Západočeská univerzita
FAKULTA APLIKOVANÝCH VĚD

Okruhy otázek ke státní závěrečné zkoušce z předmětu
Databázové technologie (DB)

Databázové systémy 1 (DB1)
Databázové systémy 2 (DB2)
Případové studie databázových systémů (PSDS)

Studijní program:	3902	Inženýrská informatika
Obor:	2612T025	Informatika a výpočetní technika – Softwarové inženýrství
	3902T031	Softwarové inženýrství

Akademický rok: 2005/2006

Obsah:

1 SŘBD, požadované vlastnosti

Systém řízení báze dat, anglicky DBMS – Data based management system

Souborově orientovaný přístup znamená, že každá aplikace má svá data a sama s nimi pracuje, což má své nevýhody.
Nutno naprogramovat základní přístup k datům, izolace dat, duplikace dat, závislost.
Využití pro nedatabázové úlohy pracují s velkým množstvím dat.

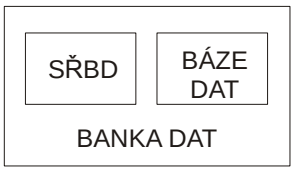
Zásady SŘBD

Data oddělena od aplikací
Definice dat provedena mimo aplikaci
V aplikaci není zabudováno řízení přístupu nebo manipulace s daty

Definice

Banka dat je organizovaná forma systému zpracování dat zahrnující bázi dat a systém řízení báze dat.
Databáze (báze dat) je množina souborů a popisu jejich dat, které jsou navzájem v určitém logickém vztahu a jsou spravovány systémem řízení báze dat.
Systém řízení báze dat (SŘBD) je programový systém, který v bázi dat zabezpečuje:

- Definování struktury dat
- Ukládání dat
- Výběr dat
- Ochranu dat
- Komunikace mezi uživatelem a systémem



Požadavky na databáze

Neredundantnost – jedna data jsou přístupná z jakékoliv aplikace SŘBD
Víceúrovňová využitelnost – možnost přístupu všem oprávněným uživatelům ke stejným datům
Integrita dat – hodnoty uložených dat nesmí být spolu ve sporu
Nezávislost dat – změna struktury dat nevyvolá změny v aplikačních programech
Implementace libovolného datového modelu

SŘBD

DDL (Data Definition Language) – uživatel specifikuje datové typy a struktury, obvykle má i možnost zadat omezující podmínky
DML (Data Manipulation Language) – je prostředkem pro správu a získávání dat
Dělí se na procedurální a neprocedurální jazyky, kde:

Procedurální jazyk zpracovává data záznam po záznamu
Neprocedurální jazyk pracuje s množinou záznamů a určuje jak data vybrat - SQL

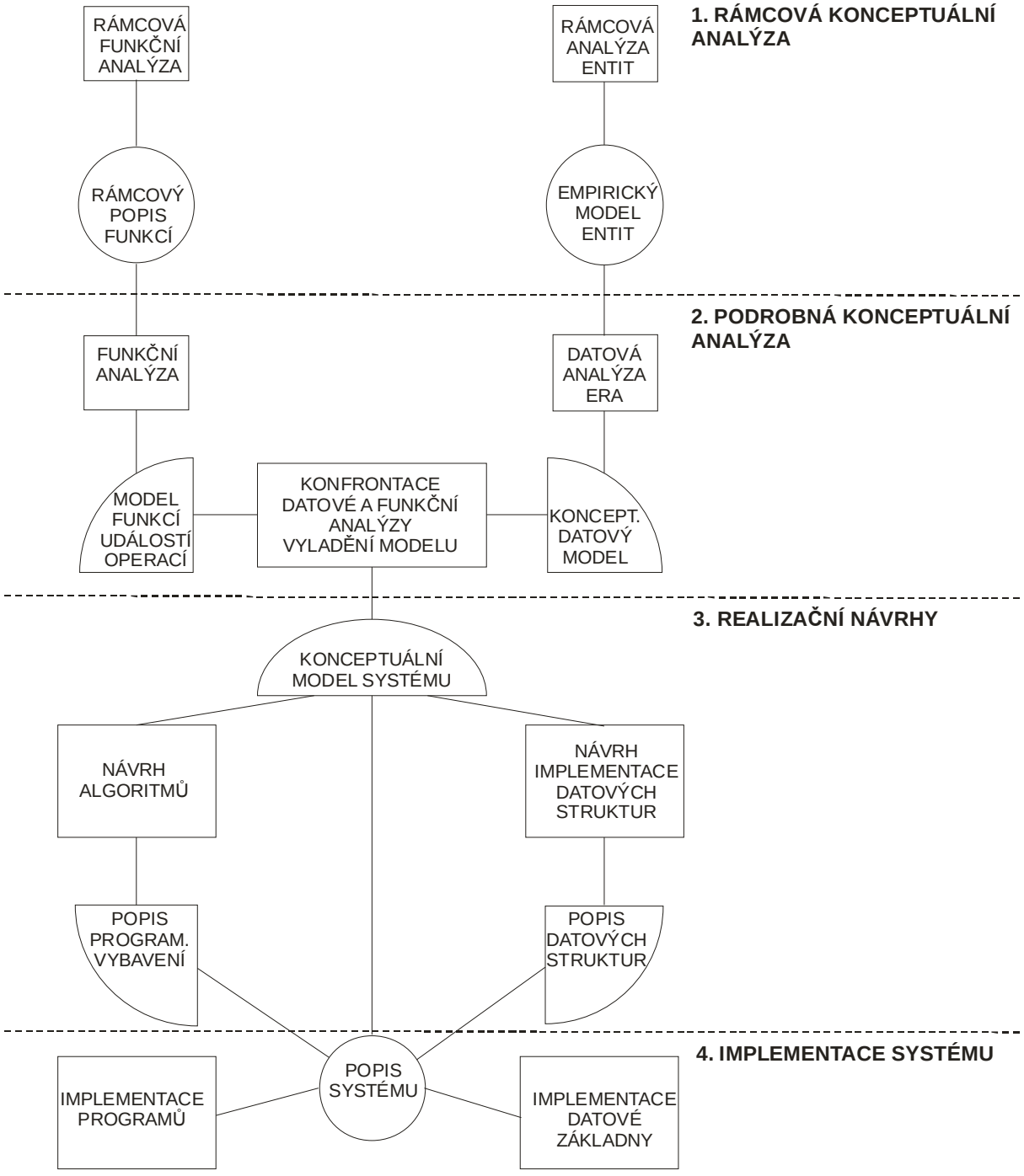
Současný přístup k datům – současný přístup více uživateli ve stejný okamžik ke stejným datům
Sdílení přístupu k datům – SŘBD přiděluje práva a role pro přístup ke společným datům
Bezpečnost systému – přístupová práva, obnova systému po chybě
Katalog dat – popis dat přístupný uživateli

Databázové okolí

- Administrátor databáze – komunikace systém x uživatel
- Administrátor dat – umí interpretovat data, zná jejich význam
- Vývojový programátor databáze – provedení modelu
- Aplikační programátor – užívá dotazovací jazyk
- Koncový uživatel – je odstíněn od práce s daty

2 Konceptuální modelování

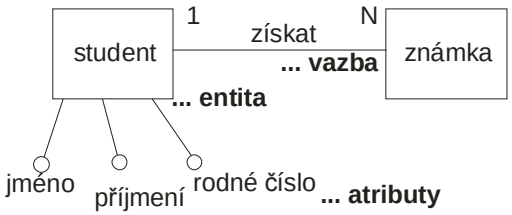
Datové modelování bez ohledu na následující realizaci



3 E-R-A modely

Entita – model z reálného světa
Relation – podchycuje vztahy mezi entitami
ERA modely = modelová analýza

Datový model	Datová struktura	Souborový přístup
Entitní množina	Tabulka	Soubor
Entita	Řádka	Záznam
Atribut	Název sloupce	Položka



Vazby

1:N - vyjadřuje, že jedné entitě E1 může příslušet více entit z entitní množiny E2
jedné entitě z E2 přísluší jen jedna entita z E1
nejlépe 1:N(0)
př.: student – známka

1:1 - na vazbě se podílí jej jedna entita z E1 a jedna z E2
vždy se ptát, proč je to rozdělené, proč to není jedna entita
vazba je zajímavá, pokud alespoň jeden konec volný (nepovinný)
př.: student – známka (student nemusí mít známku)

N:N - jedné entitě z E1 přísluší více entit z E2
jedné entitě z E2 přísluší více entit z E1
př.: student – rozvrhová akce

ER modely

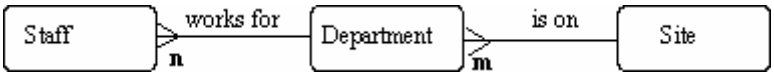
primární – minimální množina atributů, která jednozačně určuje entitu
na vazbu se díváme jako na entitu – lze k ní přidat atributy (čtenář – výpůjčka – exemplář)
vazba 1:N se při realizaci vytvoří tak, že do "podřízené" tabulky přenesu klíč (cizí klíč) z "nadřazené" tabulky
vazba M:N nelze realizovat, nelze vyřešit pomocí cizích klíčů = musí se provést rozklad vazby
mezi dvěmi entitními množinami může existovat více vazeb
vazba nemusí být binární, ale může být n-ární
vazba může být i unární (sama na sebe)

Některé datové modely rozlišují entitní množiny regulární a slabé
Slabá entitní množina je taková, u níž nelze určit, či nemůžeme zjistit nadřazenou množinu

FAN

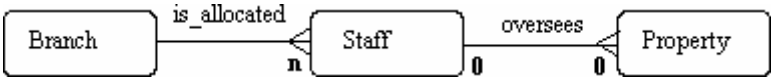
Problém: 2 vazby 1:N se větví z jedné entity
datové položky ve dvou složkách nejsou v přímém vztahu, ale mají vazbu založenou na datových položkách ve třetí složce

Řešení:

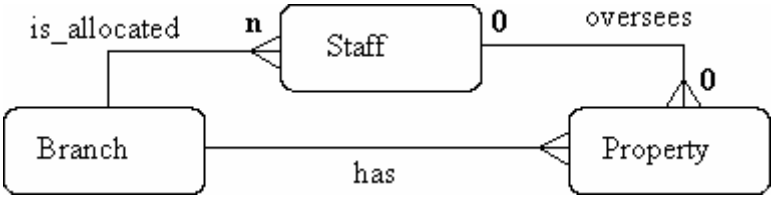


CHASM trap

Problém: Existuje vztah mezi entitami, ale chybí vazba.



Řešení:



4 Závislost atributů, normální formy

Název relace: výuka R(přednáška, učitel, místnost, hodina, student, známka) Relace je navržena nevhodně, protože obsahuje duplicity. Přednáška, učitel, místnost, hodina se musí opakovat tolikrát, kolik studentů přednášku navštěvuje.

Řešení tohoto problému se nazývá normální formy. Při zjišťování normálních forem (NF) nebo při převodu do NF se musíme zabývat tzv. **závislostí atributů**.

Vymezení: Populací relace R budeme rozumět hodnoty atributů relace v daném čase (konkrétní naplnění).

Funkční závislostí rozumíme vztah mezi atributy jedné relace.

Definice

Nechť A a B jsou atributy relace R, budeme říkat, že atribut B funkčně závisí na atributu A, jestliže pro všechny populace relace R platí pro libovolnou n-tici $u,v \in R$:
$u.A = v.A \Rightarrow u.B = v.B$, označení $A \rightarrow B$. Ke každé hodnotě atributu A existuje nejvýše jedna hodnota atributu B.

Příklad: vyuka(P,U,M,H,S,Z)
P->U (jestliže předmět učí jeden učitel)
HM->P
HU->M
PS->Z
HS->M.

Z jedné populace nelze dokázat, že nějaká funkční závislost platí, ale lze dokázat, že neplatí. Funkční závislosti zjistíme z analýzy úlohy. Pomocí závislostí lze definovat **klíč**.

Je-li dáno R(Ω) kde Ω je množina atributů a K, K $\subset$ Ω, pak K je klíčem schémat R, jestliže splňuje obě vlastnosti:

- $K \rightarrow \Omega$
- neexistují $K', K' \subset K, K' \rightarrow \Omega$

Význam definice

Na klíči jsou závislé všechny ostatní atributy relace.

Pravidla

Dávají podklad k postupu, jak z daných odhalených závislostí odvodit další závislosti.
Armstrongova pravidla jsou korektní, úplná a nezávislá.

1. **Reflexivnost (triviální funkční závislost) –** nechť je zadána množina $X, Y \subseteq X$, $x \rightarrow X$ nebo XY (množina definuje svoji libovolnou podmnožinu).

2. **Tranzitivita –** $X \rightarrow Y \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$

3. **Spojování (kompozice) –** $X \rightarrow Y \wedge X \rightarrow Z \Rightarrow X \rightarrow YZ$

4. **Projekce (dekompozice) –** $X \to YZ \Rightarrow X \to Y \wedge X \to Z$

   4.1. **Doplnitelnost –** $X \to Y \wedge X \in W \Rightarrow W \to Y$
   4.2. **Jiná varianta spojování –** $X \to Y \wedge W \to Z \Rightarrow XW \to YZ$

Normální formy

1.normální forma (NF)
Řekneme, že relace se nachází v 1.NF, pokud všechny komponenty n-tice jsou atomické (tj. nejsou opět relací).

2. normální forma (NF)
Relace R je v 2.NF, pokud je v 1.NF a pokud každý atribut, který nepatří k žádnému klíči relace R silně závisí na klíči relace R.

Silná funkční závislost: Nechť A->B, říkáme, že atribut B silně závisí na A, jestliže neexistuje žádná vlastní podmnožina A' složeného atributu A taková, že platí A' -> B.

Pokud hledám NF, tak 2.NF vyžaduje nalezení klíče a zabývání se závislostmi. Pokud je klíč jednoduchý atribut, je tabulka automaticky v 2.NF.

3. normální forma (NF)
Relace je v 3.NF, pokud je ve 2.NF a žádný atribut, který není složkou klíče relace X, není tranzitivně závislý na klíči relace R.

Boyce-Codova normální forma
Relace R se nachází v BCNF, jestliže pro každou funkční závislost tvaru X->Y, kde Y není v X, platí, že X je nadmnožinou nějakého klíče X v R klíče. Jestliže je v 3.NF nemusí být v BCNF. Zkoumá atributy, ať jsou v klíči nebo nejsou v klíči (rozdíl v 3.NF)
Řešení: vazbu, která kazí BCNF dám do samostatné tabulky.

4. normální forma
Multizávislost – tak jako ve funkční závislosti odpovídá dané hodnotě atributu 1 hodnota jiného atributu u multizávislosti odpovídá 1 hodnotě množina hodnot.

Multizávislost (množina hodnot závisejících na 1 hodnotě jiného atributu) nám vadí jedině v kontextu na jiné atributy relace.

Definice – označme nejprve Y(x) množinu Y-hodnot přiřazených v daný moment k Y-hodnotě x. Nechť A,B,C jsou podmnožiny O (O – atributy relace) takové, že C = O – A – B. Pak B multizávisí na A, jestliže pro každou AC-hodnotu ac je B(ac)=B(a). Daný fakt či tvrzení nazýváme multizávislost B na A a označujeme A ->> B. Je-li C prázdná množina, nazývá se multizávislost A->>B triviální.

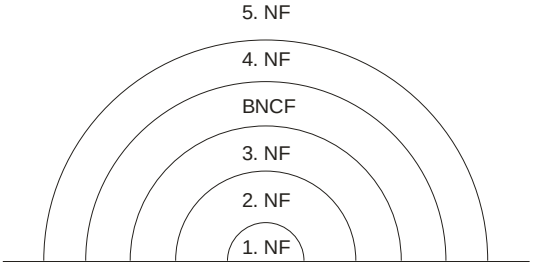
Definice 4.NF – říkáme, že schéma relace R je ve 4.NF, jestliže pro každou netriviální multizávislost X->>platí, že X je nadmnožinou nějakého klíče schématu R.

Věta – mějme schéma R(A,B,C), kde A,B,C jsou množiny atributů a funkční závislost B->C. Rozložíme-li R na schémata R1(B,C) a R2(A,B), takto provedená dekompozice je bezztrátová.

Bezztrátovost – je-li dána relace R a její projekce na atributy BC a AB, pak spojením těchto projekcí získáme zpátky relaci R.

5. normální forma (NF)
Vychází z principu, že relaci můžeme dekomponovat na více tabulek a samozřejmě dekompozice musí být bezztrátová. Relace se nachází v 5.NF, pokud neobsahuje závislost podle spojení. Pokud tam závislost je, měli bychom ji rozložit.
Poznámky k normálním formám

V praxi se zabývat návrhem do 3.NF. NF ano, ale přece jen zvážit, zda důslednost není přehnaná (v žádném případě nechceme velké množství 2 sloupečkových tabulek) – zpomalení práce s databází. Pokud bychom v návrhu vyrobili tabulku, která porušuje 3.NF, přehlédli jsme v návrhu nějakou vazbu.Převod do 2.(3.NF) znamená v ERA modelu přidat entitní množinu připojenou vazbou 1:N.



5 Relační model dat, relační algebra

3 operace pro práci s daty

Projekce relace R s atributem A na množinu atributů B, kde $B \subseteq A$

Operace vytvoří relaci se schématem B a prvky, které vzniknou z původní relace odstraněním hodnot atributů A\B. Odstraněny jsou i případné duplicitní prvky (řádky)

~~a~~	b	c	~~d~~	Student (Jméno, Příjmení, …)
~~b~~	b	c	~~d~~	S = Student [město]

Selekce relace s atributy A podle logické podmínky Φ

Operace vytvoří relaci s týmž schematem a ponechá prvky z původní relace, které splňují podmínku Φ. Formule Φ je Booleovský výraz, jehož atomické formule mají tvar $t_1 \Theta t_2$, kde $\Theta \in \{<, \leq, =, \geq, >\}$, tj je buď konstanta, nebo jméno atributu.

Značení $R(\Phi)$ S=Student (město=’Plzeň’)

Spojení relací R a S se schématy (atributy) A, resp. B

Operace vytvoří relaci se schématem $A \cup B$, jejíž projekce na A je relace R a projekce na B je relace S.

Značení $R \circ S$

Spojení $R[A \otimes B]S$

Z Kratézského součinu jsou vybrány pouze ty n-tice, které splňují Θ podmínku

- Θ **spojení** vytvoří „poměrně značný počet prvků“, což jsou všechny prvky z R a S, pro které platí spojení
  Speciálním typem Θ spojení je spojení přes rovnost
- **přímé spojení** přes rovnost, kdy se jeden z těchto dvou atributů vypustí
  obvykle, když provádíme spojení, tak relace R a S mají stejné atributy
- **kompozice** je spojení, kde se vypustí oba atributy, kterými se srovnává

Některé operace lze vyjádřit jinými operacemi a může se nazývat minimální množinou operací.

Příklad

Exemplář (ISBN, INV. Č., DATUM NÁKUPU, CENA, ZEMĚ VYDÁNÍ)
Výpůjčka (INV. Č., Č. ČTENÁŘE, DATUM VRÁCENÍ)

Selekce
Exemplář (ZEMĚ VYDÁNÍ=’GB’)

Projekce
T[ISBN, INV. Č.]

Spojení
Výsledkem v relační algebře není nikdy číslo, ale přinejmenším tabulka s jedním sloupcem a jedním řádkem
Nalezení ISBN všech vypůjčených knih:
[Exemplář (ZEMĚ VYDÁNÍ=’GB’)] [ISBN, INV. Č.]] * Výpůjčka [ISBN]

6 Síťový model dat

Jiná varianta pohledu na data. Začátek databázových systémů (70. léta).

Základní pojmy

Typ záznamu
popíšeme položky, z jakých se seznam skládá
u typu záznamu určujeme klíč

Spojka
ukazatel na záznam logicky související
spojka definuje spojení mezi 2 typy záznamu
s informací, bez informace

DBTG CODALSYL

(1971 – DataBase Task Group of Conference On Data Systems Languages) – z této skupiny vypadla forma síťového modelu dat $\rightarrow$ COBOL

Síťový datový model (dle DBTG CODALSYS) je množina typů záznamů definovaných takto:

1) existuje množina typů záznamů
2) existuje množina pojmenovaných spojek spojujících typy záznamů v diagramu datové struktury
3) každá spojka je funkcí alespoň v jednom směru
4) spojka L_{ii} není přípustná

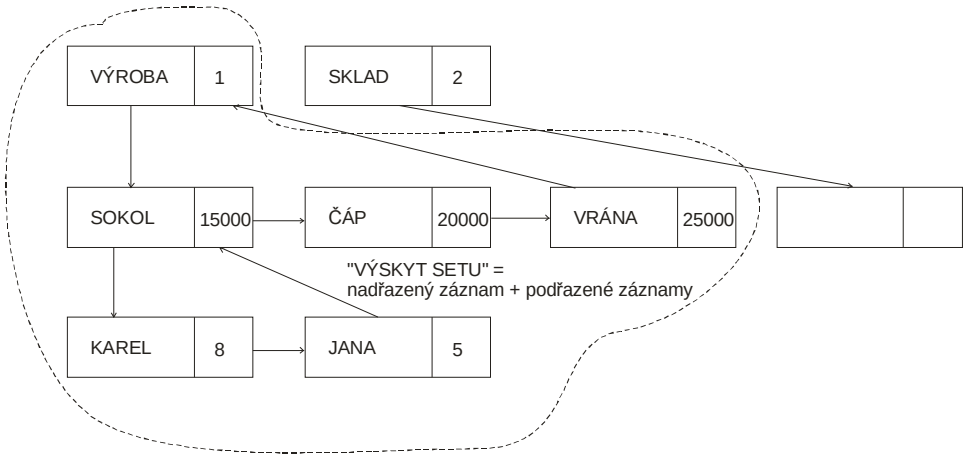
Vymezení
Mějme zobrazení n typu N:1 ze záznamů R2 do R1, každému záznamu r typu R1 můžeme přiřadit množinu S2 záznamů typu R2 takových, že m(s)=n. Je-li m funkcí, pak množiny S a Sr2 jsou disjunktní, jestliže r1 se nerovná r2. Nechť X je jméno setu reprezentující zobrazení m, potom každá množina Sr splu se záznamem r se nazývá **výskyt setu**. Záznam r se nazývá vlastník setu X a každé s, pro které platí m(s)=r se nazývá **členem výskytu setu** (member).

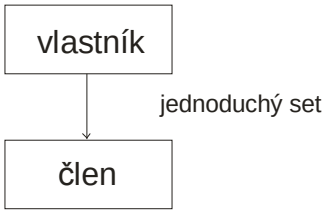
Typ záznamu R1 se nazývá typ vlastníka setu X. Typ záznamu R2 se nazývá typ člena setu X. Typy záznamů R1 a R2 musí být různé.

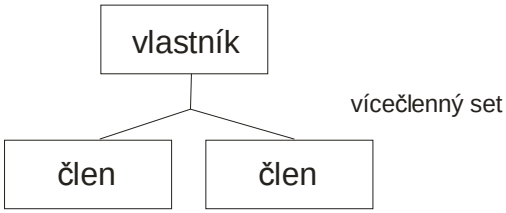
Ani v síťovém modelu neumíme realizovat vazbu M:N, musíme provést rozklad.

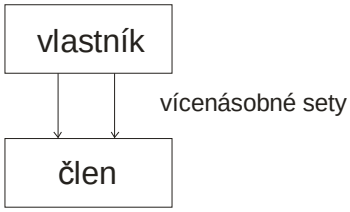
Typ členství

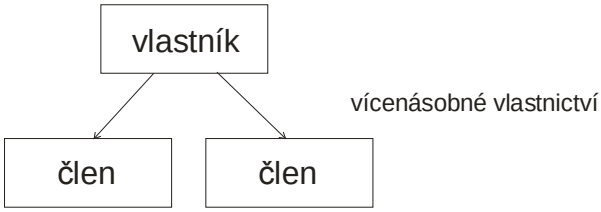
manuální – přidaný záznam se stane členem setu až po příkazu v jazyce
automatické – záznam se stane členem setu automaticky hned při výskytu
volitelné – nemusí být členem setu
povinné – musí být členem setu

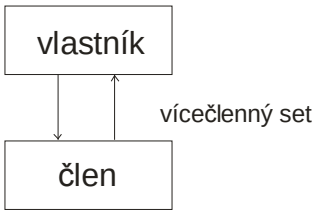

VÝROBA
1
SKLAD
2
SOKOL
15000
ČÁP
20000
VRÁNA
25000
"VÝSKYT SETU" =
nadřazený záznam + podřazené záznamy
KAREL
8
JANA
5

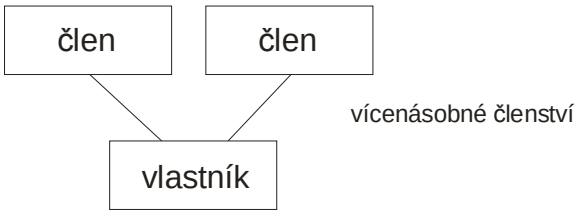

vlastník
jednoduchý set
člen


vlastník
vícečlenný set
člen
člen


vlastník
vícenásobné sety
člen


vlastník
vícenásobné vlastnictví
člen
člen


vlastník
vícečlenný set
člen


člen
člen
vícenásobné členství
vlastník

7 Základy SQL, definování datových struktur v SQL, příkaz SELECT

ANSI SQL 89, atd. – norma definuje, co mají umět jazyky pracující s SQL – Structured Query Language

- DDL – Data Definition Language
- DCL – Data Control Language (někdy i TCC – Transaction Control Commands)
- DML – Data Manipulation Language

Předepisujeme, jaká data chceme dostat. Jdná se o **neproceduráln**í jazyk,

Užití: Admin (DBA), Management, Vývojář, Uživatel. Nutnou podmínkou je znalost datové struktury.

DDL

Těmito příkazy **vytváříme struktury databáze** – tabulky, indexy, pohledy a další objekty. Vytvořené struktury můžeme upravovat, doplňovat a mazat.

CREATE – vytváření nových objektů.
ALTER – změny existujících objektů.
DROP – odstraňování objektů.

DCL

Do této skupiny patří příkazy pro nastavování **přístupových práv a řízení transakcí**.

GRANT – příkaz pro přidělení oprávnění uživateli k určitým objektům.
REVOKE – příkaz pro odnětí práv uživateli.
BEGIN – zahájení transakce.
COMMIT – potvrzení transakce.
ROLLBACK – zrušení transakce, návrat do původního stavu.

DML

Jsou to příkazy **pro získání da**t z databáze **a** pro jejich **úpravy**.

SELECT – vybírá data z databáze, umožňuje výběr podmnožiny a řazení dat.
INSERT – vkládá do databáze nová data.
UPDATE – opravuje data v databázi (editace).
DELETE – odstraňuje data (záznamy) z databáze.
EXPLAIN PLAN FOR – speciální příkaz, který zobrazuje postup zpracování SQL příkazu. Pomáhá optimalizovat příkazy tak, aby byly rychlejší.

SELECT

Nejpoužívanější příkaz
SELECT [DISTINCT | ALL] {* | [sloupcový_výraz [AS nový_název]] [,…]} **FROM** název_tabulky [alias] [,…]
[**WHERE** podmínka]
[**GROUP BY** seznam_sloupců] [**HAVING** podmínka]
[**ORDER BY** seznam_sloupců]

Selekce

SELECT isbn, inv_č, d_nakupu, cena, zeme_vydani FROM exemplar;
SELECT * FROM exemplar;

* = všechny

Projekce

SELECT zeme_vydani FROM exemplar;
SELECT DISTINCT zeme_vydani FROM exemplar;

DISTINCT = odstranění duplicitních řádků

Vypočítaná položka

SELECT cena/1.05 AS bez_dph FROM exemplar;

Podmínka

SELECT * FROM exemplar WHERE zeme_vydani='GB';

Hodnoty, datumy a texty se dávají do apostrofů. Čísla nikoliv.
V podmínkách lze využít klasické logické operátory =, !=, <>, <, <=, >, >=, AND, OR, NOT

Výčet hodnot

SELECT * FROM exemplar WHERE zeme_vydani IN ('GB','USA');

Interval

SELECT * FROM exemplar WHERE cena BETWEEN 300 AND 900;

Výběr podle vzoru

SELECT * FROM exemplar WHERE zeme_vydani LIKE 'G%';

… LIKE 'G%'; – libovolný řetězec začínající na G
… LIKE 'G_'; – G a 1 povinný znak
… LIKE 'U_ _; – U a 2 povinné znaky
… LIKE '%A'; – končí na A
… LIKE '%A%'; – klidně i jen A

… NOT LIKE 'G%'; – nezačíná na G

Test prázdné hodnoty

SELECT * FROM exemplar WHERE zeme_vydani IS NULL;

… IS NOT NULL; – zeme_vydani je vyplnena

Řazení

SELECT * FROM exemplar ORDER BY cena;

…ORDER BY cena, zeme_vydani DESC; – sestupně

Agregované funkce

COUNT – poče hodnot ve sloupci (počet řádků)

SUM – součet hodnot ve sloupci
AVG – průměr hodnot ve sloupci
MIN
MAX

Vnořený select

SELECT cislo_ctenare FROM ctenar WHERE adresa LIKE ‘%Plzeň’ AND cislo_ctenare IN (SELECT cislo_ctenare FROM vypujcka);

… čtenáři bydlící v Plzni s půjčenou knihou

SELECT jmeno FROM ctenar WHERE cislo_ctenare IN (SELECT cislo_ctenare FROM rezervace WHERE isbn=(SELECT isbn FROM kniha WHERE titul=’Babička’));

… jména čtenářů s rezervovanou knihou Babička – právě jeden záznam díky podmínce na ISBN

Grupování

SELECT COUNT(inv_cislo) SUM(cena) AS celkem.zeme_vydani FROM exemplar;

Agregované funkce ve vnořeném selectu

Sjednocení

UNION – sjednocení dvou tabulek se stejnými sloupci

SELECT cislo_ctenare FROM vypujcka WHERE datum_vraceni <= ’31.12.2003’
UNION
SELECT cislo_ctenare FROM rezervace WHERE datum_rezervace <= ’31.12.2003’

Průnik

INTERSECT

Rozdíl

EXCEPT

LEFT, RIGHT JOIN a FULL JOIN

SELECT a.* b.* FROM ctenar a LEFT JOIN vypujcka b ON a.cislo_ctenare=b.cislo_ctenare;

LEFT JOIN poskytne nejen ty čtenáře, ke kterým byla nalezena výpujčka a to každého tolikrát, kolik výpujček mají, ale i ty čtenáře, kteří výpujčku nemají. Hodnoty z výpujčky jsou null.
Obecně LEFT JOIN připojí i ty řádky, které z levé (první) tabulky nemají v pravé (druhé) relaci (tabulce) odpovídající prvek (řádku).

RIGHT JOIN – připojuje neporovnané řádky z pravé tabulky

FULL JOIN – připojuje neporovnané řádky jak z první tak druhé tabulky

EXIST, NOT EXIST – test existence

```
SELECT jmeno FROM ctenar WHERE EXISTS (SELECT * FROM rezervace WHERE
cislo_ctenare=ctenar.cislo_ctenare);
```

… najdi jména čtenářů, kteří mají rezervovanou nějakou knihu

8 Transakce, žurnál, konzistence databáze

Vyjadřuje stav databáze. Stav, který by mohl odpovídat reálné situaci.
Na zajištění konzistentního stavu databáze slouží integritní omezení.

Selhání

- Fyzické zařízení
- Operační systém
- Chybný zásah obsluhy
- Chybná data

Ochrana

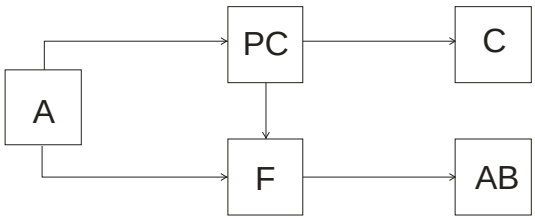
Prostředky OS
Tvorba kopií, základní prostředek ochrany
Záloha = uložení konzistentního stavu databáze, kontrolní bod – **checkpoint**

Prostředky SŘBD – transakce – posloupnost operací nad objekty databáze, která realizuje jednu ucelenou operaci z pohledu uživatele

Transakce

Říkáme, že je logickou jednotkou práce a je také jednotkou zotavení z chyb, příklad: převod peněz z účtu na účet – dva zápisy do databáze
Během provádění transakce není databáze v konzistentním stavu

Transakce se může dostat do jednoho z pěti + jednoho stavů
A = aktivní – vznik transakce, provádění
F = chybný – nelze dál pokračovat
C = částečně potvrzený – provedení poslední elementární transakce – transakce nebyla potvrzena
C = potvrzení – **commit**
AB = zrušení – objekty se uvedou do stavu před započetím transakce
PC $\rightarrow$ F = částečně potvrzená transakce se také může dostat do chybného stavu



Pořizuje se kopie databáze v kontrolním bodu
Od kontrolního bodu se zapisují změny do žurnálu
žurnál = tabulka – transakce, objekt transakce
kdo spustil transakci
kdy spustil transakci
nová hodnota objektu
stará hodnota objektu

ID transakce je obdoba ID procesu

Použití žurnálu

REDO – po chybě se vezme konzistentní stav DB z checkpointu, beru žurnál krok za krokem a provádím transakce

ROLLBACK, UNDO – zpětné použití žurnálu, vychází se ze současného stavu DB, využívá se žurnálu – zpětným odvíjením se dostanu až do konzistentního stavu, význam pouze pro nedokončené transakce

Dvoufázové potvrzování

Tvorba žurnálu s odloženými realizacemi změn

Zásady

- Transakce nesmí měnit hodnoty objektů dříve než je částečně potvrzena
- Transakce může být částečně potvrzena, až když je kompletní záznam transakce v žurnálovém souboru

Přímý zápis do báze dat

Tvorba žurnálu s bezprostředním zapsáním změn
Do žurnálu zapisuje staré hodnoty
Po obnově se provádí rollback

1. způsob – uzamčené objekty po celou dobu transakce – nevyžaduje ukládání starých hodnot

2. způsob – umožňuje otevření po zapsání – lepší pro paralelní běhy – vyžaduje ukládat staré hodnoty
nevýhoda – při vycouvání musí vycouvat i všechny transakce, které s objekty dále pracovaly (kaskádové rušení transakcí)

9 Paralelní zpracování transakcí, zamykání záznamů, časové značky

SŘBD obsahuje 3 moduly pro paralelní zpracování:

Modul řízení transakcí (RT) – na který se transakce obracejí s žádostí o provedení operace

Modul řízení dat (RD) – v databázi vykonává čtení, zápis podle požadavků plánovače

Plánovač – zabezpečuje synchronizaci požadavků více transakcí (z modulu RT), požadavky zpracovává do tzv. plánů

LOCK (X)

U paralelního běhu se hlídá uspořádanost – aby výsledek dopadl stejně jako při sériovém uspořádání. Chyba nastane, když transakce nastanou navzájem – transakce se musí řídit.

UNLOCK (X)

Objekt smí být zamčen pouze jednou transakcí, a pokud jiná transakce chce číst hodnotu x, tak je pozastavena.

Musí být proveden transakcí, která hodnotu x zamkla.

Lze dokázat, že uspořadatelnost je zaručena

1) Objekt může být v jednom okamžiku uzamčen pouze pro jednu transakci.
2) Jakmile transakce odemkne alespoň jeden objekt, nesmí již žádný jiný objekt zamknout.

Výlučný zámek x sdílený zámek. Sdílený zámek dovolí číst. Jen některé SŘBD.

Jiné než dvoufázové uzamykací protokoly

Mohou být jiné, ale vyžaduje to další znalost o databázích.
V případě stromu – hierarchická struktura (strom nemusí být vyvážený).
Hierarchické databáze – závislost mezi entitami vytváří strom.

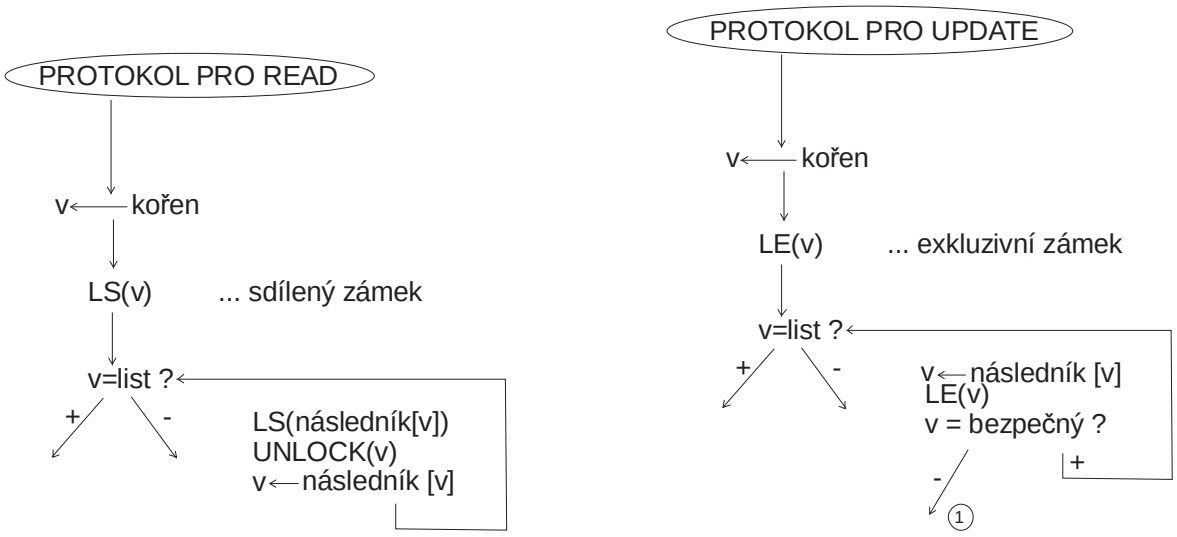
Stromový protokol pro hierarchickou databázi

1. Výlučný zámek transakce T lze použít na jakýkoliv objekt X.
2. Další objekt může být uzamčen transakcí T, pokud byl v T uzamčen jeho předchůdce.
3. Objekty mohou být odemknuty kdykoliv.
4. Objekt, který byl transakcí T zamknut a odemknut, nesmí být znovu zamknut transakcí T.

Lze dokázat, že tento protokol zaručí, že všechny rozvrhy jsou uspořádatelné a navíc nemůže dojít k uvíznutí.

Uzamykací protokol pro databázi typu B-stromu

Uzel, který při operaci INSERT nebo operaci DELETE nezpůsobí zásah do nadřízeného uzlu nebo sousedů, se nazývá bezpečný.



Porovnání stromových a dvoufázových protokolů

Pokud chci hodnotit protokol, tak sleduji, kolik objektů zamkne a na jak dlouho je zamkne. Stromové protokoly zamykají více objektů, ale na kratší dobu. U stromových protokolů nedojde k uvíznutí.

Časové značky
sledujeme časovou značku u transakce, což je okamžik, kdy transakce vznikla
u každého objektu sledujeme časovou značku posledního čtení (R/ČZ) a časovou značku posledního zápisu (W/ČZ)

Princip: Transakce vznikne v určitém okamžiku a nemůže pracovat s objekty, jejichž hodnoty se změnily v "budoucnosti".

10 Integritní omezení

Vymezují hodnoty objektů databáze tak, aby mohly mít v reálném světě smysl.

Tři typy integritních omezení

- entitní (co musí splňovat entita)
- doménové (omezení atributů)
- referenční

Entitní integrita

Požadavek na jednoznačnou identifikaci řádky v tabulce:
definicí **primárního klíče** (jedinečný, vyplněn)

Doménová integrita

Omezuje hodnoty položek (intervalem, výčtem)
V normě SQL je příkaz CREATE DOMAIN

Referenční integrita

Popisuje, jaké vztahy platí mezi tabulkami
Obvykle se zavádí cizí klíč

Způsoby zajištění integrity

Restriktivní
Nedovolí zrušit řádku nadřízené tabulce, pokud k ní existují řádky v podřízené tabulce
Rovněž nepřipustí změnu hodnoty klíče

Kaskádovitý
Zruší záznam a zruší všechny jeho podřízené záznamy
Povolí přepsat klíč a všude povolí přepsat podřízené klíče

Dosazení prázdné hodnoty

Historie v SQL

ANSI SQL 86
NOT NULL
UNIQUE

ANSI SQL 89
PRIMARY KEY
CHECK – CREATE TABLE kredit (počet kreditů INTEGER CHECK BETWEEN 1 AND 6);
REFERENCES a FOREIGN KEY – CREATE TABLE predmet (... FOREIGN KEY (garant) REFERENCES ucitel(cislo_ucitel));

ANSI SQL 92
Definice cizího klíče doplněna o
ON UPDATE CASCADE
ON DELETE CASCADE

ON DELETE NULL
ON UPDATE NULL
ALTER TABLE ADD CONSTRAINT
ALTER TABLE DROP CONSTRAINT

Lze provést změnu v definici tabulky omezujících podmínek – SET CONSTRAINT OFF (ON)

Integritní omezení na úrovni definice dat lze i pomocí napsaných "programů" – uložené procedury, triggery.

Uložená procedura – program uložený na serveru
Trigger – program uložený na serveru, který se spouští při určité činnosti

11 Oprávněnost přístupu k datům dle SQL

Uživatel může mít více rolí.
Kdo vytvoří tabulku, tak tomu patří.

Přiřazení práv – GRANT
GRANT
- ALL RIGHTS – všechna práva
- <privilegia>
- ALL NOT <privilegia>

ON <tabulka> TO <user>

[WITH GRANT OPTION] – práva může předávat dál

Odebrání práv – REVOKE

Privilegia: READ, INSERT, UPDATE, DELETE, DROP

Pokud odebereme práva uživateli s GRANT OPTION, odeberou se i všem, komu je předal dál – řešeno pomocí časových značek v tabulce práv.

Uživatelé – USERS

Vytvoření nového uživatele

```
CREATE USER uživatel
   IDENTIFIED [BY heslo | EXTERNALLY |
   GLOBALLY AS 'external_name']
   [DEFAULT TABLESPACE tabulkovy_prostor]
   [TEMPORARY TABLESPACE tabulkovy_prostor ]
   [QUOTA (N | [ K| M] UNLIMITED) ON TABLESPACE]
   [PROFILE profil]
   [ACCOUNT [LOCK|UNLOCK]
[PASSWORD EXPIRE]
```

Nutno určit minimálně:
Jméno a zpusob identifikace
Typ overení v klausuli IDENTIFIED: Global, External, Password

Vynucená změna po prvním přihlášení
PASSWORD EXPIRE

Změna hesla
ALTER USER jméno_uživatele IDENTIFIED BY nové_heslo

Role a práva uživatelů

GRANT
Umožňuje přiřadit či změnit
Aktuální prirazení rolí – GRANT ROLE TO uživatel | PUBLIC [WITH ADMIN OPTION]

Aktuální prirazení oprávnení – GRANT SYSTEM_PRIVILEG TO uživatel | PUBLIC [WITH ADMIN OPTION]

Oprávnění typu SELECT nebo UPDATE může poskytnout pouze vlastník objektu, nebo databázový administrátor (i ten však musí být oprávněn) – GRANT oprávnení ON objekt TO jméno_administrátora [WITH GRANT OPTION]

Klausule WITH GRANT OPTION znamená, že uživatel může poskytnout toto oprávnění i ostatním uživatelům nebo rolím.

REVOKE
Pomocí příkazu REVOKE odeberete roli nebo oprávnění uživateli nebo roli:

REVOKE systémové_právo | role FROM uživatel |role| PUBLIC ;

Pomocí následujícího příkazu objektová oprávnění nebo roli:

REVOKE objektové_právo | ALL ON schéma.objekt FROM uživatel |role| PUBLIC CASCADE CONSTRAINT;

ROLE
Vytvoření nové role (sady oprávnění)

CREATE ROLE role [NOT IDENTIFIED | IDENTIFIED]
[BY heslo | EXTERNALLY|GLOBALLY] NOT IDENTIFIED – bez zadání hesla
IDENTIFIED EXTERNALLY pomocí operačního systému
IDENTIFIED GLOBALLY – jedinečná role v několika databázích

Příkaz může zadávat buď oprávněný uživatel, nebo administrátor databáze. Přiřazení role uživatelům (po jejím vytvoření): GRANT role TO uživatel;