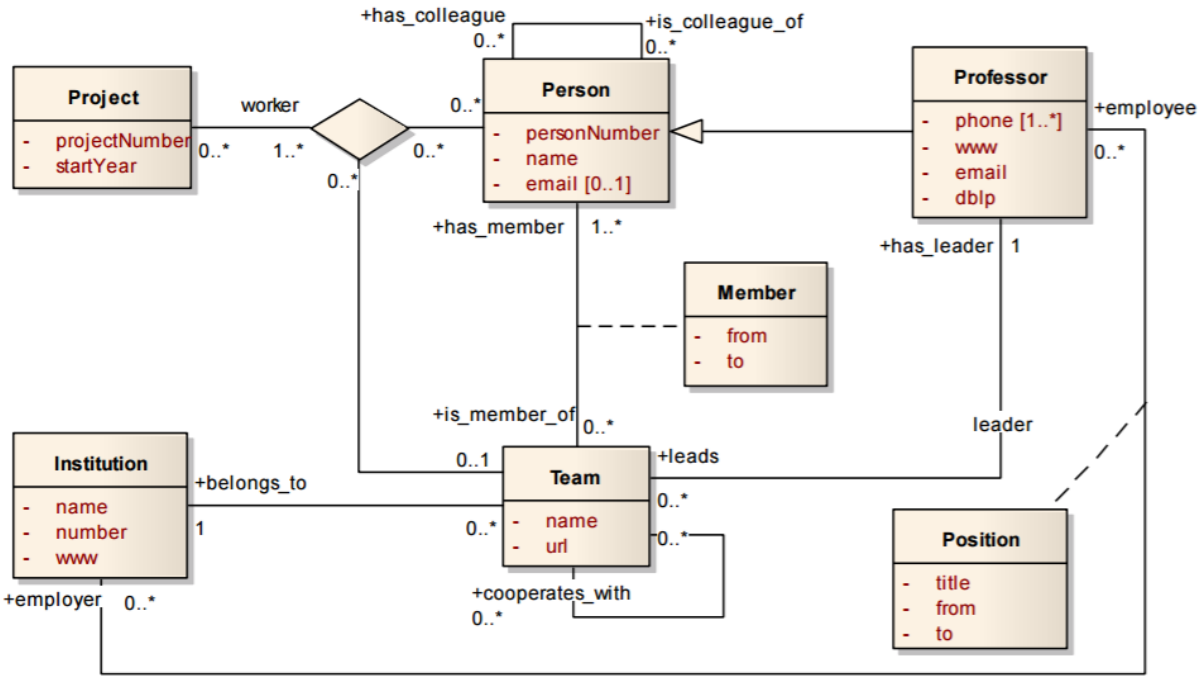
Entita – konkrétní instance třídy

- Každá entita musí mít specifikovaný typ, ale maximálně jeden

Kardinalita – mohutnost množiny

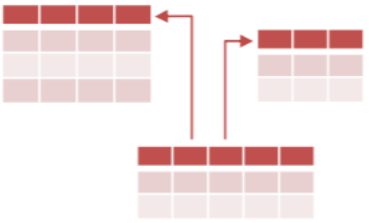
1. Typy entit
2. Charakteristiky entit
3. Typy vztahů
4. Kardinality ve vztazích
5. Charakteristiky vztahů
6. Generalizace/specializace
7. Složené atributy
8. Rekurzivní vztahy
9. N-ární vztahové typy
10. Identifikátory
    a. Každý typ entity musí být identifikovatelný
    b. Kardinalita – jen 1-1 povolena

	UML		ER	
1	**Třída**	Jméno	**Typ entity**	Jméno
2	**Atribut třídy**	Jméno a kardinalita	**Atribut typu**	Jméno a kardinalita
3	**Binární asociace**	jméno a dva účastníci se jmény a kardinalitami	**Binární vztahový typ**	Jméno a dva účastníci s kardinalitami
4				
5	**Atribut binární asociace**	Jméno a kardinalita	**Atribut vztahového typu**	Jméno a kardinalita
6	**Generalizace**	Specifická asociace, beze jména, role a kardinality	**ISA hierarchie**	Specifický vztah beze jména a kardinality
7	**Žádná specifická konstrukce**		**Složené atributy**	Jméno, kardinalita, podatributy
8	**Normální asociace**	Stejní účastníci	**Normální vztah**	Stejní účastníci
9	**N-arní asociace**	Jako binární, ale se třemi a více účastníky	**N-ární vztahové typy**	Jako binární, ale se třemi a více účastníky
10	**N/A**		**Atribut nebo skupina atributů jako Identifikátor**	



Relační model

Logické modely

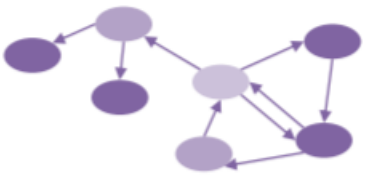


Modely založené na tabulkách

- Struktura
  - Řádky pro entity
  - Sloupce pro atributy
- Operace
  - Selekce, projekce, …
- Příklad
  - Relační model, tabulkové modely (SQL…)

Modely založené na objektech

- Struktura
  - Objekty s atributy
  - Ukazatel mezi objekty
- Motivace
  - OOP
  - Zapouzdření, dědičnost
- Operace
  - Navigace

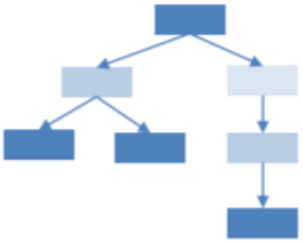


Modely založené na grafech

- Struktura
  - Hrany, vrcholy, atributy
- Operace
  - Průchod, pattern matching, grafové algoritmy
- Příklady
  - Síťový model
  - RDF (Resource description framework)
  - Neo4j, InfiniteGraph, OrientDB…

Modely založené na stromech

- Struktura
  - Vrcholy s atributy
  - Hrany mezi vrcholy
- Motivace
  - Hierarchie, kategorizace, data
- Příklady
  - Hierarchické modely
  - XML dokumenty
  - JSON dokumenty



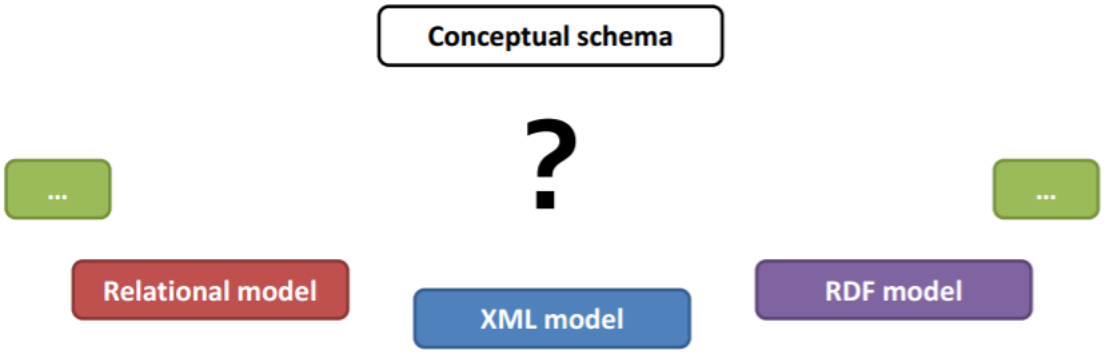
semi-strukturovaná

Hodně různých modelů – hierarchický, síťový, relační, objektový, objektově-relační, XML, dokumentově orientovaný, graf…

- Různé modely dobré pro různé situace

Logické modelování

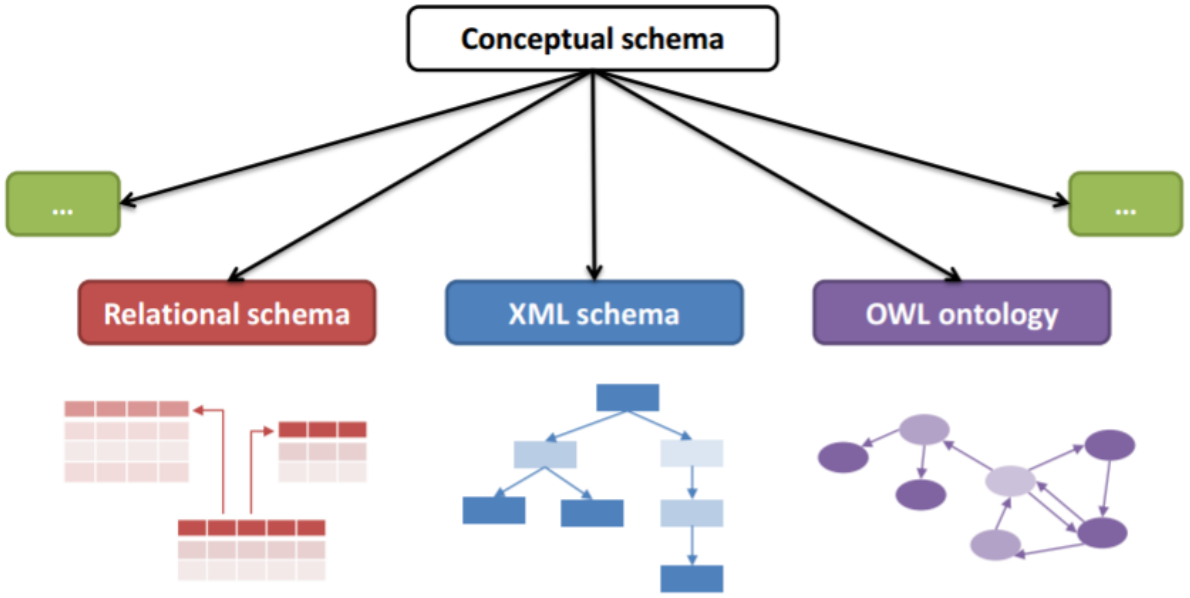
Krok 1 – Výběr správného logického modelu



Důležité – relační modely nejsou vždy tou nejlepší možností
Určení podle:

- Charakteristika dat
- Možností dotazů
- Zamýšlené použití (úložiště, výměna, publikace)
- Určené požadavky

Krok 2 – Vytvoření logického schématu



Cíl – transformace konceptuálního schématu do logického
Aplikace často potřebují více schémat
Může toho být dosaženo automaticky?
 MDD

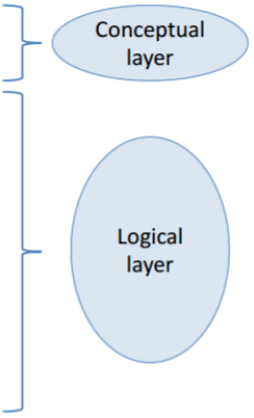
MDD – Modelově řízený vývoj

- Spustitelná schémata místo spustitelného kódu (spustitelná schémata automaticky/poloautomaticky převáděna do spustitelného kódu)
- Jen teorie

Terminologie

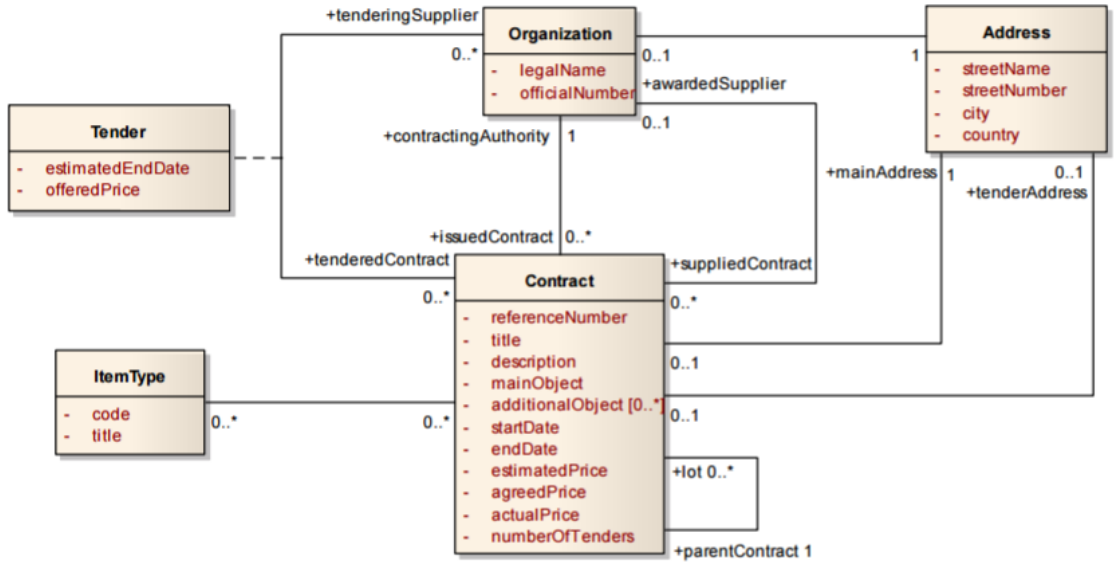
Úrovně abstrakce

1. Na platformě nezávislá
    a. Skryje konkrétní detaily o platformě
2. Specifická platforma
    a. Mapuje konceptuální schéma (nebo jeho část) k danému logickému modelu
    b. Přidá detaily o platformě
3. Kód
    a. Zobrazí schéma ve zvoleném strojově interpretovatelném jazyce
    b. SQL, XML schéma, OWL…



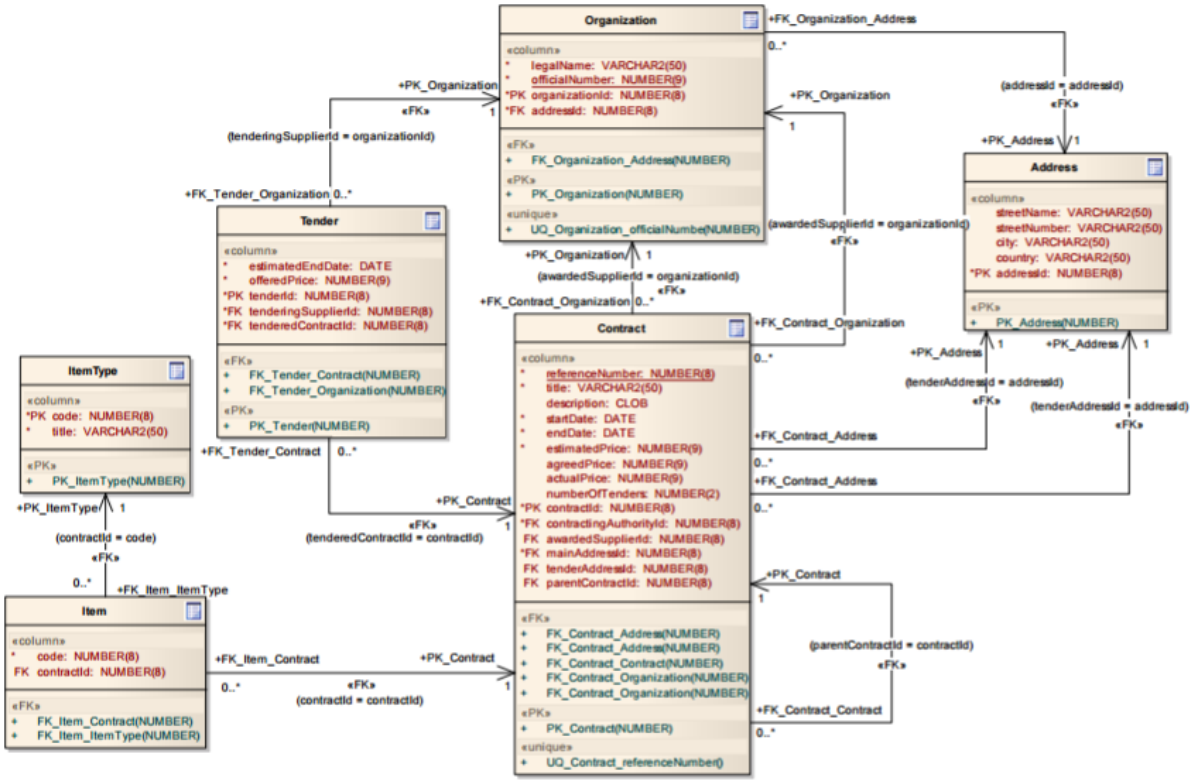
Real-World example

Nezávislé na platformě



Specifická platforma – relační model

- UML class diagram



Kód – SQL (snippet)

- Kód vygenerován automaticky (ze specifické platformy – musí obsahovat všechny nezbytné informace)

```
CREATE TABLE Contract (
  referenceNumber NUMBER(8) NOT NULL,
  title VARCHAR2(50) NOT NULL,
  description CLOB,
  startDate DATE NOT NULL,
  endDate DATE NOT NULL,
  estimatedPrice NUMBER(9) NOT NULL,
  ...
);

ALTER TABLE Contract ADD CONSTRAINT PK_Contract
  PRIMARY KEY (contractId);
ALTER TABLE Contract ADD CONSTRAINT FK_Contract_Address
  FOREIGN KEY (mainAddressId) REFERENCES Address (addressId);
...

CREATE TABLE Organization(...);
...
```

Relační model

- Umožňuje ukládat entity, vztahy a jejich atributy do vztahů (relations)
- Relační schéma
  - Popis relační struktury (všechno kromě dat)
  - S (A_1:T_1, A_2:T_2, ..., A_n:T_n)
    - S = jméno schématu, A_i = jméno atributu, T_i typ atributu

Základní požadavky

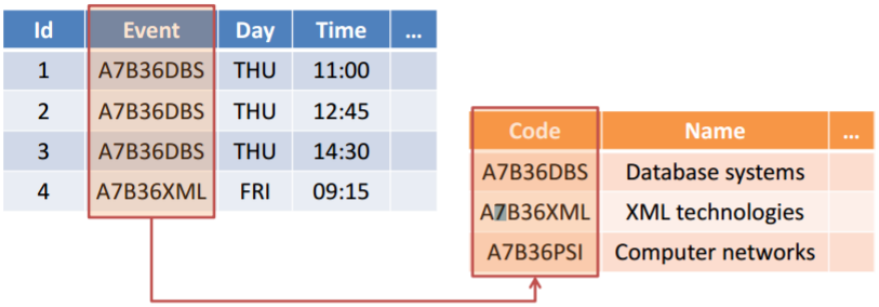
- Atomicita atributů – mohou být použity jen jednoduché typy atributů
- Jedinečnost položek
- Nedefinované pořadí
- Úplnost hodnot - všechny hodnoty musí být specifikovány

Omezení integrity

- Identifikace – každá položka musí být definována jedním nebo více atributy
  - Superkey – set takových atributů
  - Key – superkey s minimálním počtem atributů
- Referenční integrita = nástroj, který pomáhá udržovat vztahy v relačně propojených databázových tabulkách
  - Cizí klíč – množina atributů odkazujících na vztah, který odpovídá klíči referenčního vztahu

Vzorová relační databáze

- Schéma
  - Course(Code, Name, …)
  - Schedule(Id, Event, Day, Time, …), Event ⊆ Course.Code
- Data



Id	Event	Day	Time	…
1	A7B36DBS	THU	11:00	
2	A7B36DBS	THU	12:45	
3	A7B36DBS	THU	14:30	
4	A7B36XML	FRI	09:15	

Code	Name	…
A7B36DBS	Database systems	
A7B36XML	XML technologies	
A7B36PSI	Computer networks	

Relace vs. Tabulky

- Hlavička tabulky ~ relační schéma
- Řádek ~ položka
- Sloupec ~ atribut

Objekt vs. Objektově relační model

- Relační model – data uložená v tabulkách, vhodné pro datově náročné operace
- Objektový model – data uložena jako grafy objektů, vhodné pro individuální přístup k entitám
- Objektově relační mode – relační model obohacený o prvky objektu
  - Atributy mohou být komplexní datové typy
  - Metody mohou být definované jako datové typy

Transformace UML – ER do RM

Konceptuální schéma transformace

Co máme:

- ER – entitní typ, atributy, identifikátory, vztahové typy, ISA hierarchie
- UML – třídy, atributy, asociace

Co potřebujeme:

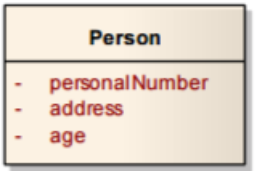
- Schéma relací s atributy, klíče, cizí klíče

Jak to udělat:

- Třídy s atributy relační schéma
- Asociace samostatná relační schémata nebo spolu se třídami (závisí na kardinalitách)

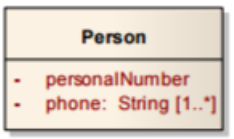
Třídy

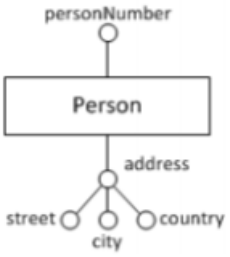
- Třída – samostatná tabulka
  - Person (<u>personalNumber</u>, address, age)
- Umělé klíče – uměle přidané číselné identifikátory
  - Obvykle automaticky generovány a přiřazené
  - Person (<u>personId</u>, <u>personNumber</u>, address, age)



Atributy

- Atribut s více hodnotami – samostatná tabulka
  - Person (<u>personalNumber</u>)
  - Phone (<u>personalNumber</u>, <u>phone</u>)
  - Phone.personalNumber ⊆ Person.personalNumber

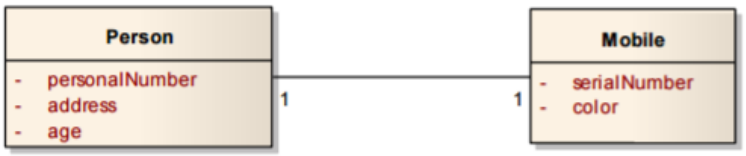




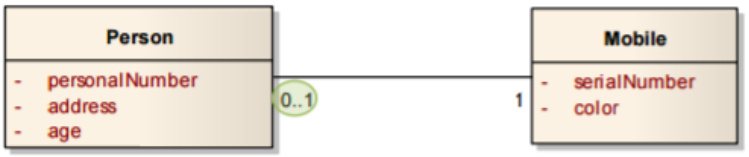
- Kompozitní atribut – samostatná tabulka
    - Person (<u>personalNumber</u>)
    - Address (<u>personalNumber</u>, <u>street</u>, <u>city</u>, <u>country</u>)
    - Address.personalNumber ⊆ Person.personalNumber

Binární asociace

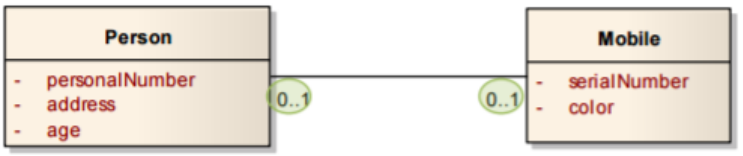
- Multiplicita (1,1):(1,1)



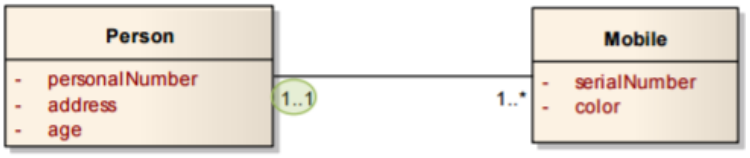
    - Person (<u>personalNumber</u>, address, age, <u>serialNumber</u>, color)
- Multiplicita (1,1):(0,1)



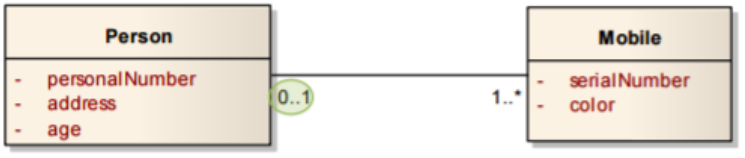
    - Person(<u>personalNumber</u>, address, age, <u>serialNumber</u>)
    - Person.serialNumber ⊆ Mobile.serialNumber
    - Mobile(<u>serialNumber</u>, color)
- Multiplicita (0,1):(0,1)



    - Person(<u>personalNumber</u>, address, age)
    - Mobile(<u>serialNumber</u>, color)
    - Ownership(<u>personalNumber</u>, <u>serialNumber</u>)
    - Ownership.personalNumber ⊆ Person.personalNumber
    - Ownership.serialNumber ⊆ Mobile.serialNumber
- Multiplicita (1,n)/(0,n):(1,1)



    - Person(<u>personalNumber</u>, address, age)
    - Mobile(<u>serialNumber</u>, color, personalNumber)
    - Mobile.personalNumber ⊆ Person.personalNumber
- Multiplicita (1,n)/(0,n):(0,1)



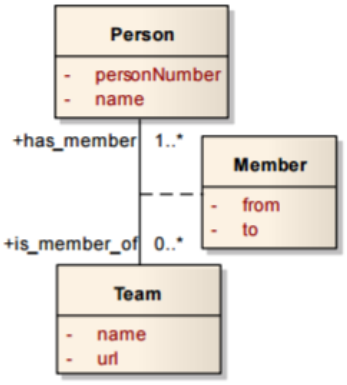
    - Person(<u>personalNumber</u>, address, age)
    - Mobile(<u>serialNumber</u>, color)
    - Ownership(personalNumber, <u>serialNumber</u>)
    - Ownership.personalNumber ⊆ Person.personalNumber

- Ownership.serialNumber ⊆ Mobile.serialNumber

- Multiplicita (1,n)/(0,n):(1,n)/(0,n)

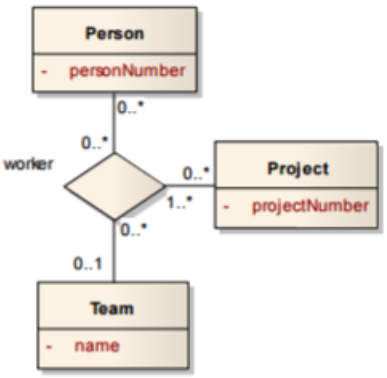
  - Person(<u>personalNumber</u>, address, age)
  - Mobile(<u>serialNumber</u>, color)
  - Ownership(<u>personalNumber</u>, <u>serialNumber</u>)
  - Ownership.personalNumber ⊆ Person.personalNumber
  - Ownership.serialNumber ⊆ Mobile.serialNumber

Atributy asociací



- Atribut asociace
  - Uloženy dohromady s danými asociačními tabulkami
    - Person(<u>personNumber</u>, name)
    - Team(<u>name</u>, url)
    - Member(<u>personNumber</u>, <u>name</u>, from, to)
    - Member.personNumber ⊆ Person.personNumber
    - Member.name ⊆ Team.name

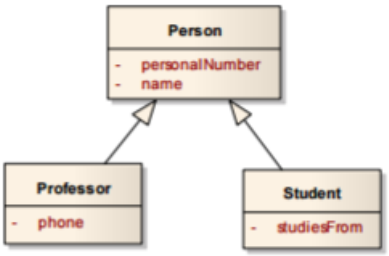
Obecné asociace



- N-ární asociace
  - Univerzální řešení – n tabulek pro třídy a jedna asociační tabulka
  - Person(personNumber)
  - Project(projectNumber)
  - Team(name)
  - Worker(personNumber, projectNumber, name)
  - Worker.personNumber ⊆ Person.personNumber
  - Worker.projectNumber ⊆ Project.projectNumber
  - Worker.name ⊆ Team.name

Hierarchie

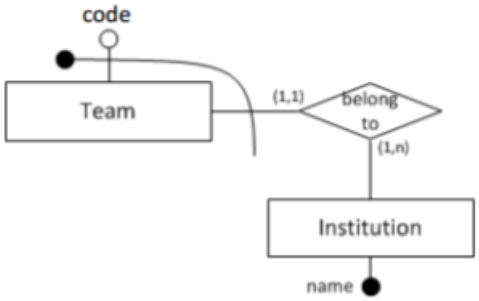
- ISA hierarchie
  - Univerzální řešení – samostatné tabulky pro každý typ se specifickými atributy
    - Person(personalNumber, name)
    - Professor(personalNumber, phone)
    - Student(personalNumber, studiesFrom)
    - Professor.personalNumber ⊆ Person.personalNumber
    - Student.personalNumber ⊆ Person.personalNumber
    - + flexibilita (při změně atributů)
    - – Joiny (když jsou obnoveny úplné údaje)



  - Jen jedna tabulka pro zdroj hierarchie
    - Person(<u>personalNumber</u>, name, phone, studiesFrom, type)
    - Univerzální, ale ne vždy vhodné – typy instance se vyznačují umělým atributem
    - + Žádné joiny
    - - NULL hodnoty vyžadovány

- Oddělená tabulka pro každý typ
  - Professor(personalNumber, name, phone)
  - Student(personalNumber, name, studiesFrom)
  - Není vždy aplikovatelné
  - + Žádné joiny
  - - redundance, integrita (jedinečnost osobního čísla)

Slabé entitní typy



Funkční závislosti

- Výsledek relačního designu = soubor relačních schémat
- Problémy
  - Nadbytek dat
    - Nepotřebné uložení více stejných dat
    - Zvyšuje se se cena místa
  - Vkládání/update/mazání – anomálie
    - Vkládání a update dat musí bránit nadbytečným datům
    - Mazání může způsobit ztrátu dat
  - Nulové hodnoty
    - Nepotřebné prázdné místo, zvyšuje se cena místa
- Řešení normalizace relačního schématu

Příklad „abnormálního“ schématu

EmpId	Name	Position	Hourly salary	Hours completed
1	John Goodman	accountant	200	50
2	Paul Newman	salesman	500	30
3	David Houseman	salesman	500	45
4	Brad Pittman	accountant	200	70
5	Peter Hitman	accountant	200	66
6	Adam Batman	lecturer	300	10

1) From functional analysis we know that position determines hourly salary:
   However, hourly salary data is stored multiple times – redundancy.
2) If we delete employee 6, we lose the information on lecturer salary.
3) If we change the accountant hourly salary, we must do that in three places.

Funkční závislosti

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.
X → Y – hodnoty v X společně určují hodnoty v Y
Pokud X → Y a Y → X, X a Y jsou funkčně ekvivalentní
Pokud X → a, kde a náleží A, pak X je elementární funkční závislost

Uzávěr X^+ je množina všech atributů funkčně závislá na atributech množiny X.

Pokrytí

Pokrytí množiny závislostí X je množina Y, jestliže platí $X^+ = Y^+$
Kanonické pokrytí

- Jeli F' množina, která vznikne z F dekompozicí jejích neelementárních závislostí, platí $F'^+ = F^+$

Redundantní závislost

- Závislost f je redundantní v F, pokud platí $(F-\{f\})^+ = F^+$

Armstrongovy axiomy

Armstrongovy axiomy jsou odvozovací pravidla funkčních závislostí. Pomocí axiomu získáme uzávěr funkčních závislostí spolu s klíči schémat.
Uzávěr F^+ je množina všech funkčních závislostí odvozených od F za použití Armstrongových axiomů
Pokrytí množiny závislostí F je množina G, jestliže platí $F^+ = G^+$
Armstrongova pravidla jsou:

1. Korektní – co odvodíme, platí ve všech relacích
2. Úplná – lze jimi odvodit všechny funkční závislost, které platí na každé relaci
3. 1, 2, 3 jsou nezávislá – odstraněním jakéhokoliv z nich porušíme vlastnosti úplnosti

A = množina všech atributů, F = množina všech funkčních závislostí, XY = XUY (X nebo Y):

Reflexivita – je-li Y ⊂ X, pak X → Y
Tranzitivita – pokud je X → Y a Y → Z, pak X → Z
Kompozice – pokud je X → Y a X → Z, pak X → YZ
Pseudotranzitivita - pokud je X → Y a WY → Z, pak XW → Z
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

Uzávěr množiny atributů

Množina atributů X a množina funkčních závislostí F
Uzávěr X vzhledem k F je množina všech funkčně závislých atributů na X, značíme X^+

Klíč schématu relace

Relační schéma R a množina atributů K
K je klíčem R, pokud jsou všechny atributy z R funkčně závislé na K a K neobsahuje redundantní atributy

Funkční závislosti		Atributy	
Mohou být redundantní	„nepotřebujeme to“	Mohou být redundantní	„nepotřebujeme to“
Mají uzávěr	Všechny odvoditelné závislosti	Mají uzávěr	Všechny odvoditelné atributy
Mohou být elementární	Jediný atribut na pravé straně	Mohou formovat klíče	
Mohou být redukované	Žádné redundance na levé straně		

Minimální neredundantní pokrytí

- Skládá se pouze z redukovaných závislostí
- Příklad

abcd ⟶ e, e ⟶ d, a ⟶ b, ac ⟶ d

b, d jsou redundantní v abcd ⟶ e, vymažeme je ⍰ ac ⟶ e

Normální formy

Pojem normálních forem se používá ve spojitosti s dobře navrženými tabulkami. Správně vytvořené tabulky splňují 4 základní normální formy.

1NF

Každý atribut v relačním schématu má jednoduchý nestrukturovaný typ.

2 dimenzionální pole (tabulka)

- Neobsahuje pole, podtabulky, stromy, struktury…

Person (Id: Integer, Name: String, Birth: Date)

2NF

Neexistují částečné závislosti neklíčových atributů na jakémkoliv klíči.

Zakazuje míchání různých údajů v jedné tabulce.

Company	DB server	HQ	Purchase date
John's firm	Oracle	Paris	1995
John's firm	MS SQL	Paris	2001
Paul's firm	IBM DB2	London	2004
Paul's firm	MS SQL	London	2002
Paul's firm	Oracle	London	2005

Company, DB Server ⟶ everything

Company ⟶ HQ

Špatně

Company	DB server	Purchase date
John's firm	Oracle	1995
John's firm	MS SQL	2001
Paul's firm	IBM DB2	2004
Paul's firm	MS SQL	2002
Paul's firm	Oracle	2005

Company, DB Server → *everything*

Company	HQ
John's firm	Paris
Paul's firm	London

Company → HQ

Dobře

Tranzitivní závislost na klíči

Mějme R(A). Nechť X ⊂ A, Y ⊂ A a C ∈ A, C ∉ X a C ∉ Y.
Nechť dále X → Y → C a neplatí, že Y → X.
Pak říkáme, že C je tranzitivně závislý na X.

3NF

Neklíčové atributy nejsou tranzitivně závislé na klíči.

Company	HQ	ZIPcode
John's firm	Prague	CZ 11800
Paul's firm	Ostrava	CZ 70833
Martin's firm	Brno	CZ 22012
David's firm	Prague	CZ 11000
Peter's firm	Brno	CZ 22012

Company → everything
ZIP code → HQ
Špatně

Company	ZIPcode
John's firm	CZ 11800
Paul's firm	CZ 70833
Martin's firm	CZ 22012
David's firm	CZ 11000
Peter's firm	CZ 22012

ZIPcode	HQ
CZ 11800	Prague
CZ 70833	Ostrava
CZ 22012	Brno
CZ 11000	Prague

Company → everything
ZIP code → everything
Dobře

BCNF – Boyce-Coddova normální forma

Každý atribut je netranzitivně závislý na klíči, pokud jsou klíče více hodnot, tak pokud se nepřekrývají, je to v BCNF.
Každé schéma, které je v BCNF, je i v 3NF. Obráceně neplatí.

Destination	Pilot	Plane	Day
Paris	cpt. Oiseau	Boeing #1	Monday
Paris	cpt. Oiseau	Boeing #2	Tuesday
Berlin	cpt. Vogel	Airbus #1	Monday

Pilot, Day → everything
Plane, Day → everything
Destination → Pilot
Špatně

Destination	Pilot
Paris	cpt. Oiseau
Berlin	cpt. Vogel

Destination	Plane	Day
Paris	Boeing #1	Monday
Paris	Boeing #2	Tuesday
Berlin	Airbus #1	Monday

Destination → Pilot
Plane, Day → everything

Dobře