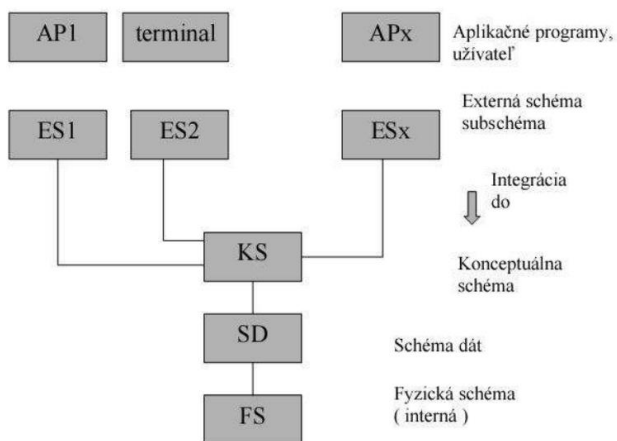


## Úvod do spracovania údajov

- spracovanie údajov – najrozšírenejšia aplikácia počítačov
- problémy hromadného spracovania údajov: – redundancia a nekonzistentnosť (každá aplikácia má svoju údajovú základňu)
  - obtiažnosť prístupu
  - každá nová požiadavka vyžaduje nový program
  - izolácia údajov
  - viac užívateľské aktualizácie
  - možnosť vzniku nekonzistencie

## Architektúra ANSI/SPARC

- pôvodne trojúrovňová architektúra
- (1975) americkým národným výborom pre štandardy (ANSI/SPARC - American National Standards Institute - Group on Data Base Management Systems - Committee on Computers and Information Processing / Standard Planning and Requirements Committee)
- **základný problém:** – počítač pracuje s bitmi (100110011100110)
  - užívateľ s pojmami a abstrakciami reálneho sveta (študent, predmet, ...)
- Databázové systémy umožňujú pracovať so všetkými úrovňami abstrakcie – hoci v skutočnosti procesor pracuje len s **fyzickou úrovňou**



- **Externé schémy (ES)** - Existuje skupina užívateľov s rôznymi prístupmi do databázy. Používa sa tiež pojem externé organizácie, poskytuje vlastne pohľad na tú časť bázy dát, ktorá je pre konkrétneho užívateľa prístupná a poskytuje mu určitý komfort v rámci využívania funkcií.
- **ES1** - výsek reality vnímaný skupinou užívateľov (napr. študijné oddelenie)
- **Konceptuálna schéma (KS)**
  - Je určená na návrh štruktúry celej bázy dát a jej zobrazenie pre všetkých zainteresovaných.
  - Konceptuálnu schému možno tvoriť alebo tzv. podnikovým prístupom, ktorý je nezávislý na jednotlivých používateľských pohľadoch a mal by čo najvernejšie odrážať danú realitu systému, alebo tzv. **integračným prístupom, kedy je zjednotením rôznych používateľských pohľadov na dáta**
  - KS nie je súčtom jednotlivých ES, vzniká ich **integráciou**
  - Musíme vylúčiť **duplicitu** spoločných údajov
  - KS popisuje aké údajové štruktúry existujú a ako sú prepojené (podstatné mená popisujú štruktúru, slovesá predstavujú funkcie).
  - **Schéma dát** - je vlastne transformovaná KS, vzniká **logický model** – závisí od použitého dátového modelu
  - **Fyzická schéma** vzniká modifikáciou SD (logického modelu) pridaním špecifických charakteristík prostriedku, ktorým je realizovaný dátový model v počítači.

## Konceptuálne modelovanie

- Konceptuálne modely (conceptual models) – prostriedky pre návrh dátových schém
- Použitím konceptuálneho modelu vznikne konceptuálna schéma (KS)
- KS **nerieši funkčnú analýzu aplikácie**
- Defacto štandardom v nástrojoch konceptuálneho modelovania sú **E-R modely** (Chen 1976)

## Entitno- relačný model

- Entito-relačné modely (ERM) sú prostriedkom na **modelovanie reality**.
- Každý model je založený na odrazoch skutočného sveta.
- ER model pozostáva z množiny základných objektov - **entít** a ich vzájomných vzťahov - **relácií**
- Metóda entito - relačného modelovania je všeobecne uplatniteľná, nezávisí od typu dátového modelu.
- Vizuálne reprezentuje dátový model ako sústavu grafických znakov
- **Entito-relačné modelovanie zahŕňa nasledujúce kroky:**
  - definície nezávislých typov entít a ich identifikačných kľúčov – definície typov vzťahov
  - definície typov slabých entít
  - formulácie atribútov, ako u entitných tak u vzťahových typov

## Entita

- **Entity** sú jasne navzájom rozlíšiteľné objekty reálneho sveta, ktoré sú schopné nezávislej existencie, sú jednoznačne odlišné od ostatných objektov a majú súvislosť s realitou, ktorú modelujeme. Môžu byť hmatateľné alebo abstraktné, ako osoby, miesta, veci či udalosti,
- entita môže byť:
  - konkrétna** – kniha, osoba
  - abstraktná** – prázdнины, čas Entita
- V ER modeli sa používa každá entita výlučne v **jednotnom čísle**, pretože definuje typ a nie konkrétne objekty.
- Množina entít - obsahuje entity toho istého typu. Množiny entít sa **môžu prekrývať** t.j. nemusia byť disjunktné.
- zamestnanec banky a zákazník banky môže byť tá istá osoba

## Atribút

- Entita je reprezentovaná množinou atribútov.
- príklady atribútov pre entitu človek: - meno, R.č., adresa
- Atribúty popisujú entity, ktorým sú priradené. Atribútom (attribute) budeme rozumieť funkciu priradujúcu entitám alebo vzťahom hodnotu (tu popisného typu), určujúcu niektorú podstatnú vlastnosť entity alebo vzťahu.
- Formálne: **atribút je funkcia**, ktorá **zobrazuje množinu entít do domény**. Preto každá entita je popísaná množinou párov (atribút, hodnota), jeden pár pre každý atribút množiny entít.

## Doména

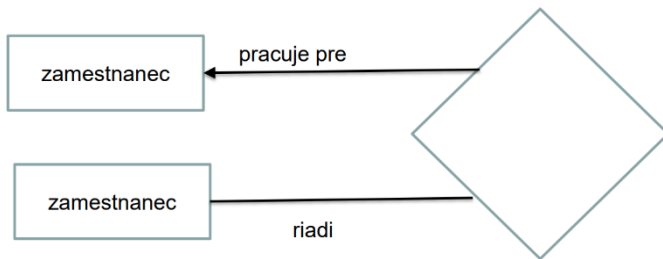
- Doména – je množina dovolených hodnôt atribútu. Jej definovanie nám umožňuje kontrolu správnosti typov údajov pri zápise do systému
- Príklad: doména dňa v mesiaci sú čísla 1 - 31

## Relácie

- Relácia vyjadruje vzťah medzi rôznymi entitami, napr. zákazník Novák vlastní účet č. 401
- Množinu relácií tvoria relácie **toho istého typu**. Formálne je to matematická relácia na  $n \geq 2$  množinách entít
- Ak  $E_1, E_2, \dots, E_n$  sú množiny entít, potom množina relácií  $R$  je podmnožinou  $\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$  kde  $(e_1, e_2, \dots, e_n)$  je relácia

## Rola entity v relácii

- Úloha, ktorú hrá entita v relácii sa nazýva rola.
- Normálne je rola implicitne známa a netreba ju špecifikovať.
- V prípade, že relácia vyjadruje vzťahy medzi dvoma entitami tej istej množiny entít, je možné vyjadriť rolu každej entity podieľajúcej sa na relácii.
- príklad: vzťah medzi dvoma zamestnancami jeden hrá rolu manažéra, druhý podriadeného Rola entity v relácii



## Relácie

- relácie môžu (ale nemusia) mať deskriptívne (popisné) atribúty
- Relácia (vzťah) môže byť klasifikovaná podľa stupňa, kardinality alebo existencie

## Stupeň relácie

- Stupeň relácie je určený počtom entít, ktoré tvoria danú reláciu.
- Najobvyklejším prípadom sú **binárne relácie**, ktorých stupeň je 2 (dve entity vytvárajú reláciu)
- Menej časté sú **ternárne relácie**, ktorých stupeň je 3 (tri entity vytvárajú reláciu)

## Kardinalita

- Kardinalita vyjadruje počet entít, ktoré môžu byť viazané s inou entitou na základe relácie.
- Je to vlastne maximálny počet inštancií jednej entity, ktoré môžeme asociovať s jednou inštanciou inej entity.
- Existujú tri všeobecné typy kardinality a to **1:1**, **1:N** a **M:N**.

## Kardinalita 1:1

- Relácia jedna-k-jednej (1:1) je vzťah keď je jedna inštancia entity A vo vzťahu s jednou inštanciou entity B.
- Príklad európske manželstvo - Každý manžel má práve jednu unikátnu manželku a každá manželka má práve jedeného unikátneho manžela.
- (modelujeme aktuálny stav, nie históriu obyvateľa, v prípade modelovania histórie nie je možný vzťah 1:1 vzhľadom na rozvody)

1 : 1

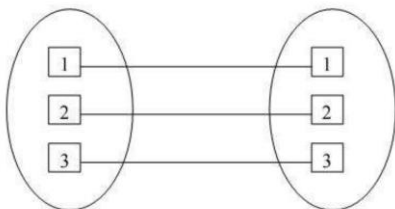


Schéma kardinality 1:1

## Kardinalita 1:N

- Vzťah jedna-k-mnohým (1:N) reprezentuje prípad, keď je na jednej strane jedna inštancia triedy A vo vzťahu s viacerými inštanciami entity B.
- Pre jednu inštanciu entity A je priradených nula, jedna alebo viac inštancií triedy B, ale pre inštanciu B je priradená práve jedna inštancia entity A.

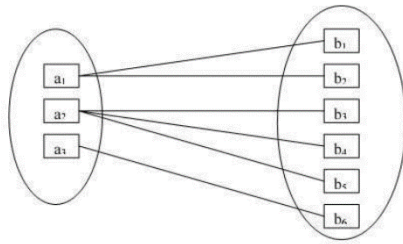
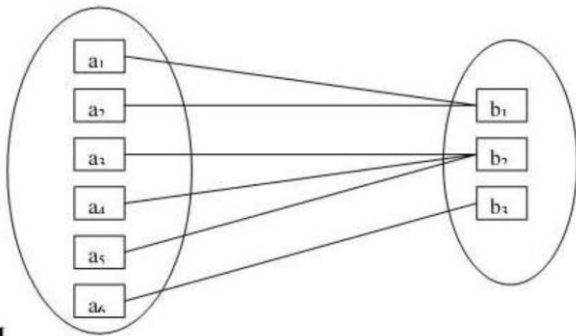


Schéma kardinality 1:N

- Príkladom kardinality relácie 1:N : – jedno oddelenie má viac pracovníkov – každý pracovník je pridelený práve do jedného oddelenia – vzťah trieda – žiak • vzťah jedna-k-mnohým patrí medzi najčastejšie existujúce kardinality.

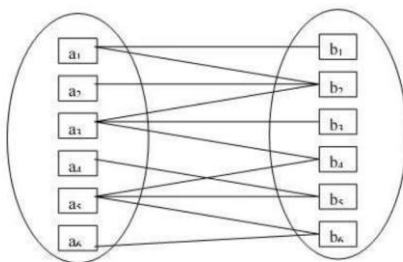
## Kardinalita N:1



## Kardinalita N:M

- Relácia mnohé-k-mnohým (M:N) je vzťah, kde vystupuje nula, jedna alebo viac inštancií entít na oboch stranách.
- početnosti m a n NEMUSIA byť rovnaké čísla!
- Určenie kardinality zvyčajne závisí od reálneho sveta, ktorý modelujeme.

N : M



- Príkladom takéhoto vzťahu N:M sú entity študent a predmet, nakoľko jeden študent si môže zapísať m predmetov a na jeden predmet je zapísaných n študentov, čo znamená že ide o vzťah M:N

## Existencia

- Dôležitou triedou obmedzení je závislosť medzi entitami.
- Ak existencia entity **x závisí od existencie entity y**, potom hovoríme, že **x existenčne závisí od y**.
- Znamená to, že **ak zmažeme y, musíme zmazať aj x**
- y = dominantná entita (účet)
- x = podriadená entita (transakcia na účte)

## Primárne kľúče

- Dôležitá úloha pri modelovaní – rozlíšenie inštancií entít alebo relácií navzájom.
- Na rozlíšenie inštancií entity vyberieme z atribútov tzv. **superkľúč** t.j. jeden alebo viac atribútov, ktoré umožnia identifikovať individuálnu inštanciu entity.
- Rodné číslo je superkľúč, meno nie.
- Najmenší možný superkľúč sa **nazýva kandidát na kľúč**. Môže ich existovať viacero.
- rodné číslo alebo meno a adresa
- **Primárny** kľúč je potom vybraný z kandidátov na kľúč tvorcom databázy
- Niektoré entity nemajú dostatok atribútov na vytvorenie primárneho kľúča (PK). (transakcie na rôznych účtoch môžu mať to isté číslo) • **Entita bez PK sa nazýva slabá entita** • **Entita s PK sa nazýva silná entita**
- **Koncept slabých a silných entít súvisí s existenčnou závislosťou**
- silná = dominantná, slabá = podriadená
- Aj inštalácie slabej entity potrebujeme identifikovať
- **diskriminátor slabej entity** je množina atribútov, ktorá umožňuje identifikáciu (číslo transakcie)
- PK slabej entity potom tvorí PK silnej entity, na ktorej je slabá entita existenčne závislá a diskriminátor slabej entity.
- PK relácie je tvorený zo všetkých atribútov, ktoré tvoria PK množín entít definujúcich množinu relácií.

## Entity vs. inštalácie

- Entita môže byť:
  - Hmotná, napr. osoba alebo výrobok
  - Nehmotná napríklad úroveň jazykových poznatkov
  - Udalosť napr. koncert, svadba
- Príklady inštancií
  - Entita zvieratá má inštalácie ako dalmatín, Siamská mačka, tiger
  - Študent má inštalácie ako Ján Mrk, Michal Hup, ...

## Atribúty

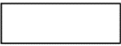



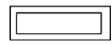

- Opisujú entitu – Atribút vs. Hodnota atribútu - Farba vs. modrá
  - Typ zvieratá vs. pes
- Môže mať najviac jednu hodnotu v danom čase
- Jeden z atribútov **musí byť definovaný ako jednoznačný identifikátor** unique identifier (UID) – primárny kľúč

## Atribút













- **Nestály** (volatile) sa môže meniť v čase, napr. vek – radšej používajte **stále** (non-volatile) atribúty ako dátum narodenia namiesto veku
- **Povinný** (mandatory) vs. **voliteľný** (optional) – emailová adresa je povinný atribút pre ZAMESTNANCA
  - emailová adresa je voliteľný atribút pre ZÁKAZNÍKA ak modelujeme online katalóg Členstvo v relácii
- **Je povinné** (Mandatory) alebo **nepovinné** (Optional) – povinné
  - Na opis použijte slovo **MUSÍ**
  - Označuje sa \* a plnou čiarou
  - nepovinné
    - Na opis použijte slovo **môže**
    - Označuje sa ° a čiarkovanou čiarou
  - príklad:
    - Každé ODDELENIE musí mať jedného alebo viacero ZAMESTNANCOV
    - Každé ODDELENIE môže mať jedného alebo viacero ZAMESTNANCOV

## ENTITO-RELAČNÝ DIAGRAM (ERD)

### E-R diagram, Chenova notácia

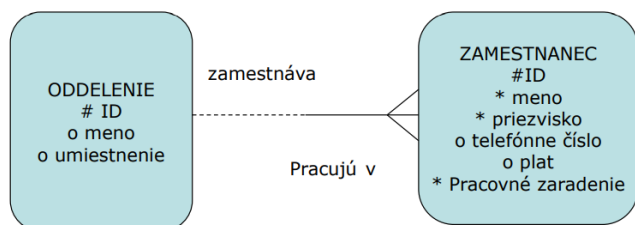
- entita 
- atribút 
- relácia 
- spojnica 
- každý komponent je označený menom
- slabá entita 
- šípka sa   
používa na označenie kardinality

### Notácia „Crow's Foot“

ORACLE	IE	
		1
		M
		práve 1
		0 alebo 1
		1 až N
		0 až N

### ERD, Notácia „Crow's Foot“

- Entity predstavujú boxy s oblými rohmi



### ERD konvencie - prehľad

- Entity predstavujú zaoblené obdĺžniky
- Názvy entít sú v **jednotnom čísle a písané veľkými písmenami**
- Atribúty sa píšú pod názov entity

# označuje UID (unique identifier – primárny kľúč)

\* označuje povinný atribút

o označuje voliteľný atribút

#### Relácie znázorňujú čiary

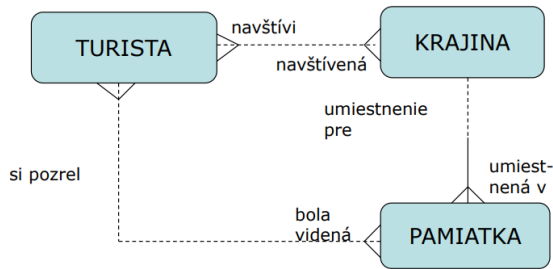
– plné čiary predstavujú povinnú reláciu

– čiarkované čiary predstavujú voliteľnú reláciu

- Ukončenie čiar vyjadruje kardinalitu – “jednoduchá päta” označuje “práve jedna”  
– “crow's foot” označuje “jedna alebo viac”

## ERD príklad (Oracle)

### Konceptuálny ERD



## Maticový Diagram

	TURISTA	KRAJINA	PAMIATKA
TURISTA		navštívi	si pozrel
KRAJINA	navštívená		umiestnenie pre
PAMIATKA	bola videná	umiestnená v	

## Zovšeobecnie

- **Zovšeobecnenie** je výsledkom spojenia dvoch alebo viacerých množín entít do množiny entít vyššej úrovne.
- študent, učiteľ → človek

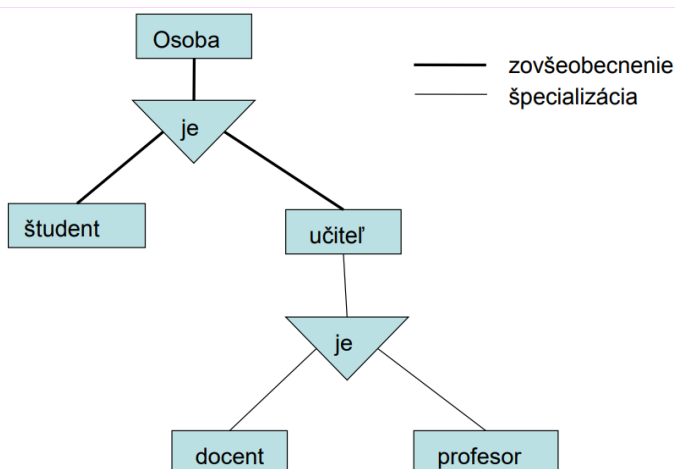
## Špecializácia

- **Špecializácia** je výberom podmnožiny z množiny entít vyššej úrovne do množiny entít.
- študent → končiaci študent

## Rozdiel medzi zovšeobecnením a špecializáciou

- Pri **zovšeobecnení** každá entita vyššej úrovne musí byť zároveň z jednej z množín entít nižšej úrovne
- Pri **špecializácii** neplatí toto obmedzenie – môže existovať entita, ktorá nie je z množiny entít nižšej úrovne

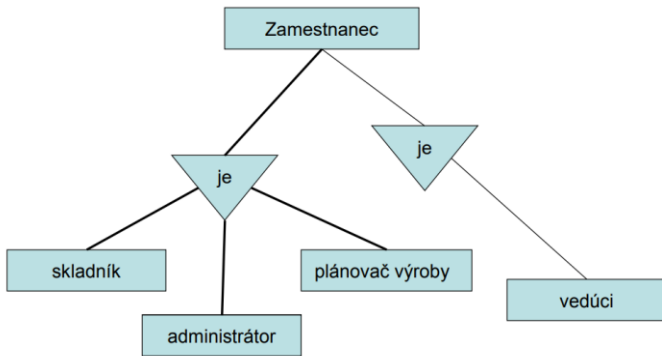
Príklad



## Zovšeobecnenie a špecializácia

- **Zovšeobecnenie** sa používa na zvýraznenie podobnosti medzi typmi entít a skrytie ich rozdielov.
- **Špecializácia** zdôrazňuje rozdiely medzi množinami entít vyššej a nižšej úrovne. Atribúty sa používajú ako prostriedky pre rozlíšenie, zabezpečuje sa to dedením.

## Príklad

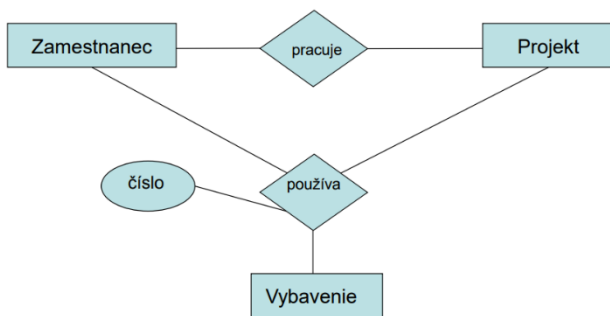


## Supertyp a subtyp

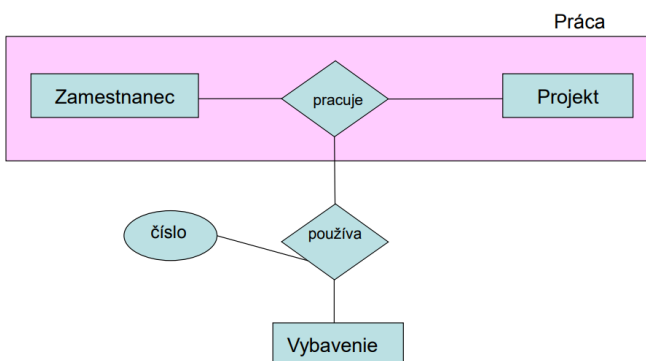
- Pri notácii „cows 'foot“ sa používa terminológia supertyp pre zovšeobecnenie a subtyp pre špecializáciu

## Agregácia

- Jedno z obmedzení ER modelu je, že **nie je schopný znázorniť relácie (vzťahy) medzi reláciami (vzťahmi)**.

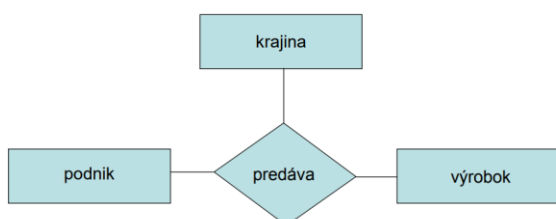


- Potrebujeme vyjadriť vzťah medzi pracuje a používa (nedá sa použiť ternárna relácia, lebo na jednom projekte pracuje veľa zamestnancov, každý používajú veľa vybavenia, to isté vybavenie môže využívať viac zamestnancov vo viac projektoch)
- Riešením je agregácia
- **Agregácia** je abstrakcia, pomocou ktorej sa na relácie môžeme pozeráť ako na entity vyššej úrovne



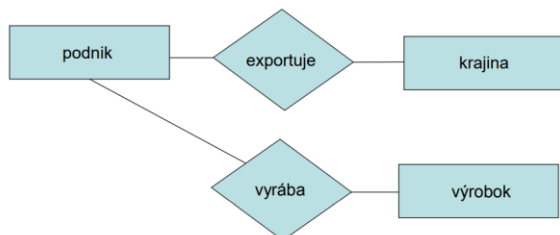
## Ternárna relácia

- Pre daný pár (podnik, výrobok) existuje vo všeobecnosti viacej krajín, kam sa dodáva. Pre daný pár (krajina, výrobok) existuje niekoľko podnikov, ktoré exportujú výrobok. Pre daný pár (podnik, krajina) existuje veľa výrobkov, ktoré podnik exportuje do danej krajiny. (M:N:P)

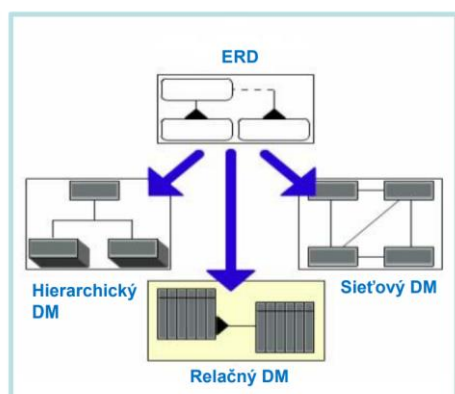




- Pozor pri definíciách ternárnych relácií!!
- **Mali by sa používať iba v prípade, že sa vzťahy nedajú vyjadriť pomocou binárnych relácií.**
- Napríklad: Ak podnik vyrába viac výrobkov a všetky exportuje do rôznych krajín, potom môžeme pôvodnú ternárnu reláciu rozdeliť na dve binárne



## Dátové modely



- **Sieťový model** -> hierarchický model
- **Relačný model** – Transformácia E-R diagramov do relačných schém  
– Normálové formy

## Dátový model (DM)

- Po vytvorení ER modelu (reprezentovaného ER - diagramov) treba určiť spôsob reprezentácie tohto modelu v báze dát.
- Organizácia dát sa navrhuje prostredníctvom dátového modelu.
- **DM definuje množinu pravidiel, podľa ktorých sú organizované logické vzťahy medzi údajmi v databáze.**
- **DM zobrazuje štruktúru dát na úrovni ich typov.**
- DM je prostriedok, pomocou ktorého sú údaje organizované na logickej úrovni.
- DM tvoria pomenované logické jednotky údajov a vyjadruje vzťahy medzi nimi.
- Poznáme niekoľko dátových modelov
- Rozdiel medzi nimi spočíva v spôsobe reprezentácie vzťahov medzi údajmi.
- **Dva druhy vzťahov:** – Vzťah atribútov  
– Vzťah asociácií (medzi entitami)
- **Dva základné modely** – Sieťový model  
– Relačný model

## Sieťový model

- Tvorí ho orientovaný graf, v ktorom sú entity zobrazené pomocou uzlov a asociácie pomocou hrán.

## Relačný model

- Založený na **množinovom pojme relácia** – ako podmnožina karteziánskeho súčinu množiny domén – reprezentácia je vo forme tabuliek.
- Navrhol ho E.F. Codd (1970).

- Ukázal, že **súbor tabuliek alebo relácií sa môže použiť na modelovanie** vzťahov medzi objektami reálneho sveta.
- Forma relácií bola zvolená preto, lebo je pomerne jednoduchá a je schopná ľahko vyjadriť a uchovať komplikované údajové štruktúry.
- Postavený na teoretických základoch (matematická teória relácií) a predikátovej logike, ale pre používateľa je ľahko pochopiteľný, pretože údaje sú vo forme dvojrozmerných tabuliek
- Silné teoretické základy modelu dovoľujú aplikáciu systematických metód, založených na vysoko-úrovňovej abstrakcii, pri návrhu a práci s relačnými databázami

### Definícia relácie

- Nech je daný systém  $\{D_i \mid 1 \leq i \leq n\}$  neprázdnych množín (domén). Potom podmnožinu karteziánskeho súčinu  $R \subseteq D_1 \times D_2 \times \dots \times D_n$  nazývame reláciou stupňa  $n$  ( $n$ -árnou reláciou) nad  $D_1, D_2, \dots, D_n$ . Prvky  $R$  sú usporiadané  $n$ -tice  $\langle d_1, d_2, \dots, d_n \rangle$  také, že  $d_i \in D_i$  pre  $1 \leq i \leq n$

### Relácia

- sa **reprezentuje tabuľkou** (môžeme si ju predstaviť ako maticu s  $m$  riadkami a s  $n$  stĺpcami).
- Každý riadok zodpovedá  $n$ -tici relácie (inštancii entity). Každý stĺpec reprezentuje atribút (entity).
- **Každý atribút má svoje meno**, rôzne od iných, a vždy **sa naň odkazuje pomocou toho mena**, nikdy nie pomocou stĺpca v tabuľke.
- **Poradie stĺpcov je preto nevýznamné**.
- **Každý atribút má doménu**, ktorá pozostáva zo všetkých povolených hodnôt atribútu (význam domény je tu úzko spojený s chápaním pojmu dátového typu programovacieho jazyka).
- Počet stĺpcov kľúčových atribútov tabuľky predstavujúcej reláciu ERM, udáva stupeň tejto relácie.
- **Každý riadok je navzájom odlišný** t.j.: nemôžu byť dva rovnaké riadky (dva rovnaké prvky jednej podmnožiny, dve inštalácie jednej entity).
- **Kardinalita relácie – počet riadkov tabuľky v danom čase**.

<u>Relácia</u>	<u>Súbor</u>	<u>Tabuľka</u>
$n$ -tica	záznam	riadok
názov atribútu	názov poľa	názov stĺpca
názov domény	typ poľa	typ stĺpca

### Zápis relácie

- Názov relácie (názov atribútu 1, ... $n$ )
- kľúčový atribút (primárny kľúč) podčiarknutý.
- Takýto zápis nazveme relačná schéma Študent (meno, priezvisko, rod. číslo, ulica, mesto)

### Cudzí kľúč

- Každá relácia môže obsahovať aj atribút, ktorý je primárnym kľúčom inej relácie.
- Takýto atribút nazývame cudzí kľúč.
- Aby sa zaistila integrita a konzistentnosť bázy dát, každá hodnota atribútu použitá ako cudzí kľúč, musí mať svojho dvojníka v inej relácii ako primárny kľúč.
- Relácie sa navzájom prepoja podľa rovnakých hodnôt týchto atribútov.
- Pri výbere informácií z bázy dát sa použije hodnota cudzieho kľúča na lokalizáciu relevantných informácií z druhej relácie.
- Používa sa spájanie podľa hodnôt a nie na základe smerníkov. Toto zabezpečuje nezávislosť relačného modelu od uvažovania prístupu a fyzického uloženia (implementačné detaily)

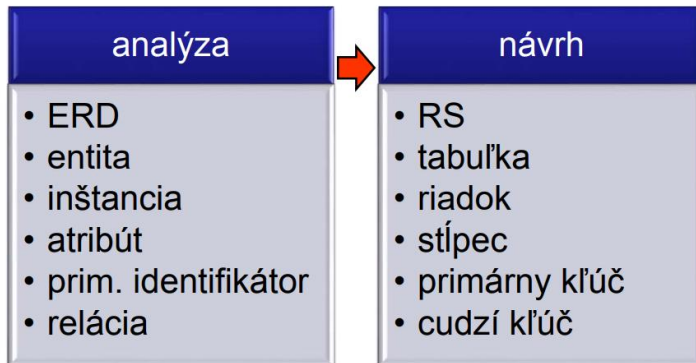
Kniha (ISBN, Autor, EV\_C)

Čitateľ (RC, Meno,....)

- Čitateľ si môže požičať naraz viac kníh a jedna kniha môže byť požičaná len u jedného čitateľa
- Cudzí kľúč vložíme tam, kde je z relácie 1:N obmedzenie na množstvo -> 1kniha 1čitateľ Kniha (ISBN, Autor, EV\_C, RC, ... ) Čitateľ (RC, Meno,....)

### Transformácia ER diagramov do relačných schém

- Konceptuálny model – ER diagram – Entity a relácie
- Dátový model – relačné schémy
- z KM → DM Mapovanie terminológie



### Reprezentácia entít

- Pre každú množinu entít vytvoríme jednu tabuľku, v ktorej počet stĺpcov zodpovedá počtu atribútov
- Názov tabuľky → názov množiny entít
- Názov stĺpca → názov atribútov entity

### Silné množiny entity

- Primárny kľúč v relačnej schéme zodpovedá primárnemu kľúču entity
- Tabuľka môže obsahovať viac stĺpcov ako je atribútov entity → cudzie kľúče

### Slabé množiny entít

- Pre každý atribút slabej entity musí tabuľka obsahovať stĺpec pre každý vlastný atribút a stĺpce pre atribúty, ktoré tvoria primárny kľúč silnej entity, na ktorej je slabá entita existenčne závislá

### Reprezentácia množín relačných vzťahov

- Technika transformácie relačných vzťahov závisí od **funkcionality** týchto vzťahov a od počtu entít participujúcich na týchto vzťahoch

### Povinný člen vzťahu

- Ak je entita E2 **povinným členom** vzťahu 1:N s entitou E1 , potom relačná schéma E2 obsahuje kľúčové atribúty E1 ako cudzí kľúč.

### Nepovinný člen vzťahu

- Ak je E2 **nepovinným členom** vzťahu N:1 s entitou E1 , potom je vzťah zvyčajne reprezentovaný ďalšou relačnou schémou obsahujúcou kľúčové atribúty E1 , E2 a atribúty tohto vzťah

### Reprezentácia binárnych vzťahov M:N

- Takéto vzťahy sú vždy reprezentované samostatnou relačnou schémou (tabuľkou) obsahujúcou Primárny kľúč E1 , primárny kľúč E2 , a vlastné atribúty relačného vzťahu.

## Reprezentácia relačných vzťahov tej istej množiny entít

- 1:1 • Osoba (R.č.#, Meno, adresa, ....)
- Manželstvo(mužR.č.#, ženaR.č.#, dátum svadby)
- príklad na nepovinné členstvo -> samostatná tabuľka
- každý má len jedného partnera -> ako primárny kľúč stačí jedno R.č.
- ak chceme uchovávať aj historické údaje - > obe R.č. tvoria PK
- 1:N • Zamestnanec(Rč.#, vedúciRč.#,meno,..)
- Zamestnanec(Rč.#, meno,..)
- Riadi(Rč.#, vedúciRč.#, ...)
- M:N
- Súčiastka (č#, názov, opis, ...)
- Montuje\_sa (zostava#, súčiastka#, počet)

## Reprezentácia ternárnych relačných vzťahov

- samostatná relačná schéma
- PK závisí od typu vzťahov
- Ak M:N:P -> všetky tri atribúty tvoria PK

## Reprezentácia zovšeobecnenia a špecializácie

- A) – dve relačné schémy
- B) – jedna spoločná relačná schéma

## Reprezentácia agregácie

- relačné schémy pre entity a vzťahy agregovanej entity
- relačná schéma relačného vzťahu obsahuje všetky kľúčové atribúty entít, relačného vzťahu a vlastné atribúty

## Integritné pravidlá (obmedzenia)

Typ obmedzenia	Vysvetlenie	Príklad
Entitná integrita	Primárny kľúč musí byť jedinečný a žiadna jeho časť nesmie byť prázdna („null“)	#ID v tabuľke ZAMESTNANCI nesmie byť prázdne
Referenčná integrita	Hodnota cudzieho kľúča musí byť alebo rovná hodnote primárneho kľúča entity, na ktorú odkazuje, alebo musí byť prázdna	Hodnota #odd v tabuľke ZAMESTNANCI a tabuľke ODDELENIE musí byť rovnaká
Stĺpcová (doménová) integrita	Stĺpec môže obsahovať len hodnoty v súlade s definovaným typom dát pre stĺpec (doménou)	Hodnota v stĺpci stav_na_ucte musí byť numerického typu
Užívateľom definovaná integrita	Údaje uložené v DB musia spĺňať užívateľom predpísané obmedzenia	Pri prekročení debetného limitu na účte sa pošle oznam užívateľovi (rieši sa to programovaním)

## Normalizácia relácií

- **Normalizácia** (normalization) je proces úpravy a korekcie štruktúry tabuliek v databáze tak, aby sa minimalizovala dátová redundancia a dátové anomálie.
- **Hlavné zásady:** – atribúty **závislé len na celom PK** -> ak nie treba ich rozdeliť  
– ak možno dekomponovať reláciu bezstratovo – treba to urobiť Normalizácia relácií
- Formálne je normalizácia založená na funkčných závislostiach.
- Teória normalizácie je formalizácia princípov správneho návrhu štruktúry DB.

## Funkčné závislosti

- Pre danú reláciu R, atribút B z R je funkčne závislý na atribúte A z R ( $A \rightarrow B$ ), ak v dvoch n-ticiach sa rovnajú hodnoty A, potom sa rovnajú aj hodnoty B. V danom časovom okamžiku každá hodnota A má priradenú práve jednu hodnotu B. A aj B môžu byť zložené atribúty.

## Funkčné závislosti – Pr.

- Hodnotenie (štud#, kurz#, názov, menouč, miestnosť#, známky)
- Študent dostal z predmetu (kurz) s názvom a učil ho učiteľ v miestnosti #. Predpokladajme, že každý kurz učí len jeden učiteľ a každý učiteľ má pridelenú práve jednu miestnosť.
- Potom pre daný pár štud#, kurz# existuje len jedna hodnota známky štud#, kurz#  $\rightarrow$  známky
- pre danú hodnotu kurz# existuje práve jedna hodnota kurz#  $\rightarrow$  názov kurz#  $\rightarrow$  menouč kurz#  $\rightarrow$  miestnosť#
- Pre danú hodnotu menouč existuje menouč  $\rightarrow$  miestnosť#

## Funkčné závislosti

- Atribút známky je plne funkčne závislý na PK, pretože je závislý na dvoch atribútoch tvoriacich kľúč a nie je závislý na nich oddelene
- Atribút B je plne funkčne závislý na A, ak je funkčne závislý na A a nie je funkčne závislý na žiadnej podmnožine A.
- Atribúty názov, menouč, miestnosť# sú čiastočne závislé na PK, pretože sú závislé len od kurz# a nie aj od štud#.
- Atribút miestnosť# je tranzitívne závislý na kurz#, pretože je závislý na menouč, ktoré je závislé na kurz#.
- Čiastočná a tranzitívna závislosť, ak sa vyskytuje v relačných schémach spôsobuje problémy pri práci s DB, a preto sa musia odstrániť pred implementáciou.

## Anomália vkladania

- Ak potrebujeme vložiť do DB údaje o novom kurze, nemôžeme tak spraviť pokiaľ sa doň neprihlási aspoň jeden študent (to isté platí pre učiteľa a číslo miestnosti  $\rightarrow$  pokiaľ nie sú priradení kurzu a neprihlási sa študent)
- Ide o anomáliu vkladania

## Anomália aktualizácie

- Ak chceme zmeniť názov kurzu č. 361 z Databázovej technológie na DBS, potom musíme vyhľadať všetky n-tice (riadky), ktoré obsahujú kurz#=361 a aktualizovať ich toľkokrát, koľko je študentov zapísaných na kurz DBS.

## Anomália mazania

- Ak sa všetci zapísaní študenti do kurzu (DBS) rozhodnú nedokončiť ho, potom všetky informácie o tomto kurze sa nenávratne stratia z DB.

## Prvá normálová forma (1.NF)

- Relácia je v 1.NF ak domény všetkých atribútov obsahujú len atomické (ďalej nedeliteľné) hodnoty.
- Znamená to, že žiaden atribút nemôže byť reláciou.

## Prvá normálová forma (1.NF)

- Hodnotenie (štud#, kurz#, názov, menouč, miestnosť#, známky)
- atribút známky je zložený
- známka (1.test, 2.test, výsledná\_zn)

## Druhá normálová forma (2.NF)

- Relácia je v 2.NF ak je v 1.NF a každý neklúčový atribút je plne funkčne závislý na PK. Druhá normálová forma (2.NF)
- Hodnotenie (štud#, kurz#, názov, menouč, miestnosť#, známka)
- musíme rozdeliť tak, aby v každej relácii všetky neklúčové atribúty boli plne funkčne závislé na PK
- Hodnotenie (štud#, kurz#, známka) • Kurz (kurz#, názov, menouč, miestnosť#)

### Tretia normálová forma (3.NF)

- Relácia je v 3.NF ak je v 2.NF a každý neklúčový atribút je netranzitívne závislý na PK. Tretia normálová forma (3.NF) • Hodnotenie (štud#, kurz#, známka)
- Kurz (kurz#, názov, menouč, miestnosť#)
- treba dekomponovať
- Hodnotenie (štud#, kurz#, známka)
- Kurz (kurz#, názov, menouč)
- Učiteľ (menouč, miestnosť#)
- Sú už plne normalizované

### Dekompozícia

- Výsledné relácie sme dostali dokompozíciou pôvodnej a pritom sme nestratili žiadne informácie -> hovoríme o bezstratovej dekompozícii.

### Heath-ova teorema

- Relácia  $R(A, B, C)$ , v ktorej existuje funkčná závislosť  $A \rightarrow B$  môže byť bezstratovo dekomponovaná do jej projekcií  $R_1(A, B)$  a  $R_2(A, C)$
- Žiadne informácie sa nestratia, lebo pôvodná relácia sa dá rekonštruovať prirodzeným spojením jej projekcií.

### Boyce-Coddova normálna forma (BCNF)

- Pôvodný návrh E. F. Codd obsahoval tri normálové formy, ktoré nazývame prvá, druhá a tretia normálová forma (1NF, 2NF, 3NF). Neskôr bola navrhnutá prísnejšia definícia 3NF, nazývaná Boyce-Coddova normálna forma (BCNF).
- 3NF v prípade relácií, ktoré majú viac kandidátov na kľúč a tieto sú zložené a prekrývajú sa (najmenej jeden atribút je spoločný) je nepostačujúca
- Determinant je atribút alebo skupina atribútov, na ktorých sú ostatné atribúty plne funkčne závislé.
- Relácia  $R$  je v BCNF vtedy a len vtedy, ak každý determinant je kandidátom na kľúč. Boyce-Coddova normálna forma (BCNF)
- Uvažujeme fakultu, v ktorej každý predmet môže určiť niekoľko učiteľov, ale každý učiteľ učí len jeden predmet. Každý študent má zapísaných niekoľko predmetov a má len jedného učiteľa pre každý predmet. Boyce-Coddova normálna forma (BCNF)
- Zapísaný (št#, menouč, názov predmetu)
- Nie je v 2NF menouč -> názov predmetu
- Zapísaný (št#, názov predmetu, menouč)
- Je v 3NF, ale nie v BCNF
- Menouč -> názov predmetu a menouč nie je kandidátom na kľúč
- Takto nemôžeme vložiť fakt, že niektorý učiteľ učí daný predmet, pokiaľ si ho nezapíše aspoň 1 študent
- Trieda (št#, menouč)
- Učí (menouč, názov predmetu)
- Je v BCNF

### Ďalšie normálové formy

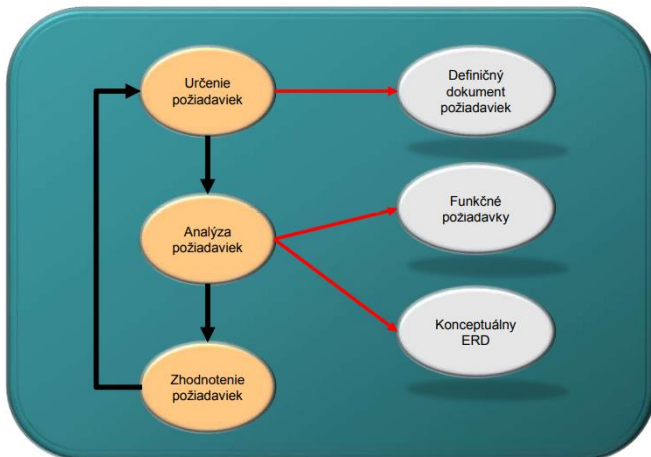
- Štvrtá NF – 4NF
- Piata NF – 5NF
- riešia niektoré špeciálne prípady viachodnotových závislostí
- Pre praktické použitie spravidla nie sú potrebné a uspokojíme sa s normalizáciou do 3N

## Metodika návrhu relačných databáz

### Postup

- Pri návrhu postupujeme v 4 krokoch: 1) Získanie a analýza požiadaviek
  - 2) Tvorba konceptuálneho modelu (ERD)
  - 3) Tvorba logického modelu - transformácia ERD do relačných schém (relačného modelu) a normalizácia relácií
  - 4) Špecifikácia fyzickej konfigurácie databázy – fyzický model
  - 5) Implementácia návrhu

### Získanie a analýza požiadaviek



### Proces hodnotenia požiadaviek

- Po analýze a zdokumentovaní požiadaviek, je potrebné výsledky analýzy vyhodnotiť.
- Tento trojstupňový proces sa opakuje v iteráciách.
- V prípade, že analýza je nesprávna, je nutné zopakovať celý proces, kým sa nepodarí definovať schodné riešenie.

### Dokumentácia požiadaviek

- existujúce **užívateľské požiadavky**
  - ◇ požiadavky na navrhovaný systém
  - ◇ funkčné požiadavky: – procesné modely  
– diagramy tokov dát (DTD).
- Konceptuálny dátový model

### Zhodnotenie požiadaviek

- Aby sa zistilo, či boli splnené požiadavky a či navrhnutá databáza bude úplná a funkčná, je užitočné vykonať krížovú kontrolu procesných modelov a DFD s ERD.
- V prípade zistenia nezrovnalostí medzi procesnými modelmi, DFD a ERD, je nutné ERD upraviť.

## POŽIADAVKY – ZÍSKANIE A ANALÝZA

### Identifikovanie požiadaviek

- Medzi **základné prvky**, ktoré je potrebné určiť, možno zahrnúť:
  - biznis pravidlá;
  - údaje;
  - procesy, ktoré definujú biznis.

### Identifikovanie požiadaviek

- Požiadavky sa dajú určiť zhromažďovaním informácií od užívateľov o ich úlohách.
- Zbierajú sa informácie o tom: – **ako sa v rámci organizácie menia údaje;**
  - ako sú zdieľané údaje;
  - aké faktory ovplyvňujú pracovné (biznis) procesy.

## Zber informácií

- Informácie o práci a úlohách užívateľov je možné zhromažďovať prostredníctvom:
  - rozhovorov (interview),
  - dotazníkmi,
  - brainstormingom,
  - pozorovaním užívateľov pri práci,
  - štúdiom firemných dokumentov,

predpisov a pod.

- Zberom informácií sa dá zistiť:
  - s akými informáciami pracuje každé oddelenie;
  - aké typy procesov sa používajú pre splnenie úloh;
  - akými biznis pravidlami sa riadia oddelenia (útvary organizácie).

## Biznis pravidlá

- Potom, čo sú identifikované informácie a s nimi súvisiace procesy, je potrebné pochopiť a zdokumentovať s nimi súvisiace biznis pravidlá.
- Biznis pravidlá často predurčujú referenčnú integritu v databáze.

## Analýza požiadaviek

- Následne je potrebné analyzovať požiadavky, aby sa identifikovali tie kritické procesy, ktoré je možné automatizovať alebo zjednodušiť.

## Tvorba konceptuálneho modelu (ERD)

### Identifikovanie a analýza požiadaviek na dáta

- Identifikovať požiadavky na údaje, popis informácií a dátových objektov a ich vzťahov.
- Spojiť používateľské pohľady do globálneho pohľadu (konceptuálnej schémy), eliminovať redundanciu, nekonzistentnosť.**
- Treba rozpoznať synonymá (rôzna forma ten istý význam) a homonymá (tá istá forma rôzny význam).

### Konceptuálny model

- Konstruujeme ER model, ktorý popisuje dátové objekty a ich vzťahy.
- Identifikujeme kľúče (atribúty), funkcionality relačných vzťahov, typ členstva vo vzťahoch (povinné, nepovinné).
- Výsledkom je ER model = diagram, ktorý dáva celkový pohľad na bázu dát.

### Vyhodnotenie požiadaviek

- Počas tejto fázy je **potrebné zistiť**:
  - či boli zahrnuté všetky informácie;
  - či boli získané informácie interpretované správne.
- Je potrebné taktiež skontrolovať všetky zhromaždené a analyzované informácie, či nie sú vo vzájomnom konflikte.
- Ak by nastal takýto prípad, je potrebné revidovať proces analýzy požiadaviek.

## TVORBA LOGICKÉHO MODELU

### Transformácia ER modelu do relačných schém

- Vykonáme transformáciu ER diagramu do relačných schém.
- Pozor musíme dávať na unárnu reláciu, zovšeobecnenie a špecializáciu, agregáciu a ternárne relácie

### Normalizácia relačných schém

- Pre každú reláciu odvodíme zoznam funkčných závislostí a multizávislostí.
- Redukujeme každú reláciu do požadovaného stupňa



## FYZICKÝ MODEL A JEHO IMPLEMENTÁCIA

### Transformácia LDM -> FDM

1. Definícia tabuliek a stĺpcov;
2. Vytvorenie pohľadov (view) a indexov;
3. Rozdeľovanie a zhlukovanie (Partitioning and clustering)
4. Denormalizácia tabuliek (Denormalizing tables)

### Úlohy transformácie LDM - FDM

- Pred vytvorením fyzického modelu, je nutné znovu preskúmať logický model a identifikovať potrebné dátové prvky.
- Pri fyzickom vytváraní tabuliek je možné, v závislosti na zvolenom SRBD, použiť nástroj, aby sa automatizovala implementácia, alebo použiť príkazy Data Definition Language (DDL), pri manuálnom vytváraní modelu.
- overiť entity a atribúty a ich mapovanie do tabuliek a stĺpcov tabuľky;
- overiť primárne a cudzie kľúče v tabuľkách pre ukladanie a prístup k údajom;
- overiť vzťahy medzi tabuľkami a ich obmedzenia;
- na vytvorenie databázy použiť nástroj (database designer) alebo DDL príkazy;
- definovať pohľady a indexy

### Fyzická implementácia

- Počas tohto procesu, je potrebné rozhodnúť aké množstvo úložného priestoru a typ úložiska použiť, vychádzajúc z tabuliek, indexov, triggerov a ďalších procedúr, ktoré sa budú vytvárať.
- Pri vytváraní novej databázy je možné vytvoriť núdzové postupy, slúžiace potom k náprave chýb implementácie.
- zvýšenie bezpečnosti databázy sa dosiahne návrhom bezpečnostnej politiky.

### Tabuľky

- Existujú tri typy tabuliek používané na usporiadanie informácií v databáze:
  - **tabuľky údajov (data tables);**
  - **tabuľky prepojení (join tables);**
  - **vyhľadávacie tabuľky (lookup tables).**

### Údajové tabuľky

- Sú tabuľky pre ukladanie informácií o jedinej triede alebo entite so spoločnými atribútmi.
- Údaje v dátových tabuľkách nie sú získané alebo modifikované pomocou výpočtov, alebo reťazením.
- Každá tabuľka má primárny kľúč, ktorý jednoznačne reprezentuje uložené informácie.
- Napríklad informácie o zamestnancoch organizácie sú v údajovej tabuľke.
- Tabuľka Zamestnanci sa skladá zo stĺpcov, ako sú Zamestnanec\_ID, Zamestnanec\_Meno, a Oddelenie\_No

### Tabuľky prepojení

- Vznikajú spojením atribútov (primárnych kľúčov) dvoch alebo viacerých tabuliek do samostatnej tabuľky;
- používajú sa na reprezentáciu vzťahov M:N medzi dvoma entitami (reprezentovanými tabuľkami), prípadne viacerých entít;

### Vyhľadávacie tabuľky

- tiež známe ako overovacie alebo kódové tabuľky;
- obsahujú zoznam statických hodnôt, ktoré sú uložené v stĺpcoch iných tabuliek;
- Napríklad vytvorením vyhľadávacej tabuľky PSC, obsahujúcej názvy obcí a PSC, je možné overiť údaje v tabuľke obsahujúcej adresy: overiť názov obce alebo PSC

## Špecifikácia stĺpcov

- **Všeobecná úroveň:** – je potrebné, aby stĺpce zodpovedali súvisiacim tabuľkám, pretože používatelia potrebujú prístup k stĺpcom prostredníctvom súvisiacich tabuliek: Zamestnanec.Meno
- **Logická úroveň** – určiť, či sa stĺpec používa ako primárny alebo cudzí kľúč, či môže obsahovať NULL alebo iné obmedzenia.
- **Fyzická úroveň:** – Definícia typu dát ako napr.: znaky, numerické hodnoty, dátum, čas - umožňujú určiť potrebu alokácie priestoru pre fyzické ukladanie údajov
  - Napr.: stĺpec Zamestnanec\_ID má definovaný ako dátový typ NUMBER (8)
  - Každý SRBD používa vlastné špecifické dátové typy

## Vytváranie pohľadov a indexov

- Pohľady a indexy poskytujú alternatívny spôsob ukladania a vyberania údajov z databázy. Pomáhajú tiež zrýchliť prístup k údajom.
- Pohľady sú virtuálne tabuľky, ktoré neobsahujú vlastné údaje, ale obsahujú výsledky realizácie príkazov SELECT pre skutočné tabuľky.

### Pohľady

- Pohľady vyžadujú menej miesta v databáze, pretože sa ukladá len definícia pohľadu.
- Pohľad:
  - bráni neoprávneným zmenám tabuľky s údajmi;
  - umožňuje užívateľom zobraziť len tie stĺpce podkladovej tabuľky, pre ktoré majú oprávnenie;
  - vo virtuálnych stĺpcoch môže obsahovať dynamické hodnoty, ako sú napríklad vypočítané hodnoty alebo zreťazené súčty;
  - umožňuje generovať správy (reporty) z rôznych uhlov pohľadu na údaje.

### Indexovanie

- indexy sa navrhujú počas fyzického návrhu databázy.
- Poskytujú ďalší spôsob prístupu k záznamom, bez ohľadu na ich umiestnenie na disku.
- Napríklad, ak je potrebné pristupovať k údajom z tabuľky Zamestnanci na základe primárneho kľúča, je vhodné vytvoriť index pre primárny kľúč.
- Index sa skladá z **ukazovateľa** a hodnoty **primárneho kľúča**.
- V prípade dotazu, pristupuje sa najskôr na index. Tento ukazuje priamo na fyzickú adresu záznamov, ktoré zodpovedajú požadovanej hodnote primárnych kľúčov.
- Ak by index nebol k dispozícii, musí sa prehľadať celá tabuľka na zhodu hodnôt primárnych kľúčov s vyhľadávanou hodnotou.
- Toto je časovo náročné v prípade, ak je tabuľka veľká.

### Typy indexov

- **hlavné typy indexov:** – (B+tree) B+strom
  - (Clustered) zhukový
  - (Hash) hašovací
  - (Bitmap) bitmapový

### B+strom

- Používa sa na **prístup k údajom v hierarchickej štruktúre**;
- Načítanie dát je rýchlejšie, pretože dáta sú usporiadané sekvenčne.
- Strom obsahuje viacero uzlov s koreňovým uzlom, ktorý sa vetví do prvej a druhej úrovne uzlov.
- Každý uzol obsahuje primárny kľúč riadku a ukazovateľ na fyzickú adresu riadku tabuľky.
- B + Tree indexy sú užitočné pre tabuľky s viacerými riadkami;

- môže byť buď **výlučný** (unique), alebo **nevýlučný** (nonunique).
- **Výlučný index** obsahuje jedinečnú hodnotu kľúča pre každý riadok. Kľúčové hodnoty sú zoradené ako hodnoty ASCII.
- **Nonunique indexy** používajú kľúče, ktoré majú rovnakú hodnotu vo viacerých záznamoch.

### Zhlukový index

- **Organizuje záznamy s podobnými hodnotami** atribútov blízko seba.
- Klastrové indexovanie zvyšuje rýchlosť načítania dát, pretože sa prehľadávajú tabuľky, ktoré sú súčasťou klastra.
- Pre tabuľku môže existovať **len jeden klastrový index** - toto napomáha k fixovaniu fyzického usporiadania tabuľky.

### Hašovací index

- je to účinný spôsob, ako **organizovať neusporiadané záznamy v databáze**.
- Hašovacie indexovanie používa **unikátny kľúč** na zoskupenie súvisiacich hodnôt, na základe zvoleného kritéria.
- Hodnota kľúča je vstupom matematického výpočtu, zvaného hašovacia funkcia, ktorý určí fyzickú adresu.

### Bitmapový index

- Bitmapový index sa používa **na usporiadanie záznamov s viacnásobnými hodnotami atribútov**.
- Kľúčové hodnoty sú tu reprezentované ako **bitové mapy**.
- Každý vektor v bitovej mape odkazuje na hodnotu atribútu.
- Bitový vektor má hodnotu 0, ak konkrétny záznam neobsahuje danú hodnotu a 1, ak áno.
- Bitmapové indexy sú užitočné, keď počet možných hodnôt atribútov je malý.

### Rozdeľovanie a zhlukovanie

- Dve stratégie, ktoré sa používajú s cieľom zlepšiť výkon databázy sú: – **rozdeľovanie** (fragmentácia) tabuliek – **zhlukovanie**.

### Rozdeľovanie

- Je možné deliť databázu rozdelením **veľkých tabuliek do menších tabuliek**, alebo **uložiť údaje z tabuľky na viacero samostatných diskov**.
- **Fyzické rozdelenie** umožňuje obmedziť zaťaženie konkrétneho hardvéru, ako napr.: procesora alebo pevného disku.
- To umožní využiť hardvérové prostriedky efektívne.

### Rozdeľovaním sa zlepši:

- **Výkonnosť (Performance)**
- **Udržiavateľnosť (Maintainability)**
- **Dostupnosť (Availability)**

### Rozdeľovanie - Výkonnosť

- Pre zvýšenie výkonu, je možné rozdeliť veľké tabuľky do malých tabuliek.
- Realizácia dotazu trvá kratšie pretože sa prehľadávajú menšie tabuľky.
- Použitie oddelených diskov zvyšuje výkon realizácie dotazov, ktoré obsahujú spojenia (join), pretože všetky disky sa môžu prehľadávať v rovnakom čase (súbežne).

### Rozdeľovanie - Udržiavateľnosť

- **Rozdeľovanie** zlepšuje **udržiavateľnosť** databázy, pretože je ľahšie robiť analýzu dát v menších tabuľkách.
- Tiež to uľahčuje údržbu, napríklad vytváranie zálohy alebo vytváranie indexov je jednoduchšie.

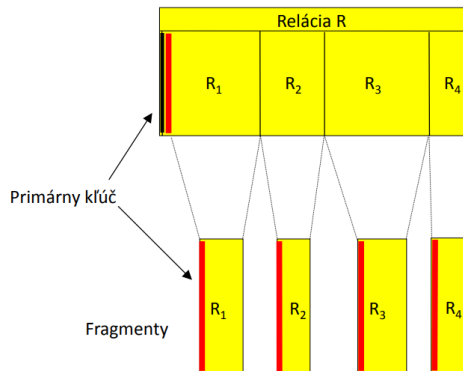
### Rozdeľovanie - Dostupnosť

- Rozdeľovanie zvyšuje dostupnosť údajov, pretože sa oddelia často používané údaje od statických dát, pričom tieto sa uložia v rôznych samostatných tabuľkách.

## Rozdeľovanie tabuliek

- **Horizontálne** – Výber riadkov v určitom rozsahu do oddelených tabuliek na základe požiadaviek analýzy dát.
- **Vertikálne** – Výber niektorých stĺpcov do oddelených tabuliek na základe požiadaviek analýzy dát  
– Na základe normalizácie: umožňuje oddeliť najmenej používané stĺpce do samostatných tabuliek a spojiť ich s hlavnou tabuľkou pomocou primárnych a cudzích kľúčov.
- **Vertikálne delenie** pomáha skrátiť dobu realizácie dotazu, ale môže mať vplyv na výkon v prípade, keď sa použije viacero spojení (join), alebo sa vyžaduje veľká fragmentácia.

## Vertikálna fragmentácia relácie R na fragmenty R1, R2, R3, R4



## Zhlukovanie

- Po vytvorení databázy, ďalšou úlohou je zabezpečiť, že databáza beží bez prerušenia.
- Na tento účel je vhodné použiť zhlukovanie.
- Zhuk (Klaster) je súbor počítačov spojených dohromady pre súbežné spracovanie úloh.
- Počítače v klastri fungujú ako jeden počítač a tak pomáhajú zlepšiť výkon databázy.
- HA klastre sú počítače alebo uzly, ktoré sú zoskupené tak, aby fungovali serverové aplikácie bez prerušenia.
- HA klastre pomáhajú obmedziť výpadky databázy: zistením poruchy na jednom počítači v klastri sa reštartuje transakcia na inom počítači klastra.
- Tento proces sa nazýva zálohovanie (failover)

## Denormalizácia tabuliek

- Nie vždy je nutné normalizovať databázu, aby boli odstránené redundantné údaje.
- **Normalizácia** zvyšuje dobu odozvy dotazu vo veľkých databázach, pretože dopyt pre získanie výsledku musí prehľadávať viacero tabuliek a pohľadov.
- Rovnaký problém nastáva pri aktualizácii údajov.

## Denormalizácia tabuliek

- Zahŕňa **zámerné pridanie redundantných dát alebo iné zoskupovanie normalizovaných tabuliek tak, aby sa znížila doba odozvy na dotaz.**
- Denormalizácia však **odstráni výhody tretej normálovej formy (3NF).**
- Denormalizácia **závisí na tom, ako sú subjekty namapované v ERD.**
- Je to vytvorenie alternatívnej logickej štruktúry, ktorá umožní načítať dáta rýchlejšie.
- Existujú dva typy denormalizácie pre vzťahy, ktoré zahŕňajú dve entity: – použitie jedinej tabuľky pre obe entity  
– denormalizácia jednej tabuľky

## Jediná tabuľka pre obe entity

- ak sú dve entity mapované navzájom vo vzťahu s kardinalitou 1:1 alebo 1:N, je možné použiť jednu tabuľku pre reprezentáciu oboch entít.
- To skráti dobu odozvy dotazu tým, že sa nemusí realizovať spojenie.

## Denormalizácia jednej tabuľky

- Ak sú entity mapované vo vzťahu s kardinalitou 1:N, je možné jednu tabuľku denormalizovať a druhú udržiavať ako nadbytočnú.
- Táto metóda zvyšuje nárok na úložný priestor potrebný pre redundantné tabuľky a pri aktualizácii údajov sa musia aktualizovať obe tabuľky: základná aj denormalizovaná tabuľka.

## Dôsledky denormalizácie

- skrátenie doby odozvy na dotazy;
- zvýšený nárok na úložný priestor;
- predĺženie doby aktualizácie dát;
- reštrukturalizované tabuľky;
- aktualizáciu údajov v dôsledku zmien dotazu;
- strata dát v dôsledku mazania.

Pri implementácii databázy je potrebné zvážiť výhody a nevýhody denormalizácie!!!

## Príklad: Databáza nemocnice

- Navrhnuť štruktúru bázy dát pre nemocnicu na pomoc pri správe nemocničných izieb a operačných sál, udržiavanie informácií o pacientoch, lekároch a sestrách

### Krok 1: Analýza požiadaviek

- Pacienti obývajú každú izbu. Na izbe môže byť viac pacientov. Primári môžu mať súkromných pacientov a títo sú v súkromných izbách. (pacient – č. zdrav. poistky, meno, adresa..)
- Zdravotné sestry majú na starosť každú izbu, sestra ale nemusí mať na starosť izbu, sestra môže mať na starosť najviac jednu izbu, izbu môže mať na starosť viac sestier. (sestra – os. Číslo, meno, kvalifikáciu..., izba – č. izby, počet postelí)
- Na pacientovi sa vykonávajú operácie, pacient môže podstúpiť viacero operácií (operácia – typ, lekár, dátum, čas, miestnosť...)
- Len jeden lekár operuje, ostatní asistujú. Lekárov riadi primár. Primár môže operovať, alebo asistovať. Každý primár je špecialista – má odbornosť. (lekár – os. Číslo, meno, adresa, tel. ....)
- Daná operácia sa vykonáva len v jednej miestnosti, ale jedna miestnosť slúži na vykonanie viacerých operácií. Každá oper. sála má identifikačné číslo, niektoré sú špeciálne vybavené pre istý druh operácie.
- Sestra môže, ale nemusí byť pridelená na operačnú sálu, nemôže mať službu na viac ako v jednej sále. Na jednu sálu môže byť pridelených viacero sestier.

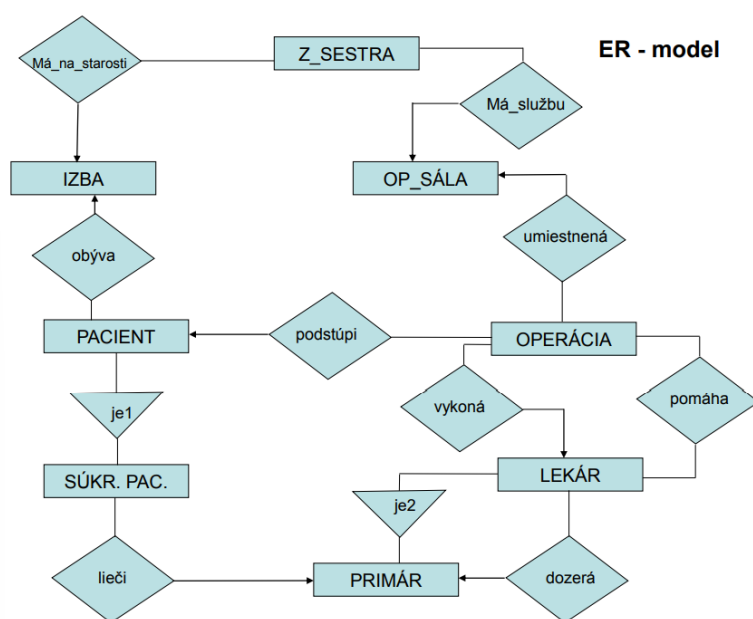
### Krok 2: Entito – relačné modelovanie

- LEKÁR (C\_L, Lmeno, Ladresa, ....)
- PRIMÁR – špecializácia entity LEKÁR (špecializácia)
- PACIENT (C\_P, Pmeno, Padresa, dat\_narodenia, pohlavie,...)
- SÚKROMNÝ PACIENT – špecializácia entity PACIENT (c\_izby)
- Z\_SESTRA (S#, Smeno, stupen,...) • IZBA (c\_izby, typ, pocet\_posteli,...)
- OPER\_SÁLA(c\_saly, typ\_saly,...)
- OPERÁCIA (operacia #, typ\_operacie, datum, cas,...)

## Relačné vzťahy

Vykoná	1:N	lekár a operácia	povinné členstvo
Pomáha	N:M	lekár a operácia	povinné členstvo
Dozerá	1:N	primár a lekár	nepovinné členstvo
Lieči	1:N	primár a súkromný p.	povinné členstvo
Podstúpi	1:N	pacient a operácia	povinné členstvo
Obýva	1:N	izba a pacient	povinné členstvo
Umiestnená	1:N	oper_sála a operácia	povinné členstvo
Má_na_starost'	1:N	izba a z_sestra	nepovinné členstvo
Má_službu	1:N	oper_sála a z_sestra	nepovinné členstvo

- Špecializácia – LEKÁR → PRIMÁR
- PACIENT → SÚKROMNÝ PACIENT



## Krok 3: Transformácia do relácií

- LEKÁR (C\_L, Lmeno, Ladresa, C\_Tel)
- PRIMÁR – (C\_L, špecializácia)
- PACIENT (C\_P, c\_izby, Pmeno, Padresa, dat\_narodenia, pohlavie)
- SÚKROMNÝ PACIENT – (C\_P, Lmeno, c\_izby)
- Z\_SESTRA (S#, Smeno, stupen)
- IZBA (c\_izby, typ, pocet\_posteli)
- OPER\_SÁLA(c\_saly, typ\_saly)
- OPERÁCIA (operacia #, c\_saly, C\_L, C\_P, typ\_operacie, datum, cas)
- Má\_na\_starosti - relácia je realizovaná pomocou novej relačnej schémy Má\_na\_starosti (S#, c\_izby, dátum\_odkedy)
- Má\_službu - relácia je realizovaná pomocou novej relačnej schémy Má\_službu (S#, c\_saly, dátum\_služby)
- Dozerá - relácia je realizovaná pomocou novej relačnej schémy Dozerá(C\_L, primár\_meno)
- Pomáha - relácia je realizovaná pomocou novej relačnej schémy Pomáha (c\_saly, C\_L, úloha)

## Relačné jazyky

### Úvod do relačných jazykov

- Väčšina relačných jazykov je založená na: – **relačnej algebre** – aplikácia operátorov na relácie => procedurálny charakter

– **relačnom kalkule** – predikátový kalkul, dopyt je vyjadrený špecifikovaním predikátu, ktorý musí splniť n-tica (riadok), aby bol vybraný => neprocedurálny charakter

- jazyky potrebujeme, aby sme dokázali ukladať a získať informácie z relácií v DB
- Algebra a kalkul poskytujú len možnosti na získanie uložených informácií, jazyky pre praktické použitie musia mať schopnosť aktualizovať DB, používať agregračné funkcie, formátovať výstup a pod.

### Relačná algebra (RA)

- Navrhnutá Codd 1970
- využíva funkcie a vyhodnocovanie algebraických výrazov
- Každý operátor RA sa aplikuje na jednu alebo dve relácie ako operandy a vytvára ako výsledok novú reláciu.
- RA nebola v čistej podobe aplikovaná ako dotazovací jazyk, ale tvorí základ DB jazykov

### Čo je relačná algebra ?

- Relačná algebra **reprezentuje manipulačnú súčasť relačného modelu.**
- Na reprezentáciu relácií a manipuláciu s dátami relačná algebra obsahuje 8 operátorov :
  - Product (karteziánsky súčin)
  - Union (množinové zjednotenie)
  - Difference (množinový rozdiel)
  - Intersection (množinový prienik)
  - Divide (podiel)
  - Selection (selekcia)
  - Projection (projekcia)
  - Join (spojenie)
- Tieto operácie pracujú s reláciami a ich výsledkom je taktiež relácia.
- Prvých **5 operácií sú základné operácie**, ostatné môžu byť definované pomocou nich.
- **Relačné operácie môžu byť kombinované a navzájom vnorené.**
- Relačná algebra je **vyššou symbolickou reprezentáciou.**
- Znalosť relačnej algebry vedie k lepšiemu pochopeniu toho, čo vykoná SRBD, keď dostane SQL požiadavku.
- Relačná algebra vysvetľuje, ako sú **získavané dáta z jednotlivých tabuliek**. To slúži na zistenie najrýchlejšieho (najefektívnejšieho) spôsobu zadania požiadavky, čo pomôže pri optimalizácii požiadaviek.

### Množinové operátory RA

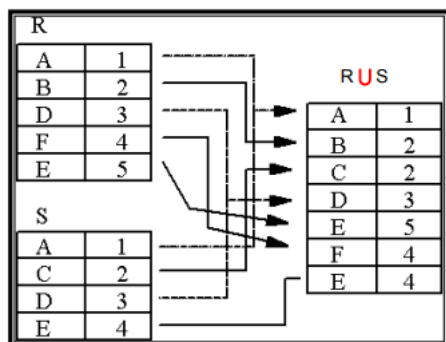
- Na reláciu sa môžeme pozerať ako na množinu n-tíc, preto môžeme použiť množinové operácie

### Základné operácie v relačnej algebre

- Zjednotenie
- Prienik
- Množinový rozdiel
- Karteziánsky súčin
- Podiel (Divide)
- Projekcia
- Reštrikcia (Selekcia)
- Spojenie (Join)

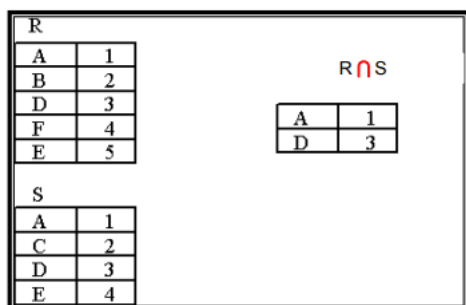
## Zjednotenie

- Množina n-tíc, ktoré sú v relácii R alebo v relácii S alebo v oboch označíme  $R \cup S$



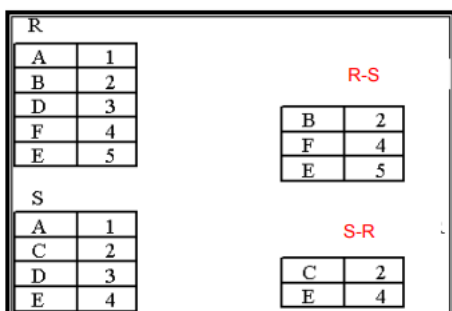
## Prienik

- Prienik relácií  $R$  a  $S$  je množina n-tíc, ktoré sú v  $R$  a zároveň v  $S$  označíme  $R \cap S$

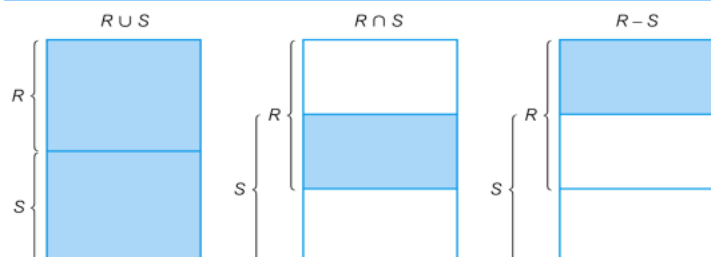


## Množinový rozdiel

- Množina n-tíc, ktoré sú v  $R$ , ale nie sú v  $S$   $R - S$
- aby platil tento výraz,  $R$  a  $S$  musia byť schopné zjednotenia t.j. musia mať ten istý stupeň, tie isté mená a typy atribútov
- platí  $R \cap S = R - (R - S)$



## Relačná algebra graficky



## Karteziánsky súčin

- Nech  $R$  a  $S$  sú dve relácie stupňa  $m$  resp.  $n$ , potom karteziánskym súčinom  $R \times S$  nazývame reláciu stupňa  $m+n$ :  
 $R \times S = \{rs \mid r \in R \wedge s \in S\}$  kde  $rs$  je  $(m+n)$ -tica tvaru  $(r_1, \dots, r_m, s_1, \dots, s_n)$



## Reštrikcia (Selekcia)

• **unárny operátor**, ktorý vyberá z relácie tie n-tice, ktoré spĺňajú **danú Boolovskú podmienku** alebo kombináciu podmienok.

• **Reštrikcia mení počet riadkov výslednej tabuľky**

$R[A \theta k]$

$\theta = \{=, <, >, \geq, \leq, \neq\}$

## SELEKCIA ( $\sigma$ )

•  **$\sigma$  (sigma)** symbol je operátor pre **SELEKCIU**

• Používa sa ako výraz na výber n-tice (riadku), ktorý vyhovuje danej podmienke

$\sigma < \text{podmienka} > (R)$

operátor SELEKCIE vyberie n-tice, ktoré vyhovujú danému predikátu

• Podmienka selekcie sa skladá z – **operátorov** ( , = ,  $\neq$  ), za ktorými nasledujú konštanty alebo mená atribútov (hodnôt)

– a **Boolovských operátorov**  $\wedge$  (AND),  $\vee$  (OR),  $\neg$  (NOT) SELEKCIA ( $\sigma$ )

## Projekcia (Projection)

• **unárny operátor**, ktorý vyberá **vertikálnu podmnožinu** z danej relácie t.j. podmnožinu obsahujúcu len niektoré špecifikované atribúty, pričom sa duplicitné n-tice vylúčia.

• Projekcia relácie R na atribútoch  $A_1, \dots, A_n$   $R[A_1, \dots, A_n]$

## PROJEKCIA( $\pi$ ) Pi

•  $\pi$  (pi) je symbol **PROJEKCIE**.

• Tento operátor zobrazí zoznam tých atribútov vo výslednej relácii, ktoré požadujeme.

$\pi < \text{zoznam atribútov} > (\text{relacia})$

## Relačná algebra graficky



## Divide

• Význam : z prvej tabuľky vyberie tie riadky, ktoré zodpovedajú všetkým riadkom v druhej tabuľke

Formát : `DIVIDE table1 BY table2 GIVING newtable`

## Spojenie (Join)

• Operátor spojenia je **binárny operátor**

• Pomocou neho sa spájajú dve relácie, ktoré majú aspoň jeden atribút spoločný (atribút s tým istým obsahom v dvoch reláciách, predpokladáme, že reprezentuje tú istú vlastnosť).

• Spojenie sa vytvára na základe hodnôt spoločného atribútu v oboch reláciách.

• kombinuje riadky dvoch alebo viacerých tabuliek, pohľadov

• Je to **kartézsky súčin**, na ktorý je aplikovaná podmienka spojenia

• vykonáva sa vždy, ak je vo FROM klauzule viac tabuliek, pohľadov

• Iné názvy pre join: – (vertikálna rekonštrukcia)  
– (disjunkcia/logický súčet)

• spojenie môže byť: – **INNER join** – do výsledku spojenia tabuliek sú zahrnuté **len tie riadky** z oboch tabuliek, pre ktoré bol nájdený odpovedajúci záznam v druhej tabuľke

– **OUTER join** – do výsledku spojenia tabuliek sú **zahrnuté aj riadky**, pre ktoré nebol nájdený odpovedajúci záznam v druhej tabuľke

- implicitná hodnota bez použitia kľúčového slova OUTER | INNER je INNER

- **ak podmienka spojenia nie je udaná, spojenie nám vráti karteziánsky súčin tabuliek**

- Majme relácie R,S obsahujúce atribúty A, resp. B; nech relácia  $T \subseteq A \times B$ . Potom spojením relácií R a S podľa T nazývame nasledujúcu reláciu

$$R \bowtie_T S = \{rs \mid r \in R \wedge s \in S \wedge (r.A, s.B) \in T\}$$

- Duplicitné stĺpce sa vylúčia

### Join variácie

- Theta join

- Equijoin (typ theta joinu)

- Natural join – Semijoin

- Antijoin

- Outer join

- Operácia Join má viacero variácií, najznámejšie z nich sú: –  **$\theta$ -join (theta join)**, všeobecný koncept joinu, spája relácie na základe podmienky, ktorá musí platiť medzi hodnotami atribútov, ak je táto podmienka rovnosť, tak ide o

- equijoin,

- natural join –spája tabuľky podľa rovnako pomenovaných atribútov,

- **semijoin** ( $\bowtie$ )( $\bowtie$ ) – spojí relácie ako join, ale výsledkom je relácia len s atribútmi prvej resp. druhej) relácie,

- **antijoin** ( $\bowtie$ )( $\bowtie$ ) – vráti riadky pôvodnej relácie, ktoré nespojí s druhou reláciou,

- **left outer join** ( $\bowtie$ )( $\bowtie$ ) – ako join, ale k nespojeným riadkom pridá prázdne hodnoty namiesto hodnôt z druhej relácie

- **right outer join** ( $\bowtie$ )( $\bowtie$ ) – k nespojeným riadkom pridá prázdne hodnoty namiesto hodnôt z prvej relácie,

- **full outer join** ( $\bowtie$ )( $\bowtie$ ) – pridá prázdne hodnoty k nespojeným riadkom.

### Theta join ( $\theta$ -join)

- $R \bowtie_F S$  – Definuje reláciu, ktorá obsahuje n-ticu spĺňajúce predikát F karteziánskeho súčinu R a S.
  - predikát F má formu:  $R.a_i \theta S.b_i$  kde  $\theta$  môže byť jedným z operátorov porovnania ( $=, \geq, \leq, \neq$ ).
- Využívajúc základné operácie Selekcie a Karteziánskeho súčinu môžeme theta join prepísať do nasledovnej formy:

$$R \bowtie_F S = \sigma_F(R \times S)$$

- Stupeň theta joinu je sumou stupňov relácií R a S. Ak predikát F obsahuje len rovnosť (equality ( $=$ )), potom používame pojem Equijoin.

### EQUIJOIN

- Najčastejšie používaný join
- Použitý operator je  $=$ ,
- EQUIJOIN používa jeden alebo viacej párov atribútov (pričom ich mená nemusia byť rovnaké), ktoré majú identické hodnoty v každom riadku (n-tice) výslednej relácie.

### NATURAL JOIN

- Podmienkou PRIRODZENÉHO SPOJENIA je rovnaký názov atribútu (atribútov) v oboch spájaných reláciách, ktoré slúžia na vytvorenie spojenia
- Týmto je možné zjednodušiť zápis
- V prípade, že mená nie sú rovnaké, je potrebné najskôr jeden atribút premenovať
- spája riadky dvoch tabuliek, ktoré majú rovnaké hodnoty v stĺpcoch s rovnakým názvom

$$R \bowtie S$$

- Je to Equijoin dvoch relácií R a S cez všetky spoločné atribúty.
- Jeden výskyt každého spoločného atribútu sa z výsledku odstráni.
- Pri použití dajte pozor, aby atribúty s rôznymi vlastnosťami nemali rovnaký názov.

### Vonkajšie spojenie (Outer join)

- OUTER JOIN sa používa k výpisu aj takých záznamov, ktoré nespĺňajú spojovacie kritérium.
- Podľa tabuľky, z ktorej sa všetky záznamy objavajú vo výslednej relácii, môže byť spojenie
  - ľavé (LEFT OUTER JOIN),
  - pravé (RIGHT OUTER JOIN),
  - Úplné (FULL OUTER JOIN)

### Outer Join

- Existujú tri formy VONKAJŠIEHO SOJENIA, závisí to na tom ktoré údaje sa nachádzajú vo výslednej relácii.
  - ĽAVÉ VONKAJŠIE SPOJENIE (LEFT OUTER JOIN) – Údaje z ľavej tabuľky
  - PRAVÉ VONKAJŠIE SPOJENIE (RIGHT OUTER JOIN) - Údaje z pravej tabuľky
  - ÚPLNÉ VONKAJŠIE SPOJENIE (FULL OUTER JOIN) - Údaje z oboch tabuliek
- (LEFT) OUTER JOIN je spojenie, v rámci ktorého riadky z R, ktoré nemajú zodpovedajúcu hodnotu v prepájacom stĺpci v S, sa tiež zahrnú do výslednej relácie.
- Označujeme:  $R \bowtie S$
- Analogicky to platí aj pre pravé a úplné vonkajšie spojenie.

### Semijoin

- $R \bowtie_F S$ 
  - Definuje reláciu, ktorá obsahuje riadky (n-tice) z R, ktoré sa zúčastňujú na spojení R s S.
  - Vykonáva sa spojenie dvoch relácií a potom projekcia atribútov prvého operandu.

U Semijoin je možné zapísať využijúc PROJEKCIU a SPOJENIE:  $R \bowtie_F S = \Pi_A(R \bowtie S)$

### Relačná algebra graficky

$T$	$U$	$T \bowtie U$	$T \bowtie_B U$	$T \bowtie_C U$																																							
<table border="1"> <tr><th>A</th><th>B</th></tr> <tr><td>a</td><td>1</td></tr> <tr><td>b</td><td>2</td></tr> </table>	A	B	a	1	b	2	<table border="1"> <tr><th>B</th><th>C</th></tr> <tr><td>1</td><td>x</td></tr> <tr><td>1</td><td>y</td></tr> <tr><td>3</td><td>z</td></tr> </table>	B	C	1	x	1	y	3	z	<table border="1"> <tr><th>A</th><th>B</th><th>C</th></tr> <tr><td>a</td><td>1</td><td>x</td></tr> <tr><td>a</td><td>1</td><td>y</td></tr> </table>	A	B	C	a	1	x	a	1	y	<table border="1"> <tr><th>A</th><th>B</th></tr> <tr><td>a</td><td>1</td></tr> </table>	A	B	a	1	<table border="1"> <tr><th>A</th><th>B</th><th>C</th></tr> <tr><td>a</td><td>1</td><td>x</td></tr> <tr><td>a</td><td>1</td><td>y</td></tr> <tr><td>b</td><td>2</td><td></td></tr> </table>	A	B	C	a	1	x	a	1	y	b	2	
A	B																																										
a	1																																										
b	2																																										
B	C																																										
1	x																																										
1	y																																										
3	z																																										
A	B	C																																									
a	1	x																																									
a	1	y																																									
A	B																																										
a	1																																										
A	B	C																																									
a	1	x																																									
a	1	y																																									
b	2																																										
		(g) Natural join	(h) Semijoin	(i) Left Outer join																																							

### Ďalšie relačné operácie

- Niektoré bežne používané dotazy nemôžu byť vyjadrené štandardnými operáciami relačnej algebry.
- Preto sa zavádzajú dodatočné operácie, ktoré zvyšujú vyjadrovaciu schopnosť relačnej algebry.

### Agregačné funkcie

- Sú operácie, ktoré sa vykonávajú nad množinou záznamov vybraných z databázy.
- Medzi tieto funkcie patria:
  - SUM - súčet všetkých vybraných hodnôt atribútu
  - AVERAGE - priemerná hodnota všetkých vybraných hodnôt atribútu
  - MAXIMUM - maximálna hodnota atribútu zo všetkých vybraných hodnôt
  - MINIMUM - minimálna hodnota atribútu zo všetkých vybraných hodnôt
  - COUNT - počet vybraných záznamov (n-tíc)
- Ďalšou častou požiadavkou je združenie záznamov podľa hodnoty niektorých jeho atribútov a aplikovanie agregáčnych funkcií nezávisle na každú skupinu

- Všeobecne operáciu s použitím funkcie môžeme zapísať:

< atributy pre zdruzenie> $\varnothing$  <zoznam funkcii> (<nazov relacie>)

- Ak nie sú špecifikované žiadne atribúty zdruzenia, potom je funkcia aplikovaná na hodnoty atribútu nad všetkými n-ticami.

Výsledok aplikácie agregovanej funkcie je relácia a nie skalárna hodnota, a to dokonca i vtedy, keď má len jednu hodnotu !

### Relačný kalkul

- Výraz relačného kalkulu vytvára novú reláciu, ktorá je špecifikovaná riadkovou (riadkový kalkul) alebo stĺpcovou premennou (doménový kalkul).
- V kalkule nie je určené poradie vykonania operácií k získaniu výsledku – výraz kalkulu len špecifikuje akú informáciu má obsahovať výsledok. – Toto je hlavný rozdiel medzi relačnou algebrou a relačným kalkulom.
- Relačný kalkul je považovaný za neprocedurálny jazyk.
- Na rozdiel od toho v relačnej algebre je potrebné špecifikovať poradie operácií; preto je relačná algebra považovaná za procedurálny spôsob špecifikovania dotazu.

### Riadkovo-orientovaný relačný kalkul

- Je založený na špecifikácii riadkových premenných.
- Každá riadková premenná je zvyčajne zviazaná s konkrétnou reláciou (tabuľkou), čo znamená, že premenná môže nadobúdať ako svoju hodnotu akýkoľvek riadok z danej relácie (tabuľky).
- Jednoduchý výraz kalkulu má tvar:  $\{t \mid \text{COND}(t)\}$ 
  - Kde **t** je riadková premenná a COND (t) je **podmienený výraz obsahujúci t**.
  - Výsledkom je množina všetkých riadkov (n-tíc) t, ktoré vyhovujú podmienke COND (t).

### Existenčné a univerzálne kvantifikátory

- Vo výrazoch sa môžu objaviť kvalifikátory:
  - univerzálny kvalifikátor ( $\forall$ )
  - existenčný kvalifikátor ( $\exists$ ).
- Riadková premenná t je viazaná, ak je kvantifikovaná, prostredníctvom ( $\forall t$ ) alebo ( $\exists t$ ) klauzuly.
- Ak F je výraz, tak aj ( $\exists t$ )(F) a ( $\forall t$ )(F) sú výrazy, kde t je riadková premenná.
  - Výraz ( $\exists t$ )(F) je pravdivý, ak F je pravdivý pre (aspoň jeden) riadok (n-ticu) priradenú premennej t v F; inak ( $\exists t$ )(F) je nepravdivý.
  - Výraz ( $\forall t$ )(F) je pravdivý, ak F je pravdivý pre každý riadok (n-ticu) priradenú premennej t v F; inak ( $\forall t$ )(F) je nepravdivý.

### Jazyky založené na riadkovo-orientovanom kalkule

- Jazyk SQL používa základnú blokovú štruktúru na vyjadrenie dotazov v riadkovom kalkule:
  - SELECT <zoznam atributov>
  - FROM <zoznam relácii>
  - WHERE <podmienky>
- Klauzula SELECT je vyjadrením projekcie, klauzula FROM špecifikuje relácie (tabuľky) potrebné pre realizáciu dotazu a klauzula WHERE vyjadruje reštrikciu (selection), ako aj podmienky spojenia.
- Ďalším jazykom založeným na riadkovom kalkule je **QUEL**, ktorý používa riadkovú premennú:
  - RANGE OF <meno premennej> IS <nazov relacie>
- Potom – RETRIEVE <zoznam atributov z riadkovej premennej >
  - WHERE < podmienky >
- Tento jazyk bol navrhnutý pre SRBD INGRES.

### Doménový relačný kalkul

- Doménový kalkul je rovnocenný riadkovému kalkulu a relačnej algebre.

- **Jazyk QBE** (Query-By-Example), ktorý je založený na doménovom kalkule, bol vyvinutý skoro súčasne s SQL v IBM Research, Yorktown Heights, New York.
- Doménový kalkul sa líši od riadkového kalkulu v type premennej použitej vo výrazoch: – Premenné nenadobúdajú hodnotu jednotlivých riadkov relácie, ale hodnoty atribútu z danej domény.
- Aby sme mohli vytvoriť reláciu stupňa n ako výsledok dotazu, je potrebné zaviesť n doménových premenných – pre každý atribút jednu.
- Výraz doménového kalkulu má tvar:  $\{x_1, x_2, \dots, x_n \mid \text{COND}(x_1, x_2, \dots, x_n, x_{n+1}, x_{n+2}, \dots, x_{n+m})\}$ 
  - Kde  $x_1, x_2, \dots, x_n, x_{n+1}, x_{n+2}, \dots, x_{n+m}$  sú doménové premenné s oborom hodnôt príslušnej domény (atribútov)
  - a COND je podmienka alebo výraz doménového kalkulu.

## SQL

- **Structured Query Language (SQL)**
  - Štandardizovaný ANSI
  - Podporovaný modernými SRBD
- Príkazy sa delia do troch skupín
  - Data Definition Language (DDL) • vytváranie a modifikácia štruktúry,
  - Data Manipulation Language (DML) • výber a modifikácia údajov
  - Data Control Language (DCL) • riadenie prístupu k údajom – prideľovanie

a odoberanie práv užívateľov

## Bezpečnosť a ochrana v DBS

### Pojmy

- **Konzistentný stav** – ak je stav BD odrazom možnej situácie v reálnom svete (dvt = nedeľa a prednáška)
- **Integrita** (celistvosť) – BD je v konzistentnom stave a je disponibilná
- **Ohrozenie** (threat) – nevyžiadaný náhodný, alebo zámerný zásah do DBS, modifikujúci, alebo mažúci údaje v DBS

### Typy ohrození

- **Náhodné ohrozenia:** – Prírodné katastrofy, havárie alebo nešťastia
  - Chyby hardvéru a softvéru
  - Ľudské chyby
- **Zámerné ohrozenia:** – Autorizovaní používatelia – zneužijú svoje práva
  - Nepriateľské ohrozenia – neautorizované zmeny údajov prostredníctvom napadnutia hw či sw

### Príklady zámerných ohrození

- SQL Injection
- Neautorizovaný prístup
- Ukradnutie prístupových údajov
- Odpočúvanie sieťovej komunikácie

## Metódy ochrany

- **Šifrovanie** – citlivých údajov

- **Autorizácia** – pohľady, prístupové práva
- **Autentifikácia** – heslá
- **Sieťová ochrana** – firewaly, net proxies, ..

## Šifrovanie

- Šifrovanie (encryption) – Šifrovanie si kladie za cieľ transformovať vstupné dáta do podoby, v ktorej sú pre potenciálneho útočníka nezrozumiteľné a nie je schopný rekonštruovať ich pôvodný tvar.
- Zároveň požadujeme, aby oprávnené entity mohli pôvodné dáta rekonštruovať.

## Autorizácia

- Autorizácia (authorization) – proces identifikujúci, ktorá identita má právo pristupovať ku ktorej informácii a akým spôsobom.
- Príkladom je pridelenie práv databázovému používateľovi na vkladanie záznamov do nejakej databázovej tabuľky

## Autentifikácia

- Identifikácia (identification) a Autentifikácia (authentication) – proces na rozpoznanie a overenie entity, ktorá chce pristupovať k určitej informácii.
- Takáto entita sa nazýva autentifikovanou identitou, ak je dokázané napríklad podľa mena a hesla, že je to presne ona.
- Príkladom môže byť autentifikácia používateľa do DBS pomocou používateľského mena a hesla. **BEZPEČNOSŤ V DBS**

## Bezpečnosť v DBS

- Bezpečnosť databázy znamená chrániť databázu od zámerných alebo náhodných ohrození.
- To zahŕňa zabezpečenie:
  - hardvéru,
  - softvéru,
  - ľudí a – údajov, vzťahujúcich sa k databáze.
- Základným účelom udržania bezpečnosti je minimalizovať straty, ktoré môžu nastať v dôsledku rôznych bezpečnostných hrozieb.

## Ohrozenia bezpečnosti

- Niektoré dôležité faktory, týkajúce sa bezpečnosti databázy sú nasledovné:
  - **Neautorizovaný prístup k údajom** - krádeže a zneužitie údajov: krádeže údajov a podvody ovplyvňujú prostredie databázy, ako aj celej organizácie. Riadenie prístupu je podmienkou obmedzenia neautorizovanej manipulácie s údajmi.
  - **Strata utajenia**: utajenia požiadavka, aby ostali údaje bezpečné.
  - **Strata ochrany súkromia**: súkromie je požiadavka ochrany osobných údajov ľudí. Problém v ochrane súkromia, môže mať za následok právne kroky proti organizácii.
  - **Strata dostupnosti**: znamená to, že údaje alebo systém nemusia byť dostupné. To môže mať za následok vážne poškodenie finančnej výkonnosti organizácie.

## Požiadavky na bezpečnostné služby

- **Dôvernosť** (utajenie) (confidentiality) údajov

- **Zaistenie integrity** (integrity) BD
- **Zaistenie dostupnosti** (availability) BD
- **Výkon sledovateľnosti** (auditing) v BD

### Dôvernoscť údajov

- Prístup k jednotlivým informáciám musí byť povolený jedine autorizovaným entitám – ochrana pre neautorizovanými užívateľmi.
- Splnenie tejto požiadavky je najčastejšie v DBS realizované pomocou autentifikačných a autorizačných mechanizmov, pričom citlivé informácie musia byť šifrované a databázové súbory musia byť umiestnené na zabezpečenom serveri.
- Zabezpečí, že užívatelia majú oprávnenie robiť to, o čo sa pokúšajú

### Integrita BD

- Informácia z hľadiska bezpečnosti spĺňa vlastnosť integrity, ak nebola zmenená žiadnou entitou, ktorá na jej zmenu nemá právo
- Je to ochrana údajov v BD pred autorizovanými užívateľmi
- Zabezpečí, že to, čo sa pokúšajú užívatelia spraviť, je správne a v súlade s realitou

### Integrita

- zahŕňa viacero aspektov:
  - **riadenie prístupu** (access control) – Prístup k databázovým objektom musí byť riadený pomocou oprávnení, takže iba oprávnené entity môžu meniť dáta.
  - **ochrana DBS** – DBS musí byť chránený proti škodlivému kódu, ktorý by mohol poškodiť dáta (napr. vírusy).
  - **ochrana sieťovej komunikácie** – sieťová komunikácia medzi DBS a inou entitou musí byť chránená proti odstráneniu, poškodeniu a odpočúvaniu.

### Dostupnosť informácií

- Informácie by mali byť autorizovanej entite dostupné kedykoľvek ich bude potrebovať.
- Dostupnosť informácií môžu ohroziť napríklad útoky typu DoS (Denial of service)

### Sledovateľnosť

- Mala by existovať možnosť uchovávanía záznamov o prístupoch (oprávnených ako aj neoprávnených) k informáciám a následná analýza týchto záznamov, ktorá môže prispieť k zvýšeniu bezpečnosti.
- V niektorých DBS môžeme napríklad sledovať, kto a kedy databázové objekty vytváral, odstraňoval, editoval a podobne.

### Bezpečnostné politiky

- Politika je množina pravidiel, ktorými by sa mali riadiť všetci používatelia, a ktoré predstavujú smernicu potrebnú pre implementáciu zabezpečenia.
- **Globálne**
- **Lokálne**

### Globálne bezpečnostné politiky

- definujú množinu všeobecných pravidiel, ktoré platia pre celú organizáciu a obsahujú všeobecnejšie a abstraktnejšie postupy, čím nastavujú úroveň zabezpečenia

### Lokálne bezpečnostné politiky

- dopĺňajú globálne politiky tým, že definujú technické detaily.
- Lokálne politiky sú konkrétnejšie ako globálne politiky.
- Lokálne politiky môžu stanoviť podmienky, za ktorých sú možné výnimky z globálnej politiky

### Analýza rizík

- Každá bezpečnostná politika musí začať analýzou a manažmentom zistených rizík.
- Analýza rizík je jednou z kľúčových aktivít zvyšovania bezpečnosti.
- Cieľom je zistiť, čo ohrozuje DBS a údaje v ňom, ako je DBS proti takýmto hrozbám chránený, a kde sú zraniteľné miesta v bezpečnosti DBS.
- Pozostáva z nasledujúcich štyroch krokov:
  - **Identifikácia hrozieb**
    - Externé hrozby – útočníci.
    - Interné hrozby – zamestnanci s prístupom k databázovej aplikácii, OS alebo DBS.
    - Prírodné katastrofy – povodeň, zemetrasenie, atď.
    - Technické poruchy – napríklad výpadok elektrickej energie
  - **Identifikácia zraniteľných miest pre každú aplikáciu** (vrátane DBS a OS).
  - **Určenie pravdepodobnosti výskytu hrozby**, a na základe toho určenie priority implementácie protiopatrení.
  - **Identifikácia následkov**, ktoré nastanú v dôsledku využitia zraniteľného miesta útočníkom a identifikácie možností zamedzenia využitia zraniteľného miesta.

## OCHRANA INTEGRITY

### Možnosti narušenia integrity

- uviaznutie
- chyba transakcie
- chyba aplikačného programu
- chyba systémového programu
- porucha hardvéru
- chybné údaje
- zlá logika spracovania

### Možnosti ochrany

- ochrana súborov
- ochrana transakcií

## OCHRANA SÚBOROV



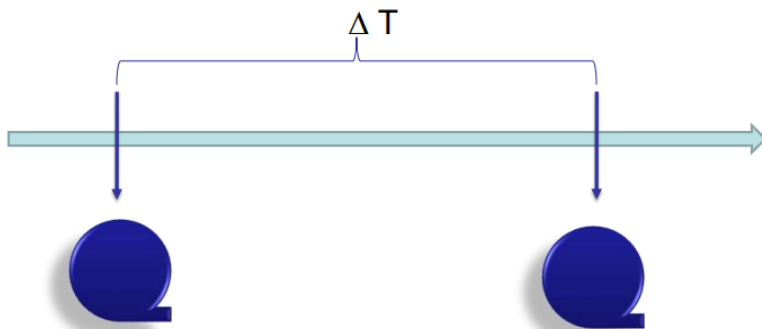
## OS 1

- slúži najmä ako ochrana pred fyzickým poškodením
- Viacnásobné kópie – paralelné zaznamenávanie zmien do súborov na viacerých nosičoch; zrkadlenie disku, kľúčové súbory, slovník dát

## OS2

- Kopírovanie súborov – periodické vytváranie kópií napr. na magnetické pásky.
- Pri poruche sa obnoví stav z kópie, stratia sa zmeny medzi archiváciou a časom poruchy

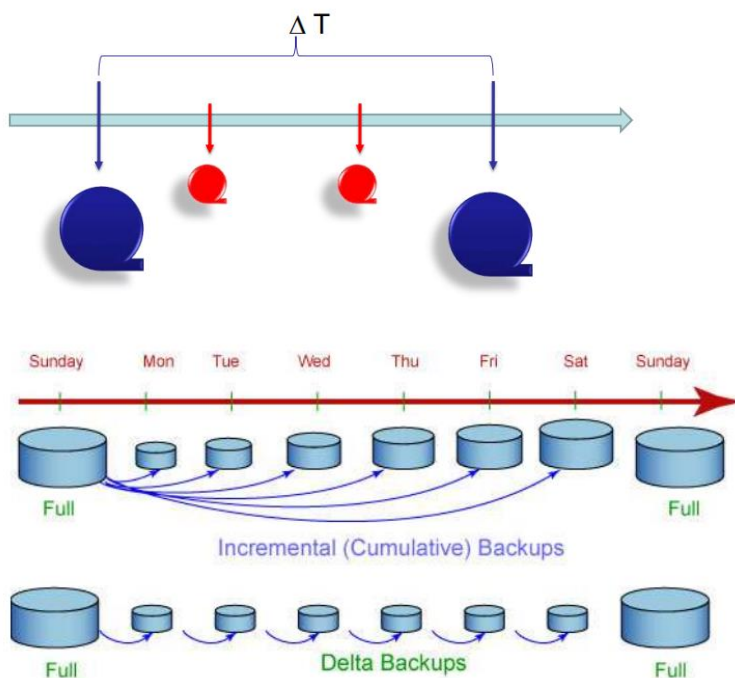
### Kopírovanie súborov



## OS3

- Inkrementálne kopírovanie – vytváranie kópií v dlhších časových intervaloch, medzitým sa archivujú zmenené bloky.
- Pri poruche sa obnoví stav z kópie a všetky zmeny po archivácii, stratia sa zmeny medzi archiváciou poslednej a časom poruchy Inkrementálne kopírovanie

### Inkrementálne kopírovanie



## Zálohovanie

- **Offline zálohovanie:** – databáza nie je prístupná užívateľom.  
– Databázový administrátor musí naplánovať čas na zálohovanie.
- **Online zálohovanie:** – databáza je prístupná užívateľom  
– Nie je potrebné plánovať výluky z dostupnosti databázy

## OS4

- Zaznamenávanie zmien – priebežné zaznamenávanie zmien do žurnálového súboru = protokol o zmenách.
- Pri poruche sa obnoví stav z kópie a všetky zmeny po archivácii, stratia sa zmeny medzi archiváciou poslednej zmeny a časom poruchy

## Transakcia

- je postupnosť operácií nad objektami BD, ktorá z hľadiska užívateľa tvorí ucelenú operáciu (jednotku).
- môžu mať veľkosť od jednoduchých operácií k mnohopočetným komplexným operáciám (tisíce operácií)

## Vlastnosť transakcie

- transformuje BD z jedného konzistentného stavu do iného konzistentného stavu.
- počas vykonávania transakcie BD nemusí byť v konzistentnom stave => alebo sa vykoná celá transakcia alebo sa nevykoná ani jedna operácia == transakcia je teda atomická
- počas vykonávania transakcie nesmú byť objekty, nad ktorými sa realizuje transakcia, viditeľné pre iné transakcie
- pri zlyhaní dokončenia transakcie je potrebné vrátiť všetky dotknuté objekty do stavu pred transakciou -> návrat do východného stavu
- príklad: rezervácia leteniek pre viacčlennú rodinu z A do B s prestupmi

## Príčina zlyhania transakcie

- Možné dôvody zničenia transakcie:
  - technická porucha
  - chyba softvéru
  - iné transakcie ...
- Technika potvrdzovania transakcií

## Zlyhanie transakcie

- V prípade poruchy je potrebné:
  - obnoviť čo najviac potvrdených transakcií
  - odstrániť následky nepotvrdených a zničených transakcií

## Technika ochrany transakcií

- základný prostriedok == žurnálový súbor (log file)
- záznam v žurnále obsahuje najmä:
  - identifikátor transakcie
  - adresu objektu
  - starú hodnotu objektu
  - novú hodnotu objektu
  - kontrolné body transakcie (ak existujú)

## Technika ochrany transakcií

- žurnálový súbor umožňuje **dva typy činností** v prípade poruchy:
  - zopakovanie transakcie
  - vycúvanie transakcie

## Typy operácií na objektami BD

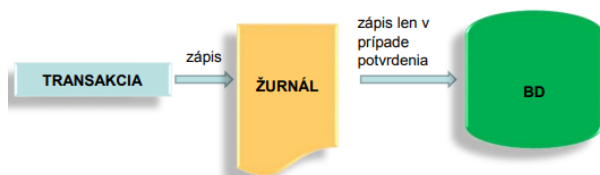
- **nechránené** (nemenia hodnotu objektu napr. čítanie)
- **chránené** (menia hodnotu objektu napr. zápis)

### Dvojfázové potvrdzovanie

- Následky potvrdených transakcií je nutné opravovať pomocou kompenzačných transakcií

**A.** transakcia nesmie meniť hodnoty objektov BD skôr, ako je potvrdená

**B.** transakcia nesmie byť potvrdená skôr, ako sa zapíše do žurnálového súboru



- **fáza 1** : zápis do žurnálu
- **fáza 2** : prepis zmien zo žurnálu do BD
- pre obe fázy je potrebné zamknúť dotknuté objekty BD

### Priamy zápis do BD

- pri poruche treba vycúvať všetky ovplyvnené transakcie
- transakcia postupne uvoľňuje objekty pred svojim potvrdením, tým umožňuje iným transakciám prácu

### Obnova z poruchového stavu

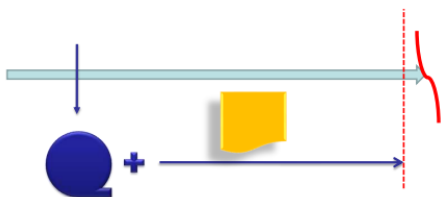
- odstránenie zmien spôsobených zničenou transakciou – vycúvanie (rollback)
- zopakovanie potvrdených transakcií – (rollforward)

### Rollback

1. prehľadanie žurnálu, identifikovanie pôvodných stavov objektov danej transakcie
  2. prepis týchto stavov do BD
- ak je potrebné vycúvať potvrdenú transakciu, potom sa musia vycúvať aj všetky transakcie, ktoré pracovali s dotknutými objektami po nej.
  - vycúvanie je efektívne pri malých poruchách, uviaznutí, poruchách softvéru, ktoré ovplyvnia malý počet transakcií

### Rollforward

1. určenie posledného kontrolného bodu pred poruchou
  2. zavedenie poslednej BD pred poruchou
  3. použitím žurnálu sa aplikujú nové hodnoty objektov potvrdených transakcií od bodu potvrdenia
- používa sa pri poruchách hardvéru, systémových poruchách a poruchách veľkého rozsahu. rollforward



### Paralelizmy v DBS

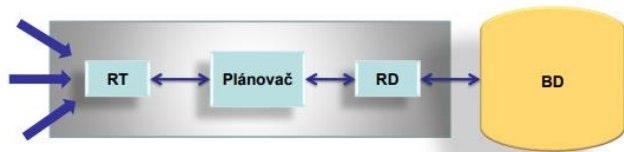
## Paralelné spracovanie dát

- Pri realizácii niekoľkých transakcií súbežne, pričom zdieľajú tie isté objekty BD, musí byť ich vykonávanie synchronizované.
- Výsledok ich vykonania musí byť taký istý, ako keby sa vykonávali za sebou.

## Pojmy

- Sérializovateľnosť rozvrhov – také paralelné vykonanie transakcií, že po vykonaní sérializovateľného rozvrhu sa dosiahne taký istý stav BD, ako po vykonaní sériového rozvrhu týchto transakcií. (postupné vykonávanie)

Vytváranie rozvrhov v SRBD RT Plánovač RD BD



## Štruktúra SRBD

- RT – modul riadenia transakcií – označuje objekty, ktoré požadujú transakcie
- RD – modul riadenia dát – vykonáva operácie
- plánovač – zabezpečuje synchronizáciu požiadaviek transakcií a vytvára rozvrh t.j. určuje poradie vykonávania operácií

## Strata aktualizácie

1. T1 číta záznam R
  2. T2 číta záznam R
  3. T1 aktualizuje R a zapíše ho do BD
  4. T2 aktualizuje R a zapíše ho do BD
- Aktualizácia vykonaná T1 sa stratila

## Neaktuálne vyhľadávanie

1. T1 číta záznam R
  2. T2 číta záznam R
  3. T1 modifikuje R a zapíše ho do BD
  4. T2 vyhľadávala podľa neaktuálnej hodnoty R
- aktualizácia vykonaná T1 sa stratila

## TECHNIKY VYTVÁRANIA SÉRIALIZOVATEĽNÝCH ROZVRHOV

### Zamykanie

- prevencia chýb, vyplývajúcich z čítania hodnoty objektu transakciou, keď iná transakcia mení hodnotu objektu
- dva typy uzamknutia: – zamknutie pre čítanie – hodnotu objektu môže čítať viacero transakcií, ale žiadna nemôže meniť jeho hodnotu

– zamknutie pre zápis – len jedna transakcia môže čítať a meniť hodnotu objektu  
(exkluzívne zamknutie)

- objektom zamknutia môže byť:
  - tabuľka
  - riadok
  - atribút



## Uviaznutie

- T1 zamkne A
- T2 zamkne B
- T1 požaduje zamknúť B -> musí čakať na T2
- T2 požaduje zamknúť A -> musí čakať na T1
- Viacero stratégií riešenia problému uviaznutia:
  - najskôr všetko zamknúť, potom spustiť transakciu == dôsledkom je čakanie na realizáciu.
  - umožniť uviaznutia, rozpoznať ho a vyriešiť == jedna z transakcií sa zničí a spustí sa ešte raz neskôr. Uviaznutie je možné detekovať z grafu čakania.
  - umožniť zamykanie objektov rešpektujúc lineárne usporiadanie

### Dvojfázové zamykanie

- **zabezpečuje sérializovateľnosť rozvrhov**
- Každá transakcia:
  - pred operáciou zamkne objekt
  - nezamyká objekt, ktorý už zamkla
  - pred ukončením všetky objekty odomkne
- pri dvojfázovom zamykaní:
  - a) objekt môže byť zamknutý **len pre jednu transakciu**
  - b) ak transakcia odomkne aspoň jeden objekt, nesmie už žiadny ďalší zamknúť

**I. fáza:** zamykajú sa všetky objekty bez ich odomykania

**II. fáza:** objekty sa postupne odomykajú

### Úlohy plánovača

- riadiť **zamykanie** objektov
- **čítanie a zápis povoliť len pre transakciu**, ktorá má dané objekty zamknuté
- sledovať **dodržiavanie dvojfázového protokolu**, pri nedodržaní transakciu zničiť
- riešiť stavy **uviaznutia**

### Časové pečiatky

- Ďalší zo spôsobov riadenia súbežných transakcií
- časová pečiatka (čp) je číslo priradené transakcii alebo objektu bázy dát.
- čp tvoria rastúcu postupnosť == funkcia času
- prideluje ich **modul riadenia transakcií (RT)**, aby pomocou **nich riadil vykonávanie konfliktných transakcií**
- **princíp použitia čp:** všetky páry konfliktných operácií treba vykonať v poradí ich čp == vytvárajú sa len sérializovateľné rozvrhy

### Základný plánovač

- registruje najvyššiu čp operácie čítania a zápisu nad daným objektom.
- ak transakcia s nižšou čp požaduje daný objekt, je zničená a spustí sa znova s vyššou čp
- Dôsledok: časté ničenie transakcií

### Konzervatívny plánovač

- modifikácia základného plánovača
- požiadavky z RT ukladá do fronty a vyberá z nej operácie s najnižšou čp
- Dôsledok – zníženie počtu odmietnutí, ale zdržuje sa realizácia transakcií

### Thomasov plánovač

- Ak má operácia zápisu nižšiu čp ako už realizovaná operácia zápisu a túto hodnotu ešte nikto nečítal, môžeme ignorovať túto operáciu, pretože by priradila už zastaralú hodnotu

### Objektovo orientované

#### Kritériá delenia DBS

- Dátový model
- Dotazovací jazyk
- Výpočtový model (tj. navigácia a prístup k údajom)

### Relačné systémy

- návrh: E.F.Codd
- implementácia IBM a i.
- Štandard popísaný ANSI a ISO normou,

### Dátový model

- RSRBD údaje v databáze vo forme tabuliek.
- Riadok zodpovedá záznamu (record, tuple);
- stĺpec zodpovedá atribútom (poliam záznamu).
- Každý stĺpec má určený dátový typ.
- Dátových typov je obmedzené množstvo, typicky 6 a viac (napr. znak, reťazec, dátum, číslo...).
- Každý atribút (pole) záznamu môže uchovávať jedinú hodnotu.
- Vzťahy nie sú explicitné, ale vyplývajú z hodnôt v špeciálnych poliach, tzv. cudzie kľúče (foreign keys) v jednej tabuľke, ktoré sa rovnajú hodnotám v inej tabuľke.

### Jazyk

- SQL
- Pohľad (view) je podmnožinou databázy, ktorá je výsledkom vyhodnotenia dopytu. V RSRBD je pohľadom tabuľka. RSRBD využíva SQL na definíciu údajov, riadenie údajov a prístupu a získavanie údajov.
- Údaje sú získavané na základe hodnoty v určitom poli záznamu.

### Výpočtový model

- Celé spracovanie je založené na hodnotách polí záznamov.
- Záznamy nemajú jednotné identifikátory.
- Neexistujú žiadne odkazy z jedného záznamu na iný.

### Výpočtový model

- Výsledok je vytváraný pod kontrolou kurzoru, ktorý umožňuje užívateľovi sekvenčne prechádzať výsledok po jednotlivých záznamoch.
- To isté platí aj pre update.

## Objektovo – orientované DBS

- Uplatňujú sa v „nových“ oblastiach ako napr. navrhovanie artefaktov, spracovanie dokumentov, softwarové inžinierstvo, vedecké databázy (medicína)
- Priniesli bohatšie dátové a procedurálne modelovanie:
  - spájanie príbuzných dát – lepšia reprezentácia zložitých dát. štruktúr (vektory)
  - spájanie dát s funkciami – ukrývanie dát, typové dedenie, polymorfizmus
  - spájanie databáz. a programovacieho jazyka – integrácia datab. prostredia s program. prostredím
- Poznáme dve historické vývojové cesty OODBS:
  1. Pridávanie databáz. črt do O – O programovacích jazykov
  2. Rozšírenie relačných databáz. systémov o O – O črty

■ **Nevýhoda OODBS** – v porovnaní s relačnými databázovými systémami sa znížila jednoduchosť a matematická precíznosť

## OO systémy

- Pre objektové databázy neexistuje žiadny oficiálny štandard. Štandardom je de facto kniha Morgana Kaufmana The Object Database Standard: ODMG-V2.0.
- **Dôraz** v OOSRBD je na :
  - Objektoch a objektových vzťahoch v aplikáciách napísaných v OO jazykoch
  - spôsobe ich uchovávaní v databáze.

## Dátový model

- OO systémy využívajú dátový model, ktorý má objektovo-orientované aspekty, ako sú triedy s atribútmi a metódami a integritnými obmedzeniami;
- poskytujú objektové identifikátory (OID) pre každú trvalú inštanciu triedy;
- podporujú
  - **zapuzdrenie (encapsulation);**
  - **násobnú dedičnosť (multiple inheritance)**
  - **abstraktné údajové typy.**
- kombinujú prvky objektovo-orientovaného programovania s databázami.
- Rozširujú funkčnosť objektových programovacích jazykov (C++, Smalltalk, Java,...)
- poskytujú úplnú schopnosť programovania databázy.
- Dátový model aplikácie a dátový model databázy sa vo výsledku temer zhodujú
- výsledný kód sa dá efektívnejšie udržiavať.

## Jazyk

- Objektovo orientovaný jazyk (C++, Java, Smalltalk) je jazykom ako pre aplikáciu, tak aj pre databázu.
- Poskytuje úzky vzťah medzi objektom aplikácie a uloženým objektom. Názorne je to vidieť v definícii a manipulácii s údajmi a v dopytoch.

## Výpočtový model

- V relačných systémoch sa rozumie dotazovacím jazykom vytváranie, prístup a aktualizácia objektov, ale v objektových, hoci je to stále možné, je toto realizované priamo pomocou objektového jazyka (C++, Java, Smalltalk) využitím jeho vlastnej syntaxe.
- Naviac každý objekt v systéme automaticky obdrží identifikátor (OID), ktorý je jednoznačný a nemenný počas existencie objektu.
- **Objekt môže mať buď vlastné OID,**
- **alebo môže ukazovať na iný objekt.**

## Objektovo relačné systémy

- "Rozšírené relačné", „Post-relačné" či "objektovo-relačné", sú synonymá pre databázové systémy, ktoré sa snažia zjednotiť rysy ako relačných, tak aj objektových systémov.
- ORSRBD je špecifikovaný v rozšírení **SQL štandardu — SQL3**.
- Do tejto kategórie patrí napr. Informix, IBM, Oracle a Unisys.

## Dátový model

- ORSRBD využívajú dátový model tak, že "pridávajú objektovosť do tabuliek".
- Všetky trvalé údaje sú v tabuľkách, ale niektoré položky môžu mať bohatšiu dátovú štruktúru, nazývanú abstraktné dátové typy (ADT).
- ADT je dátový typ, ktorý vznikne skombinovaním základných dátových typov.
- Podpora ADT je atraktívna, pretože operácie a funkcie asociované s novými dátovými typmi môžu byť použité k indexovaniu, ukladaniu a získavaniu záznamov na základe obsahu nového dátového typu.
- ORSRBD sú nadmnožinou RSRBD. Pokiaľ nevyužijeme žiadne objektové rozšírenie sú ekvivalentné SQL2.
- Preto majú obmedzenú podporu dedičnosti, polymorfizmu, referencií a integrácie s programovacím jazykom.

## Jazyk

- ORSRBD podporuje rozšírenú verziu SQL — SQL3.
- Dôvodom je podpora objektov (tj. dotazy obsahujúce atribúty objektov).
- Typické rozšírenie zahŕňa dotazy obsahujúce vnorené objekty, atribúty, abstraktné dátové typy a použitie metód.
- ORSRBD je **stále relačný**, pretože údaje sú uložené v tabuľkách a SQL, vrátane spomenutých rozšírení, pracuje práve s nimi.

## Výpočtový model

- Jazyk SQL s rozšírením pre prístup k ADT je stále hlavným rozhraním pre prácu s databázou. Priama podpora objektových jazykov stále chýba, čo núti programátorov k prekladu medzi objektami a tabuľkami.

## POROVNANIE

### I

- Relačné systémy sú dobré pre spravovanie veľkého množstva údajov;
- OO programovacie jazyky sú dobré vo vyjadrovaní zložitých vzťahov medzi objektami.
- RSRBD sú dobré pre vyhľadávanie údajov, ale poskytujú malú podporu pre manipuláciu s nimi;

### II



- OO programovacie jazyky sú výborné pri manipulácii s údajmi, ale poskytujú malú alebo žiadnu podporu pre nemennosť a vyhľadávanie údajov.
- Tieto prístupy sú protichodné a preto je ich skĺbenie trochu komplikované.

### III

- OO model poskytuje základné vlastnosti objektov — zapúzdrenie, dedičnosť a polymorfizmus. Navyše, každý objekt má jednoznačnú identifikáciu, ktorá umožňuje používanie odkazov.
- RSRBD poskytujú vlastnosti, ktoré OO programovacie jazyky nemajú, ako napr. rýchle vyhľadávanie, zdieľanie objektov, prepracovaný systém opráv chýb pre databázové operácie, trvalé uloženie, atď.

### Vznik

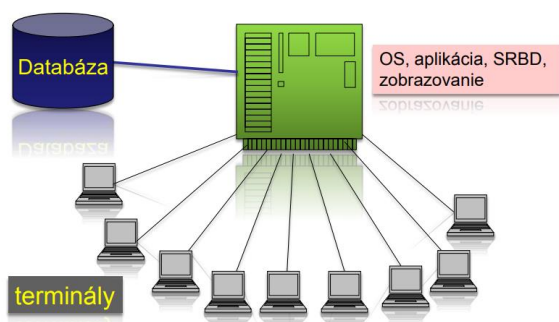
- pridávaním databázových črt do O-O programovacích jazykov - takýmto spôsobom na báze jazyka smalltalk vznikol ODBMS GemStone, z jazyka C++ systémy ObjectStore, Ontos, Versant, Objectivity, rozšírením lispu systém Orion a iné;
- rozšírením relačných SRBD o O-O črty (uvedené systémy sa preto niekedy nazývajú tiež „post-relačné systémy“) - autori relačného SRBD Ingres vyvinuli na jeho základe SRBD Postgres, ako iné príklady slúžia systémy Starbust alebo UniSQL.

### Typy architektúr DBS

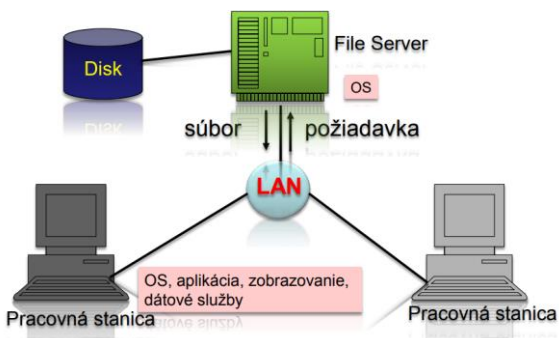
- Centralizované
- Klient-server
- Paralelné
- Distribuované

### CENTRALIZOVANÉ

#### Centrálny počítač



#### Súborový server

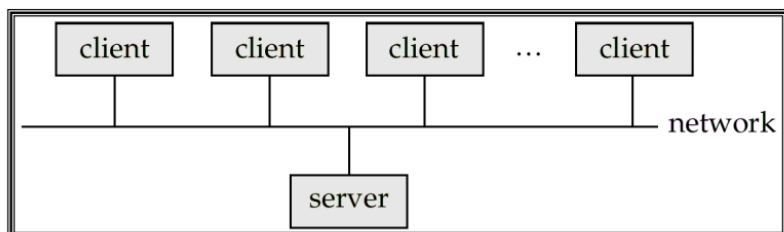


### Centralizované

- Implementované na jednom počítači
- Jedno používateľské systémy (napr., osobný počítač alebo pracovná stanica).
- Viacpoužívateľský systém – využíva viacero užívateľov prostredníctvom terminálov.

### Všeobecná schéma klient-server

- Server obsluhuje požiadavky generované klientami :



### Delenie funkcionality

- Databázovú funkcionality môžeme rozdeliť na: – **Back-end**: riadi prístup k údajom, vyhodnocuje a optimalizuje dopyty, riadi súbežné spracovanie a zotavenie z chýb.

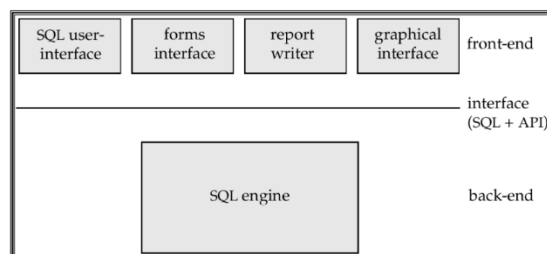
– **Front-end**: tvoria ho nástroje ako forms, report-writers a zariadenia grafického užívateľského rozhrania.

- Rozhranie medzi front-endom a back-endom je realizované prostredníctvom SQL alebo aplikačného programového rozhrania.

### Delenie funkcionality

#### Výhody klient-server architektúry

- Výhodou náhrady sálového počítača sieťou pracovných staníc alebo osobných počítačov pripojených na back-end server:
  - Lepšia funkcionality za nižšie náklady
  - Pružnosť v alokácii zdrojov
  - Lepšie užívateľské rozhranie
  - Ľahšia údržba



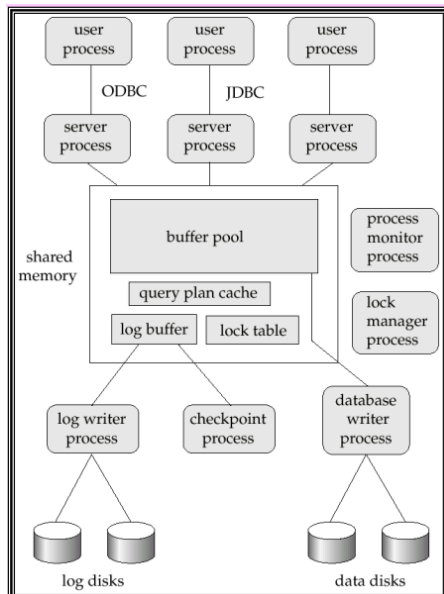
### Kategórie serverov

- Transakčné servery často používané v relačných DBS
- dátové servery používané v objektovoorientovaných DBS

### Transakčné servery

- Tiež nazývané query server systémy alebo SQL server systémy; klienti pošlú dopyt na server, kde sa realizuje transakcia a výsledok je doručený naspäť klientovi.
- Dopyty sú špecifikované v SQL a komunikované na server prostredníctvom mechanizmu vzdialeného volania (remote procedure call (RPC)).
- Open Database Connectivity (ODBC) je štandardizované rozhranie pre aplikačné programy v jazyku C od Microsoftu pre pripojenie sa na server, poslanie SQL dopytu a získanie výsledkov.
- JDBC je podobný štandard ako ODBC pre jazyk Java

### Transaction System Processes (Cont.)

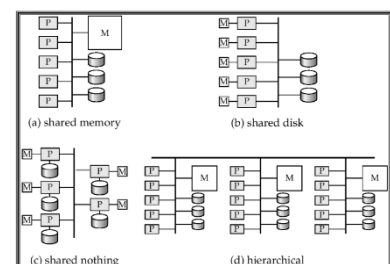


## Dátové servery

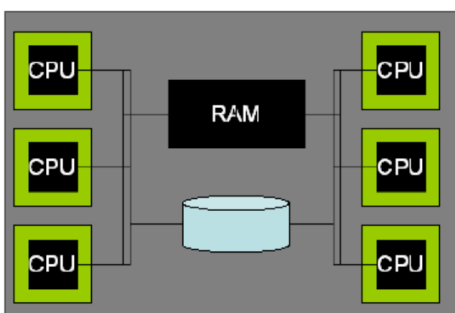
- Používaný v LAN, kde je vysokorýchlostné prepojenie medzi klientom a serverom, klientske počítače sú výkonnostne porovnateľné so serverom
- Údaje sa spracujú na klientovi a výsledok je poslaný späť na server.
- Táto architektúra vyžaduje mať na klientovi plnú **funkcionalitu back-endu**.
- Používané vo **viacerých objektovoorientovaných databázových systémoch**

## Paralelné architektúry

- Viacero procesorov a diskov zabezpečuje činnosť databázového systému.
- Procesory sú umiestnené fyzicky pohromade
- Dva hlavné parametre výkonu:
  - **Priepustnosť** (throughput) --- počet úloh ukončených v danom časovom intervale
  - **Čas odozvy** (response time) --- časový interval realizácie úlohy
- **Zdieľaná pamäť** -- procesory zdieľajú spoločnú pamäť
- **Zdieľaný disk** -- procesory zdieľajú spoločný disk
- **Nezdieľaná architektúra** (klaster) -- procesory nezdieľajú ani disk ani pamäť
- **Hierarchická** -- hybrid z predchádzajúcich troch architektúr



## Architektúra so zdieľanou pamäťou



## Zdieľaná pamäť

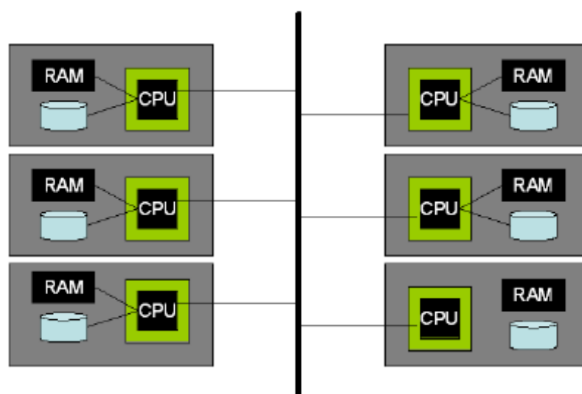
- Procesory a disky majú prístup k spoločnej pamäti, typicky prostredníctvom zbernice alebo sieťového prepojenia
- Efektívna komunikácia medzi procesormi — k údajom v pamäti môže pristupovať ľubovoľný procesor bez nutnosti ich presúvania prostredníctvom softvéru
- Často používaná architektúra **pre nízky stupeň paralelizmu** (4 až 8). architektúra so zdieľaným diskom

### Zdieľaný disk

- **Všetky procesory môžu priamo pristupovať na všetky disky prostredníctvom prepojenia**, ale každý procesor má vlastnú pamäť — Pamäťová zbernica nie je úzkym miestom
  - Takáto architektúra poskytuje určitú ochranu pred zlyhaním — ak zlyhá jeden procesor, jeho úlohu môže prebrať iný procesor, pretože databáza je uložená na diskoch prístupných pre všetky procesory

### DBS klaster

- Uzol tvorí procesor, pamäť a jeden alebo viacero diskov. Procesor z jedného uzla komunikuje s procesorom z iného uzla, využívajúc prepojenie.
- Uzol slúži ako server pre dáta na disku alebo diskoch v danom uzle.
- Na prístup k lokálnym údajom nie je využívaná sieťová infraštruktúra. Klaster preto môže obsahovať tisíce procesorov bez toho, aby si navzájom prekážali.



- Hlavná nevýhoda: náklady komunikácie a prístup k diskom iných uzlov.
- Posielanie údajov zahŕňa softvérovú interakciu na oboch koncoch DBS klaster

### Hierarchická architektúra

- Kombinácia architektúr klastrov so zdieľanou pamäťou, alebo diskom.
- Na najvyššej úrovni ide o klaster — prepojené uzly bez zdieľania pamäte či diskov. Každý uzol na nižšej úrovni môže tvoriť niekoľko procesorov so zdieľanou pamäťou.
- Alternatívne: uzly môžu zdieľať disky a v každom uzli na nižšej úrovni môže byť niekoľko procesorov so zdieľanou pamäťou.

### Distribúované DBS

- DB sa nachádza v rôznych miestach (tiež nazývaných uzly (sites či nodes)) spravovaná rôznymi procesormi, uzly sú navzájom prepojené.

### Základné pojmy

- distribuované spracovanie: aplikácia využíva DBS na inom procesore
- distribuovaná BD: množina BD spravovaná množinou CPU tak, že užívateľovi sa javí ako jedna veľká BD. Každá BD je spravovaná lokálne vlastným SRBD, tento využíva softvér na prístup k iným BD. Neexistuje centrálna koordinácia BD.
- distribuovaný DBS — zahŕňa distribuované spracovanie aj distribuovanú BD.

### Distribúovaný DBS

- distribuovaný databázový systém je **množina uzlov počítačovej siete**, navzájom prepojených v komunikačnej sieti pričom:
  - každý z uzlov je samostatný databázový systém, ale
  - tieto uzly navzájom spolupracujú tak, že z každého uzla je možné sprístupniť údaje uložené na inom uzle presne tak, akoby boli umiestnené na vlastnom uzle.
- je rozložený do uzlov siete prepojených komunikačnou infraštruktúrou
- v uzloch je možné lokálne uloženie a spracovanie dát (autonómne)
- prostriedky v uzloch môžu byť nehomogénne, ale rovnocenné (úlohy je možné prenášať z uzla do uzla)
- užívateľ nevie o existencii ostatných uzlov siete (transparentnosť)
- SRDBD podporuje spracovanie globálnych transakcií t.j. takých, ktoré vyžadujú pri spracovaní údaje z iných, alebo viacerých uzlov.
- **Homogénne** distribuované DBS
  - Ten istý softvér/schéma vo všetkých uzloch, údaje sú rozdelené medzi uzly
  - cieľ: poskytnúť užívateľovi dojem, že pracuje s jednou BD, skryjúc detaily uloženia údajov
- **Heterogénne** distribuované DBS
  - Rôzny softvér/schéma v rôznych uzloch
  - cieľ: integrovať existujúce DBS a poskytnúť užitočnú funkčnosť užívateľom

- udržiavanie identických kópií vo viacerých uzloch siete
- zabudovaný optimizér vyberie tú kópiu, pre ktorú sú minimalizované náklady alebo čas získania odpovede
- problémy s konzistenciou – všetky kópie musia byť stále identické

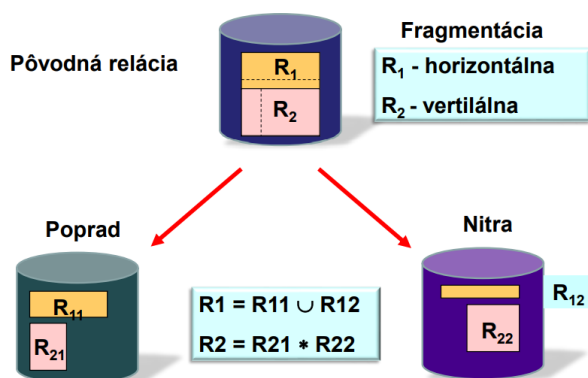
### Techniky udržiavania konzistencie

- Master/slave
- dvojfázové potvrdzovanie

### Fragmentácia

- horizontálna
- vertikálna
- Ich kombináciou je možné vytvoriť tzv. hybridnú fragmentáciu CVCVC

### Fragmentácie relácií



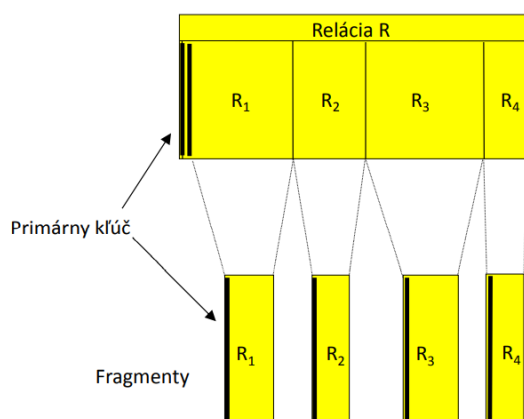
### Horizontálna fragmentácia

- relácia sa rozdelí do fragmentov tak, že  $n$ -tica môže patriť do viacerých fragmentov
- **výhoda** = redukcia nákladov na prenos, zvýšená bezpečnosť
- **nevýhoda** = archivácia

### Vertikálna fragmentácia

- dekompozícia relácie na projekcie
- ak je potrebné funkčné delenie relácie miesto geografického delenia

### Vertikálna fragmentácia relácie $R$ na fragmenty $R_1, R_2, R_3, R_4$



### Fragmentácie relácií

- Samotná fragmentácia sa vykoná podobne ako proces normalizácie relácií, ale okrem toho je nutné zabezpečiť tri podmienky:
  - Kompletnosť
  - Rekonštruovateľnosť
  - Nespojiteľnosť

### Kompletnosť

- Ak výskyt relácie je dekomponovaný na fragmenty, potom každá položka sa vyskytuje aspoň v jednom z fragmentov.
  - V prípade horizontálnej fragmentácie je položkou celá n-tica a v prípade vertikálnej fragmentácie atribút.
  - Táto vlastnosť je známa ako vlastnosť bezstratovej dekompozície čo zaručuje, že údaje sú z globálnej relácie transformované do fragmentov bez akejkoľvek straty.

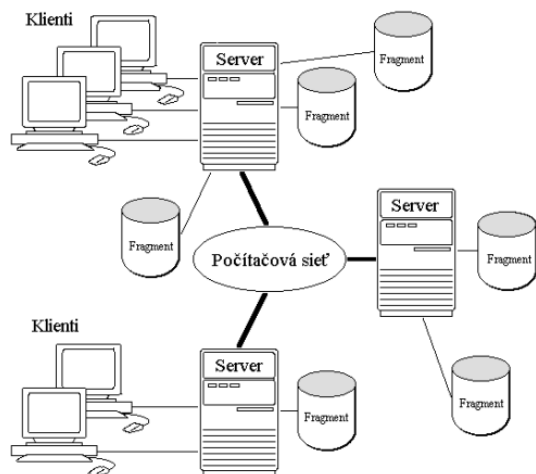
### Rekonštruovateľnosť

- Ak výskyt relácie je dekomponovaný na fragmenty, potom každá položka sa vyskytuje aspoň v jednom z fragmentov.
  - V prípade horizontálnej fragmentácie je položkou celá n-tica a v prípade vertikálnej fragmentácie atribút.
  - Táto vlastnosť je známa ako vlastnosť bezstratovej dekompozície čo zaručuje, že dáta sú z globálnej relácie transformované do fragmentov bez akejkoľvek straty.

### Nespojiteľnosť

- Ak relácia je horizontálne dekomponovaná na fragmenty a dátová položka patrí do fragmentu, potom sa nevyskytuje v žiadnom z fragmentov ( $k < j$ ).
- Toto kritérium zaručuje, že fragmenty relácie po horizontálnej fragmentácii sú nespojiteľné. V prípade vertikálnej fragmentácie, atribúty tvoriace primárny kľúč sa vyskytujú vo všetkých fragmentoch. Preto pri vertikálnej fragmentácii nespojiteľnosť sa týka atribútov, ktoré netvoria primárny kľúč relácie.

### Počítačová sieť s príkladom umiestnenia fragmentov



### Distribúovaný systém riadenia bázy dát

- hlavné funkcie sú:
  - Riadenie globálnych katalógov dát o distribuovaných dátach.
  - Definovanie distribuovaných dát.
  - Distribuovaná sémantická kontrola.
  - Distribuované spracovanie dotazov vrátane optimalizácie dotazov vrátane vzdialeného prístupu k dátam.
  - Distribuované riadenie transakcií vrátane zabezpečenia paralelného spracovania.

### 12 vlastností DDBS

• C.J. Date [ Date96] definoval 12 vlastností, ktoré by mal splňovať ideálny distribuovaný databázový systém, čo predstavuje akýsi cieľový stav výskumu a vývoja v tejto oblasti. Sú to nasledovné vlastnosti:

1. Lokálna autonómnosť.
2. Nezávislosť na centrálnom uzle.
3. Súvislá prevádzka.
4. Nezávislosť na umiestnení dát.
5. Nezávislosť na fragmentácii dát.
6. Nezávislosť na replikácii dát.
7. Spracovanie distribuovaných dotazov.
8. Distribuované riadenie transakcií.
9. Nezávislosť na technickom vybavení.
10. Nezávislosť na operačnom systéme.
11. Nezávislosť na sieti.
12. Nezávislosť na lokálnych databázových systémoch.

#### **Výhody DDBS**

- Lokálna autonómnosť,
- Zvýšenie výkonu,
- Zvýšenie spoľahlivosti,
- Zvýšenie dostupnosti dát,
- Vyššia zdieľateľnosť dát,
- Rozširowateľnosť,
- Ekonomická efektívnosť.

#### **Nevýhody DDBS**

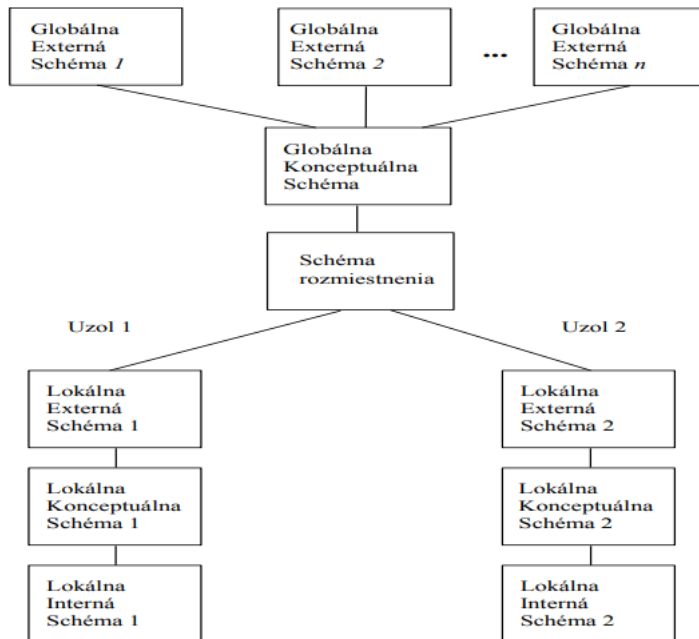
- Nedostatok skúseností,
- Zložitosť nového systému,
- Zvýšenie ceny,
- Riadenie distribuovaného spracovania,
- Vyššie nároky na bezpečnosť.

#### **Klasifikácia problémov**

- Návrh distribuovaných databáz.
- Distribuované spracovanie dopytov.
- Distribuované riadenie katalógov.
- Distribuované riadenie transakcií.
- Distribuované riadenie paralelizmu.
- Zabezpečenie spoľahlivosti.

#### **Architektúra databáz pre distribuované spracovanie**





## KLASIFIKÁCIA SRDBD

označenie	logicky	fyzicky
1	centralizované	centralizované
2	centralizované	decentralizované
3	decentralizované	centralizované
4	decentralizované	decentralizované

### ad1

- klasické SRBD

### ad 2

- fyzicky decentralizované systémy,
- klasické SRBD
- existuje globálna schéma
- **homogénne** -> v uzloch ten istý SRBD
- **heterogénne** -> rôzne SRBD
- **centrálne sa riadi:**
  - prístup do BD
  - zmena štruktúry
  - synchronizácia transakcií
- **výhoda** – ľahké riadenie (administrácia)
- **nevýhoda** – veľké komunikačné zaťaženie, pomalosť, slabá robustnosť

### ad 3 a 4

- logicky decentralizované systémy
- **výhoda:** - vysoký stupeň autonómnosti individuálnych BD, neprispôsobujú sa globálnej schéme
- netreba DBA pre koordináciu lokálnych schém, definujú sa len rozhrania medzi komponentami == vysoká odolnosť voči poruchám
- **nevýhoda:** zložité riadenie transakcií