# Teorie

## Armstrongovy axiomy

Vyvozují všechny funkční závislosti na relační databázi.

- **Primární**

  Reflexivita $Y \subseteq X \Rightarrow X \rightarrow Y$

  Tranzitivita $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$

  Rozšíření $X \rightarrow Y \Rightarrow XZ \rightarrow YZ \forall Z$

- **Sekundární**

  Dekompozice $X \rightarrow YZ \Rightarrow X \rightarrow Y$ a $X \rightarrow Z$

  Kompozice $X \rightarrow Y, Z \rightarrow W \Rightarrow XZ \rightarrow YW$

  Union $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$

  Pseudotransitivita $X \rightarrow Y, WY \rightarrow Z \Rightarrow WX \rightarrow Z$

  Sebeurčení $I \rightarrow I$

  Extensita $X \rightarrow Y \Rightarrow X \rightarrow XY$

## Transakce

Sekvence operací považována jako celek, vytvořená pro účel zajistit správnost vytvoření / smazání / editace dat (během transakce může dojít k selhání). Transakce přistupuje ke kopii dat, změny se potvrdí / zahodí při COMMIT / ROLLBACK.

Transakce splňuje

- Atomicity - buď všechny změny, nebo žádná
- Consistency - před a po transakci jsou data v konzistentním stavu
- Isolation - změny transakce nejsou viditelné jiným transakcím
- Durability - změny jsou trvalé

## Coffmanovy podmínky

Při splnění všech podmínek najednou dojde k deadlocku.

- Vzájemné vyloučení (mutual exclusion) - každý prostředek může v danou chvíli používat pouze jeden proces
- Drž a čekej (hold and wait) - proces může zažádat o další prostředky, i když už nějaké drží
- Neodnímatelnost (no preemption) - jen proces samotný se prostředku může vzdát
- Cyklické čekání (circular wait) - procesy čekají na sebe navzájem v kruhu

## Cover/Překryv dvou množin funkčních závislostí

Máme dvě množiny funkčních závislostí F a G. Říkáme, že G je překryv (cover) F, pokud každá funkční závislost v F může být odvozena z G a naopak, každá funkční závislost v G může být odvozena z F.
Tedy to je nástroj pro určení zda dvě množiny funkčních závislostí vyjadřují stejná omezení na data v databázi.

## Porovnání OLTP a OLAP

**Data:** OLTP systémy pracují s aktuálními daty a vyžadují časté aktualizace, zatímco OLAP systémy pracují s historickými daty a jsou aktualizovány pravidelně, ale ne neustále.

**Dotazy:** OLTP dotazy jsou jednoduché a často modifikují data, zatímco OLAP dotazy jsou složité a slouží k čtení a analýze dat.

**Uživatelé:** OLTP systémy jsou typicky používány velkým počtem uživatelů pro každodenní operace, zatímco OLAP systémy jsou používány méně uživateli, ale těmi, kteří provádějí hloubkové analýzy.

## Persistentní kontext

- **Definice**: Prostředí/stav, kde jsou data uložena trvale (přežívají ukončení aplikace).
- **Příklady**:
    - Relační databáze (např. MySQL, PostgreSQL)
    - NoSQL databáze (např. MongoDB)
    - Souborové systémy
    - Cloudová úložiště

### Operace nad persistentním kontextem

1. **Create (Vytvoření)**
2. **Read (Čtení)**
3. **Update (Aktualizace)**
4. **Delete (Mazání)**
5. **Transakce**
    - Složené operace s vlastnostmi ACID (Atomicity, Consistency, Isolation, Durability).

- **Indexování**
- Vytváření indexů pro rychlejší přístup k datům.
- **Zálohování a obnova**
- **Správa schématu**

## ORM (Object-Relational Mapping)

**Definice**

- **ORM (Object-Relational Mapping)**: Technologie nebo knihovna, která mapuje objektově orientovaný model na relační databázi.

**Funkce a výhody**

- **Automatizace**: Automatické generování SQL dotazů z objektů.
- **Abstrakce**: Abstrahuje relační databázi do objektově orientovaných tříd.
- **Zjednodušení kódu**: Méně psaní SQL dotazů, více práce s objekty.
- **Údržba**: Snadnější údržba kódu díky použití objektově orientovaného přístupu.

**Hlavní operace ORM**

1. **CRUD operace**
2. **Transakce**

## Klíčové pojmy

- **Entity**: Třída, která reprezentuje tabulku v databázi.
- **Session**: Kontext pro provádění operací s databází.
- **Mapping**: Propojení mezi třídami a databázovými tabulkami.

**SQLAlchemy, Hibernate, …**

## NoSQL

**Definice**

- **NoSQL**: Typ databázových systémů, které se liší od tradičních relačních databází a jsou navrženy pro specifické potřeby jako vysoká škálovatelnost, flexibilní datový model, a rychlý výkon.

**Charakteristiky**

- **Nestrukturovaná a semi-strukturovaná data**: Umožňuje ukládat data, která nemají pevně dané schéma.
- **Horizontální škálovatelnost**: Snadné rozšiřování kapacity přidáním více serverů.
- **Vysoký výkon**: Optimalizováno pro rychlé čtení a zápisy, ideální pro velké objemy dat.
- **Flexibilita**: Schéma může být dynamické, což umožňuje snadnější změny struktury dat.

**MongoDB**