# Databázové systémy

**Databáze** – soubor informací (znaky, čísla, diagramy…), jejichž systematická struktura umožňuje vyhledávat tyto informace pomocí počítače

* Velké množství dat, nevejdou se do operační paměti
* Perzistentní – přetrvávající na disku
* Spolehlivé – ochrana proti výpadku systému
* Prvky se většinou v databázi neopakují
* Nezávislost

**Databázový systém** – skládá se ze systému řízení báze dat a z databáze

**Systém řízení báze dat** – software, který umožňuje spravovat databázi, manipulaci s daty, DB engine (MySQL, Oracle, MS SQL Server)

**Databáze** – obsahuje samotná data

**Vývoj DBS:**

1. Plánování – stanovení cílů a požadavků, analýza potřeb uživatelů, sestavení plánu
2. Analýza požadavků – shromáždit a analyzovat detailní požadavky na systém
3. Návrh (Design) – detailní návrh DBS, diagramy, entity, vztahy a atributy, normalizace datových struktur, volba SŘBD, optimalizace výkonu
4. Implementace – tvorba databázových schémat, datových struktur, aplikační logiky
5. Testování
6. Nasazení – do produkčního prostředí
7. Údržba

Relační datový model – způsob uchování dat v tabulkách, tabulka se definuje přes relaci, popisuje veškeré vlastnosti tabulky za pomoci matematických pojmů (logika a teorie množin), definuje jazyky, pomocí kterých se v databázi dají vyhledávat informace, normalizuje zacházení s databázemi, v tabulce řádek = entita; sloupec = atribut entity

* Entitní typ – kategorie nebo třída objektů se stejným typem atributů, např. „Student“, „Předmět
* Entita – konkrétní instance entitního typu, např. Jan Novák, Matematika
* Vztahová množina – kolekce vztahů mezi entitními typy, např. vztah „Zapisuje“
* Vztah – asociace mezi entitami, např. „Jan Novák zapisuje Matematiku“ mezi entitami „Student“ a „Matematika“
* Atribut – vlastnost entity, např. „Jméno“, „Příjmení“, „Věk“
* Kardinalita – určuje počet instancí jedné entity, např. 1:N – 1 entita A může být spojena s více entitami B
* Primární klíč – atribut, který jednoznačně identifikuje entitu v tabulce, např. „ID“
* Cizí klíč – atribut, který odkazuje na primární klíč v jiné tabulce

**Konceptuální model** – vysoká úroveň abstrakce, zaměření na strukturu dat bez ohledu na fyzické aspekty nebo technologie, které se použijí při implementaci, často se vyjadřuje pomocí ER (entity-relation) diagramů, cílem je jasně definovat obchodní požadavky a porozumění datovým entitám a jejich vztahům

**Logický model** – detailnější reprezentace DB systému, vychází z konceptuálního modelu, ale stále nezohledňuje fyzickou implementaci, udává strukturu databáze v rámci konkrétního datového modelu (obvykle relačního), zahrnuje detaily jako tabulky, sloupce a klíče

Konceptuální model se často vytváří v počátečních fázích návrhu databáze a slouží jako základ pro logický model, který se následně překládá do fyzického modelu během implementace.

**Význam normalizace** – redukce redundance – opakování stejných dat na různých místech, integrita dat – ochrana před anomáliemi při vkládání nebo úpravách dat, zjednodušení údržby

**Definice závislostí** – důležité pro pochopení a implementaci normalizace na databázi, funkční závislost určuje to, že jedna hodnota atributu určuje jednoznačně hodnotu druhé sady atributů

* Jednoduchá závislost (X -> Y): Atribut Y je funkčně závislý na atributu X, pokud pro každou hodnotu X existuje přesně jedna hodnota Y.
* Tranzitivní závislost: Pokud X → Y a Y → Z, pak X → Z.
* Parciální závislost: Pokud atribut Y je funkčně závislý na části složeného klíče X.

Pravidla normálních forem

Normalizace se provádí prostřednictvím aplikace normálních forem (NF), které jsou pravidly určujícími úroveň strukturování databáze.

1. První normální forma (1NF)

**Pravidlo:** Všechny atributy tabulky musí obsahovat pouze atomické (nedělitelné) hodnoty. Tabulka musí být bez opakujících se skupin nebo vícenásobných hodnot.

**Příklad:** Tabulka nesmí obsahovat sloupec "Telefonní čísla" s více čísly v jedné buňce; místo toho každé telefonní číslo musí být ve své vlastní řádce.

1. Druhá normální forma (2NF)

**Pravidlo:** Tabulka musí být v 1NF a každý neklíčový atribut musí být plně funkčně závislý na primárním klíči. To znamená, že nesmí existovat parciální závislosti.

**Příklad:** Pokud máme tabulku "Zaměstnanec-Projekt" se složeným primárním klíčem (ZaměstnanecID, ProjektID), pak atribut "Jméno projektu" by neměl být závislý pouze na ProjektID.

1. Třetí normální forma (3NF)

**Pravidlo:** Tabulka musí být ve 2NF a žádný neklíčový atribut nesmí být tranzitivně závislý na primárním klíči. To znamená, že každý neklíčový atribut musí být přímo závislý na primárním klíči a ne na jiném neklíčovém atributu.

**Příklad:** Pokud máme tabulku "Student" s atributy (StudentID, Jméno, Katedra, Vedoucí katedry), atribut "Vedoucí katedry" by měl být přesunut do jiné tabulky, protože je tranzitivně závislý na "Katedra".

1. Čtvrtá normální forma (4NF)

**Pravidlo:** Tabulka musí být v 3NF a nesmí obsahovat vícenásobné nezávislé multihodnotové závislosti.

**Příklad:** Pokud máme tabulku s atributy (Student, Jazyk, Sport), kde každý student může znát několik jazyků a hrát několik sportů, měla by být rozdělena do dvou tabulek (Student-Jazyk a Student-Sport), aby se předešlo multihodnotovým závislostem.

1. Pátá normální forma (5NF)

**Pravidlo:** Tabulka musí být v 4NF a všechny závislosti musí být výsledkem kandidátních klíčů. Zajišťuje, že každá join-dependency je implikována klíčovými závislostmi.

**Příklad:** Tabulka musí být dekomponována tak, aby při rekonstrukci původní tabulky z jejích projekcí nedocházelo ke ztrátě informací.

**Integritní** **omezení** – pravidla a podmínky k zajištění správnosti dat v databázi, data musí splňovat vždy určitá pravidla, typy integritních omezení zahrnují doménovou integritu, entitní integritu, referenční integritu, unikátnost, NOT NULL omezení a CHECK omezení

**Vztah** **mezi** **objekty** **a** **relačními** **tabulkami** – proces mapování objektů na relační tabulky je znám jako Object-Relational Mapping (ORM). ORM nástroje automatizují převod dat mezi objektově orientovanými jazyky (např. Java, C#, Python) a relačními databázemi

**Jazyk** **SQL** – dotazovací jazyk, standardizovaný jazyk pro správu a manipulaci s relačními databázemi. SQL poskytuje příkazy pro provádění různých operací, jako je definice dat, manipulace s daty, kontrola přístupu a transakční operace

**Transakce** – sekvence jednoho nebo více SQL příkazů, které jsou prováděny jako jeden celek

1. Aktivní – transakce začala, ale ještě nebyla dokončena
2. Částečně potvrzená – Všechny příkazy transakce byly vykonány, ale výsledky ještě nebyly zapsány na disk
3. Potvrzená – Všechny změny provedené transakcí jsou trvale uloženy v databázi
4. Selhala – Transakce nemohla být dokončena kvůli chybě
5. Zrušena – Transakce byla zrušena a všechny provedené změny byly vráceny zpět

ACID – zkratka, která popisuje čtyři klíčové vlastnosti transakcí v DB systému, které zajišťují jejich spolehlivost

1. **Atomicity (Atomicita):** Transakce je nedělitelná jednotka práce, která je buď provedena celá, nebo vůbec
2. **Consistency (Konzistence):** Transakce převádí databázi z jednoho konzistentního stavu do jiného konzistentního stavu.
3. **Isolation (Izolace):** Výsledky transakce nejsou viditelné ostatním transakcím, dokud není dokončena.
4. **Durability (Trvalost):** Po potvrzení transakce jsou její změny trvale uloženy i v případě výpadku systému.

**Paralelní zpracování** – vykonávání více operací (transakcí) zároveň, zvyšuje to výkon a efektivitu systému, využívají často uzamykací protokoly k předejetí problémům uvíznutí

**Uzamykací protokoly** – mechanismy, které zajišťují správnou izolaci transakcí pomocí zámků (locks)

1. **Zámky na úrovni řádku (Row-level locks):** Zamykají jednotlivé řádky tabulky, což umožňuje vyšší souběžnost.
2. **Zámky na úrovni tabulky (Table-level locks):** Zamykají celou tabulku, což snižuje souběžnost, ale zjednodušuje správu zámků.
3. **Zámky na úrovni databáze (Database-level locks):** Zamykají celou databázi, což je nejméně granularita, ale také nejméně efektivní pro souběžné zpracování.

**NoSQL databáze** – typy, které nepoužívají relační model a SQL k manipulaci s daty, nabízí vysokou škálovatelnost, flexibilní schéma a práce s velkým množstvím nestrukturovaných dat

1. **Dokumentové databáze:** Ukládají data ve formátu dokumentů (např. JSON, BSON) (MongoDB)
2. **Sloupcové databáze:** Ukládají data v sloupcích namísto řádků, což umožňuje efektivní dotazy na velké množství dat (Apache Cassandra)
3. **Klíč-hodnota databáze:** Ukládají data jako páry klíč-hodnota (Redis)
4. **Grafové databáze:** Ukládají data ve formě grafů, což je vhodné pro modelování vztahů mezi daty (Neo4j)