KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO

DATABÁZOVÉ SYSTÉMY

JIŘÍ HRONEK

Studium databázových systémů (DBS) můžeme členit do tří širokých oblastí podle úhlu uživatelského nebo vývojářského pohledu a hloubce a rozsahu řešených problémů.

***1.Návrh databáze** – Řeší problém, jak navrhnout strukturu informací, jaký model pro popis vztahů, typů, jaké hodnoty ukládat.*

***2.Programování databáze** - Řeší problém, jak konkrétně, jakým programovacím jazykem, manipulovat s daty a získávat informace, pracovat s transakcemi, atd. v aplikacích i v návaznosti na konvenční programovací jazyky.*

***3.Implementace databáze** - Řeší problém, jak navrhnout program, který efektivně řeší všechny důležité databázové procesy a operace, jako je fyzická struktura uložení dat, rychlý přístup k datům, efektivní zpracování dotazu, transakcí, atd...*

1.1 Úvod do databázové technologie

1.1.1 Pojem databáze

Informací se data a vztahy mezi nimi stávají vhodnou interpretací pro uživatele vytvořením struktur, které odhalují uspořádání, vzory, tendence a trendy.

Moderní pojem informační technologie představuje unifikovaný soubor metod a nástrojů pro vývoj software informačních systémů, které zpracovávají data (realizují sběr, uložení a uchování, zpracování, vyhledávání) a většinou ve své architektuře používají databázovou technologii. Od historicky prvních síťových a hierarchických databázových systémů se přešlo na relační systémy. Na relačních komerčních systémech v průběhu několika desetiletí vývoje a využití došlo k odstranění hlavních problémů, většina postupů byla správně definována i úspěšně a efektivně implementována. Tím se tato klasická technologie stala nesmírně robustní a pro jistou třídu aplikací i do budoucna perspektivní. Další vývoj pokračuje paralelně ve formě čistě objektových systémů, které se prosazují hlavně ve speciálních aplikacích (CAD, grafické systémy) a největší perspektiva se dává evolučnímu pokračování relačních systémů - relačně-objektovým systémům. I klasická relační technologie se dále rozvíjí – příkladem jsou paralelní architektury pro zvýšení výkonu SŘBD, deduktivní databáze a expertní systémy.

S rozvojem technologie krystalizují za klasickými IS s databázovými aplikacemi (katalogy, bankovnictví, knihovny, sklady, doprava, …) další aplikace se složitě strukturovanými daty :

- Multimediální databáze – informace ve formě (i kombinované) dokumentů - textů, obrázků, zvuků, videa
- Geografické informační systémy – data ve formě map

- Podnikové systémy pro podporu analýzy, řízení a rozhodování, využívající technologii datových skladů a OLAP s možností dolování dat (data minig)
- Komerční obchodování na internetu , práce s XML daty
- Řízení podnikových procesů – workflow

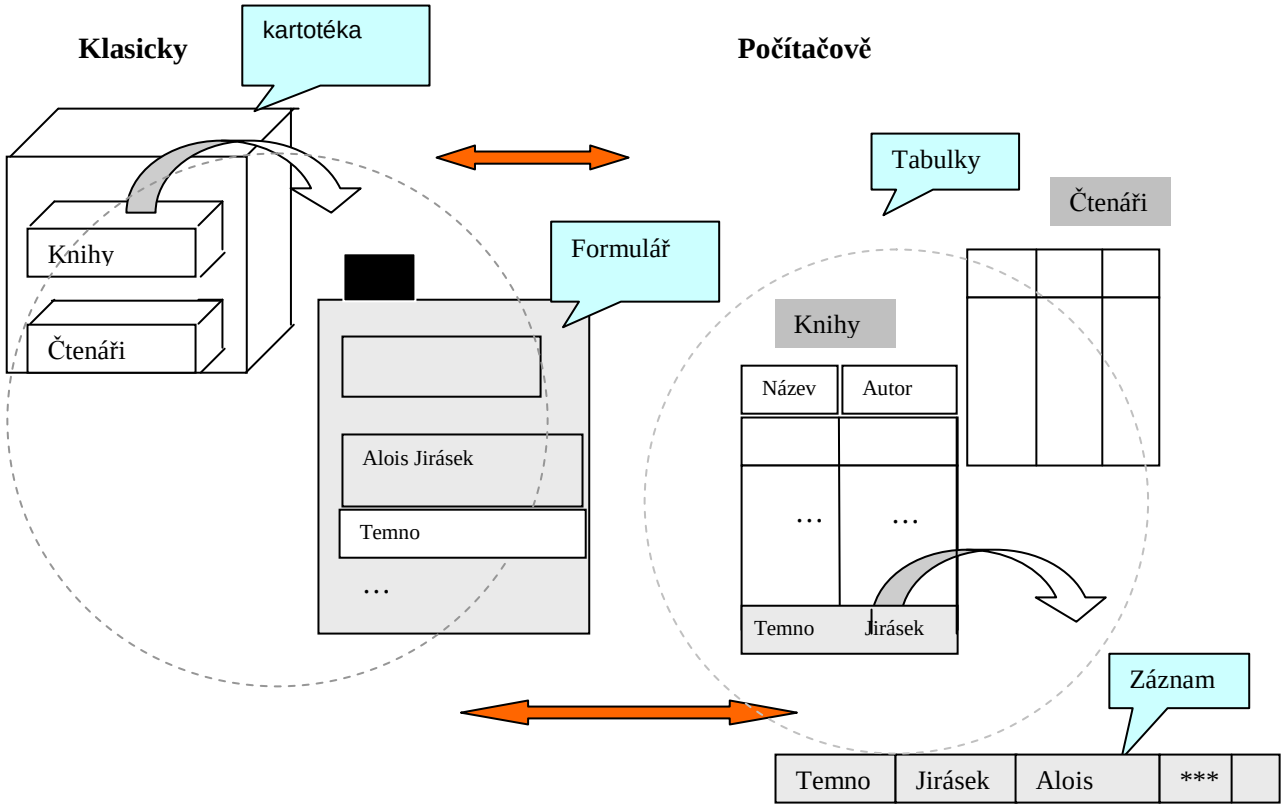
Databáze – registr v elektronické podobě

Rozvoj databázové technologie je reakcí na potřebu efektivně, za pomoci počítačů, zpracovávat různé rozsáhlé agendy, se zaměřením na vyhledávání a aktualizaci dat. Vyhledávání představuje nalezení takových informací - záznamů, které vyhovují podmínkám na požadovaná data. Podmínky jsou formulovány ve formě dotazu ve vhodném dotazovacím jazyce a odpovědí je typicky podmnožina z uložených záznamů (případně ještě zpracovaných – výpočtem z uložených dat získáme odvozené informace). Výsledné údaje můžeme třídit dle různých kritérií, prezentovat ve formě tištěných výstupních sestav. Aktualizace zajišťuje změnu hodnot vlastností objektů nebo zrušení či přidání nového objektu (záznamu) tak, aby informace korespondovaly s realitou. Aby popsané operace byly efektivní, musí být data vhodně uspořádaná a organizovaná na vhodném médiu.

Existuje mnoho způsobů, jak definovat pojem databáze, například: Databáze je úložiště informací, udržované v čase, v počítačově zpracovatelné formě.

> *Databáze – sdílená kolekce logicky souvisejících dat i s popisem své datové struktury, organizovaná pro optimální manipulaci s perzistentními daty a získávání informací pro potřeby informačního systému.*

Pro základní představu porovnejme zjednodušené analogie klasické a elektronické verze na příkladu kartotéky části knihovny (je použit relační model dat, který data uchovává ve formě tabulek):



Obr. 1 Porovnání klasické a elektronické technologie

Výše uvedený příklad je typický svou „plochou“ datovou strukturou, přirozeně transformující data aplikace do dvourozměrných tabulek. Takto pojatá data – tabulky- jsou častou základní logickou datovou strukturou počítačem podporovaných informačních technologií. Programové vybavení, zajišťující perzistentní uložení a bezchybnou údržbu dat na médiích, programové rozhraní pro bezpečný přístup více uživatelů nebo aplikací k manipulacím s daty, získávání informací z kolekcí dat vhodným dotazovacím jazykem, správu transakcí a další funkce, se nazývá systém řízení báze dat – SŘBD. Vše podstatné se v databázové technologii točí kolem dat.

Databázová technologie se zabývá řízením velkého množství perzistentních, spolehlivých a sdílených dat

Data v databázi si můžeme představit jako známá fakta, která nás zajímají, s poměrně pevnou strukturou, uložená trvale v počítači. Mezi nejdůležitější charakteristiky dat v databázích patří

- **Perzistence** – data přetrvávají dlouhodobě od jedné operace ke druhé, nezávisle na použitých programech
- **Velké množství** – operace typicky nevystačí s vnitřní pamětí, proto použití sofistikovaných algoritmů při manipulaci s daty
- **Správnost, nerozpornost** – snaha odhalením nejrůznějších chyb v datech při vkládání nebo úpravě databáze zachovat korespondenci s realitou, vztaženou ke konkrétnímu času, ne nutně k nejaktuálnějšímu (realizováno pomocí integritních omezení)
- **Spolehlivost** – data je možné po poruše počítače zrekonstruovat
- **Sdílení** – s daty pracuje typicky více uživatelů
- **Bezpečnost** – možnost omezit přístup k datům a operacím s nimi
- **Integrace** – spojení několika požadovaných pohledů do komplexní datové struktury
- **Konzistence** – identická data mohou být dočasně nebo trvale uložena na více místech, ale musí mít stejnou hodnotu

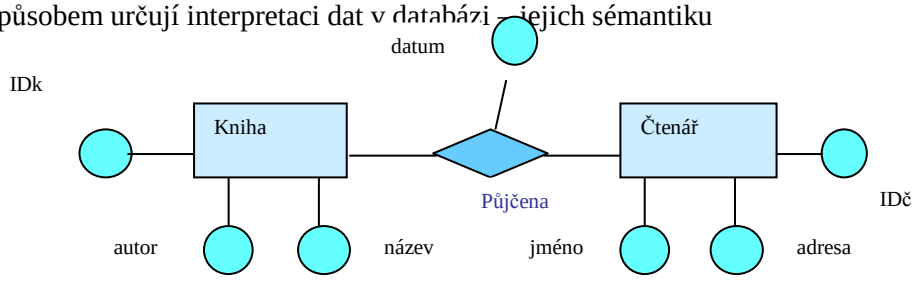
S efektivitou – rychlostí operací – je spojena organizace dat. Klasicky je základem zpracování na fyzické úrovni soubor, každý objekt reality je popsán záznamem souboru, vlastnost objektu je položkou záznamu, která je uložená ve formátu vybraného předdefinovaného typu. Množinu datových souborů, uchovávajících data o nějakém vymezeném úseku reality, nazýváme *databází*. *Instance databáze* je kolekce informací uložených v databázi, nerozporná v konkrétním čase, definuje stav.

Schéma konkrétní *databáze* – informace o metadatech (data o datech, uložena odděleně v katalogu dat), popisuje strukturu dat v databázi - je definováno prostředky použitého datového modelu, se kterým pracuje SŘBD.

Datový model je soubor prostředků a konceptů, popisujících data (sémantiku, strukturu, vztahy, integritní omezení) na určité úrovni abstrakce. Obvykle rozlišujeme tři komponenty – strukturální, operační a specifikace integritních omezení. Konkrétní datový model souvisí s úrovní abstrakce, s pohledem na data v procesu vývoje aplikace – od vymezení požadované části reality ze zadání až po fyzické uložení v počítači. V průběhu vývoje IS se prakticky může použít mnoho různých modelů v závislosti na metodologiích, informačních technologiích, architektuře IS, atd., typická je potřeba přechodů z jednoho typu modelů do druhého v průběhu vývoje softwaru IS s použitím vhodných transformačních pravidel. Možné rozdělení databázových modelů :

- *konceptuální* (data na úrovni pohledů a konceptů) - založené na objektech

(ER model, sémantický model, OO model, funkcionální datový model), na vysoké úrovni abstrakce, bez bližší specifikace budoucí implementace. Je výsledkem datové analýzy, prostředníkem mezi zadavatelem a analytikem v procesu formulace a zpřesňování zadání. Spolu s funkčními závislostmi mezi atributy rozhodujícím způsobem určují interpretaci dat v databázi – jejich sémantiku



Obr. 2 Příklad ER diagramu

- *logické* - založené na záznamech (znalosti seznamů vlastností - atributů <a1, a2, …, an >), které tvoří logický celek (n-tici) jako obraz vlastností abstraktního objektu, prostředky a formu určuje typ datového modelu použitého SŘBD. Mezi historicky první řadíme modely

hierarchický, ***síťový*** – vztah mezi záznamy je v implementaci definován pomocí ukazatelů a data tvoří skupiny záznamů s topologií příslušných grafů – stromu, nebo obecnější sítě.

Stále nejrozšířenější je následující model

relační – Záznamy stejného entitního typu jsou logicky organizovány ve formě dvoudimenzionálních tabulek, vztah mezi záznamy je definován hodnotami vazebních atributů (cizích klíčů), obecně v samostatných tabulkách. Relační databázi tvoří jedna, nebo několik tabulek. Tabulka uchovává informace o skupině podobných objektů reálného světa, např. o knihách. Informace o jednom objektu je na jednom řádku tabulky. Pořadí řádků v tabulce není důležité, nenese žádnou informaci. Sloupec tabulky uchovává informace o jedné nestrukturované vlastnosti objektu.

Př. Definice relací – tabulek (představuje databázové relační schéma, vzniklé transformací z předchozího ER diagramu)

Kniha (IDk : int, autor : char(20), název : char(20))

Půjčena (IDk : int, IDč : int, datum : date)

Čtenář (IDč : int, jméno : char(20), adresa_ulice : char(20), adresa_číslo_pop : char(20))

Kniha

IDk	autor	název
65	Němcová	Babička
3	Jirásek	Temno

Půjčena

IDk	IDč	datum
3	103	1.3.1999
…	…	….

Čtenář

IDč	jméno	adresa_ulice	adresa_číslo_pop
6	Krátká	Okružní	3
103	Novák	Zelená	26

Základní operace v databázi, manipulující s daty, jsou například:

- vložení informací o nové knize (INSERT INTO Kniha VALUES (6, ‘Čapek’, ‘Matka’)
- odstranění informací o vyřazené knize (DELETE FROM Kniha WHERE IDk = 5)
- oprava, aktualizace údaje existující položky (UPDATE Kniha SET stav = ‘zapůjčen’ WHERE IDk = 63)
- dotaz na výběr knih s jistou vlastností – např. rok vydání (SELECT * FROM Kniha WHERE vydání = 1992)

- **fyzické** (data na fyzické úrovni, struktura uložení v paměti – na disku, pomocné podpůrné vyhledávací struktury – indexy, …)

Databázové systémy můžeme rozdělit na klasické, souborově orientované s navigací pomocí ukazatelů – hierarchické a síťové, pracující s tabulkami – relační a na nové směry a přístupy v databázové technologii, což mohou reprezentovat rozšířené relační systémy (relačně- objektové), čistě objektově orientované, XML databáze, deduktivní databáze (Datalog) a distribuované databáze

Databázový systém (DBS) zahrnuje:

- technické prostředky – spolu s dalšími faktory a požadavky uživatele limitují možnou složitost architektury IS nebo častěji je HW návrhem určen. Komerční DBS pokrývají širokou škálu možností s různým stupněm úplnosti a efektivity splnění požadavků, kladených na SŘBD, výkonem, cenou, charakterem aplikace, atd.. Setkáváme se s jednoduššími souborovými systémy (např. dBASE, FoxPro, Microsoft Access) na jedné straně až po komplexní (a nákladné) systémy (DB2, Oracle, Microsoft SQL server)
- programové vybavení (SŘBD, vývojové nástroje)
- data uložená v databázi (DB)
- uživatele – ty můžeme klasifikovat podle různých kritérií (oprávnění k operacím, znalost a úroveň řízení DBS i aplikace, …) do typových skupin např.
    1. administrátor, správce dat - koordinuje všechny aktivity v databázovém systému, zakládá, modifikuje uživatele, rozhoduje o tom, která data a jak budou v bázi uložena – definuje schéma databáze a integritní omezení, určuje schéma uložení dat a metody přístupu k datům, pokud je to nutné, realizuje požadované změny, modifikuje struktury dat, přiděluje přístupová práva k datům i operacím, sleduje výkon a chování DB serveru, zálohuje, rekonstruuje databáze v případě jejího poškození, …
    2. aplikační programátor (tvůrce aplikací) - programuje aplikační programy nad definovanými datovými strukturami, složitější dotazy a transakce použitím DML v hostitelském jazyku nebo jazyky 4. generace.
    3. příležitostný uživatel - umí prostřednictvím dotazovacího jazyka formulovat vlastní specifický dotaz nebo jinak manipuluje s daty
    4. naivní uživatel - (obvykle neprogramátor), který prostřednictvím aplikačních programů pracuje s databází a používá databázi jako informační systém pro ukládání, zpracování a vyhledávání informací

  Pro úplnost - jedno z dalších kritérií dělení DBS je na jedno-uživatelské (hlavně dříve na PC) a více-uživatelské.

Schematicky zkracujeme definici jako spojení SŘBD a dat uložených v databázi

databázový systém = systém řízení bází dat + databáze *DBS = SŘBD + DB*

Základní paradigma – existence dat v databázi je nezávislá na aplikačních programech. To umožňuje na aplikaci nezávislý popis dat v datovém slovníku (např. v systémových tabulkách relačních systémů)

1.1.2 Historický vývoj zpracování dat

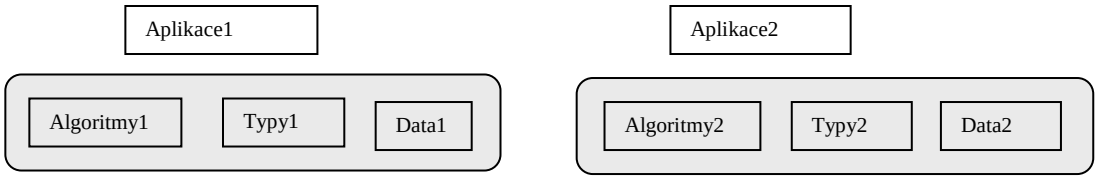
Ilustrativní z pohledu funkce a vývoje programové vrstvy SŘBD je historický přehled používaných metod, který také úzce souvisí se stupněm rozvoje hardwaru a architektury výpočetních systémů.

> *Souborový systém – kolekce aplikačních programů, které zajišťují služby pro uživatele. Každý program definuje a udržuje svá vlastní data.*

50. léta :

Počátečním etapám programového řešení úloh tohoto typu se říká agendové zpracování dat. Vybrané charakteristiky tohoto přístupu:

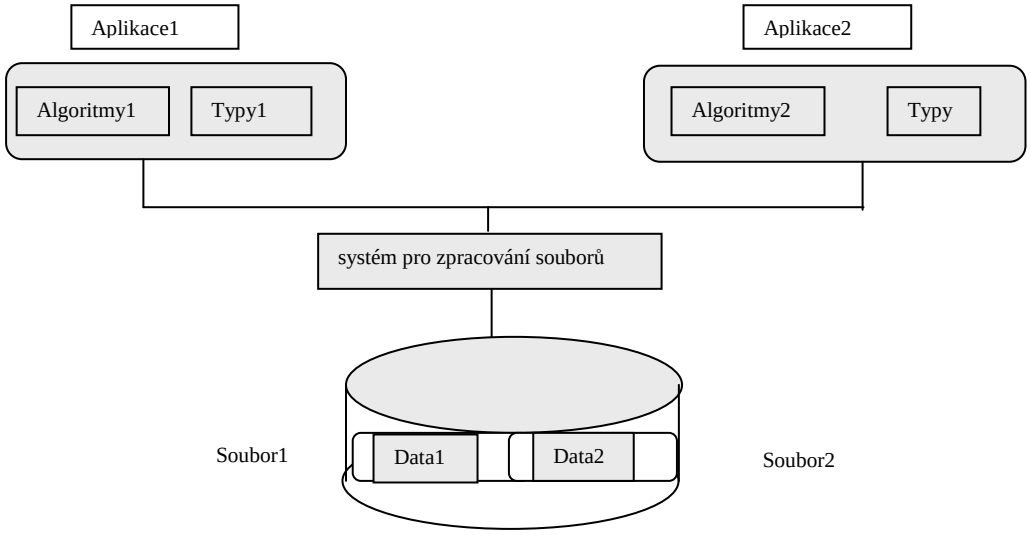
- aplikační programy řeší jednotlivé úlohy - uložení dat na médium, zpracování dat, tisk sestav, …
- soubor programů tvoří ucelenou agendu
- plná závislost dat a programů (každý program řeší nejen vlastní aplikační problém, ale i otázky fyzického uložení dat na médiu; navazující úlohy musí respektovat již vytvořené fyzické struktury dat)
- nízká efektivnost datových struktur i programů
- zpracování v dávkách : - data se ručně zapisují do formulářů
  - z formulářů se zaznamenávají na vstupní médium pro počítač
  - formou primárního zpracování se data načtou do počítače
  - řadou sekundárních zpracování se pak nad daty provádějí výpočty, výběry, tisky sestav …
- řešení ucelených problémových oblastí v jedné agendě (data se sbírají speciálně pro tuto agendu)
- mezi různými agendami nejsou žádné nebo jen minimální vazby
- typická architektura aplikace (vše v programu)
- Použití specializovaných jazyků (Pl /1, COBOL)



Obr. 3 Struktura dat agendového typu

První polovina 60. let

Systémy pracující s interními organizacemi dat s přímým přístupem a interaktivním stykem s uživatelem, vytvoření **systémů pro zpracování souborů,** závazná doporučení CODASYL



Obr. 4 Struktura dat systémů pro zpracování souborů

Nevýhody dosavadních řešení:

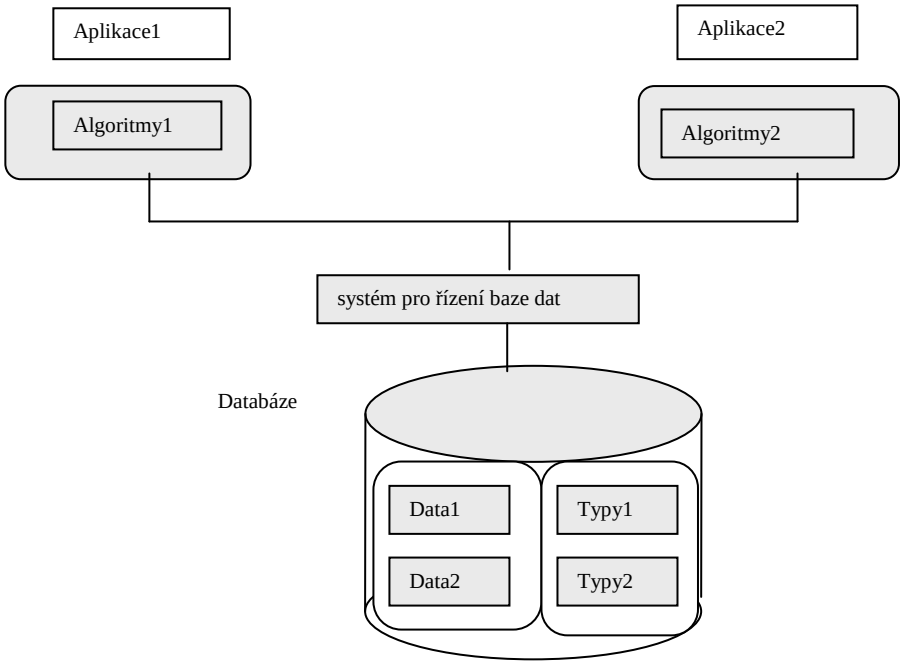
- Soubory navrženy podle potřeb konkrétních programů – malá flexibilita, nízká úroveň abstrakce při pohledu na data – jednoduché datové modely
- Velká redundance dat (aplikační programy vytvářené různými programátory způsobí opakování informací ve více souborech)
- Nekonzistence dat (při změnách hodnot se oprava položky neprovede na všech místech, kde je opakující se informace zapsána)
- Problémy se zabezpečením integrity dat (uložená data musí být většinou aktuální, vyjadřovat skutečnost z reálného světa, vztaženou k jistému časovému okamžiku, implementace integritních omezení)
- Špatná dosažitelnost dat – izolovanost dat (data rozptýlena v různých agendách , různé formáty ekvivalentních dat, problémy s neplánovanými dotazy …)
- Problémy se specifikací a zajištěním ochrany dat (proti získání informací z vnějšku systému, ale i mezi uživateli), souběžného přístupu více uživatelů

Reakcí ve vývoji je základní princip - tendence oddělení dat a jejich definic od aplikačních programů.

Druhá polovina 60. let a 70. léta

vytvoření prvních systémů pro řízení baze dat – SŘBD, vývojem ze souborových systémů. DBS podporovaly různé datové modely - síťový a hierarchický, později relační model dat (Edgar Codd)

- data centralizovaná
- jednotný přístup k informacím, rozvoj speciálních jazyků a rozhraní
- popis dat oddělen od aplikačních programů



Obr. 5 Struktura dat SŘBD

1.1.3 Systém Řízení Báze Dat – SŘBD

SŘBD je kolekce programů, které tvoří rozhraní mezi aplikačními programy a uloženými daty.

Základní funkce:

- na základě použitého datového modelu:

  - umožňuje vytvořit novou databázi a definuje její schéma a data (popis souborů, záznamů, položek, typů, velikostí, vztahů mezi záznamy, indexů),

  - provádí validaci dat (kontrola typu, rozsahu, konzistence, nerozpornosti),

  - případně modifikuje schéma, strukturu dat (vytváří, modifikuje slovník dat)

- určuje strukturu uložení (i pro velké množství) dat
- manipuluje s daty, hlavně umožňuje dotazování, volí metody přístupu (optimalizace), zajišťuje výkonnost
- zajišťuje autorizaci a bezpečnost
- souběžný přístup
- zajišťuje zotavení po poruše
- kontroluje integritu dat
- zajišťuje správu transakcí

SŘBD – softwarový systém, umožňující definovat, vytvořit, udržovat a řídit přístup do databáze.

Hlavní výhody a požadavky na SŘBD

1. Vyšší datová abstrakce – manipulace s formalizovanými strukturami na vyšší, logické úrovni abstrakce
2. Nezávislost dat – schopnost modifikovat definici schématu bez vlivu na schéma vyšší úrovně abstrakce. Analogie s abstraktními datovými typy, detaily implementace jsou skryty.

Nezávislost dat:
Fyzická nezávislost dat - změna fyzického schématu neovlivní aplikační programy (sem patří rovněž např. přidání a zrušení indexů, změna klastrů)
Logická nezávislost dat - změna logického schématu neovlivní aplikační programy (sem patří rovněž např. přidání, modifikace a zrušení entitních typů nebo vazeb)

3. Centralizovaná administrace dat a popis struktury.
4. Možnost formulovat ad hoc dotazy mimo aplikační programy.

Většina SŘBD má vlastní speciální *databázové jazyky*, které zajišťují funkčnost prostřednictvím příkazů a předdefinovaných standardních funkcí. V relačních systémech je prakticky standardem jazyk SQL. Možné dělení jazyků (případně příkazů komplexního jazyka) do kategorií :

1. jazyk pro definici dat (**JDD**) - definice, modifikace a rušení entitního typu, vazby mezi entitními typy a atributů, s použitím logických jmen a datových typů nebo domén (předdefinovaných nebo uživatelských), definuje některá systémová integritní omezení
2. jazyky pro manipulaci dat (**JMD**) - manipulace s atributy, entitami a jejich vazbami, množinami entit, realizují operace typu INSERT, UPDATE, DELETE na logické úrovni. Výběr dat z databáze (operace SELECT) zajišťuje dotazovací jazyk, který je buď součástí JMD, nebo funguje samostatně. Je většinou neprocedurální (formulujeme jen požadavky dotazu, ne postup, algoritmus získání informací)
3. jazyk pro řízení přístupu k datům – prostředky pro realizaci změn schématu a dat databáze se zaměřením na definici oprávnění operací pro každého uživatele
4. jazyk pro řízení transakcí
5. jazyky pro zápis algoritmu
   - v hostitelském jazyce (Cobol, C, Java, ...), pak jsou výše uvedené JDD a JDM vytvořeny jako procedury v hostitelském jazyce a celý SŘBD tvoří nadstavbu tohoto jazyka;
   - vlastní speciální jazyk SŘBD, obsahující příkazy JDD a JMD a navíc programové struktury pro provádění algoritmů - příkazy pro větvení a cykly, někdy podporující i vývoj prezentační vrstvy aplikace.
6. jazyky čtvrté generace (4GLs) – se stávají pravidelnou součástí hlavně aplikačních vývojových prostředků. Patří sem generátory celých aplikací, formulářů, dotazů, výstupních sestav, grafických výstupů, ale i datových tabulek.

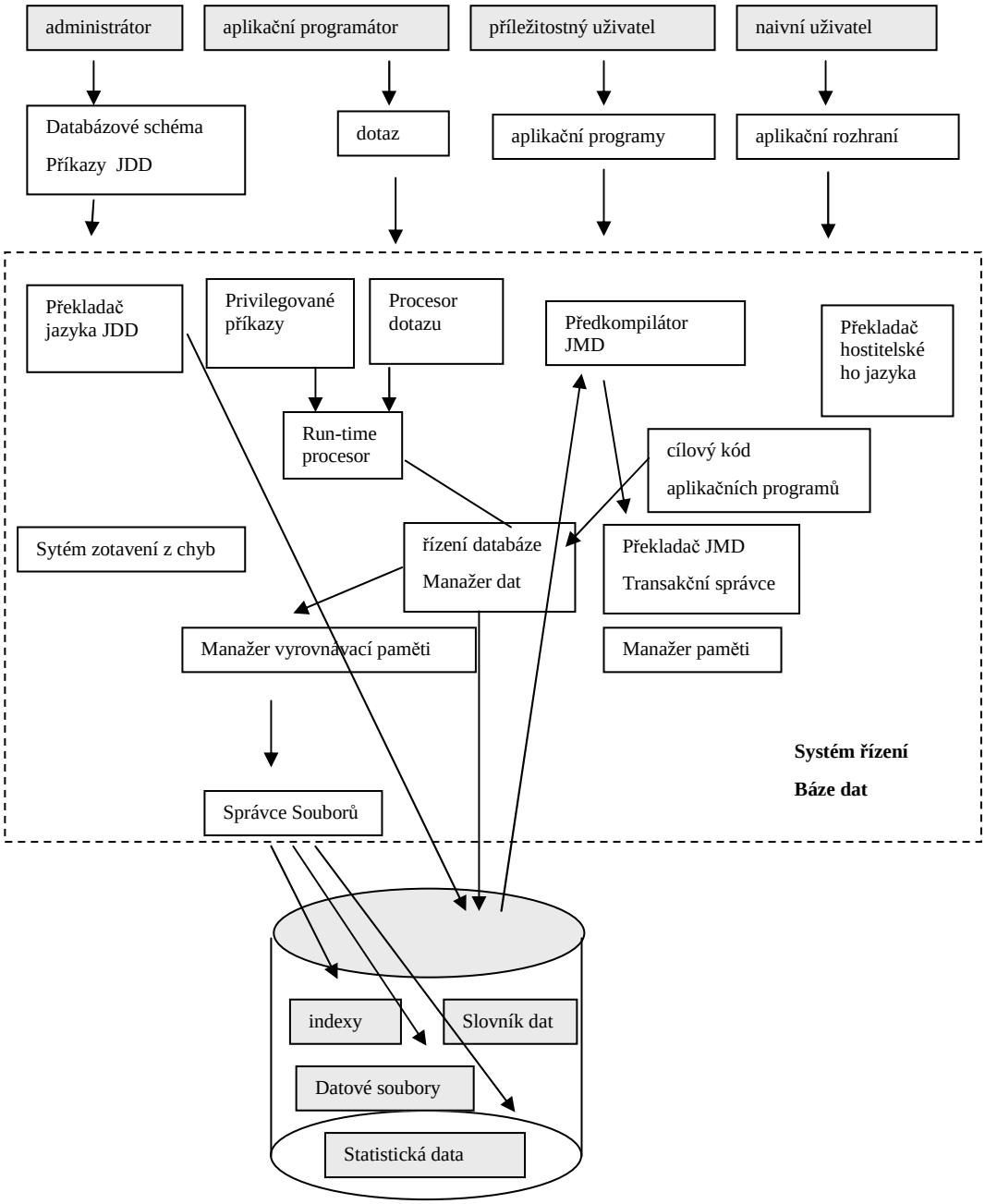
Původně byly SŘBD velké a drahé programové systémy na rozsáhlých a vybavených počítačích. Vývoj sekundárních pamětí a jejich cena dovoluje nasazení vhodných SŘBD na všechny třídy počítačů – od nejmenších systémů, většinou nad jednoduchými datovými soubory, až po nejvýkonnější paralelní architektury, zpracovávající TB informací.

Proč použít databáze?
- pro datovou nezávislost a efektivní přístup
- urychlují a standardizují návrh aplikací
- integrují data a zajišťují bezpečnost
- zajišťují snadnou administraci a minimální redundanci
- umožňují souběžný přístup a zotavení po poruše

1.1.4 Architektura DBS

Pojem architektura DBS, případně IS zahrnuje mnoho možných úhlů pohledu v závislosti na kontextu a etapě návrhu. Tradiční je některá varianta centralizovaného modulárního funkčního schématu relačního databázového stroje, ve které můžeme rozlišit vrstvy - pro optimalizaci a provedení dotazu, relační operace, metody přístupu k souborům, správce vyrovnávací paměti a správce disku.

Například:



Obr. 6 Funkční architektura DBS

Překladač JDD zpracovává definici a změny schématu databáze a ukládá je do katalogu dat. Run-time procesor pracuje s databází při běhu programu. Manažer dat spolupracuje s operačním systémem případně podsystémy vnitřní a vyrovnávací paměti a řízení disků na přenosu dat. Procesor dotazu interpretuje nebo překládá dotaz do optimalizované podoby a předá ho na vyhodnocení. Nejnižší úroveň tvoří subsystém pro ovládání souborů. Zahrnuje fyzickou

organizaci datových souborů, vlastní uložení dat na vnějším médiu a realizaci přenosů dat s pamětí prostřednictvím příslušných manažerů. Na disku jsou uloženy informace čtyř kategorií

1. Data – obsah vlastní databáze
2. metadata ve slovníku dat – popis schématu databáze a integritních omezení
3. statistiky – informace o vlastnostech uložených dat, jako je velikost, charakter hodnot, vzájemné vazby
4. indexy – podpora efektivního přístupu k datům

Mezi nejdůležitější patří architektura z hlediska topologie rozdělení základních typů služeb ve vrstvách programového vybavení IS. Služby můžeme rozdělit na :

1. prezentační – vstup / výstupní zařízení zobrazuje informace(určuje co a jak uživatel vidí), reakce myši, klávesnice, …
2. prezentační logika – interakce uživatele s aplikací (reprezentuje hierarchii formulářů a menu, logiku jejich vztahů)
3. logika aplikace – realizuje aplikační operace a funkce(výpočty, rozhodování) , „prostředky (jazykem) aplikace“
4. logika dat – podpora logiky aplikace operacemi, které mají být prováděny s databází, vyjádřená jazykem SŘBD(SQL – SELECT, INSERT, UPDATE, DELETE)
5. datové služby – operace s databází vně logiky dat, např. definice dat, transakce
6. zpracování souborů – operace na fyzické úrovni, získání dat z disku, práce s vyrovnávací pamětí, … (většinou poskytuje operační systém)

Typický SŘBD se skládá s jednotlivých vrstev – např. 1. Optimalizace a provádění dotazů, 2. Provádění relačních operátorů, 3. Správa souborů a přístupové metody, 4. Správa vyrovnávacích pamětí, 5. Správa disku. Vrstvy 2-5 musí umožnit paralelizmus řízení a zotavení po poruše.

Typické architektury mají historický kontext s návazností na stupeň vývoje HW i SW včetně operačních a síťových systémů. Některé vybrané příklady:

- *Centrální architektura:*V/V (neinteligentní) terminál (služby **1**) – sálový počítač (služby **2-6**): sdílení systémových prostředků, ale primitivní textové rozhraní, problematická rozšiřitelnost o další klienty, zátěž sítě o prezentační data
- *File-server, databáze jako soubory*: stanice, např. PC (služby **1-5**) - file-server (služba **6**) : umožňují rozšiřitelnost o nové klienty, ale velké zatížení sítě, neefektivní souběžný přístup, citlivé na změnu logiky dat.

  Klient – server má několik variant, je nejpoužívanější, znamená dekompozici funkcionality a jistou distribuci dat a databázového softwaru mezi databázovým serverem a jeho klientem. To umožňuje aplikacím škálování zdrojů – horizontální (aplikaci lze zpřístupnit více DB serverů) nebo vertikální (nasazení levnějšího méně výkonného počítače pro klienta a výkonného pro server). Komunikace mezi klientem a serverem probíhá pomocí příkazů SQL s odpovědí typicky ve formě relačních dat. Centralizace architektury podporuje ochranu dat před ztrátou, nebo zneužitím.

- Klient – server : stanice (služby **1-4**)- DB server (služby **5-6**) – mnoho variant architektury, ve které požadavek jednoho procesu (klienta) je poslán k provedení na druhý proces (server), problémy s efektivitou při souběžném přístupu více uživatelů : typicky heterogenní prostředí

- Klient – server se třemi vrstvami : stanice (služby **1**-**2**) - aplikační server (služba **3**)- DB-server (služby **4**-**6**)
- Klient – server s více vrstvami : tenký web klient (služby **1**) – web-server (služba **2)** - aplikační server (služba **3**)- DB-server (služby **4**-**6**)

Dále existuje mnoho možností, jak služby rozdělit nebo netypicky přesunout (např. služba **3** implementována v uložených procedurách na DB serveru nebo na web serveru). Samostatnou kapitolou jsou distribuované DBS, které umožňují fyzické rozdělení nebo replikaci dat na více uzlech sítě.

Shrnutí Databázové systémy tvoří většinou základ pro datovou úroveň v architektuře informačních systémů. Předchůdci moderních DBS byly souborové systémy s aplikačními programy se službami zajišťujícími například tvorbu výstupních sestav, s izolovanými vlastními daty v každém aplikačním programu. Problémy tohoto řešení – např. velká redundance a izolovanost dat, závislost na fyzické struktuře uložení, nedostatečná podpora paralelního transakčního zpracování, nemožnost jednoduše modifikovat menší části dat, atd. vedly postupně k vývoji systému řízení báze dat. SŘBD je programová vrstva, která podporuje efektivní správu velkých perzistentních dat, umožňuje získání informací z databáze prostřednictvím dotazovacích jazyků, s možností pracovat souběžně v transakcích s maximální vzájemnou podporou nezávislosti, izolovanosti a konzistence. Uživatel komunikuje se SŘBD prostřednictvím databázových jazyků, ve kterých jsou jednotlivé příkazy podle účelu děleny na části jazyka definující data (JDD) s možností vytvořit a modifikovat databázové objekty a jazyka manipulující s daty (JMD), který umožňuje provádění operací insert, update, delete a select. Do kontextu DBS můžeme zahrnout hardware – počítač na kterém DBS pracuje, software – SŘBD, operační systém, aplikační programy, dále data a procedury a nakonec různé skupiny uživatelů – administrátor databáze, aplikační programátor, běžný uživatel a podobně. Mezi nejdůležitější moduly SŘBD patří manažer paměti a transakcí a modul zpracování a optimalizace dotazu. Historicky první generaci databázových, souborově orientovaných systémů reprezentují hierarchické a síťové – podle CODASYL modely dat. Nejrozšířenější datový model databázových systémů je relační model, který informace organizuje ve formě tabulek a pro programování nejčastěji používá jazyk SQL, který umožňuje definovat a modifikovat datovou strukturu databáze, manipulovat s daty a administrovat databázový systém s možností vytvářet i modifikovat všechny databázové objekty a určovat oprávnění k operacím. Perzistence dat je zajištěna uložením databáze na disku. Nejčastější architektura databázových systémů je typu klient – server, s podporou uživatelského rozhraní pro přístup do databáze na obou stranách, na klientovi i na serveru.

Pojmy k zapamatování

- Databázový systém, vlastnosti databázových dat, historický přístup
- Systém řízení báze dat, transakce
- Datový model, úrovně abstrakce
- Databázové jazyky, SQL
- Architektura systému

Kontrolní otázky

1. *Jaké specifické vlastnosti mají data v databázích?*
2. *Co je databáze?*
3. *Jaké jsou základní databázové operace?*
4. *Jaký byl historický vývoj hromadného zpracování dat?*

5. *Jaké základní funkce podporuje SŘBD a z jakých logických modulů se skládá?*
6. *K čemu slouží databázové programovací jazyky?*
7. *S jakými datovými modely se setkáme na jednotlivých úrovních abstrakce?*
8. *Jak je možné charakterizovat jednotlivé skupiny uživatelů?*
9. *Jaké typy architektury DBS znáte?*

Úkoly k textu

Kontaktujte uživatele databázových systémů nebo administrátora a pokuste se o charakteristiku jednotlivých komerčních systémů, získání přístupu do databázového systému a seznamte se podle možností s prostředím a ovládáním DBS.

2 Konceptuální modelování

Studijní cíle: Po prostudování kapitoly by studující měl porozumět modelování datové struktury databázové aplikace pomocí ER modelu, popsat základní koncept ER modelu. Měl by definovat a na jednoduchých úlohách použít konstrukty ER modelu při návrhu datové struktury ze zadání, popsat integritní omezení v ER modelu. Měl by vysvětlit pojem slabá entita a ISA hierarchie.

Klíčová slova: ER model, entita, atribut, vztah (relace), identifikační (primární) klíč, kardinalita, povinné členství ve vztahu, ISA hierarchie.

Potřebný čas: 2.hodiny

Rozsáhlejší projekt IS i malá aplikace prochází při vývoji typickými fázemi životního cyklu:

- Analýza (datová a funkční) – odpovídá na otázky : Proč vyvíjíme aplikaci? Kdo ji bude používat? Jaký bude mít přínos pro uživatele? Jaké funkce a potřeby bude splňovat? Kompletní funkčnost systému získáme z analýzy požadavků, z formalizovaného zadání, konzultacemi s uživateli.
- Návrh – je nejdůležitější fáze. Návrh databáze je prováděn modelováním datové struktury použitím ER diagramu tak, aby vyhovoval funkčním požadavkům IS, logické schéma databáze se nakonec transformuje do fyzických struktur databázového systému.
- Vývoj, Implementace – v této fázi se tvoří programy a datové struktury podle návrhu, na návrh a vývoj databáze navazuje vývoj aplikačních programů. Vytváří se prototypy z návrhu, po ověření správnosti následuje implementace aplikace.
- Testování – je důležitá fáze ověřování očekávaných funkcí systému v reálných podmínkách. Případné chyby jsou opraveny a znovu je prováděno testování.
- Údržba - je fáze, ve které se sledují vlastnosti systému, dolaďuje se, v průběhu času se mění požadavky a systém se modifikuje, reorganizuje.

S rozvojem informačních technologií se mění metodologie i metody a prostředky. Konceptuální návrh databáze začíná analýzou, ze které vyplyne, jaké informace je nutno uložit v databázi a v jaké struktuře. Komplexnější přístup k problematice analýzy a návrhu softwaru je jistě náplní speciálních předmětů, následující řádky jsou úvodem k typicky databázovým prostředkům pro konceptuální modelování formou ER-modelu.

Dá se říci, že v databázových aplikacích (soustředíme se na řešení datové struktury na úrovni relačních SŘBD) se často setkáváme s klasickými přístupy – konceptuální ER model nebo rozšířený EER model se transformuje do relační databáze, ale rozšiřuje se i objektově orientovaný návrh, používající UML a nové metody návrhu (diagram tříd, …). Výhodné je transformovat objektový návrh do objektového nebo objektově-relačního databázového systému. Důvody obliby ER modelu hledejme hlavně v převažujících plochých typech aplikací, optimálně využívající relační systémy a setrvačnosti v používání osvědčených postupů, atd.. V této kapitole se soustředíme na první dvě fáze, které řeší základní problémy – jak v databázi popsat sémantiku dat a jak data strukturovat. Jednoznačnost a smysluplnost sémantiky konceptuálního schématu určuje jeho korektnost.

konceptuální schéma databáze je implementačně nezávislý popis části reality, která se týká IS, prostřednictvím modelového pohledu

Konceptuální návrh řeší, prostřednictvím ER modelu otázky:
Jaké entity (objekty) a v jakých vztazích a struktuře jsou v analyzovaném systému ?
Jaké informace o těchto entitách, případně vztazích se mají uložit do databáze ?

Jakým integritním omezením musí vyhovovat data v databázi ?
Také prakticky - jak navrhnout ER diagram tak, aby byl srozumitelný a přehledný (aby na některých úrovních zakrýval příliš velké detaily), aby jeho transformace do relačního modelu byla optimální ?

2.1 ER Model

ER model chápe realitu, případně její sledovanou část, jako množinu objektů (entity) a vztahů mezi nimi (relationship). To představuje přirozený, ale zjednodušený pohled na svět. Model pracuje s těmito konstrukty a pojmy:

Entity odpovídají objektům reálného světa (osoba, věc, …)a jsou popsány pomocí hodnot svých vlastností. *Entita* musí být rozlišitelná od ostatních entit a existovat nezávisle na nich.

Relace – vztah, případně typ vztahu, je vazba mezi dvěma nebo více entitami. Vztahová množina R je určena:

$R \subset E_1 \times E_2 \times \ldots \times E_n = \{ (e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n \}$, kde e_i je entita, E_i je entitní typ, n je stupeň relace.

Atribut je vlastnost entity nebo vztahu.

Popisný typ (doménu atributu) definujeme jako jednoduchý datový typ (množina přípustných hodnot, množina operací).

Doménu atributu tvoří přípustné hodnoty atributu.

Atribut je funkce, která přiřazuje entitám nebo vztahům hodnotu vlastnosti (popisného typu, domény), je charakteristikou entity. Je zadán svým názvem (identifikátorem) a datovým typem. Každá entita je reprezentována množinou dvojic {atribut, hodnota dat}. Rozlišujeme několik typů atributů, např.

- Jednoduché – jedna atomická hodnota a skupinové (strukturované, kompozitní, složené) - struktura nemusí být jednoúrovňová, ale může vytvářet obecně celou hierarchii. Skupinový atribut vznikne vytvořením složitější struktury z jednotlivých položek. V obecných úvahách je užitečné užívat skupinové atributy, pokud potřebujeme někdy přehledně popisovat celou skupinu, jindy dáme přednost podrobnější reprezentaci pomocí jednotlivých složek a zploštěním víceúrovňové struktury.

Atribut -

jednoduchý obsahuje atomickou hodnotu

kompozitní obsahuje strukturu hodnot

vícehodnotový obsahuje množinu hodnot

odvozený obsahuje hodnoty odvozené z jiných atributů

např. ADRESA proti {ULICE, ČÍSLO POPISNÉ, MĚSTO, PSČ, STÁT}

- Vícehodnotové - atributy obsahují opakující se stejné položky, tedy jsou představovány množinou hodnot.

Např. AUTOR KNIHY – {J. Ullman, J. Widom}

- Odvozené atributy – požadovaná hodnoty atributů, které nejsou uloženy v tabulkách, vypočteme z hodnot jiných atributů, uložených v tabulkách.

Např. Známe POČET OBYVATEL a PLOCHA STÁTU a vypočteme HUSTOTA OBYVATEL

HUSTOTA OBYVATEL = POČET OBYVATEL / PLOCHA STÁTU

- S nedefinovanou hodnotou (NULL), případně předdefinovanou hodnotou (default).

Typ entity - množina objektů stejného typu, abstrakce popisující typ objektu. Je definován jménem a množinou atributů. Jednotlivé entity nazýváme také výskyty, nebo instance objektů entitního typu.

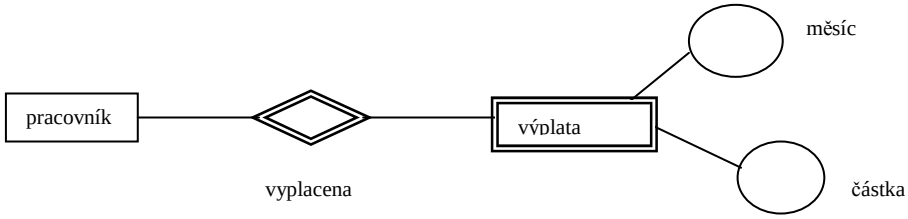
Entitní typ je skupina objektů se stejnými vlastnostmi, identifikované v informačním systému s nezávislou existencí

Silný entitní typ – Entitní typ existenčně nezávislý na jiném entitním typu.

Slabý entitní typ – Někdy nejsou dvě instance jednoho entitního typu rozlišitelné pomocí svých atributů, jsou rozlišitelné až pomocí toho, že jsou povinně v identifikačním vztahu k další entitě

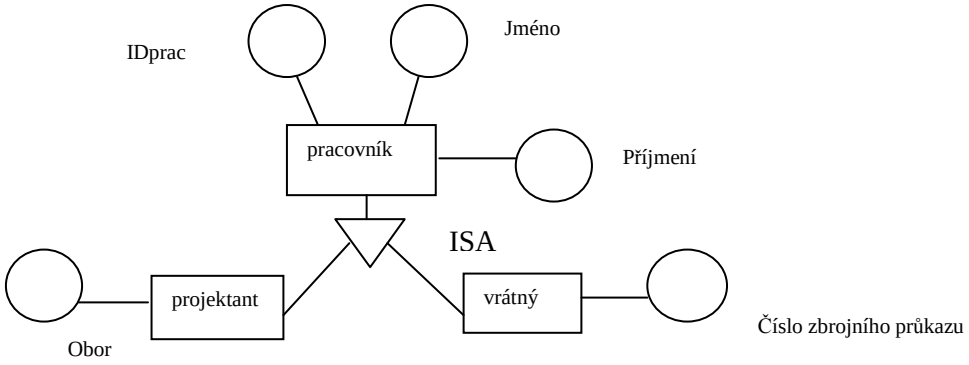
jiného typu (silné, regulární). V identifikačním klíči takového slabého entitního typu musí mýt i vazební atribut, případně atributy (cizí klíč) identifikačního vlastníka. Graficky v ER diagramu se slabý entitní typ často znázorňuje dvojitým obdélníkem, identifikační vztah dvojitým kosočtvercem.

Např.



ISA hierarchie. Někdy se mezi sledovanými objekty vyskytují objekty podobného typu, sémanticky vyjadřující asociace – generalizaci (jedním směrem) nebo specializaci (opačným směrem), popsané řadou stejných atributů a lišících se jen v některých atributech. Specializace je proces maximalizující rozdíly mezi podobnými entitními typy identifikací rozdílných rysů. Naopak generalizace je proces minimalizující rozdíly mezi podobnými entitními typy identifikací jejich společných rysů. Jestliže při většině manipulací s daty obou typů entit se provádějí akce nad společnými údaji, bude vhodné definovat společný typ entity, který bude mít atributy společné a speciální typy se speciálními atributy, které rozlišují odvozené entity. Někdy hovoříme o nadtřídě, respektive podtřídě. Takový vztah definuje ISA hierarchii.

Např. :



Agregace reprezentuje vztah ‘je částí’, ‘obsahuje’ mezi dvěma entitními typy, z nichž jeden představuje celek a druhý jeho část. Zvláštní případ agregace je *kompozice* pro případy silného, nesdíleného vlastnictví části celkem, se stejnou délkou života obou entit.

IO:

Primární klíč je vybraný kandidátní klíč entitního typu, identifikující každou entitu.

Integritní omezení ER modelu se týkají identifikačních klíčů, speciálně primárního klíče, referenční integrity, domény atributů, kardinality a členství ve vztahu.

Kardinalita vazby popisuje maximální počet entit, zúčastněných entitních typů v typu relace

Integritní omezení (IO) ER modelu jsou logická omezení na typy a hodnoty atributů, entit a vazeb taková, aby konceptuální schéma co nejlépe a nerozporně odpovídalo zobrazované realitě.

Účast ve vztahu určuje, zda se ve vazbě zúčastní všechny, nebo jen některé entity

Jeden atribut nebo množinu atributů, které jednoznačně určují entitu v množině entit, nebo vztah v množině vztahů, nazveme *identifikačním klíčem*, obecně nadmnožinou klíče (superkey). Množinu všech minimálních podmnožin atributů entity, které ji identifikují, nazýváme

kandidátní klíče. Jeden z kandidátních klíčů zvolíme za primární klíč. Vybíráme ten, který je z hlediska zpracování dat nejefektivnější, nebo přirozeně identifikující. Často se volí i uměle dodefinovaný identifikační atribut (automaticky generované přirozené číslo), jako klíč pro efektivnější provádění operací. V ER diagramu se značí obvykle podtržením.

Na hodnoty atributů mohou být kladeny omezující podmínky rozličného charakteru, které respektují meze dané sémantikou dat a které představují doménové integritní omezení.

Tabulka ekvivalentních položek ER modelu a relačního modelu

ER model	**Relační model**
Entitní typ .Entita	relace
1:1 nebo 1:N typ vztahu	cizí klíč (nebo vztahová relace)
M:N typ vztahu	vztah. relace a dva cizí klíče
n-ární typ vztahu	vztah. relace a n cizích klíčů
jednoduchý atribut	atribut
kompozitní atribut	množina atributů
vícehodnotový atribut	relace a cizí klíč
množina hodnot	Doména
klíčový atribut	Primární, kandidátní klíč

3 Relační databázové a dotazovací jazyky

Obecné požadavky na dotazovací jazyk můžeme formulovat jako

- blízkost – výsledek dotazu musí být reprezentovatelný v konceptech datového modelu,
- kompletnost – jazyk musí zajišťovat operace datového modelu – např. u OOJ konstruktory, selektory složek, dereference, operace s kolekcemi, …
- ortogonalitu – možnost různě kombinovat a vnořovat operace.

Hlavní sílou relačního modelu je matematicky formalizovaná podpora jednoduchého dotazování. Jazyky SŘBD můžeme rozdělit podle různých kritérií, například na procedurální proti neprocedurálním, nebo podle úrovně na „čisté"(matematické, interní), které tvoří základ pro vyšší uživatelské dotazovací jazyky. Dva reprezentanti interních jazyků relačního modelu :

1. jazyky založené na *relační algebře*, kde jsou výběrové požadavky vyjádřeny jako posloupnost operací prováděných nad relacemi (tím je definován algoritmus vyhodnocení dotazu – procedurální jazyk), vhodné pro reprezentaci prováděcích rozvrhů při optimalizaci.
2. jazyky založené na *predikátovém kalkulu*, které požadavky dotazu zadávají jako predikát **P** charakterizující požadovanou relaci - {a, P(a)}. Jedná se o neprocedurální jazyk. Takové jazyky dále dělíme na
   - *n-ticové relační kalkuly*
   - *doménové relační kalkuly.*

Porozumění relační algebře a relačnímu kalkulu umožní pochopení dotazování v SQL

Jazyky vyšší úrovně, např.

SQL- postavený na n-ticovém relačním kalkulu a vybraných algebraických konstruktech

QBE : využívá doménové relační kalkuly

QUEL : využívá n-ticové relační kalkuly

3.1 Jazyk SQL

Jazyk SQL byl původně navržen v roce 1975 u firmy IBM jako dotazovací jazyk (původní název Sequel v systémech R). V roce 1986 definován standard ANSI (American National Standard Institute), standard ISO – SQL/86, v roce 1989 přidán integritní dodatek –SQL/89. Přelomovým rokem byl rok 1992, standard SQL/92 se třemi úrovněmi souladu – Entry, intermediate, full. Další standardy (SQL3, SQL1999) evolučně směřují k relačně-objektovým databázím a různým rozšířením. Prakticky existují firemní dialekty SQL/92 s rozšířením.

Jazyk obsahuje příkazy pro definici databáze a vytvoření jejích objektů a integritních omezení, interaktivní jazyk pro manipulaci s daty – včetně komplexních dotazů, řízení přístupových práv, řízení transakcí, které z SQL dělají mnohem silnější prostředek, než jen dotazovací jazyk.