UČEBNÍ TEXTY OSTRAVSKÉ UNIVERZITY

Přírodovědecká fakulta

ÚVOD DO DATABÁZÍ I

(DISTANČNÍ VÝUKOVÁ OPORA)

Zdeňka Telnarová

Ostravská univerzita

1 MODUL 1

1.1 ZÁKLADNÍ SYSTÉMOVÉ POJMY

Cíl:

Cílem této kapitoly je seznámit čtenáře se základními systémovými pojmy, které budou dále využívány a na něž bude v ostatních kapitolách odkazováno. Čtenář by měl pochopit, co je to systém, informační systém, informační systém s databází a databázový systém. Čtenář získá představu o tom, k čemu slouží informační systémy a jakou úlohu v nich hrají databáze. Stěžejními pojmy této kapitoly jsou pojmy data a informace. Čtenář se naučí mezi těmito pojmy rozlišovat a bude umět definovat, co to jsou data a co to jsou informace.

Klíčová slova:

Systém, struktura, informační systém, databázový systém, data, báze dat, informace, systém řízení báze dat.

Průvodce studiem:

Milá čtenářko, milý čtenáři,

začínáte studovat disciplínu, která je ve většině literatury nazývána "Databázové systémy". Jedná se o velmi obsáhlou problematiku, která se v žádném případě nemůže vejít do jednoho předmětu či kurzu. Tento úvodní předmět (Úvod do databází) by vám měl poskytnout základní informace, které vám umožní další studium databázové problematiky a především by vám měl usnadnit orientaci v této komplexní a obsáhlé oblasti informačních technologií. Je velmi důležité, abyste si osvojili všechny pojmy, se kterými budete postupně seznamováni a to na všech úrovních abstrakce. Tato kapitola uvádí obecné pojmy, jakými jsou systém, struktura, model a dále již se věnuje pouze těm systémům, které využívají databází.

Přeji vám, aby text předmětu Úvod do databází byl pro vás čitelný, zajímavý a aby vám co nejvíce pomohl zvládnout probíranou látku. Nyní se už pohodlně usaďte a začněte studovat.

1.1.1 Systém

Systém je abstrakce, kterou tvoříme v procesu poznávání reálných objektů, přičemž bereme v úvahu pouze podstatné vlastnosti zkoumaných objektů.

Reálným objektem nazýváme určitou část toho, co skutečně existuje kolem nás. Objekty můžeme zpravidla členit na části a vymezit některé jejich vlastnosti. Účelnost definování systému na reálném objektu spočívá v tom, že abstrahujeme od některých, z hlediska zkoumání nepodstatných, vlastností objektu.

Systém je tedy odrazem - modelem - objektivní reality.

Systém **S=(P,R)** je účelově definovaná množina prvků **P={p_i}** a množina vazeb **R={r_{ij}}** mezi prvky **p_i**, **p_j**, kde **i,j** jsou prvky množiny indexů. Tato množina prvků a vazeb má jako celek určité vlastnosti.

Prvky systému **p_i** jsou nejmenší, elementární části systému, na zvolené rozlišovací úrovni dále nedělitelné. Množinu všech prvků systému nazýváme univerzum systému.

Vazby **r_{ij}** jsou vzájemné závislosti, působení, návaznosti, způsoby spojení mezi prvky **p_i**, **p_j**. Množina prvků **p_i** a vazeb **r_{ij}** se nazývá struktura systému **S**.

1.1.2 Základní rozdělení systémů

Obecné systémy jsou abstraktní modely, které jsou přesně definovány. Jsou to formální systémy, které nemají konkrétní obsah. Příkladem může být soustava lineárních rovnic.

Reálné systémy jsou zavedeny na konkrétních reálných objektech. Vymezení reálného systému závisí na účelu, pro který tento systém konstruujeme. Proces se nazývá definování reálného systému na objektu. Reálné systémy jsou popsány verbálně, graficky, někdy částečně pomocí matematických prostředků.

Okolí systému je účelově definovaná množina prvků, které nejsou prvky daného systému, avšak mají k němu vazby. Vazby, které uskutečňují působení systému na okolí jsou výstupy systému. Vazby, kterými okolí působí na systém, jsou vstupy.

Prvky okolí, které mají vazbu na systém, nazýváme hraničními prvky. Existuje-li alespoň jeden prvek systému, který působí na okolí nebo opačně, jedná se o systém otevřený. V opačném případě se jedná o systém uzavřený. Struktury systémů se popisují nejčastěji pomocí blokových schémat a jednoduchých grafů.

1.1.3 Informační systémy s databází

Systém (z hlediska teorie systémů) je množina prvků a vazeb mezi nimi, které jsou účelově definovány na určité objektivní realitě. Touto objektivní realitou budeme rozumět uživatelskou aplikaci. Uživatelská aplikace je tedy výsek reálného světa či světa objektů.

Informační systém je systém, který zpracovává data a zabezpečuje komunikaci informací mezi jeho prvky. IS se často člení na systém zpracování dat a komunikační systém.

Informačním systémem rozumíme systém pro sběr, uchovávání, vyhledávání a zpracování dat za účelem poskytnutí informace o daném vymezeném světě objektů.

Dále budeme hovořit pouze o takových IS, které obsahují automatizovanou složku, tj. budeme mít na mysli AIS. IS obsahují pouze data, která se informacemi stávají až jejich interpretací uživatelem.

Data jsou údaje získané pozorováním, měřením, atd. Jsou to např. čísla, znaky, slova, jména.

Informace jsou pouze taková data, která nám mohou být k něčemu užitečná. Taková, která se dají rozumně interpretovat.

Informace je sdělení (vygenerované na základě dat uložených v databázi), které v příjemci odstraňuje neurčitost (neznalost).

1.1.4 Vývoj k informačním systémům s databází

Vývoj informačních systémů směrem k IS s databází můžeme rozdělit do 3 etap:

1. Polovina 50.let, zpracování bylo organizováno způsobem "vše v programu", tj. v programu byly popisy dat, data, algoritmy, případně různé typy zpracování.

2. Polovina 60.let, tzv. systémy s monitorem. Data byla oddělena od programů, ale jejich popis zůstal v programu. Data mohla být zpracovávána více programy současně. Popis fyzického záznamu však stále zůstával v programech. Nevýhodou byla redundance dat (stejná data se objevovala v různých souborech) a nekonzistence dat, protože nebyla řízená aktualizace dat od různých uživatelů.
3. Druhá polovina 60.let se vyznačovala snahou vytvoření prvních systému řízení báze dat. Data byla uložena v databázi, většinou centralizovaně na jednom místě, jejich popis nebyl součástí uživatelského programu. Tím, že se programy osvobodily od popisů dat, staly se do značné míry nezávislé na fyzickém uložení dat. Data byla udržována jednotně, jejich struktury byly navrhovány centrálně, a bylo dosaženo mnohem menší redundance a lepší konzistence dat.

1.1.5 Pojem databázový systém

Velmi často se v databázové literatuře setkáme s následující „rovnicí“:

SŘBD + DB = DBS

Systém řízení báze dat + báze dat = databázový systém

SŘBD je programové vybavení, které řídí všechny přístupy k datům v bázi dat. DB je množina vzájemně propojených dat, která slouží mnoha aplikacím. Data jsou uložena v paměti způsobem vylučujícím nežádoucí redundanci.

Popis dat vytváří tzv. schéma databáze (též databázové schéma). Schéma databáze popisuje jisté objekty a vztahy mezi nimi. Databázový model je kolekce (souhrn) pojmů, na kterých je vybudován jazyk pro definici dat. Jde o formalizmus, který umožňuje popsat část schématu databáze. Této části se obvykle říká logické schéma databáze, na rozdíl od fyzického schématu databáze, které popisuje uložení dat na vnějších pamětech, resp. způsob přístupu k nim. Databázový model je prostředek modelování. Výsledkem tohoto prostředku je schéma databáze. Databázový model tedy není výsledkem modelování, ale jeho prostředkem. Zmíněné popisy se pak ukládají do katalogu dat (slovníku dat). Katalog tedy obsahuje zásadní informace o datech, není součástí databáze. Definuje typy dat a typy vztahů mezi nimi. Konkrétní datové struktury jsou definovány pomocí definičního jazyka JDD (jazyk pro definování dat). Katalog dat uchovává data o datech neboli metadata. Operace nad daty jsou univerzální pro všechny organizace dat. Pojem katalog se vztahuje jednak k programovým prostředkům pro organizování popisu dat, jednak k výsledkům tohoto popisu, tj. k samotným metadatům. Katalog je tedy jakýmsi systémem řízení báze metadat. V katalogu dat je uloženo vždy příslušné interní datové schéma báze dat. Skutečnost, že si SŘBD vede katalog dat, je charakteristickým rysem databázové technologie. K manipulaci dat v databázi slouží speciální databázové jazyky JMD (Jazyky pro manipulaci s daty). Obsahují obvykle primitivní operace pro vkládání, odebírání a změnu dat v databázi a důležité prostředky pro výběr dat.

Pojmy k zapamatování:

Systém

Struktura

Informační systém

Databázový systém

Data

Báze dat

Informace

Systém řízení báze dat

1.2 CHARAKTERISTICKÉ RYSY DATABÁZOVÉ TECHNOLOGIE

Cíl:

Cílem kapitoly je upozornit na některé vlastnosti databázové technologie a dále vysvětlit všeobecně akceptovaný pojem "tříúrovňové architektury databází". Čtenář by měl po přečtení této kapitoly při pohledu na data rozlišovat mezi jednotlivými úrovněmi abstrakce. Podstatné pro čtenáře je, aby si uvědomil, jaké základní výhody při práci s daty poskytují databázové technologie a jaké charakteristické rysy databázové technologie vykazují.

Klíčová slova:

Nezávislost dat, sdílení dat,redundance, konzistence, integrita, interní struktura, externí struktura, konceptuální model.

Databázovou technologií nazýváme soubor pojmů, prostředků a technik pro vytváření informačních systémů s databází. Přitom za hlavní paradigma databázové technologie je považována vzájemná nezávislost dat a aplikačních programů.

1.2.1 Vlastnosti databázové technologie

Mezi základní vlastnosti (charakteristické rysy) databázové technologie patří nezávislost dat, přístup k informacím, sdílení dat, ochrana a utajení dat, konzistence a integrita dat.

Nezávislost dat

Data a programy jsou vzájemně nezávislé, tzn. že změna vyvolaná v datech nevyvolá nutnost změny aplikačního programu a naopak. Fyzická nezávislost dat znamená, že pracujeme s objekty nezávisle na jejich vnitřní interpretaci. Metoda uložení dat není podstatná.

Přístup k informacím a sdílení dat

Cílem databázové technologie je poskytnout uživateli informačního systému efektivní prostředky přístupu k informacím báze dat. Možnost sdílet data různými oprávněnými uživateli je jednou z velkých výhod IS založených na databázové technologii.

Prostředky ochrany dat a jejich utajení

Jedná se o ochranu dat proti neoprávněnému přístupu, zneužití resp. zničení dat. Jedná se o problematiku archivace dat a stanovení individuálních přístupových práv k datům.

Redundance, konzistentnost a integrita dat

Redundancí rozumíme vícenásobné uchovávání týchž dat v rámci databáze. Tato vlastnost působí četné potíže. Vysoké nároky na paměť nejsou ty nejhorší. Redundantní data působí problémy při jejich aktualizaci. Data se vyskytují opakovaně a při aktualizaci je třeba všechny jejich výskyty správně aktualizovat. Chyby při aktualizaci redundantních dat vedou k narušení konzistence (slučitelnosti) dat. SŘBD disponují prostředky na hlídání integrity databáze. Integrita databáze se hlídá na základě definovaných integritních omezení, která jsou součástí schématu databáze. Integrita dat se zkoumá ve třech oblastech:

- Doménová integrita
- Entitní integrita
- Referenční integrita

Průvodce studiem:

S bližším vysvětlením těchto pojmů a hlavně s jejich užitím se budete v textu setkávat velmi často. Zatím pouze zaregistrujte, že existují uvedené tři typy integritních omezení. Zopakuji tedy a vy si zapamatujte: doménové integritní omezení (IO), entitní IO, referenční IO.

Vztahy mezi daty

Databázová technologie může vyjadřovat vztahy mezi datovými strukturami bez použití algoritmů. Logické vazby lze definovat mezi různými typy záznamů v rámci různých databázových souborů. Způsob, jakým se vazby definují je dán použitým databázovým modelem.

1.2.2 Tříúrovňová architektura databáze

Interní struktura

je nejblíže paměťové struktuře uložení dat. Využívá funkcí a vlastností konkrétního operačního systému. Uživatel se touto interní strukturou nemusí zabývat. Změna interní struktury se nesmí odrážet v koncepční struktuře databáze.

Externí struktura

představuje uživatelovy požadavky. Tyto jsou vyjadřovány v nějakém jazyku. Pomocí nich uživatel formuluje své datové objekty a vztahy mezi nimi. K jednomu internímu záznamu je možno sestavit více uživatelských externích pohledů. Externí struktura je závislá především na použitých jazycích. Změny v uživatelských pohledech by se neměly projevit ve změně koncepčního modelu.

Koncepční struktura - konceptuální model

představuje celý informační obsah databáze. Měl by být nezávislý jak na fyzickém řešení, tak na okamžitých uživatelských potřebách.

Pojmy k zapamatování:

Nezávislost dat

Sdílení dat

Redundance

Konzistence

Integrita

Interní struktura

Externí struktura

Konceptuální model.

 **Kontrolní otázky:**

1. Co je to systém, informační systém, informační systém s databází?
2. Co to jsou data a informace?
3. Co je to systém řízení báze dat?
4. Vysvětlete pojmy data a metadata.
5. Jaké potíže činí redundance v databázi?

1.3 HISTORIE DATABÁZOVÝCH MODELŮ

Cíl:

Kapitola se věnuje dvěma problematikám. Nejdříve krátce informuje o historii databázových systémů, dále se zabývá definováním báze dat. Z hlediska dalšího výkladu je problematika definování báze dat klíčovou, proto ji nedoporučuji podcenit. Čtenář by si měl osvojit postup pro definování báze dat a bude mít možnost si tento postup v dalších kapitolách ověřit. Veškeré pojmy, které jsou v kapitole "Fáze definování báze dat " zavedeny, jsou velmi důležité a pro další pochopení problematiky nepostradatelné, proto by je měl čtenář po prostudování kapitoly znát.

Klíčová slova:

Analýza uživatelských požadavků, logický návrh, fyzická realizace, konceptuální modelování, datové modelování.

Databázová technologie disponuje specifickými prostředky. Jedním z nich je modelování. Model je prostředek pro vytváření schémat. Obvykle se setkáváme s definicí modelu jako souboru prostředků pro definování báze dat. K popisu struktur dat a jejich vzájemných vazeb slouží databázové definiční jazyky.

1.3.1 Historie databázových modelů

 **Průvodce studiem:**

Kapitolu 1.3.1. a kapitolu 1.3.2. chápejte jako přehledové, které slouží k vytvoření celkového přehledu o vzniku a vývoji databázové technologie. Nejsou pro další studium nijak podstatné, takže není třeba si veškeré údaje, v nich uvedené, pamatovat. Protože do kontextu databázové problematiky patří, rozhodla jsem se je do textu zařadit.

Ze současného pohledu můžeme sledovat 4 generace modelů:

- Primitivní databázové modely.
- Klasické databázové modely.
- Sémantické databázové modely - konceptuální.
- Aplikačně orientované databázové modely.

1.3.1.1 Primitivní databázové modely

Tato generace chápe popis světa objektů v řeči počítačů, tj. ve strukturách programovacích jazyků vhodných pro hromadné zpracování dat. Jde vlastně o popis vybraných informací o světě objektů pomocí souborů skládajících se z vět, které obsahují položky. Veškeré souvislosti mezi jednotlivými záznamy, ať už různého nebo stejného typu, omezení na hodnoty atributů atd. realizoval programátor ve svých programech. Definice souborů byly stále součástí uživatelských programů.

1.3.1.2 Klasické databázové modely

Tato generace je charakterizována síťovými modely a rozvojem hierarchických modelů, který se jeví jako speciální případ síťového modelu. Jednotkou pohledu na svět objektů je opět typ záznamu a záznam. Vztahy mezi objekty světa objektů se realizují vztahy mezi záznamy. Podstatné na těchto modelech je to, že jsou vybaveny silnými manipulačními prostředky. V podstatě však představují pouze jiný styl programování. Průlomem v oblasti modelování se stal v roce 1970 návrh relačního modelu dat (RMD), který přinesl několik zcela nových pohledů na databázové modelování. Především výrazně zvýšil nezávislost dat na implementaci. Tento způsob modelování využívá prostředků predikátového kalkulu 1.řádu (interpretace predikátových symbolů jsou totiž relace) a prostředků relační algebry.

Všechny tři typy modelů (síťový, hierarchický a relační) jsou souborově orientované. Tento rys jim předurčuje jisté konceptuální meze, které mají blíže k počítačově orientovanému světu objektů než ke světu objektů danému uživatelskou aplikací.

1.3.1.3 Sémantické - konceptuální - databázové modely

Jsou založeny na kvalitativně odlišné úrovni v otázce vztahu databázových modelů ke světu objektů. Objevují konstrukty podporující přesnější vyjádření vztahů ve světě objektů. Jsou to konstrukty založené na pojmech entita, vztah, vlastnost, atribut, objekt apod. Tyto databázové modely se nazývají sémantické, případně konceptuální. Původcem těchto modelů je Chenův E-R model, založený na pojmech entita (Entity), vztah (Relationship) a atribut (Attribute). Obvykle částí těchto modelů bývají transformační algoritmy převádějící konceptuální schéma v jednom modelu na schéma v jiném modelu.

1.3.1.4 Aplikačně orientované databázové modely

Tato generace je založena na zjištění neúspěšnosti vytvořit "univerzální databázový model". Vytvářejí se speciální databázové modely pro zpracování vědeckých a experimentálních dat, modely pro grafické databáze, nestrukturované databáze textů, inženýrské aplikace (CAD/CAM) atd.

1.3.2 Programování databázových aplikací

V této oblasti můžeme vysledovat 3 proudy, respektive metodiky přístupu:

1. Konvenční přístup

Tento přístup k programování databázových aplikací je založen na vymezení potřebných datových typů pomocí konstruktorů zvaných datové struktury a na popisu funkční složky systému - algoritmu zpracování vstupních dat na data výstupní. Datové typy poskytují funkční složce nutné pozadí. Samotná data v konvenčním přístupu nemají povahu hotového výrobku, ale jsou podporou pro algoritmy. Těžiště konvenčního přístupu spočívá v rovině funkčního modelování.

2. Přístup z pohledu databázového modelování

Báze dat, která představuje strukturovaný datový typ vysoké úrovně, nemá již jen pomocnou funkci pro algoritmické zpracování dat, ale sama o sobě je hotovým výrobkem - datovou základnou informačního systému. Vytvářet aplikační programy lze v tomto přístupu až po té, kdy byla vytvořena báze dat jako nutné pozadí databázových aplikací. Programy v tomto přístupu obvykle představují manipulaci s informacemi, které jsou obsažené nebo odvoditelné z báze dat. Jazyky pro manipulaci s daty jsou hlavním nástrojem pro tvorbu uživatelských aplikací. Dalším prostředkem, který mohou uživatelské aplikace využívat jsou, dotazovací jazyky.

3. Objektově orientovaný přístup

Tento přístup představuje významnou změnu v pojetí databázového modelování. Každý objekt, představující zpravidla menší a co do funkčnosti autonomní výsek světa objektů, má sám pro sebe své pozadí i své popředí (tedy datovou úroveň a funkční úroveň). Objekt má samostatný život a možnost komunikace s dalšími objekty. Využitím objektové orientace se dosahuje větší flexibility a snadnější udržovatelnosti bází dat.

1.3.3 Fáze definování báze dat

Průvodce studiem:

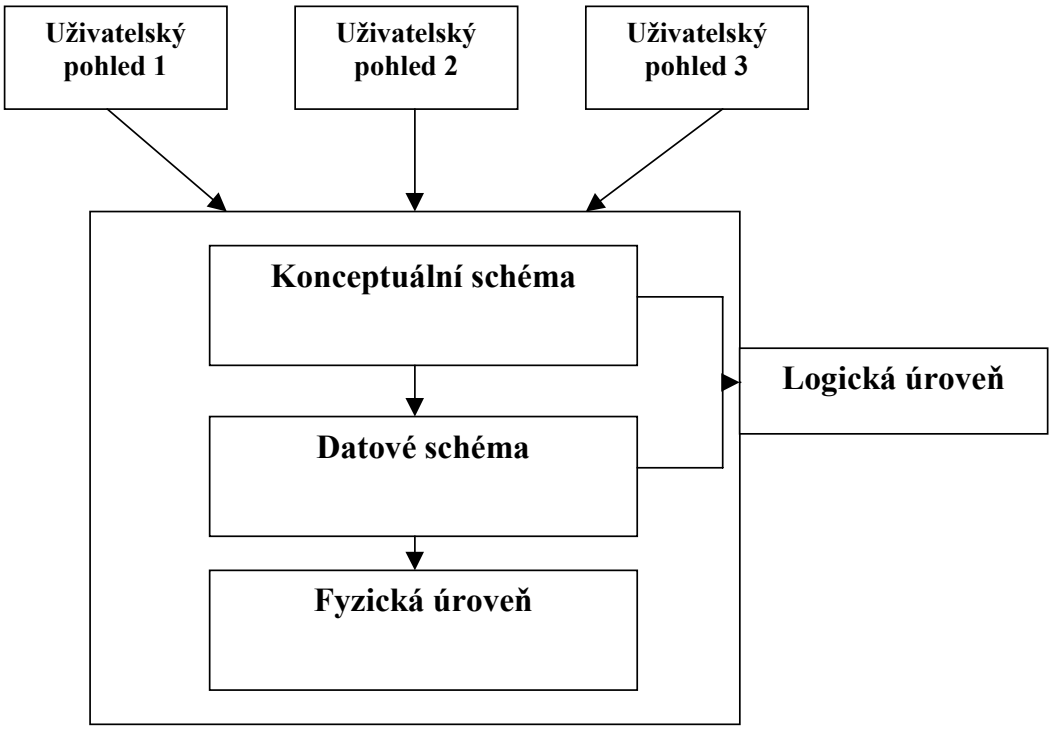
Na rozdíl od předchozích dvou kapitol, tato kapitola je velmi důležitá. Zavádí totiž základní kroky pro definování báze dat. Vytvoření báze dat, respektive její struktury, je základním krokem na cestě k budování databázových systémů. Bez správně navržené struktury není možno vytvořit kvalitní databázi. Kvalitou přitom rozumíme hlavně integritu a konzistenci databáze. Abyste strukturu báze dat správně navrhli, musíte dodržovat některé postupy. A s těmito postupy vás právě seznámí následující text.

Proces databázového modelování probíhá v těchto fázích:

- Analýza uživatelských požadavků.
- Fáze logického návrhu.
- Fáze fyzické realizace.

Fáze logického návrhu má 3 etapy:

- Konceptuální modelování.
- Výběr vhodného systému řízení báze dat (SŘBD).
- Transformace konceptuálního modelu na datový model.



Analýza požadavků

V této fázi pracuje konstruktér databázového návrhu na tzv. uživatelské úrovni. Svět objektů popisuje obvyklými vyjadřovacími prostředky. Různí uživatelé mohou mít různé nebo překrývající se požadavky na data. Tím jsou vymezeny jednotlivé uživatelské pohledy na bázi dat. Je třeba poznat modelovanou realitu a veškeré požadavky uživatelů na budoucí aplikaci.

Logický návrh báze dat

Probíhá obvykle ve dvou krocích:

1. Modelování v rámci jednotlivých uživatelských pohledů na data

Produktem modelování je popis dat a jejich vzájemných vztahů - dílčí schéma. Integrací dílčích schémat vzniká celkový formalizovaný popis báze dat bez ohledu na to, jakými programovými prostředky bude dále zpracováván. Produktem této integrace je konceptuální schéma. Definiční jazyk konceptuálního modelování (tzv. konceptuální model) používá speciálních pojmů a grafických symbolů. V současné době jsou nejvyspělejší konceptuální modely součástmi systémů CASE. Konceptuální modelování popisuje svět objektů pomocí pojmů entita, atribut, vztah.

2. Výběr vhodného SŘBD

Při výběru se řídíme mnoha různými hledisky. K nejpodstatnějším patří:

- Současný přístup k datům pro více uživatelů.
- Ochrana dat.
- Prostředky pro centrální správu dat.
- Nezávislost dat na aplikacích.
- Možnost vytváření složitých datových struktur.
- Architektura desktop/klient, server.

- Vyhledávací mechanizmy.
- Fyzická implementace souborů (jak jsou řešeny primární soubory, indexy).

3. Transformace konceptuálního modelu na datový

Konceptuální model je převeden do popisu v jazyce pro definování dat, který je již závislý na použitém SŘBD. Touto transformací vzniká datový model, který je již méně abstraktním pohledem na bázi dat. Datový model pracuje s pojmy strukturovaných datových typů - položka, záznam, soubor. Datovým modelem jsou data předepsaným způsobem strukturalizována, aby je bylo možno zobrazit ve fyzické bázi dat. Nejznámější datové modely jsou síťový, hierarchický a relační.

4. Generování schématu databáze

Posledním krokem na cestě k tvorbě schématu databáze (struktura báze dat) je vygenerování konkrétního schématu z datového modelu. Obvykle je tato činnost automatizována a provádějí ji nástroje nazývané CASE.

Fyzická úroveň

Na fyzické úrovni se zabýváme strukturou souborů dat a způsobem jejich uložení na vnějších paměťových médiích. Dále se zabýváme způsoby zpřístupnění dat (poskytování informací z databáze), vyhledávacími mechanizmy, optimalizací ukládání, čtení, vyhodnocování dotazů, atd. Velmi důležitou problematikou jsou transakce.

Pojmy k zapamatování:

Analýza uživatelských požadavků

Logický návrh

Fyzická realizace

Konceptuální modelování

Datové modelování

Generování schématu databáze.

1.4 SÍŤOVÉ A HIERARCHICKÉ DATABÁZOVÉ MODELY

Cíl:

Tato kapitola je pouze informativní a seznamuje čtenáře s některými, dnes již překonanými, přístupy k databázovému modelování. Po jejím přečtení by měl mít čtenář přehled o možnostech síťových a hierarchických datových modelů. Není nutné, aby čtenář přesně znal jednotlivé konstrukty, stačí mít o síťových a hierarchických modelech rámcovou představu.

Klíčová slova:

Síťový model, hierarchický model.

1.4.1 Síťové databázové modely

Síťové modely jsou založeny na souborech a vztazích mezi záznamy souborů. Jde o souborově orientované modely s možností zapsání schématu databáze buď vhodným jazykem pro definování dat nebo s možností použití pomocné - diagram datové struktury (též Bachmanův diagram).

1.4.1.1 Atributy záznamů v síťových modelech mohou být 4 druhů

jednoduché - dále nedělitelné (atomické)

vícehodnotové (např. knihy s více autory)

složené (např. adresa, kterou lze rozdělit na č.,ul.)

složené, opakující se (taky může být adresa, např. instituce v několika pobočkách)

Vztahy mezi záznamy se modelují pomocí tzv. spojek. Zadáním jmen položek, ev. jejich druhu získáme typ záznamu. Výskyty typu záznamu jsou reprezentovány jednotlivými záznamy danými nějakou kombinací hodnot odpovídajících položek.

1.4.1.2 Typy vztahů

Nejrozšířenějším typem vztahu je binární vztah mezi objekty. O binárním vztahu hovoříme tehdy, jestliže do vztahu vstupují dva objekty. U takových vztahů zjišťujeme jejich kardinalitu. Kardinalitu vztahů sledujeme u všech typů databázových modelů.

1.4.1.3 Kardinality

a) 1:1

 **Příklad:**

- Učitel učí maximálně jeden předmět.
- Předmět je vyučován maximálně jedním učitelem.

Tento vztah zahrnuje i případy 1:0 a 0:1

b) 1:N

 **Příklad:**

- Učitel může učit více než jeden předmět.
- Předmět může být vyučován maximálně jedním učitelem.

Tento vztah zahrnuje i případy 1:0, 0:1 a 1:1

c) M:N

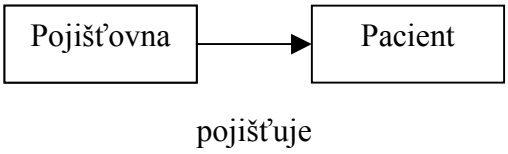
 **Příklad:**

- Učitel může učit více než jeden předmět.
- Předmět může být vyučován více než jedním učitelem.

Tento vztah zahrnuje i případy 1:0, 0:1,1:1 a 1:N

1.4.1.4 Diagram datové struktury

skládá se ze 3 komponent:



- Obdélníky představuje typ záznamu.
- Čáry představuje typy vztahů.
- Pojmenování čar představuje jména typů vztahů.

U síťového modelu jsou povoleny pouze vztahy 1:1 a 1:N, to znamená pouze funkcionální vztahy.

Příklad síťového modelu:

Příklad je z prostředí knihovny.

Čtenáři budou zobrazeni na záznamy typu:

ČTENÁŘ(Č-ČT, JMÉNO, ADRESA),

Knihy na záznamu typu:

KNIHA(ISBN, AUTOR, TITUL),

Jednotlivé exempláře na záznamu typu:

EXEMPLÁŘ(INV-Č, D-NÁKUPU),

Další typ záznamu bude:

VÝPUJČKA(Č-ČT,INV-Č,D-ZPĚT) a ZÁZNAM(ISBN, D-REZ, Č-ČT).

Dále je třeba definovat typy vztahů:

SI-PŮJČIL:(ČTENÁŘ, VÝPŮJČKA)

JE-PŮJČEN:(EXEMPLÁŘ, VÝPŮJČKA)

MÁ-KOPIE:(KNIHA, EXEMPLÁŘ)

MÁ-ZÁZNAMY:(KNIHA, ZÁZNAM)

MÁ-REZERVACE:(ČTENÁŘ, ZÁZNAM)

1.4.1.5 Schéma databáze v síťovém modelu

Schéma se skládá z typů záznamů, které obsahují popisy datových položek včetně domén a typů C-množin, které definují vztahy mezi záznamy. Každý záznam má přiřazen databázový klíč, což je disková adresa spojená se záznamem. C-množiny jsou uspořádané

dvojice typů záznamů. Typy záznamů z hlediska vztahů (C-množin) se nazývají vlastník (owner) a člen (member). C-množina se sestává z jednoho výskytu vlastníka a několika (ev. žádného) výskytu člena. Síťový model umožňuje pouze vztahy 1:N. Je-li třeba modelovat vztahy typu M:N, je třeba provést transformaci na vztahy 1:N.

Typy vztahů

Vztahy jsou v síťových modelech nazývány CS - typ nebo C - množina.

V CS - typu figuruje 1 typ záznamu (typ vlastníka) s jedním nebo více typy záznamů (typ člen)

Implicitní IO

Žádný záznam nemůže být členem ve více než jednom výskytu daného CS - typu.

1.4.1.6 Tři typy začlenění nového typu záznamu do CS - typu

FIXED

Stane-li se záznam členem ve výskytu CS-typu, nemůže být přepojen do jakéhokoli výskytu jiného CS-typu. Jedině ho lze odstranit, znovu vytvořit a zapojit do jiného CS-typu.

MANDATORY

Záznam může být členem ve výskytu více CS-typů.

OPTIONAL

Záznam může být z CS-typu rozpojen a přepojen do jiného CS-typu.

 **Příklad:**

DEFINICE SCHÉMATU DATABÁZE SÍŤOVÉHO MODELU

2 oblasti : OBLAST - STUDENTA
OBLAST - UČITELE

```
1  SCHEMA NAME IS  EVIDENCE - STUDENTA.
2  AREA NAME IS   OBLAST - STUDENTA.
3  AREA NAME IS    OBLAST - UČITELE.
4  RECORD NAME IS STUDENT;
5  LOCATION  MODE IS CALC USING CISLO - STUDENTA
6  DUPLICATES ARE NOT ALLOWED;
7  WITHIN OBLAST - STUDENTA
8  02  CISLO - STUDENTA; TYPE IS FIXED DECIMAL 9.
9  02  JMENO; TYPE IS CHARACTER 20.
10 02  ADRESA; TYPE IS CHARACTER 40.
11 02  ROCNIK ; TYPE IS FIXED DECIMAL 2.
12 RECORD NAME IS UČITEL;
13 LOCATION MODE IS CALC USING CISLO -ZAMESTNANCE
14 DUPLICATES ARE NOT ALLOWED;
```

15 WITHIN OBLAST - UČITELE.

16 02 CISLO - ZAMESTNANCE; TYPE IS FIXED DECIMAL 9.

17 02 JMENO; TYPE IS CHARACTER 20.

18 02 KATEDRA; OCCURS

19 .....

1.4.2 Hierarchické datové modely

Hierarchické modely jsou opět souborově orientované modely. Z matematického hlediska jsou hierarchie speciální případ síťových modelů. Diagram datové struktury je v tomto případě strom, případně les (několik stromových struktur).

Typy záznamů u hierarchických modelů jsou podobné síťovým, ale jsou jednodušší, protože obsahují pouze jednoduché položky, případně opakující se položky.

Datové schéma je tedy tvořeno zadáním typů záznamů a hierarchické struktury tzv. definičních stromů. Tyto databázové stromy lze vždy spojit jedním prázdným (systémovým) uzlem - kořenem - do jednoho stromu. Prázdnému kořenu může odpovídat prázdný typ záznamu v databázovém schématu.

Důležitým pojmem u hierarchických modelů je hierarchická cesta. Tato udává nějakou cestu v odpovídajícím definičním stromě od kořene k nějakému typu záznamu.

1.4.2.1 Rušení záznamů v hierarchické databázi

Každý záznam má v hierarchické databázi jednoznačně určenou množinu předchůdců. Proto při rušení záznamů mohou nastat tyto problémy:

- Existují závislé záznamy.
- Rušení záznamu mění hierarchickou strukturu.
- Při rušení je třeba zrušit všechny následníky. To je tzv. vynucené rušení (TRIGGERED DELETE).
- V případě, že chceme následníky uchovat, musíme použít prázdný záznam.

Problémy s vyhledáváním záznamů:

1. Vybírání záznamů - předchůdců = vzestupná hierarchická normalizace.
2. Vybírání záznamů - následníků = sestupná hierarchická normalizace.

Příklad:

DEFINICE SCHÉMATU DATABÁZE HIERARCHICKÉHO MODELU.

DATABAZE NAME IS STUDENT;

1 CISLO STUDENTA (INTEGER (9)):

2 JMENO STUDENTA (NON-KEY NAME X (20)):

3 ADRESA (NON-KEY NAME X(40)):

4 ROCNIK (NON-KEY INTEGER 9 (2)):

5 PREDNASKA STUDENTA (RG):

.

. POPIS SKUPINY

.

DATABAZE NAME IS UCITEL:

1 CISLO ZAMESTNANCE (INTEGER 9(9)):

2 JMENO UCITELE (NON-KEY NAME X (20)):

.

.

Příklad aktualizace:

1. Zvýšení všech bodových hodnocení přednášky 2301 o 10%.

```
CHANGE   BODOVE HODNOCENI   TO   (BODOVE HODNOCENI * 1,1)  CISLO
                PREDNASKY STUDENTA EQ  2301
```

2. Zrušení v databázi učitele s číslem 03167895.

```
REMOVE    UCITELCISLO   WHERE   UCITELCISLO EQ 03167895;
REMOVE TREE UCITEL WHERE CISLO ZAMESTNANCE EQ 03167895
```

2 MODUL 2

2.1 KONCEPTUÁLNÍ MODELOVÁNÍ

Cíl:

Cílem této kapitoly je, aby čtenář pochopil základní koncepty (pojmy) konceptuálního modelování a činnosti spojené s tvorbou konceptuálního modelu. Tvorba konceptuálního modelu má jisté formální požadavky, které by měl čtenář zvládnout. Podstatné ovšem je, aby správně pochopil význam jednotlivých konceptů, které se v modelu vyskytují a význam konceptuálního modelu jako celku. Po prostudování této kapitoly by měl být čtenář schopen vytvořit korektní konceptuální model výseku reality.

Klíčová slova:

entita, vztah, atribut, doména, instance, identifikační (primární) klíč, integritní omezení, kardinalita vztahu, ISA hierarchie, cizí klíč.

Průvodce studiem:

Konceptuální modely jsou pokusem umožnit vytvoření popisu dat v databázi nezávisle na jejich uložení. Jsou prvním krokem na cestě k vytvoření databázového schématu, jako struktury, do které budou data ukládána. Konceptuální model představuje formální popis modelované reality, která nás zajímá a o které budeme v budoucí databázi shromažďovat data. Má-li být budoucí databáze správně navržená s cílem minimalizovat nadbytečná data, bez konceptuálního modelu se neobejdeme.

2.1.1 Úvod

Na úvod je třeba se seznámit se základními koncepty (pojmy), se kterými konceptuální model pracuje. Jsou jimi tyto abstrakce:

Entita	**resp.**	**Objekt**
Relationship	**resp.**	**Vztah**
Atribut	**resp.**	**Vlastnost**

Průvodce studiem:

Hned v úvodu upozorňuji na častou chybu, které se studenti dopouštějí. Chybou je nerozlišování od abstraktního pojmu objekt, vztah, atribut a konkrétního výskytu (instance) objektu, vztahu, či atributu.

Objektem může být "Student", instancí konkrétní Student popsaný konkrétními hodnotami atributů. **Vztahem** může být "patří do" (např. Student patří do Skupiny), instancí vztahu může být Král patří do AI1. **Atributem** může být "Příjmení", instancí (hodnotou atributu) pak může být Král.

Entita je objekt reálného světa, který je schopen nezávislé existence a je jednoznačně odlišitelný od ostatních objektů. Např. "Student", "Předmět", "Kniha", "Pojišťovna". Konkrétní výskyty entit se nazývají **instance**.

Atributem budeme rozumět funkci přiřazující entitám či vztahům hodnotu popisného typu, určující některou podstatnou vlastnost entity nebo vztahu. Hodnoty atributů jsou vybírány z jednotlivých **domén**.

Doména je množina homogenních dat, spojená s konkrétním atributem. Je-li např. doména množina všech celých čísel z intervalu <0,150), může se jednat o doménu atributu věk.


Příklad:

Mějme entitu ***Pacient*** s atributy *Rodné číslo, Jméno, Příjmení, Datum narození, Pojišťovna*. Doména atributu *Rodné číslo* je množina textových řetězců délky 10 alfabetických znaků, doména atributu *Jméno* je množina textových řetězců délky max. 15 alfabetických znaků, doména atributu *Příjmení* je množina textových řetězců délky max. 15 alfabetických znaků, doména atributu *Datum narození* je množina datumových hodnot, doména atributu *Pojišťovna* je množina přirozených čísel z intervalu (100, 1000).Pro všechny atributy entity ***Pacient*** provedeme jedno možné přiřazení *jméno atributu = hodnota z odpovídající domény*. Tímto způsobem získáme jednu instanci entity ***Pacient***, která by mohla vypadat následujícím způsobem:(5562290800,Jan,Novák,29.12.1955, 111)

Vztah je vazba mezi dvěma entitami (obecně i více entitami). Např. Student studuje Předmět. "Student" a "Předmět" jsou entity, zatímco "studuje" je vztah.

 **Příklad:**

Mějme entity ***Pacient*** a ***Pojišťovna***. Strukturu entity ***Pacient*** jsme již nadefinovali, zbývá definice entity ***Pojišťovna***, která by mohla být následující: *Kod, Název, Adresa*. Bude-li atribut *Pojišťovna* entity ***Pacient*** definován nad stejnou doménou, jako atribut *Kod* entity ***Pojišťovna***, je možné mezi těmito dvěma entitami nadefinovat vztah, např. vztah ***je_pojištěn_u***.

 **Průvodce studiem:**

Vymezení pojmů entita, vztah a atribut je dosti volné. Správné nalezené entit modelované reality patří k nejtěžším úkolům a vyžaduje kromě velmi dobré znalosti modelované reality i jistou zkušenost návrháře. Vodítkem Vám v začátcích může být to, že entity označujeme obvykle podstatnými jmény a vztahy slovesy. Ne ovšem každé podstatné jméno neformálně popsané reality je entitou. Může se jednat i o atribut. Atribut, na rozdíl od entity, není schopen samostatné existence, nemá svou identitu. Atribut je např. "barva", "datum narození".

 **Kontrolní otázky:**

1. Pokuste se najít výsek reality, který je Vám blízký a neformálně jej popište.
2. Najděte na tomto výseku reality entity.
3. Pokuste se definovat, do jakých vztahů nalezené entity vstupují.

2.1.2 Tvorba konceptuálního modelu

Konceptuální model (někdy taky označován jako E-R model) je množina pojmů, které nám pomáhají, na konceptuální úrovni abstrakce, popsat realitu za účelem následně specifikovat strukturu databáze. Konceptuální úrovní abstrakce přitom máme na mysli tu skutečnost, že v popředí zájmu je pouze modelovaná reality. Na této úrovni se ještě nestaráme o to, jak bude budoucí databáze technicky spravována a řízena.

2.1.2.1 Činnosti při tvorbě konceptuálního modelu

 **Průvodce studiem**:

Následující řádky berte jako doporučení, jak postupovat při tvorbě konceptuálního modelu. Je rozumné se těchto doporučení držet, abyste se aspoň v počátcích tvorby konceptuálního modelu nedopustili zbytečných chyb.

1. Identifikace entit jako množiny objektů stejného typu.

Např. KNIHA, ABONENT_KNIHOVNY, ZAMESTNANEC označují typy entit.

2. Identifikace vztahů, do kterých entity mohou vstupovat.

Např. ABONENT(entita) MA_PUJCEN (vztah) daný EXEMPLAR (entita).

3. Na základě přiměřené úrovně abstrakce přiřazení jednotlivým entitám a vztahům popisné atributy.

Např. PRIJMENI (popisný atribut) daného ZAMESTNANCE (entita),

DATUM (popisný atribut), do kdy si daný ABONENT (entita) VYPUJCIL

(vztah) daný EXEMPLAR (entita).

4. Formulace integritních omezení (IO) vyjadřujících s větší či menší přesností soulad schématu s modelovanou realitou.

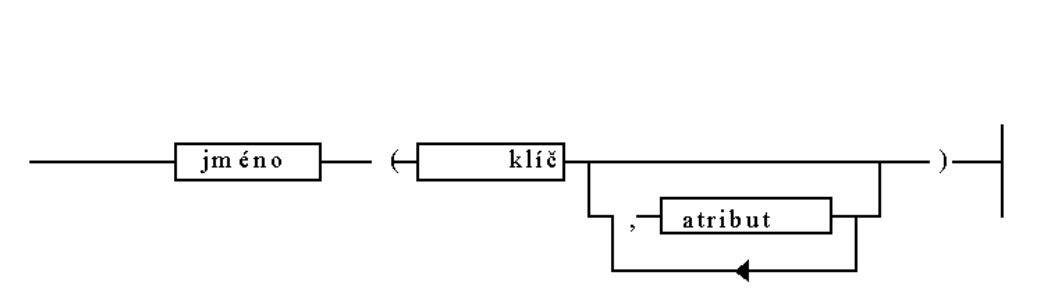
2.1.2.2 Způsoby zápisu konceptuálního modelu

Existují dva způsoby zápisu konceptuálního modelu:

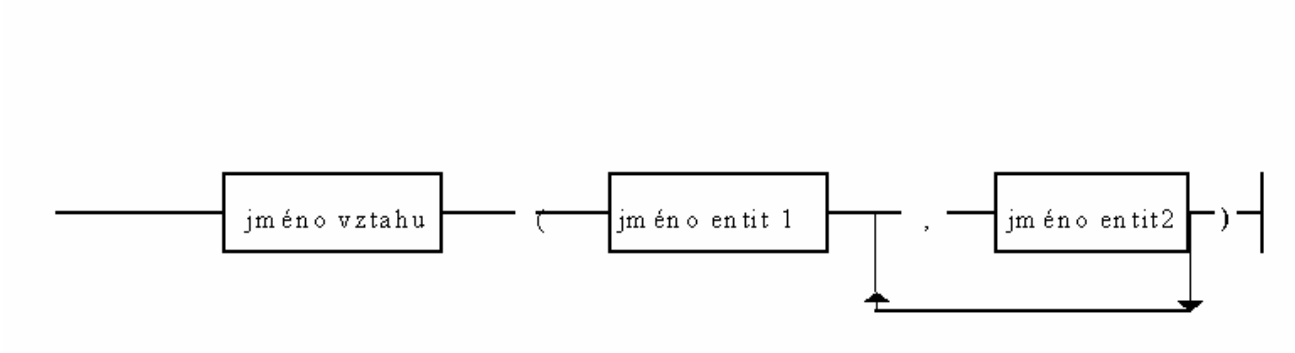
- Lineární textový.
- E-R diagramy.

Syntaxe lineárně textového zápisu

Entita



Vztah



Častěji se používá typový E-R diagram:

Entity - obdélníky

Vztahy - kosočtverce

Hrany ukazují, které entity vstupují do kterých vztahů. Každému uzlu grafu je přiděleno jméno.


Průvodce studiem:

Nyní jste se setkali se značným množstvím nových pojmů. Nenechejte se odradit, v následujícím textu se seznámíte s jejich definicemi a hlavně s příklady jejich použití. Doporučuji Vám, abyste se po prostudování kapitoly 2.2.3. ke kapitole 2.2.1.a 2.2.2 ještě jednou vrátili a přesvědčili se o tom, zda Vám jsou všechny pojmy jasné.

2.1.2.3 Integritní omezení

Integritní omezení je množina pravidel, kterou si na modelované realitě definujeme proto, abychom měli data v budoucí databázi korektní a v souladu s modelovanou realitou. Rozeznáváme tři typy integritních omezení: Entitní, doménové a referenční.

Entitní Integritní omezení

někdy taky označované jako integritní omezení pro entity, je omezením v tom smyslu, že každá entita může obsahovat pouze navzájem různé instance (tzn. výskyty entity). K definice entitního integritního omezení potřebujeme identifikační klíč. Je nutné, aby každá entita měla identifikační klíč. S jeho nalezením mohou někdy být potíže, jak vyplyne z následujícího textu (slabý entitní typ).

Identifikační klíč

Každá entita musí být jednoznačně identifikovatelná. Atribut (skupina atributů), jehož hodnota slouží k indentifikaci konkrétní entity se nazývá identifikačním klíčem. Entitní typ může mít několik kandidátů na roli identifikačního klíče. V každém konkrétním případě je třeba zvažovat a potom volit tak, aby zvolené atributy skutečně plnily úlohu identifikačního klíče a jejich použití bylo efektivní z hlediska časových a paměťových nároků. Konkrétní výskyt vztahu může být identifikován identifikačními klíči entit, které ve vztahu vystupují.

Např. vztah MA_ZAPSAN mezi STUDENTEM S1 a PREDMETEM P7 může být vyjádřen dvojicí S1, P7.

 **Příklad:**

Máme-li entitu ***Student*** s atributy *Číslo studenta*, *Jméno*, *Příjmení*, atd., atribut *Číslo studenta* zajistí, že všechny instance entity ***Student*** budou navzájem různé. Nemohou totiž existovat dva různí studenti s stejným *Číslem studenta*.

Atributy, které mají výše uvedenou vlastnost, totiž, že identifikují jednotlivé instance entit, se nazývají identifikační klíče (primární klíče).

 **Průvodce studiem:**

Při nalezení jakékoli entity se hned zamýšlejte nad tím, co (který atribut/atributy) ji bude identifikovat. Nalezený atribut nebo skupinu atributů označte jako identifikační klíč. Pokud není možné v entitě najít žádnou kombinaci atributů, které by ji jednoznačně identifikovaly, bude se patrně jednat o slabou entitu (její definici najdete níže). Taková entita bude identifikovatelná až po zakreslení vztahu. Je ovšem třeba vždy velmi pečlivě zvážit, zda skutečně neexistuje možnost definovat vlastní identifikační klíč.

Integritní omezení pro domény

nazývané doménové integritní omezení zajišťuje, že hodnota atributu smí být vybrány vždy pouze z předem definované a přiřazené domény. Základní možností pro definici doménového IO je definice datového typu. Dále pak můžeme stanovit různé podmínky platnosti.

Doména je tedy množina hodnot konkrétního atributu. Analogie z matematiky je definiční obor.

 **Příklad:**

Pro výše uvedenou entitu ***Student*** doménové integritní omezení přiřadí pro atributy *Číslo studenta*, *Jméno*, *Příjmení* jejich domény např. následujícím způsobem:

Číslo studenta Celé číslo z intervalu 1 - 10000

Jméno Text maximálně na 15 znaků, přičemž znakem může být pouze písmeno

Příjmení Text maximálně na 20 znaků, přičemž znakem může být pouze písmeno

Referenční Integritní omezení

se někdy označuje jako IO pro vztahy a má několik složek:

Typ vztahu

Většina vztahů mezi entitami, které se na úrovni konceptuálního modelu zkoumají, jsou binární vztahy, tj. vztahy mezi dvěma entitami.

V rámci binárního vztahu může být entita:

- Nezávislá (nemá povinné členství ve vztahu).

- Existenčně závislá (má povinné členství ve vztahu).
- Identifikačně závislá (má povinné členství ve vztahu a je slabou entitou).

Členství ve vztahu

O entitách, které jsou zapojeny do vztahu říkáme, že jsou členy vztahu. V této souvislosti mluvíme o povinném (obligatorním) a nepovinném členství ve vztahu.

 **Příklad:**

- Každý zaměstnanec musí být zařazen do některého oddělení.
- Oddělení může existovat i bez zaměstnanců.

Členství entity ZAMESTNANEC ve vztahu je povinné. Entita ODDELENI má členství ve vztahu nepovinné.

Existenční závislost

Povinné členství ve vztahu bývá označováno a v konceptuálním modelu modelováno jako tzv. existenční závislost. Proto si zapamatujte následující definici: Entita E1 je existenčně závislá na entitě E2, jestliže nemůže existovat ani jedna instance entity E1, která by nevstoupila do vztahu s entitou E2.

 **Příklad:**

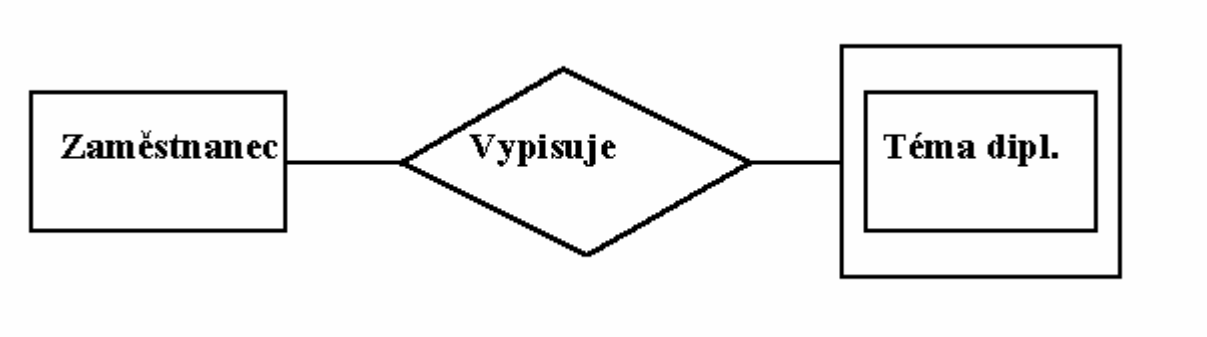
Mějme entitu Pacient, která charakterizuje jednotlivé Pacienty přicházející k lékaři a entitu Návštěva, která bude charakterizovat vlastnosti konkrétních onemocnění daného Pacienta, způsob léčby tohoto onemocnění, atd. Je zřejmé, že Pacient může existovat, aniž by byl na Návštěvě u lékaře s nějakou chorobou. Prostě se pouze u lékaře zaregistroval a zatím je zcela zdráv. Entita Návštěva je ovšem ve vztahu závislosti resp. nezávislosti v naprosto jiném postavení. Návštěva vypovídá o chorobě konkrétního Pacienta. Entita Návštěva je tedy existenčně závislá na entitě Pacient. Nemůže existovat žádný výskyt entity Návštěva, ke které by neexistoval výskyt entity Pacient. Lékař nemůže ošetřit neexistujícího Pacienta.

Identifikační závislost

Entita E1 je identifikačně závislá na entitě E2, je-li na ní existenčně závislá a navíc neexistuje žádná možnost u entity E1, jak definovat identifikační klíč s použitím pouze jejích atributů. Pro definování identifikačního klíče je nutné použít identifikační klíč entity E2. Entita E1 je nazývána slabá entita, entita E2 identifikační vlastník.

 **Příklad:**

Zaměstnanci školy mohou vypisovat témata diplomových prací, která mohou být stejná. Je nutné je rozlišit např. identifikátorem zaměstnance (Č_ZAM). Entita TEMA_DIPLOM je tedy identifikačně závislá na entitě ZAMESTNANEC. Dědí atribut Č_ZAM a teprve s ním je možno vytvořit skutečný identifikační klíč dané slabé entity. Jestliže doplníme entitě TEMA_DIPLOM nový vlastní atribut, např. Č_TEMATU, který bude schopen plnit úlohu identifikačního klíče, entita přestane být slabá, existenční závislost však zřejmě zůstane. Na následujícím obrázku je slabá entita označena dvojitým obdélníkem.



Vztahová integrita je ve většině datových modelů modelována pomocí cizího klíče. Rozhodnout, zda pojem cizí klíč je pojmem datového či konceptuálního modelu, je poměrně složité. Čistě formálně vzato, cizí klíč je pojem datového modelu, nicméně z praktických důvodů (většina CASE nástrojů to tak chápe) vysvětlíme pojem cizího klíče již na tomto místě.

Cizí klíč

je atribut nebo skupina atributů, které jsou v nezávislé entitě identifikačním klíčem. Cizí klíč u entity tedy signalizuje, že entita je závislá na jiné entitě, se kterou tato entita vstupuje do vztahu. Pokud je cizí klíč součásti identifikačního klíče této entity, jedná se o slabou entitu. Většina CASE nástrojů vytváří cizí klíče automaticky po zakreslení vztahu mezi entitami.

Kardinalita vztahu

Kardinalita vztahu je číslo, které udává počet instancí (výskytů) entity, kterými vstupuje do vztahu s instancemi druhé entity.

Konceptuální model připouští následující kardinality:

1:1 Jedna instance entity E1 vstupuje do vztahu s jednou instancí entity E2 a jedna instance entity E2 vstupuje do vztahu s jednou instancí entity E1.

1:N Jedna instance entity E1 vstupuje do vztahu s více instancemi entity E2 a jedna instance entity E2 vstupuje do vztahu s jednou instancí entity E1.

M:N Jedna instancí entity E1 vstupuje do vztahu s více instancemi entity E2 a jedna instancí entity E2 vstupuje do vztahu s více instancemi entity E2.

 **Příklad:**

Učitel učí Předmět

Kardinalita 1:1 znamená, že jeden učitel učí jeden předmět a zároveň jeden předmět je vyučován jedním učitelem.

Kardinalita 1:N znamená, že jeden učitel učí více předmětů a zároveň jeden předmět je vyučován jedním učitelem.

Kardinalita M:N znamená, že jeden učitel učí více předmětů a zároveň jeden předmět je vyučován více učiteli.

min-max io

Mějme typ vztahu **R(E1, E2)**. Potom **E1:(min1, max1)**, **E2:(min2, max2)** označuje minimální, resp. maximální počet výskytů entity daného typu ve vztahové množině **R**. Nula jako minimum vyjadřuje nepovinné členství ve vztahu. Nepovinné členství ve vztahu

vyjadřuje možnost nezávislé existence entity daného typu na entitě druhého typu. Naopak, je-li minimum > 0, je entita existenčně závislá na jiné entitě.

Příklad:

VÝPUJČKA (ČTENÁŘ:(0,4), EXEMPLÁŘ:(0,1))

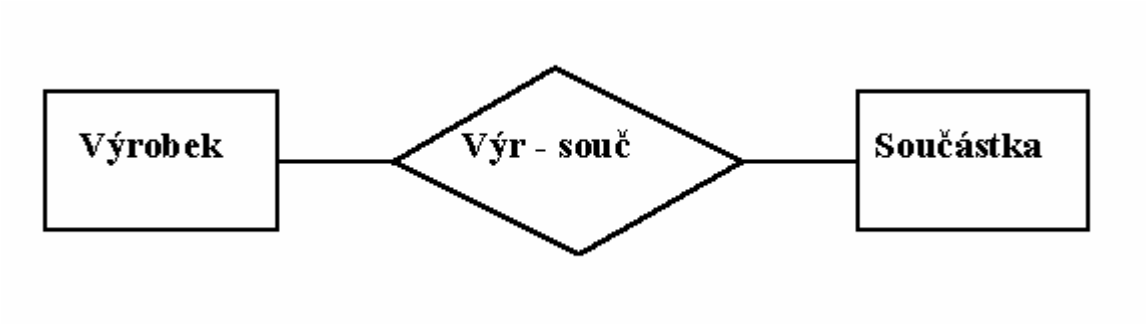
Konkrétní čtenář nemusí mít vypůjčenou žádnou knihu, může mít současně vypůjčeny až 4 exempláře.

Konkrétní exemplář může být buď v regálu nebo je půjčen vdaném okamžiku jen jednomu čtenáři.

Dekompozice M:N vztahu

Konceptuální model sice připouští kardinalitu N:M, ovšem z hlediska dalšího zpracování směrem k vytvoření schématu databáze musíme obvykle tuto kardinalitu N:M eliminovat. Situace se řeší dekompozicí, což znamená zařazení další entity. Tato entita je označována jako začlenění.

Příklad:

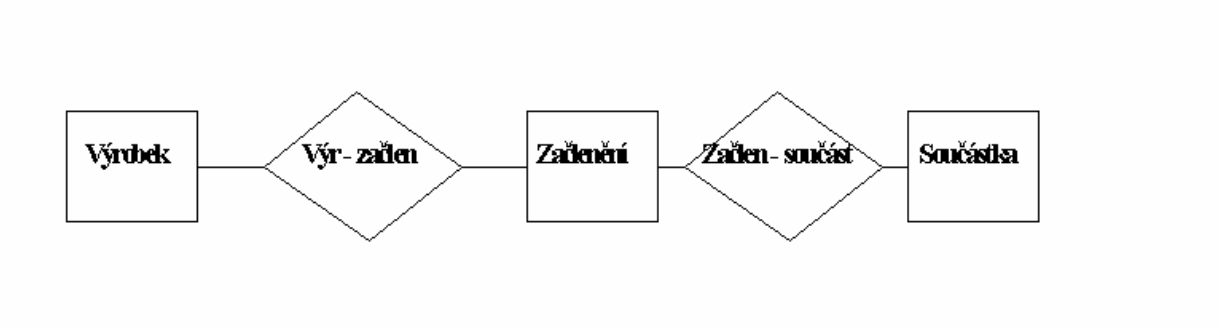


Výrobek	Součástka
V1	S1
V1	S4
V2	S4
V2	S2
V2	S3
V3	S3

Zavedeme entitu **Začlenění**

Výrobek	Začlenění	Součástka
V1	Z1	S1
V1	Z2	S4
V2	Z3	S4
V2	Z4	S2
V2	Z5	S3
V3	Z6	S3

Odpovídající E-R diagram:



2.1.2.4 Přiřazení popisných atributů

Entity a vztahy jsou popsány úplně, až když je jim přidělena množina popisných atributů.

- Pro každou entitu je nutno vytvořit samostatnou tabulku atributů, kde se uvádí jméno atributu a IO.
- Typ atributu se zadá hodnotovou množinou (doménou).
- U klíčového atributu je třeba zadat příznak pro identifikační klíč.
- Někdy se určuje, zda atribut může mít prázdnou hodnotu. Obvykle se označuje NULL.

Je zřejmé, že identifikační klíč nesmí nabývat hodnoty NULL.

Kombinace E-R diagramu a tabulek, popisujících atributy jednotlivých entit a vztahů, tvoří úplné schéma E-R modelu.

 **Příklad:**

Entity:

PACIENT(RODNE ČÍSLO, JMÉNO, ADRESA, VÁHA, VÝŠKA, POČET_LŮŽEK...)

POKOJ(ČÍSLO POKOJE,POČET_LŮŽEK,LOKALITA...)

Pro všechny entity je třeba vytvořit následující tabulku.

Tabulka pro entitu PACIENT:

PACIENT	RODNÉČÍSLO	JMÉNO	ADRESA	VÁHA	VÝŠKA ....
typ	N:10	X:25	X:25	N:3	N:3
NULL	ne	ne	ano	ano	ano
klíč	ano	ne	ne	ne	ne

Vztahy:

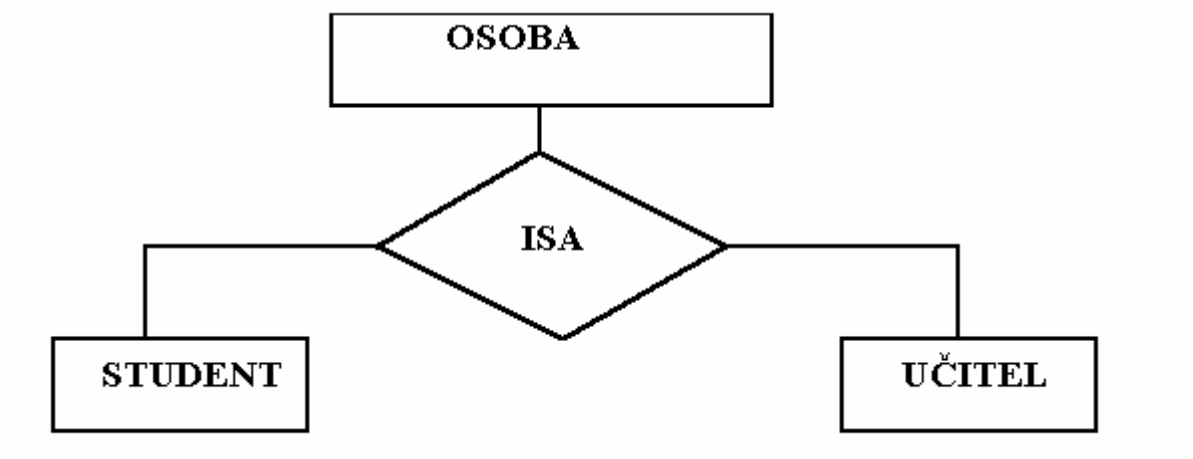
JE_UMÍSTĚN_NA (PACIENT, POKOJ, N:1)

resp.JE_UMÍSTĚN_NA(PACIENT:(1:1), POKOJ:(0:N))

2.1.2.5 ISA hierarchie, podtypy entit

Speciální atributy představují v konceptuálním modelování takové atributy, které danému typu entity přiřazují jeho nadtyp. Atribut je pak podtypem svého nadtypu. Jde o tzv. ISA-hierarchii.

Graficky se ISA-hierarchie zobrazuje takto:



Abstraktní entitní typ (entita) OSOBA se zavádí z praktických důvodů proto, že existuje řada společných atributů entit UČITEL a STUDENT a proto má smysl zavést jediný společný entitní typ. Pak je třeba připustit existenci hodnot NULL u těch atributů, které jsou irelevantní s ohledem na konkrétní typ osoby. Entitní typy (entity) STUDENT a UČITEL jsou podtypy entitního typu OSOBA. Platí STUDENT JE OSOBA (anglicky IS A). V oblasti umělé inteligence se tomuto vztahu říká ISA vztah.

Neatomické atributy

Konceptuální model, na rozdíl od např. relačního modelu (jak se dovíte později), připouští neatomické atributy. V souvislosti s návazností na příslušný databázový model, konceptuální model má nástroje, jak strukturu atomických atributů vytvořit. Atributy, které lze rozčlenit, se nazývají skupinové (grupped). Některé konceptuální modely připouštějí zavedení vícehodnotového atributu. Např. Autor knihy může být vícehodnotový, tzn., že může existovat více autorů jednoho titulu. Přitom dopředu nevíme, kolik hodnot tento atribut bude mít. Jedná se tedy o vektor proměnné délky.

Korektní konceptuální schéma

- Žádný entitní typ nemá v konceptuálním schématu více než jeden zdroj ISA hierarchie.
- ISA vztahy netvoří v E-R diagramu orientovaný cyklus.
- Identifikační typy vztahů netvoří v E-R diagramu orientovaný cyklus (jinak by jeden typ entity byl identifikován pomocí sebe sama) .
- yp entity v ISA hierarchii, který není zdrojem není identifikačně závislý na žádném entitním typu (je totiž již identifikován svým zdrojem ISA hierarchie).
- Jména typů entit a vztahů jsou jednoznačná globální jména, jména atributů jsou jednoznačná lokální jména.
- Je-li typ entity zdroj ISA hierarchie, pak má identifikační klíč. Ostatní typy v ISA hierarchii nemají identifikační klíč.

 **Shrnutí:**

Základem tvorby jakékoliv databáze je vytvoření jejího schématu. Abychom byli schopni schéma databáze vytvořit, musíme sestavit model reality. Na nejvyšší úrovni abstrakce (nejblíže k modelované realitě) se vytváří konceptuální model. Konceptuální model je tedy na začátku každé rozumně navržené databáze. Vytvořit konceptuální model znamená především znát modelovanou realitu a také znát zásady tvorby konceptuálního modelu. Koncepty (pojmy) konceptuálního modelu jsou entity, vztahy a atributy. U entit je nejdůležitější se zabývat jejich identifikací (identifikační klíč), u vztahů pak typem vztahů a kardinalit. Typy vztahů pak automaticky definují cizí klíče závislých entit. Pro jednotlivé atributy pak je nutné definovat množiny jejich možných hodnot (domény). Konceptuální model lze zapsat buď v lineárně textovém zápisu nebo pomocí tzv. E-R diagramu. Nejnáročnější pasáží tvorby konceptuálního modelu je nalezení entit. Proto je nutné se zabývat všemi výseky reality, které jeví známky samostatnosti v existenci či chování. To jsou potencionální entity. Velmi důležité je si uvědomit, co jsou ony abstrakce, o nichž budeme v budované databázi uchovávat data.

 **Kontrolní otázky:**

1. Zakreslete E-R diagram se dvěma entitami, které vstupují do existenčního typu vztahu.
2. Vysvětlete rozdíl mezi existenčním a identifikačním typem vztahu.
3. Charakterizujte entitní, doménovou a referenční integritu.

 **Pojmy k zapamatování:**

Entita

Vztah

Atribut

Doména

Instance

Identifikační (primární) klíč

Integritní omezení

Existenční a identifikační závislost

Kardinalita vztahu

ISA hierarchie

Cizí klíč

 **Průvodce studiem:**

Jste u konce velmi náročné kapitoly. Pokud jste byli schopni odpovědět na kontrolní otázky a jsou Vám jasné pojmy k zapamatování, jste na nejlepší cestě k pochopení problematiky konceptuálního modelování. Dejte si pauzu a pak se pusťte do studia CASE nástroje Erwin. CASE nástroje na datové modelování, k nimž Erwin patří, Vám umožní efektivním způsobem vytvářet konceptuální a později datové modely. Je však nutné, abyste si osvojili základní dovednosti v práci s tímto softwarovým nástrojem. Pojmový aparát, který Erwin používá je Vám již znám. Následující kapitola Vám přinese návod na to, jak s Erwinem pracovat a jak v něm konceptuální model reality zakreslit.

2.2 CASE NÁSTROJ ERWIN

Cíl:

Cílem je naučit čtenáře obsluhovat CASE nástroj pro datové modelování Erwin. Pojmy, které se vyskytují v textu této kapitoly, nejsou vysvětlovány, jejich znalost se již předpokládá a čtenář se je naučil v předchozích kapitolách. Po prostudování této kapitoly bude čtenář umět nakreslit konceptuální model v prostředí CASE nástroje Ewin. Je vhodné, ne-li nezbytné, si

při studiu tohoto textu spustit Erwin a všechny vysvětlované pasáže si rovněž ihned prakticky vyzkoušet.

Klíčová slova:

entita (objekt), vztah (relationship), nezávislé a existenčně závislé objekty (entity), identifikačně závislý objekt (entita), slabý entitní typ, členství ve vztahu, kardinalita, primární klíč, cizí klíč, datový typ, index, target server.

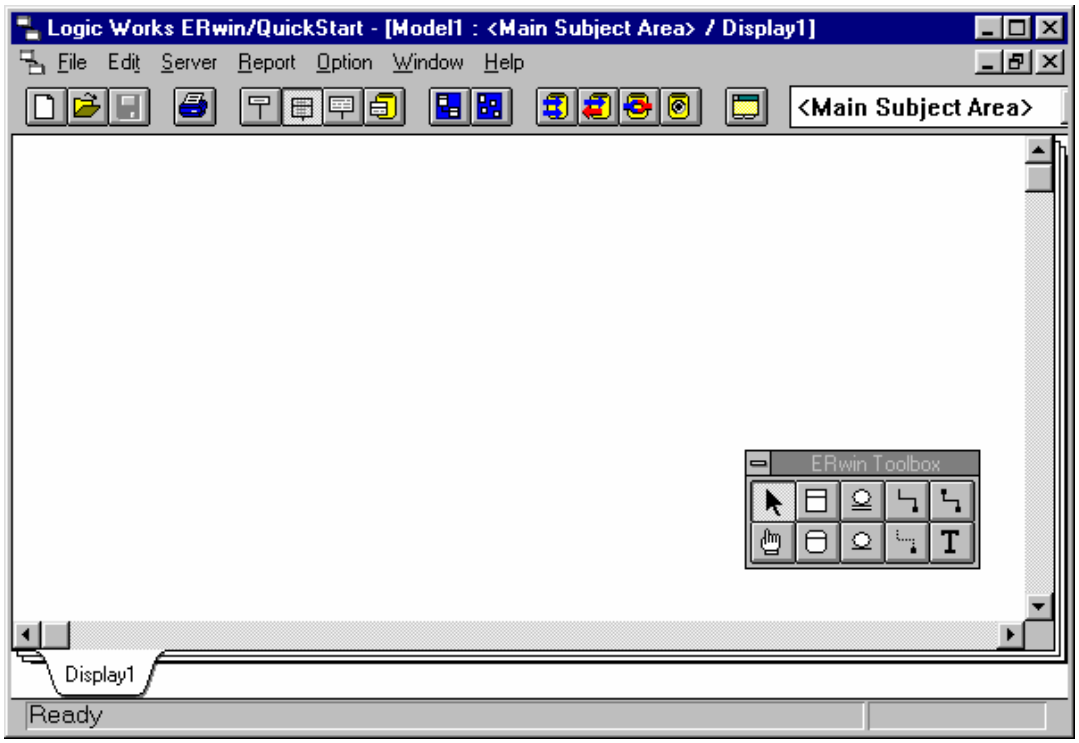
Průvodce studiem:

Nejdříve Vás musím upozornit, že by bylo ztrátou Vašeho času začít číst tuto kapitolu, pokud pojmy, které jsou uvedeny v klíčových slovech ještě neznáte nebo přesně nevíte, co znamenají. V takovém případě si raději ještě jednou přečtěte předcházející kapitolu (2.1.). Návod na obsluhu nástroje Erwin je velice stručný a popisuje jen základní charakteristiky a způsob obsluhy systému Erwin. Není možné popsat všechny funkce, kterými software disponuje. Pokud by se stalo, že příslušnou funkci v návodu nenajdete, pokuste se využít nápovědy / Helpu, který je součástí software. Pokud se v textu objeví podtržený pojem, je Vám tím dáváno na vědomí, že pojem je vysvětlen v předchozím textu a tudíž s jeho pravým významem se můžete (lépe řečeno jste se mohli) seznámit v předchozím textu. Abyste měli celý návod na obsluhu Erwina pohromadě, jsou v této kapitole popsány i aktivity (kap. 2.2.2.), které budou teoreticky vysvětleny až v následujících kapitolách. Na místo, kde máte přerušit čtení, budete upozorněni. Po vysvětlení příslušného teoretického aparátu Vás opět odkážu na dočtení této kapitoly.

Pokud ještě nemáte na svém počítači nainstalován produkt Erwin, udělejte tak a při studiu si ihned veškeré funkce ověřujte.

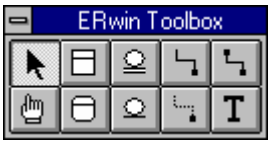
2.2.1 Konceptuální model v Erwinu

Po spuštění programu ERWIN se objeví následující obrazovka:



Budeme hlavně pracovat s těmito nástroji:

- Hlavní nabídka.
- File, Edit, Server, Report, Option, Windows.
- Toolbar.
- Jednotlivá tlačítka pod hlavní nabídkou.
- Toolbox.
- Erwin Toolbox je umístěn na pracovní ploše.



Nejdříve si vysvětlíme jednotlivá tlačítka Toolboxu, která budeme potřebovat:

 Umožňuje uchopovat objekty (entity)

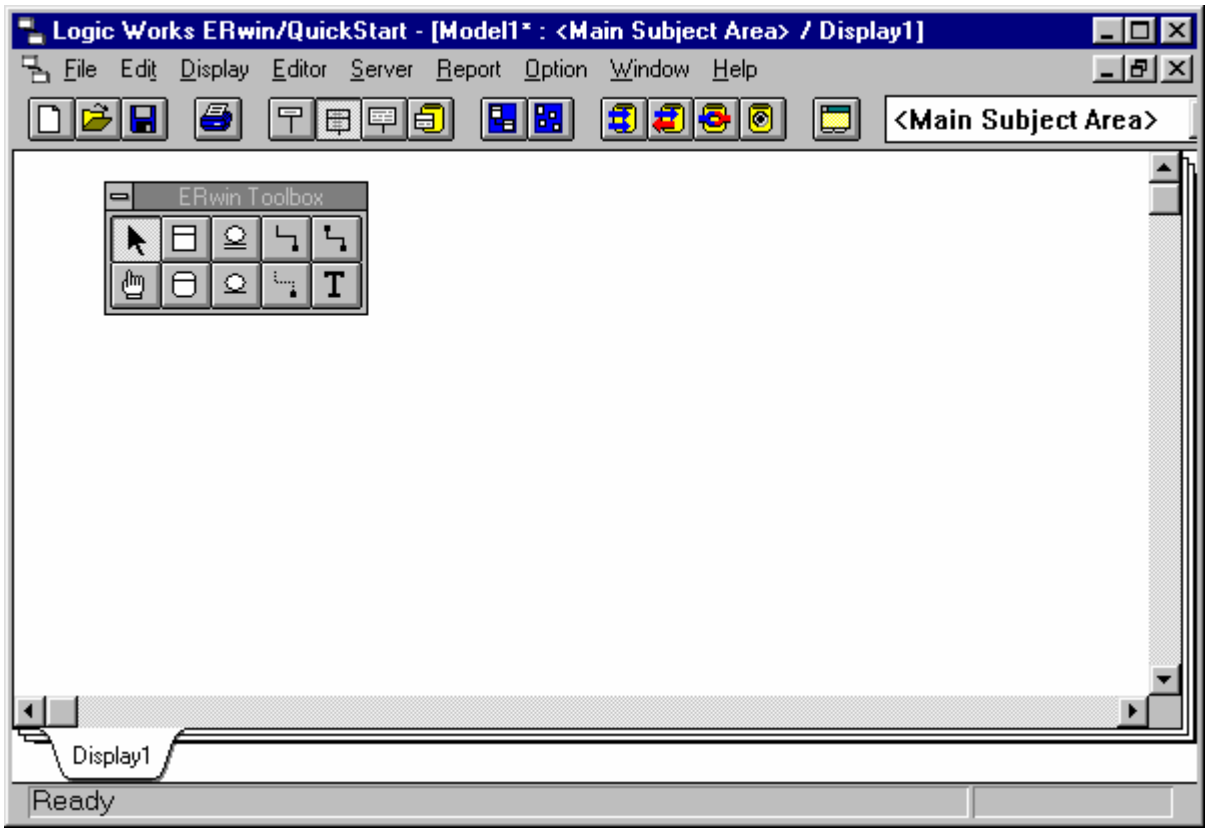
 Umožňuje zakreslovat nezávislé a existenčně závislé objekty (entity). Pokut má obdélník zakulacené rohy  , představuje identifikačně závislý objekt (entitu), někdy označovanou slabý entitní typ .

 Umožňuje zakreslit identifikační závislost. Objekt, u kterého je plné kolečko, představuje závislý objekt, objekt s prázdným kolečkem je nezávislý. Identifikačně závislý objekt má povinnou účast ve vztahu. Nezávislý objekt má nepovinnou účast ve vztahu.

 Umožňuje zakreslit existenční závislost. Objekt, u kterého je plné kolečko , představuje závislý objekt, objekt s prázdným kolečkem je nezávislý. Existenčně závislý objekt má povinnou účast ve vztahu. Nezávislý objekt má nepovinnou účast ve vztahu.

 Umožňuje zakreslit vztah, kdy o žádném objektu se nedá říci, že by byl závislý resp. nezávislý. Jedná se o objekty, které mají nepovinné členství ve vztahu. Pomocí tohoto typu vztahu modelujeme kardinalitu N:M

Rozbalení dalších nabídek v hlavní nabídce:

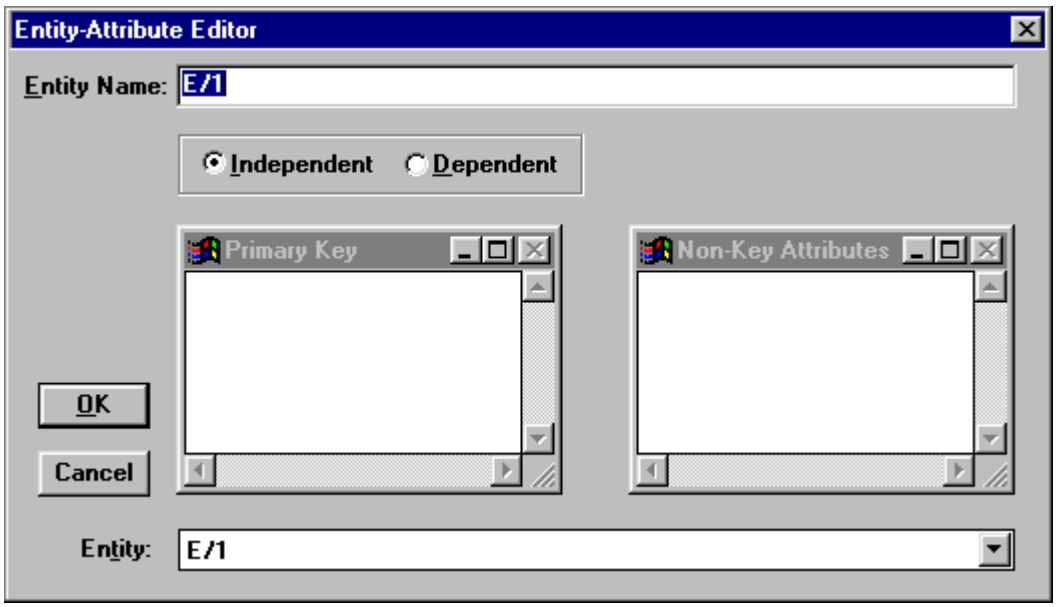


Po zmáčknutí volby Option se rozbalí nabídka. Zaškrtněte volby Show Display Menu a Show Editor Menu. Pokud to provedete správně, přibude v hlavní nabídce volba Editor a Display.

Z volby Editor budeme používat tyto volby:

- Entity-Attribute, pro definování objektů, jejich vlastností a vztahů.
- … Database Scheme, pro definování datových typů jednotlivých atributů.
- … Index, pro definování indexů.

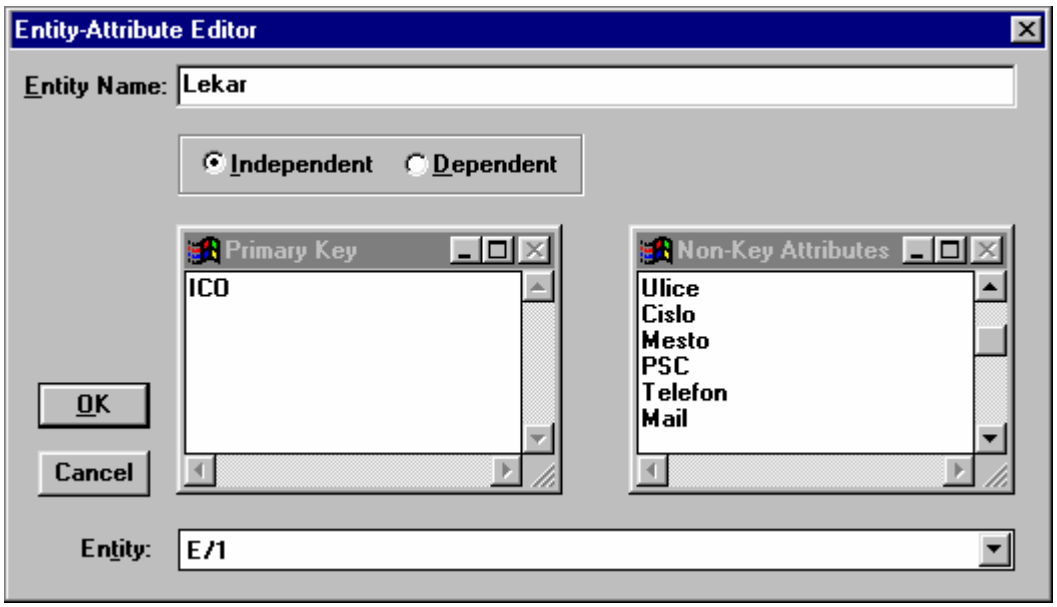
Definování objektů, vztahů a atributů:



Do pole **Entity Name** zapíšeme jméno objektu, do pole **Primary Key** napíšeme atribut nebo skupinu atributů, které budou Identifikačním klíčem, do pole **Non-Key Attributes** napíšeme ostatní, tzv. popisné atributy.

Takovým způsobem nadefinujeme všechny objekty v modelu.

Příklad nadefinovaného objektu by mohl být:



Průvodce studiem:

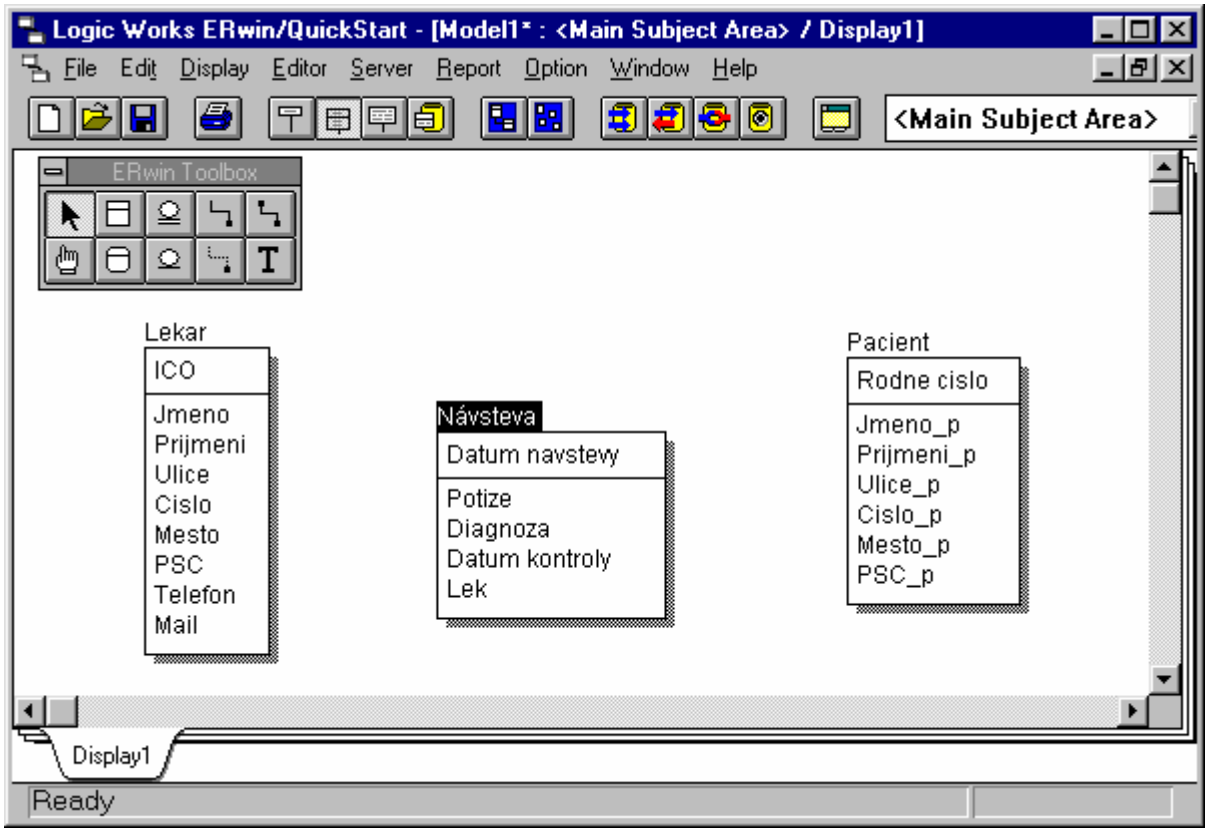
Nepoužívejte diakritiku (háčky a čárky), jedná se o demo verzi Erwina, která diakritiku neumí rozpoznat. Vyhnete se tak zbytečným potížím. Jistě jste si všimli, že Erwin používá místy trošku jiné termíny, než je standard. Proto je třeba si uvědomit:

Primary key = identifikační klíč. V dalším textu, který se bude týkat datových modelů, budeme také používat označení pro atribut/y, který identifikuje entitu pojem primární klíč. Na úrovni konceptuálního modelování je však standardně používán pojem identifikační klíč. Dále si jistě všimnete, že Erwin označuje jako Independant pouze entitu identifikačně nezávislou. Ve standardním pojetí je ovšem třeba připustit, že Independent je rovněž entita existenčně nezávislá. Snad Vás tato nejednotnost příliš nezmate.

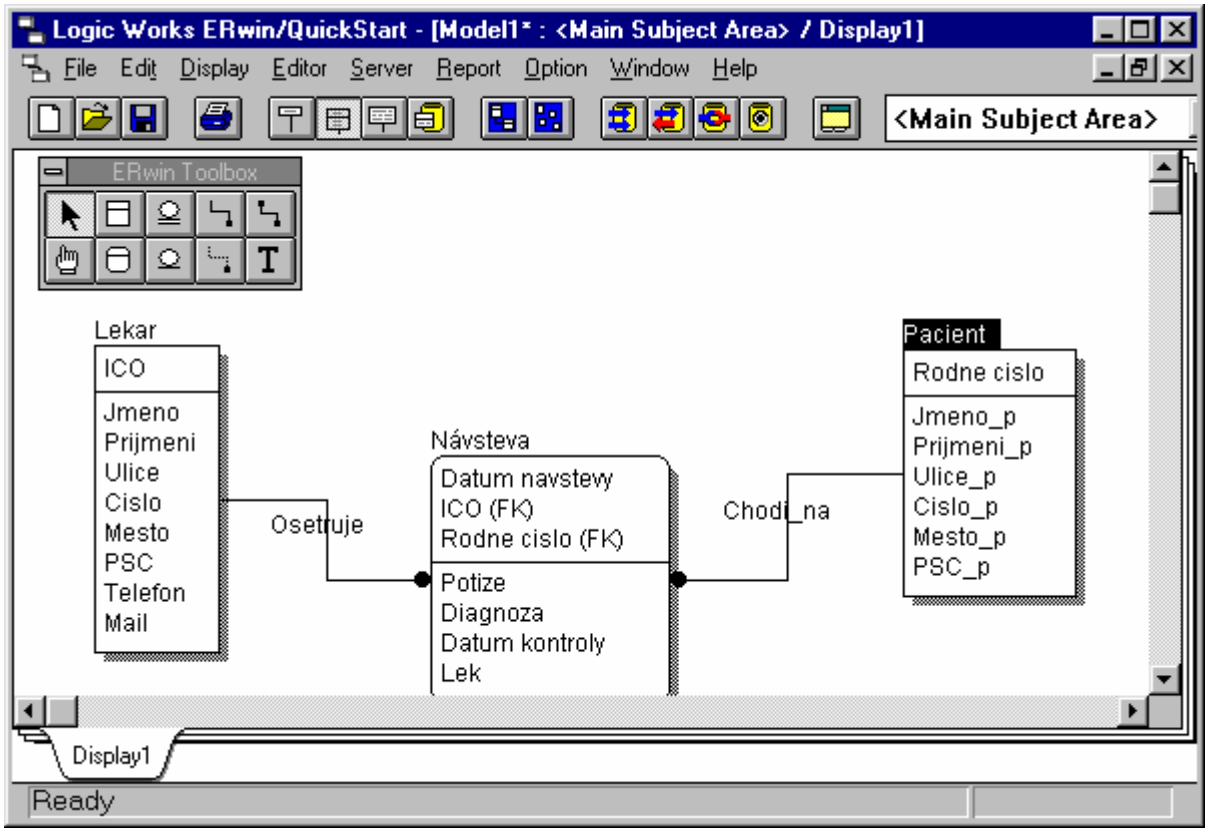
Mezi atributy objektu nezapisujte atributy jiných objektů. Ty se do objektu natáhnou automaticky po zakreslení vztahů mezi objekty (myslí se tím identifikační klíče).

Příklad:

Jednoduchý model objektů by mohl být následující:



Všimněte si, že objekt “Navsteva” nemá dostatečný Identifikační klíč. “Datum Navstevy” totiž ještě nemůže identifikovat každou instanci objektu “Navsteva”. Další klíčové atributy se do objektu “Navsteva” automaticky převedou po zakreslení vztahů mezi objekty “Lekar”, “Navsteva” a “Pacient”.V těchto vztazích bude totiž “Navsteva” závislá jak na “Pacientovi”, tak na “Lekari”. Není možné uskutečnit návštěvu, pokud není pacient resp. lékař. Tuto skutečnost zakreslíme do modelu použitím vhodných vztahů. V našem případě se bude jednat o vztahy identifikační, protože “ICO” lékaře a “Rodne cislo” pacienta budou nápomocné k identifikaci instance objektu “Navsteva”.Tento model představuje základní definici objektů, atributů a vztahů na úrovni Entity-Attribute. Pokud se Vám nezobrazují názvy vztahů, zadejte volbu: Display/Verb Phrase.



Shrnutí:

Při tvorbě konceptuálního modelu pomocí CASE nástroje Erwin je třeba nejdříve nalézt a zakreslit entity modelované reality. U každé entity najít identifikační (primární) klíč. Dále rozhodnout, které entity do jakých vztahů vstupují a zakreslit vztahy. Erwin automaticky rovněž přenese primární klíč nezávislé entity do závislé entity, kde se tento stane cizím klíčem. Pokud použijeme identifikační typ vztahu, cizí klíč se umístí do primárního klíče závislé entity (slabá entita), pokud použijeme existenční závislost, cizí klíč se umístí mezi ostatní atributy. Nakonec pro všechny entity definujeme popisné atributy. Nejdůležitější při celém procesu tvorby modelu je mít stále na paměti modelovanou realitu a snažit se model udělat co nejvěrnější. Samozřejmě s přihlédnutím k určití míře zjednodušení, která vyplývá z požadavků uživatele na budoucí databázi.

Kontrolní otázky:

1. Co je to Identifikační, resp. primární klíč?
2. Co je to cizí klíč?
3. Jaké typy vztahů znáte?

Pojmy k zapamatování:

Identifikační / primární klíč (Primary key)

Popisné / neklíčové atributy

Závislá entita (existenčně, identifikačně)

Nezávislá entita

Existenční typ vztahu

Identifikační typ vztahu

Cizí klíč (Foreign key)

Průvodce studiem:

Na této úrovni máme vytvořen konceptuální model. Následující aktivity se již budou týkat datového modelu. Proto zde přerušte čtení, dejte si pauzičku a pak si prohlédněte vyřešené příklady, které jsou uvedeny v následující kapitole.

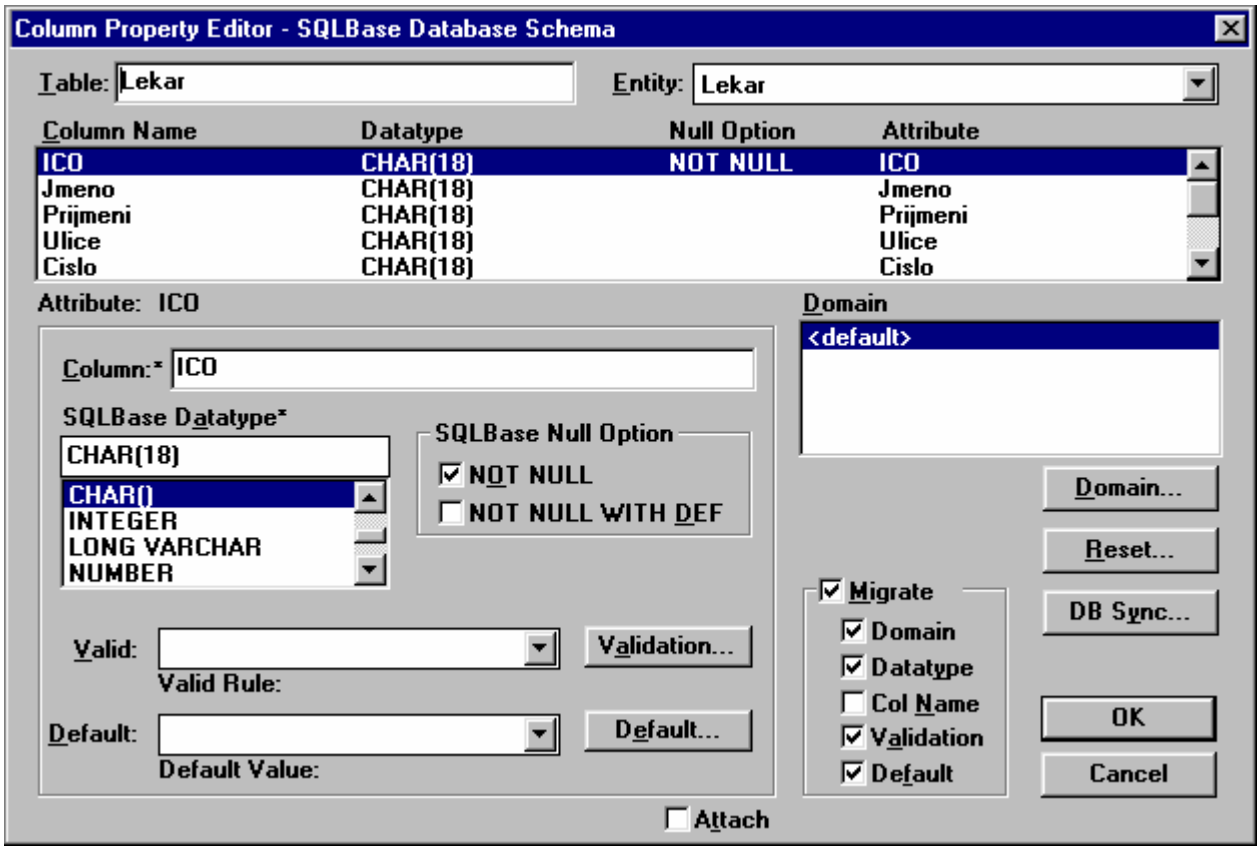
2.2.2 Datový model v Erwinu

Průvodce studiem:

Datový model je na nižší úrovni abstrakce. Znamená to, že již musíme přihlížet ke skutečnosti, v jakém prostředí (Systém Řízení Báze Dat) budou data uchovávána, aktualizována, organizována a řízena. V současné době existuje několik typů datových modelů. Z historického hlediska se jedná o modely síťové a hierarchické, v současnosti nejvíce používaný je model relační, jehož pozici se již delší dobu snaží získat model objektový. Nicméně většina současných databází je založena na relačním datovém modelu, i my se budeme v následujících textech věnovat hlavně modelům relačním. Máme-li vytvořen konceptuální model, musíme jej (jako 2. krok tvorby schématu databáze) transformovat na model datový. V našem případě to bude relační datový model. S transformací do relačního datového modelu jsou spojeny následující činnosti:

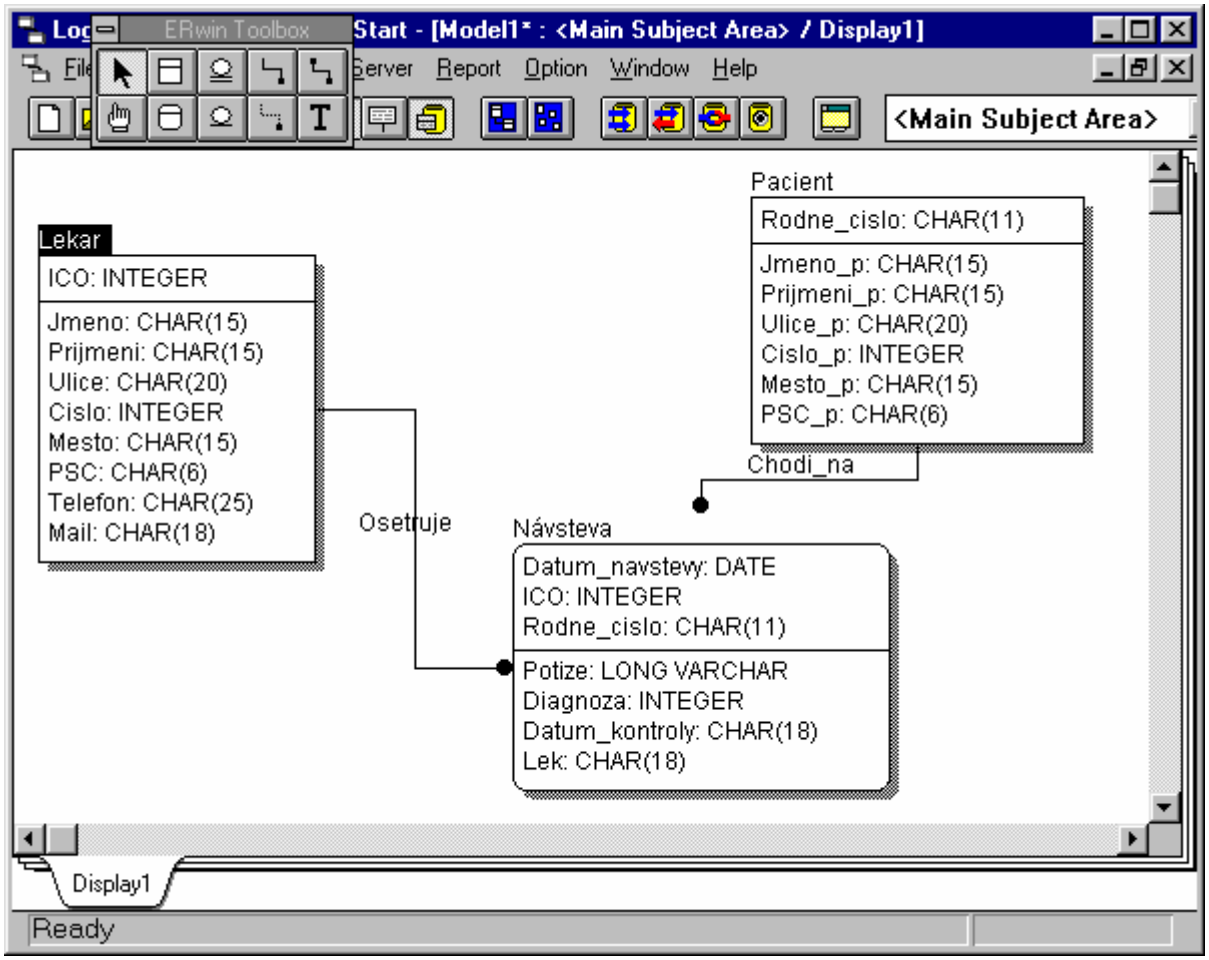
Definování datových typů jednotlivých atributů:

Pro definování datových typů je třeba se v hlavní nabídce přepnout na volbu: Editor/…Database Scheme. Tři tečky znázorňují tu skutečnost, že na tomto místě se mohou vyskytovat různé tzv. target servery. Target server je vlastně systém řízení báze dat (SŘBD), ve kterém se budou modelovaná data zpracovávat. Proto v této fázi se musíme rozhodnout, který SŘBD budeme dále používat. Výběrem SŘBD se již dostáváme k definovaní datového modelu. Předcházející kroky byly součástí tzv. konceptuálního modelování. Způsob výběru Target serveru je popsán na konci textu. Máme-li vybrán Target server, nastavenou volbu Editor/…Database Scheme a vybereme libovolný objekt, zobrazí se nám následující dialogové okno, do kterého budeme moci zapisovat pro jednotlivé atributy jejich datové typy.



Všimněte si, že Erwin implicitně nastavuje veškeré datové typy atributů na CHAR(18). To samozřejmě budeme potřebovat změnit. Proto pro každý vybraný atribut vybereme z nabídky datových typů ten, který se nám zdá pro daný atribut nejvhodnější.

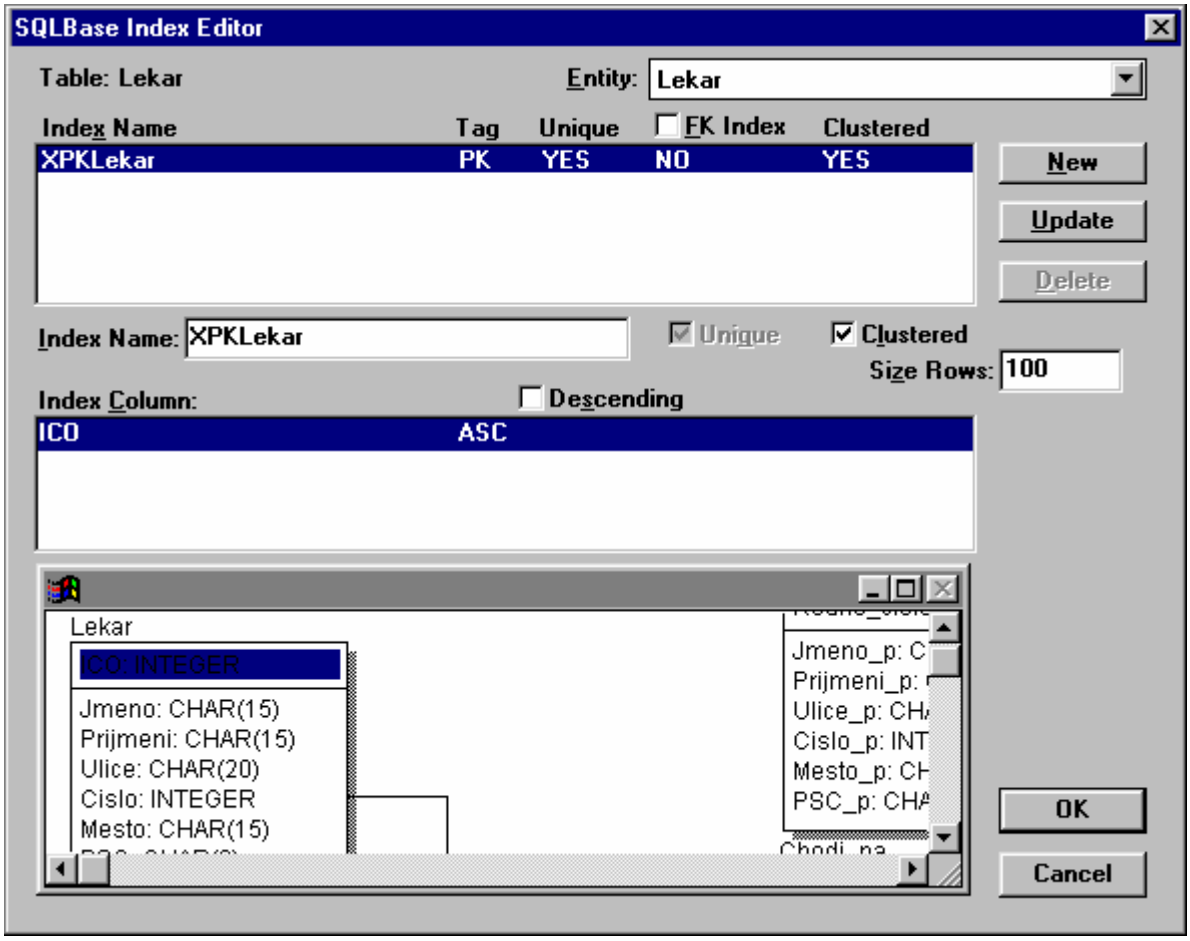
Chceme-li si nadefinované datové typy zobrazit, vybereme tlačítko Physical view z Toolbaru.



Definování indexů:

Indexy jsou pomocné datové struktury (pro jednoduchost si je můžeme představit jako tabulky), které potřebujeme pro rychlé vyhledávání v databázi. Na úrovni modelování si můžeme vybrat atributy, na které budeme indexy vytvářet. Tyto atributy se nazývají vyhledávací klíče.

Nejdříve je třeba volbou Editor nastavit na vytváření indexů a to je volba …Index. Význam tří teček je stejný, jako bylo uvedeno výše. Po zadání volby … Index a vybrání kterékoliv tabulky se nám objeví následující dialogové okno:

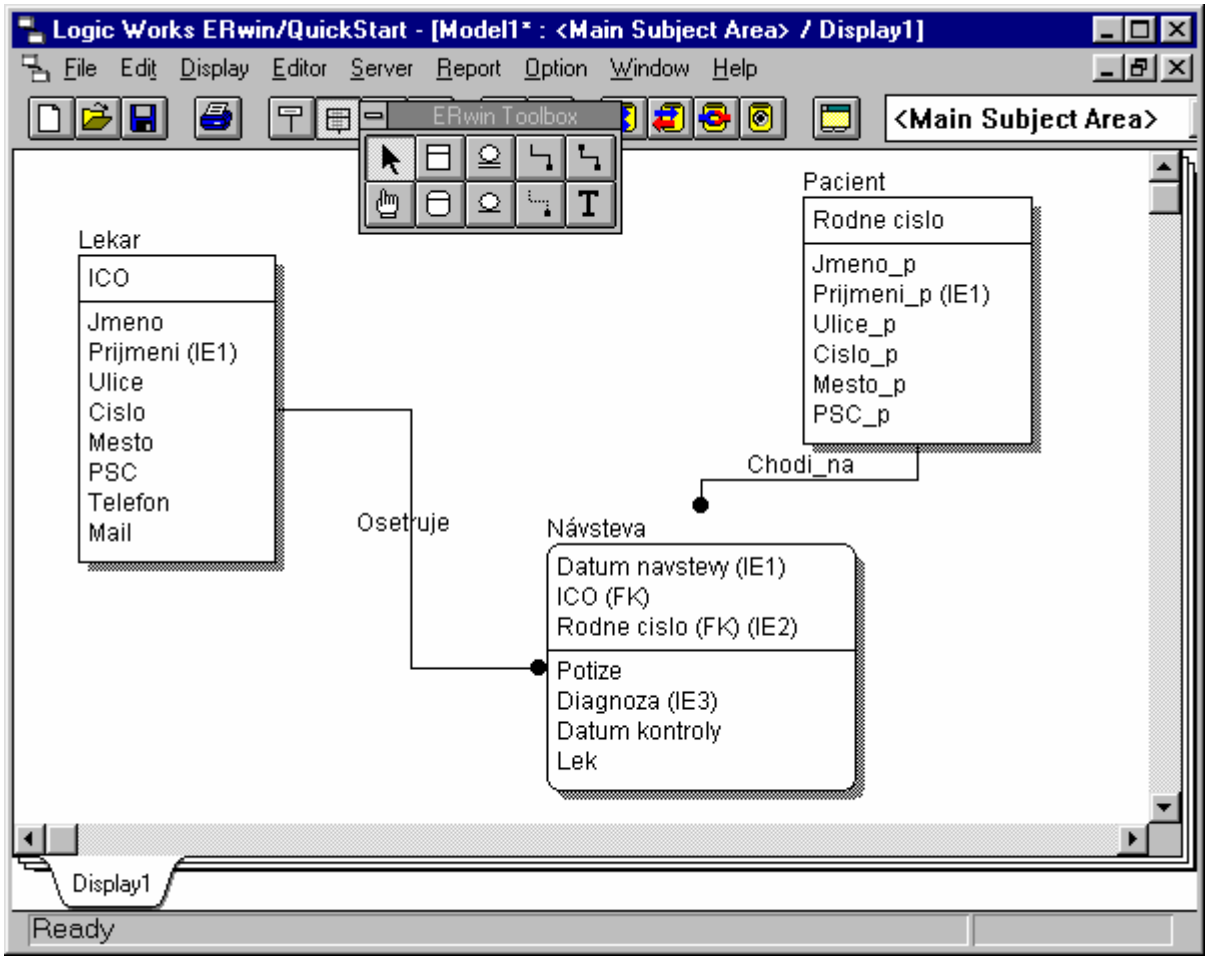


V tomto okně je možno vytvářet nové indexy volbou NEW. Po zadání NEW si vybereme atribut nebo skupinu atributů, na které chceme vytvořit index. Pro index lze nastavovat tyto charakteristiky:

- Clustered: umožňuje zadat, že daný index bude clusterovaný, hasovaný. Podrobne se s problematikou _hašování můžete seznámit v teoretické části kurzu. Volba Size Rows odkazuje na velikost alokovaného prostoru, který je vytvořen pro hašovaný soubor.
- Unique: pokud je volba unique zadána, pak se jedná o tzv. alternativní klíč a tudíž hodnoty tohoto klíče nemohou být duplicitní. Není-li volba unique vybrána, jedná se o klíč inverzní a tudíž hodnoty tohoto klíče mohou být duplicitní.
- Descenting: tato volba umožní zadat způsob třídění hodnot v indexu. Implicitně je nastavena volba Ascending (vzestupné třídění). Můžete však vybrat volbu Descending (sestupné třídění).

Po zadáví všech vyhledávacích klíčů (resp. indexů) a zadání volby Display/Alternate key si můžete nastavení indexů prohlídnout.

Jedno z možných nastavení indexů je na následujícím obrázku.



V tomto okamžiku lze říci, že datový model je hotov.

Všimněte si 2 úrovní abstrakce, kterých jsme použili. Na 1. Úrovni abstrakce jsme definovali tzv. E-R model (konceptuální model), tj. zajímaly nás pouze objekty, atributy a vztahy. Na 2. Úrovni abstrakce jsme již začali definovat datové typy, ale ty mohou být různé pro různé systémy řízení báze dat (SŘBD). Proto bylo třeba nejdříve vybrat, který SŘBD (target server) bude tím, v němž budeme data zpracovávat. Tím se ovšem posouváme na jinou úroveň abstrakce. Ve 2. Úrovni vytváříme tzv. datový model. Třetí úrovní abstrakce je generování datových schémat, což odpovídá imlementaci souborů a databází, jedná se tedy o fyzickou (implementační) úroveň.

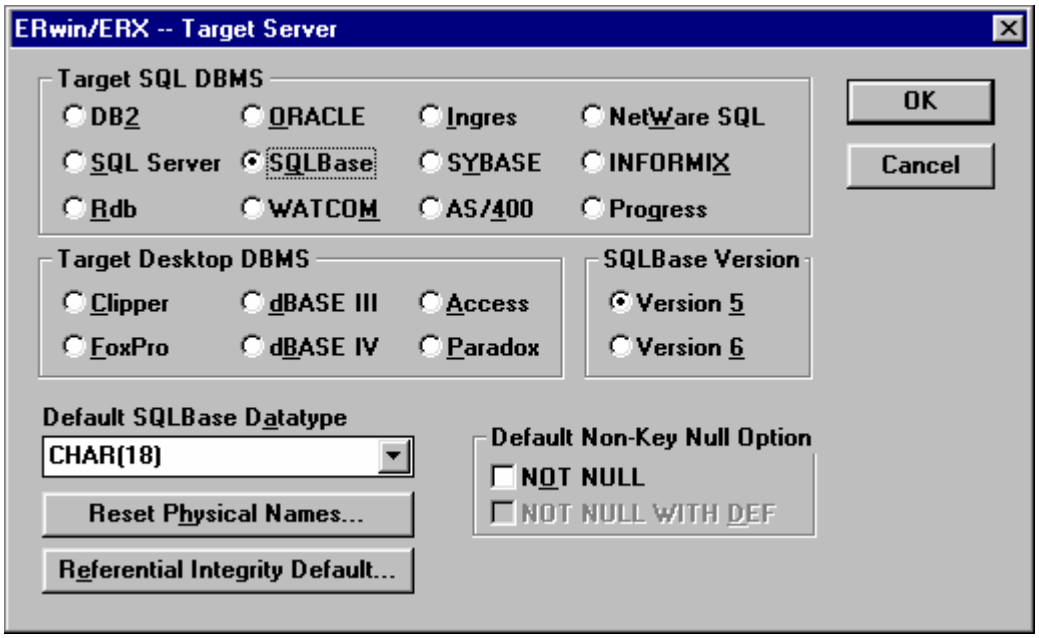
Vygenerovat schémata databáze můžeme dvěma základními způsoby:

- Přímo z Erwina volbou generovat (tato volba není v demo verzi přístupná).
- Přes tzv. skript, což je soubor příkazů jazyka SQL spustitelný v příslušném SŘBD.

Obě tyto možnosti zabezpečuje volba Server/ Schema Generation.

Definování Target serveru:

Vybráním volby Server/Target server se nám otebře následující dialogové okno:



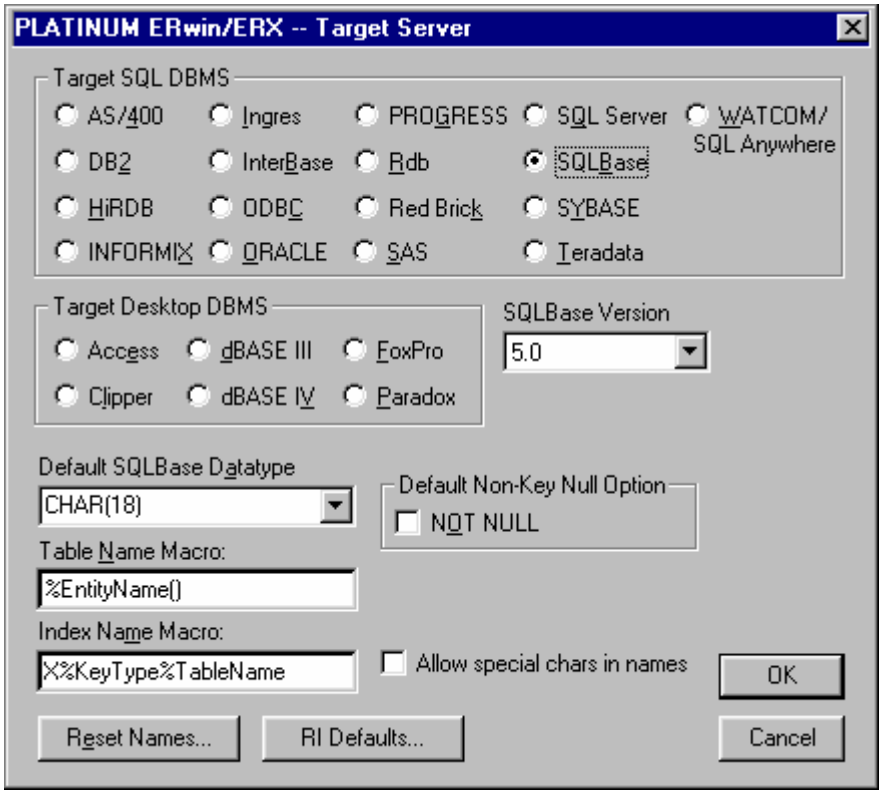
Máme možnost vybírat ze dvou skupin target serverů (SŘBD):

- SQL servery jsou takové SŘBD, které pracují na bázi architektury Klient/Server.
- Desktop servery sice umožňují sdílet databázi, uloženou na jednom počítači, ovšem toto sdílení nefunguje na principu Klient/Server.

Pro více informací musíte sáhnout po skriptech a teoretických textech.

Zajisté jste si všimli starých verzí nabízených SŘBD. Aktuální verze Erwina podporuje všechny frekventované SŘBD v nejnovějších verzích. Aktuální ostrou verzi tohoto CASE nástroje Vám bohužel nemohu poskytnout.

Pouze pro srovnání zde uvádím výběr Target serverů novější verze Erwina:



 **Kontrolní otázky:**

1. Co je to datový tap atributu a jak závisí na vybraném target serveru (SŘBD)?
2. Co je to index a jak souvisí s pojmem vyhledávací klíč?
3. Jaké typy target serverů nabízí Erwin?

 **Pojmy k zapamatování:**

Datový typ atributu

Vyhledávací klíč

Index

Target server

 **Průvodce studiem:**

Po prostudování tohoto stručného návodu a příslušných teoretických kapitol byste měli být schopni vytvořit konceptuální model zadané reality a transformovat jej do relačního datového modelu dle vybraného SŘBD (target serveru). Pro prověření znalostí si ještě pečlivě prostudujte kapitolu 3.7. - řešené příklady a pak se pusťte do řešení korespondenčního úkolu. Jestliže jste byli při čtení pozorní a všechny funkce si prakticky vyzkoušeli, jistě nebudete mít problémy s jeho řešením.

2.3 ŘEŠENÉ PŘÍKLADY

Cíl:

Cílem kapitoly Řešené úkoly je demonstrovat proces tvorby konceptuálního modelu na zadaném výseku reality. Čtenář by se měl zamýšlet hlavně nad tím, jak:

- *Hledat entity z neformálního popisu reality.*
- *Hledat identifikační klíče entit.*
- *Hledat vzájemné vztahy mezi entitami.*
- *Vybrat popisné atributy, jako podstatné vlastnosti nalezených entit.*

Klíčová slova:

Neformální popis reality, konceptuální model reality, Erwin.

Průvodce studiem:

Věnujte, prosím, této kapitole mimořádnou pozornost. Jakmile Vám bude řešení uvedených příkladů jasné, můžete přistoupit ke korespondenčnímu úkolu, jehož zadání najdete v následující kapitole.

2.3.1 Popis modelované reality:

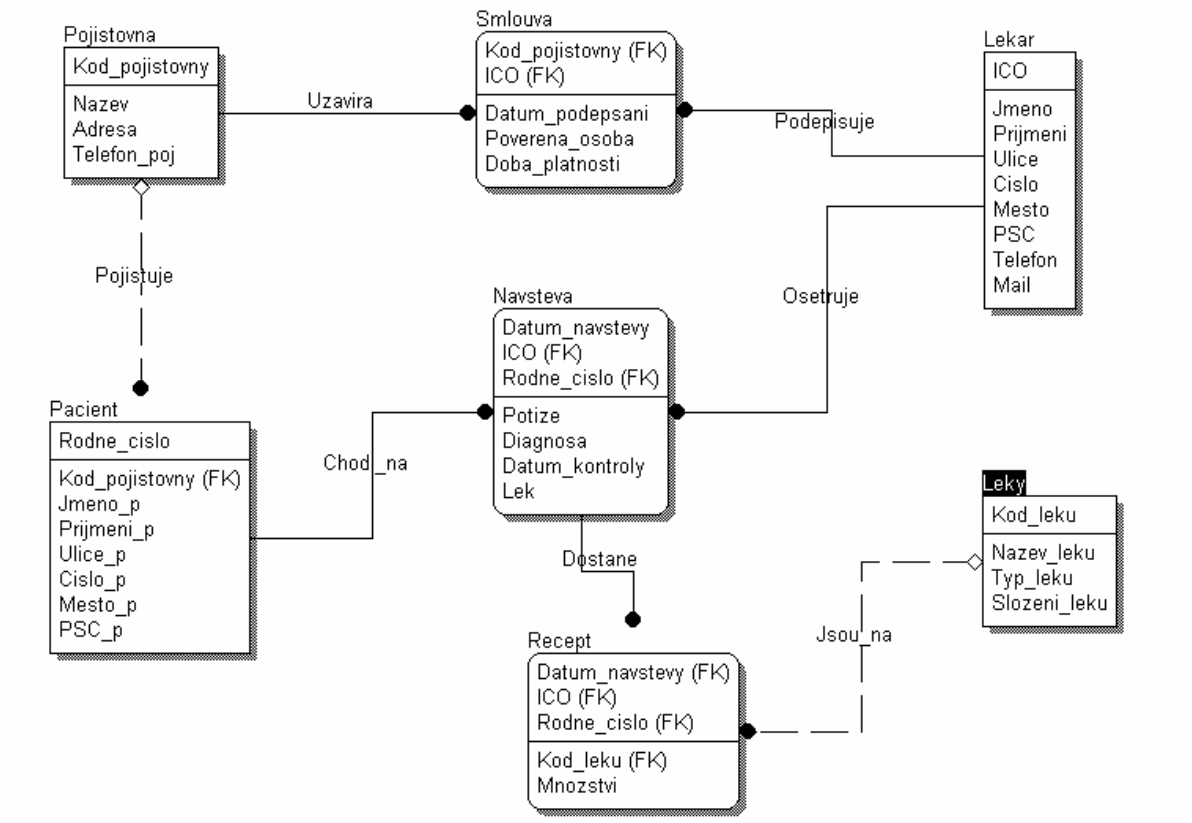
Následující text je jednoduchým popisem výseků realit, které by mohly být předmětem databázového modelování. Popis je velmi zjednodušen. Každé zadání obsahuje vždy možné řešení v prostředí Erwin.

Příklad:

Realita – ordinace u lékaře

Vytvořte model lékařské ordinace, který bude postihovat následující skutečnosti. Lékaři mají smlouvy s pojišťovnami a to tak, že více lékařů může mít smlouvu s toutéž pojišťovnou a zároveň jeden lékař může uzavřít smlouvu s více pojišťovnami. Pacienti jsou pojištěni vždy u jedné pojišťovny. Pacienti chodí na návštěvy k lékařům, na kterých lékaři stanoví diagnózu a předepíší recepty na léky. Atributy a klíče volte dle vlastní zkušenosti. Následuje jedno z možných řešení.

Konceptuální model ordinace

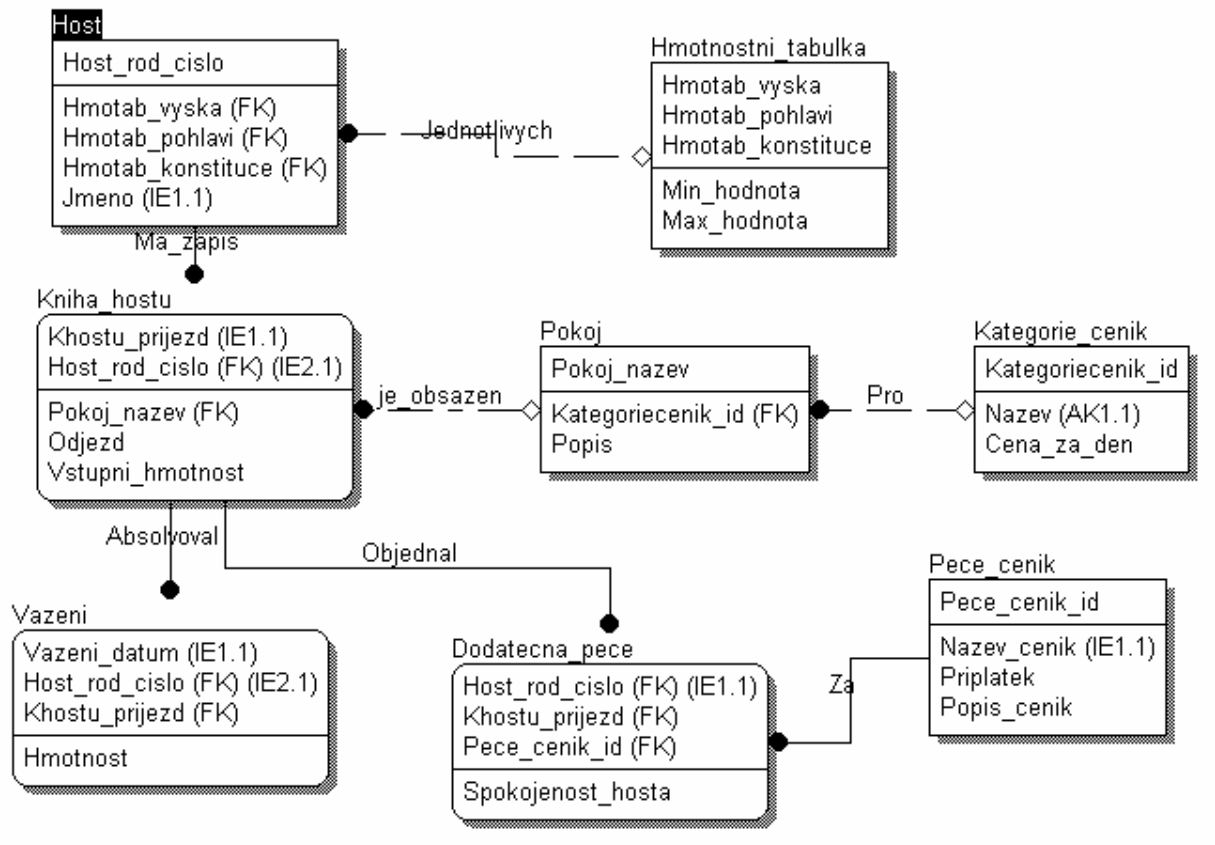


Příklad:

Realita – lázně

Vytvořte model lázeňské péče. Hosté jezdí do lázní a jejich pobyty jsou evidovány v knize hostů. Bude se jednat o lázně, kde bude u pacientů hlavně sledována jejich hmotnost v souvislosti s jejich konstitucí. Informace o různých konstitucích se nachází v hmotnostní tabulce. Hosté budou pravidelně váženi a naměřené hmotnosti budou zapisovány do tabulky vážení. Host si bude moci připlatit dodatečnou péči, která je nad rámec běžného programu. Bude nás zajímat, na kterých pokojích jsou kteří hosté umístěni. Následuje jedno z možných řešení.

Konceptuální model lázní

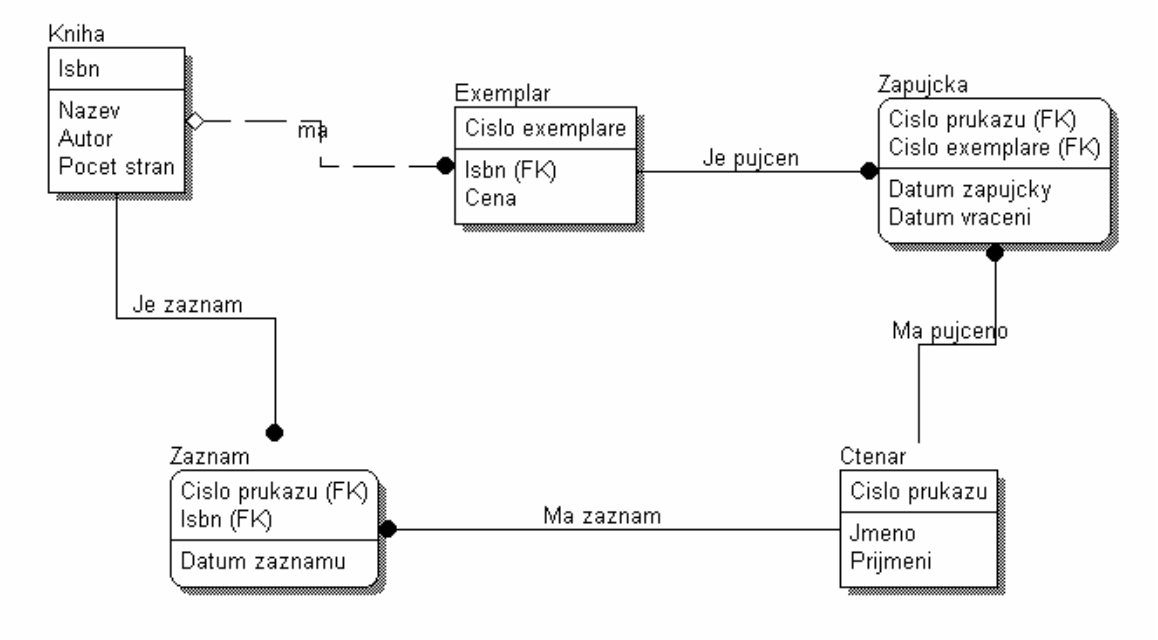


Příklad:

Realita – knihovna

Vytvořte model půjčovny knih v knihovně. Knihovna vlastní exempláře mnoha různých knih. Tyto knihy si mohou půjčovat čtenáři, kteří mají u této knihovny zřízen čtenářský průkaz. Je-li kniha k dispozici, čtenář si půjčí odpovídající exemplář. Pokud kniha momentálně k dispozici není, čtenář si může na tuto knihu vytvořit záznam, pomoci kterého si ji rezervuje, až k dispozici bude. Následuje jedno z možných řešení.

Konceptuální model knihovny



3 MODUL 3

3.1 RELAČNÍ DATOVÝ MODEL

Cíl:

Cílem této kapitoly je, aby se čtenář seznámil se zatím nejvyužívanějším datovým modelem, kterým je relační datový model. Kapitola přináší základní ideje, na kterých je model založen, definici relačního modelu dat, základní pojmy. Čtenář se rovněž seznámí se základními manipulačními prostředky, které lze v relačním datovém modelu nad daty využít. Po prostudování textu by měl mít čtenář představu o strukturách, do kterých se ukládají data v databázi, kterými jsou databázové relace. Dále by měl již naprosto přesně umět definovat pojmy uvedené v klíčových slovech. Co se týká databázových operací, čtenář by měl po přečtení této kapitoly o nich získat základní přehled. V následujících textech ještě bude databázovým operacím věnována značná pozornost.

Klíčová slova:

RMD, databázová relace, 1NF, doména, primární klíč, cizí klíč, operace relační algebry.

3.1.1 Úvod

Relační datový model (RDM) byl poprvé popsán v roce 1970 E.F. Coddem a je zatím nejrozšířenějším datovým modelem, na kterém je založen návrh a tvorba databází.

Základní ideje relačního modelu

- RMD důsledně odděluje data, která jsou chápána jako relace, od jejich implementace.
- Přístup k datům je symetrický, tj. při manipulaci s daty se nezajímáme o přístupové mechanizmy k datům.
- Pro manipulaci s daty jsou k dispozici dva silné prostředky - relační kalkul a relační algebra.
- Pro omezení redundance dat v relační databázi jsou navrženy pojmy umožňující normalizovat relace.

Průvodce studiem:

RMD má jediný konstrukt, kterým je databázová relace. Proto při transformaci konceptuálního modelu na model relační je třeba celou složitou realitu transformovat na množinu vcelku jednoduchých relací. Již tento fakt naznačuje, že bude skutečně velmi složité a někdy dokonce nemožné realitu natolik zjednodušit. Tato úvaha vede ke snaze vývoje datových modelů založených na objektových principech. To je ovšem v této kapitole ještě téma předčasné, přesto je dobré si uvedenou skutečnost uvědomit.

Základní definice RMD

Mějme množiny $\mathbf{D_1, D_2, D_3,.....D_n}$. Z každé vybereme 1 prvek. Tím vytvoříme uspořádanou n-tici. Kartézský součin $\mathbf{D_1 \times D_2...}$ je množina všech posloupností $\mathbf{(x_1,x_2,...)}$

kde x_1 je prvkem D_1 ... Relace je každá podmnožina kartézského součinu. Z hlediska databázových systémů jsou množiny D_1, D_2,.. množina hodnot atributů a označují se jako domény.

Od matematické relace se liší v několika aspektech:

- Relace je vybavena pomocnou strukturou, které se říká schéma relace. Schéma relace se skládá ze jména relace, jmen atributů a domén.
- Prvky domén, ze kterých se berou jednotlivé komponenty prvků relace, jsou atomické (dále nedělitelné) hodnoty. Tomuto omezení se říká **1.normální forma relací (1NF)**.

Schéma relace **R** se vytvoří nad množinou atributů $A_1:D_1,...A_n:D_n$, kde A_i jsou jména atributů a D_i jsou domény. Dvojici $A_i:D_i$ se říká atribut relace. Schéma relace lze zapsat $R(A_1:D_1,....A_n:D_n)$. Relace **R** nad množinou **A** je libovolná podmnožina kartézského součinu domén D_1 **x...x** D_n. Doména náležící atributu **C** se označuje jako **dom(C)**. Domény jsou obvykle primitivní typy dat (STRING, INTEGER...).

Prvkům relace se říká n-tice, přičemž **n** určuje řád relace. Relační schéma databáze je dvojice **(R,I)**, kde **R** je množina schémat relací a **I** je množina integritních omezení. Jedno z významných IO na relaci **R(A)** je existence primárního klíče.

Primární klíč je množina atributů **K** z **A**, jejichž hodnoty jednoznačně určují n-tice relace **R. K** je minimální v tom smyslu, že z ní nelze odebrat žádný atribut, aniž by to narušilo identifikační vlastnost.

Atribut, který je součástí nějakého klíče se nazývá klíčový.Atributy, které nejsou součástí žádného klíče se nazývají neklíčové. Z podstaty RMD vyplývá, že každá relace má klíč. Protože relace jsou množiny, nesmí relace obsahovat duplicitní prvky.

Dalším důležitým IO je referenční integrita. Toto omezení popisuje vztahy mezi daty ve dvou relacích. Atribut, kterého se referenční integrita týká se nazývá cizí klíč (foreign key). Takové dvě relace se obvykle nazývají master/ detail nebo parent/child nebo independent/dependent. Česky obvykle hlavní/závislá někdy taky nezávislá/závislá, nadřazená/podřízená. Cizí klíč je atribut relace, který se v nadřazené relaci primárním klíčem.

Příklad:

Hlavní relace:

UCITELE(CISLO,JMENO,PLAT,PRIPLATEK,...)

Závislá relace:

PREDMETY(ZKRATKA,NAZEV,.....,GARANT)

CISLO z relace UCITELE je primární klíč a objevuje se v relaci PREDMETY jako položka GARANT. Položka GARANT je tzv. cizí klíč.

Přípustnou relační databází se schématem **(R, I)** nazýváme množinu relací $R_1,...R_k$ takových, že jejich prvky vyhovují **I**. O takové množině relací říkáme, že je konzistentní.

Formou reprezentace relací může být dvojrozměrná tabulka.

Podmínky, které musí splňovat relační tabulka:

- Všechny hodnoty v tabulce musí být elementární - tz. dále nedělitelné - podmínka 1.NF.
- Sloupce mohou být v libovolném pořadí.
- Řádky mohou být v libovolném pořadí.
- Sloupce musí být homogenní, tzn. ve sloupci musí být údaje stejného typu.
- Každému sloupci musí být přiřazeno jednoznačné jméno (tzv. atribut).
- V relační tabulce nesmí být dva zcela stejné řádky. Tzn., že každý řádek je jednoznačně rozlišitelný.

Shrnutí:

Doména: je množina datových hodnot stejného typu. Tyto hodnoty popisují nějakou vlastnost objektu.

Relace: je každá podmnožina kartézského součinu, jejíž konkrétní n-tice vznikly vybráním příslušných prvků z domén.

Atribut: je pojmenování pro každé užití hodnoty z domény v relaci.

Záhlaví relace: obsahuje jméno relace a jména atributů v relaci. Je v čase neměnné.

Tělo relace: obsahuje v čase proměnnou množinu n-tic hodnot, jejichž pořadí je dáno záhlavím relace.

Stupeň relace: je počet atributů relace.

Kardinalita relace: je počet řádků relace.

Primární klíč: je atribut nebo množina atributů, který jednoznačně určuje n-tice relace. Pokud je třeba použít více atributů pro jednoznačné určení n-tice, potom hovoříme o tzv. složeném primárním klíči. Pokud je více atributů, které splňují pravidlo pro primární klíč,jeden zvolíme jako primární a ostatní jsou alternativní klíče.

Definice primárního klíče:

Primární klíč je podmnožina atributů relace, která

1) jednoznačně identifikuje každý prvek relace

2) není redundantní, tj. žádný její atribut nelze vynechat, aniž by podmínka 1) přestala platit.

Kandidáti primárního klíče:

primární klíč - pouze jeden

alternativní klíče - více.

Kontrolní otázky:

1. Co je to databázová relace a čím se liší od relace matematické?
2. Jak lze vytvořit každou instanci databázové relace?
3. Čím lze identifikovat jednotlivé n-tice relace?
4. Kolik může mít relace primárních klíčů?

 **Pojmy k zapamatování:**

Doména

Atribut

Relace

Kandidáti primárního klíče

Primární klíč

Alternativní klíč

Kardinalita relace

Stupeň relace

Cizí klíč

1NF

3.2 KLÍČE V DATABÁZOVÉ TECHNOLOGII

Cíl:

Cílem této kapitoly je zopakovat již zavedené pojmy týkající se klíčů v relačních datových modelech, přehledně uvést základní definice a na příkladech ozřejmit použití různých typů klíčů. Čtenář po prostudování této kapitoly musí umět definovat co to je klíč, vyjmenovat typy klíčů a charakterizovat je.

Klíčová slova:

Primární klíč, cizí klíč, alternativní klíč, inverzní klíč.

 **Průvodce studiem:**

Tuto kapitolu je třeba chápat jako kapitolu přehledovou. Nedovíte se v ní v podstatě nic nového, pouze se přehledově zopakujete čtyři již zmiňované pojmy (klíče) a na příkladem si připomenete jejich význam a použití. Pro ty, kteří studovali poctivě to bude skutečně snadná záležitost.

3.2.1 Primární klíč (Primary Key)

je atribut nebo množina atributů **K** z **A** (kde A je množina všech atributů relace R), jejichž hodnoty jednoznačně určují n-tice relace **R. K** je minimální v tom smyslu, že z ní nelze odebrat žádný atribut, aniž by to narušilo identifikační vlastnost. Pokud je atributů, které splňují tuto vlastnost v relaci R více, nazýváme je **kandidáty primárního klíče**. Kandidátů primárního klíče smí být v dané relaci R více, primární klíč však smí být pouze jeden (to nevylučuje možnost, že tento jeden primární klíč je složen z více atributů). Výběr primárního klíče je veden několika kritérii, kterým se budeme věnovat později.

 **Příklad:**

Mějme relaci *Pacient*, která má následující atributy:

Rodné číslo

Jméno

Příjmení

Datum narození

Ulice

Číslo domu

Město

Telefon

Kód pojišťovny

Nad každým atributem se zamyslíme a zvážíme, zda splňuje identifikační vlastnost, tj. zda jeho hodnota jednoznačně identifikuje každou n-tici relace *Pacient*. Je zřejmé, že můžeme mít více různých pacientů téhož jména, příjmení, data narození, atd. Je však nemožné (aspoň doufám, po všech kontrolách, které byly provedeny v souvislosti se zavedením zdravotních pojišťoven), aby dva různí pacienti měli totožné rodné číslo. Proto *Rodné číslo* je kandidátem primárního klíče. Dá se předpokládat, že rovněž složený atribut (*Jméno, Příjmení, Datum narození*) by mohl být kandidátem primárního klíče. Při rozhodování, který ze dvou kandidátů bude vhodnějším primárním klíčem, rozhoduje jednoduchost a tudíž jako PK vybereme *Rodné číslo*.

3.2.2 Cizí klíč (Foreig Key)

Cizí klíč je atribut nebo množina atributů, které jsou v nadřazené relaci primárním klíčem. Nadřazenou relací budeme označovat takovou relaci, která vznikla z nadřazené entity, podřízenou relaci, která vznikla z podřízené entity. Existence vztahu mezi entitami v konceptuálním modelu je v relačním modelu zachycena právě cizím klíčem. Obsahuje-li relace cizí klíč znamená to, že je v podřízeném vztahu s jinou relací.

 **Příklad:**

Mějme relaci *Pacient* z předchozího příkladu a relaci *Pojišťovna*, která bude mít následující atributy:

Kód pojišťovny

Název

Ulice

Číslo

Město

Telefon

V konceptuálním modelu platilo, že entita *Pacient* je existenčně závislá na entitě *Pojišťovna*, tzn., že nemůže existovat žádná instance entity *Pacient*, která by nevstupovala do vztahu s entitou *Pojišťovna*. Jednoduše řečeno všichni pacienti musí být pojištěni u nějaké pojišťovny. Tato skutečnost byla zakreslena existenčním typem vztahu.

Relační model disponuje pouze jediným konstruktem a tím je relace, vztah se modeluje cizím klíčem (FK). Proto primární klíč nadřazené relace se stane cizím klíčem podřízené relace.

Nadřazená relace je *Pojišťovna* (její PK je *Kód pojišťovny*)

Podřízená relace je *Pacient* (její PK je *Rodné číslo*). *Kód pojišťovny* je FK v relaci *Pacient*.

Výsledná definice klíčů:

Pojišťovna

Kód pojišťovny PK

Název

Ulice

Číslo

Město

Telefon

Pacient

Rodné číslo PK

Jméno

Příjmení

Datum narození

Ulice

Číslo domu

Město

Telefon

Kód pojišťovny FK

3.2.3 Vyhledávací klíče

Vyhledávací klíč je atribut nebo množina atributů, které složí k rychlému vyhledávání v databázi. Na tyto klíče se vytvoří pomocné datové struktury, tzn. indexy a ty způsobí, že záznam se zadanou hodnotou vyhledávacího klíče se najde v databázi rychleji. Více informací o indexech najdete v následujících kapitolách, proto se zde omezíme pouze na definici vyhledávacích klíčů.

Vyhledávací klíče mohou být dvojího typu:

Alternativní klíč (Alternate Key)

Je vyhledávací klíč, jehož hodnoty nesmí být duplicitní. Jedná se v podstatě o nevybraného kandidáta primárního klíče. Alternativní klíč (AK) totiž musí, stejně jako primární klíč, splňovat podmínku jednoznačné identifikace.

 **Příklad:**

Složený kandidát primárního klíče z relace *Pacient* (*Jméno, Příjmení, Datum narození*) by mohl být alternativním klíčem.

Inverzní klíč (Inverse Key)

Inverzní klíč (IE) je atribut nebo množina atributů k rychlejšímu vyhledávání v databázi. Jeho hodnoty mohou být duplicitní.

 **Příklad:**

Příkladem by mohl být atribut jakýkoliv atribut relace *Pacient*, podle jehož hodnot chceme v databázi hledat záznamy. Obvykle chceme najít pacienta podle *Příjmení*.

 **Úkol k zamyšlení:**

Popište neformálně výsek reality, ve kterém nalezněte aspoň dvě entity, které spolu vstupují do vztahu. Zamyslete se na těmito entitami a nalezněte kekich primární klíče, alternativní klíče, inverzní klíče a určete, která z nich bude obsahovat cizí klíč a jak bude tento cizí klíč vypadat.

3.3 NORMALIZACE RELAČNÍCH SCHÉMAT

Cíl:

Cílem kapitoly Normalizace relačních schémat je objasnit, jak lze nalézt v navržených relacích zdroje redundancí dat a jak tyto zdroje redundance odstranit. Čtenář po prostudování kapitoly bude schopen posuzovat jednotlivé atributy v rámci navržené relace z pohledu funkčních závislostí a bude schopen stanovit, ve které normální formě se relace nachází. Bude schopen nalézt uzávěr množiny funkčních závislostí s využitím Armstrongových axiomů a dodatečných deduktivních pravidel.

Klíčová slova:

funkční závislosti atributů, 1NF, 2NF, 3NF, BCNF, dekompozice relačních schémat, uzávěr množiny funkčních závislostí, Armstrongovy axiomy, dodatečná deduktivní pravidla.

 **Průvodce studiem:**

Relační databázové modely disponují aparátem pro odstranění redundantních dat z databáze. Redundance v databázi jsou potencionálně velkým problémem. Ne proto, že by data zbytečně zabírala místo na paměťových médiích, ale hlavně proto, že redundantně vyskytující se data se jen velmi těžko dají uchovávat v konzistentním stavu. Z tohoto pohledu

odstranění redundancí je základní podmínkou pro udržení konzistence databáze. Prostředkem pro návrh databáze bez redundancí je tzv. normalizace relačních schémat. Každý krok směrem k vyšší normalizaci relačních schémat je zároveň krokem ke kvalitnějším a spolehlivějším datům. proto je problematika normalizace velmi důležitá. K čemu by nám byla data, o kterých bychom věděli, že se na jejich správnost nedá spoléhat?

3.3.1 První normální forma relace

Báze dat je nejjednodušší a nejpřehlednější, jsou-li relace, které ji reprezentují, definovány nad doménami, jejichž prvky jsou jednoduché (nejsou to opět relace). O takových relacích se říká, že jsou v první normální formě a příslušné atributy se nazývají jednoduchými (atomickými, dále nedělitelnými).

Relace je v první normální formě (1NF), jsou-li všechny její atributy atomické (nedělitelné)

Atributy, které nejsou jednoduché, jsou složené. Některé relace v první normální formě však mají i negativní vlastnosti z hlediska redundance a konzistence dat. Problémy spojené s redundancí dat vyplývají zpravidla ze skutečnosti, že hodnota některého atributu může zcela určovat hodnoty jiných atributů. Sdružování takových atributů v jedné relaci se ukazuje jako nevhodné. Problematice dekompozice relací neboli jejich normalizaci budou proto věnovány následující odstavce.

Příklad:

Příkladem relace, která je v 1NF, ale není v žádné vyšší NF je relace, která nemá primární klíč.

Relace dodavatelů automobilů

Název	Stát	Město	Název auta	Počet
D1	ČR	Mladá Boleslav	Škoda Favorit	67
D1	ČR	Mladá Boleslav	Škoda Favorit	80
D3	SRN	Kolín n.Rýnem	Opel Vectra	9
D3	SRN	Kolín n.Rýnem	Opel Vectra	44
D3	SRN	Kolín n.Rýnem	Opel Vectra	48
D6	Francie	Paříž	Renault 19	10

Relace je nejvýše v 1NF, protože má atomické atributy, ale nemá PK.

3.3.2 Funkční závislosti atributů

Problém, který byl nastíněn v předcházejícím odstavci, spočívá ve vzájemných závislostech atributů jedné relace. Pro tuto závislost se zpravidla používá termínu funkční závislost, i když z hlediska matematického nejde o funkci, stejně jako pojem relace zcela neodpovídá matematickému pojetí (v databázovém pojetí se totiž relace stejně jako funkční závislosti mění v čase).

Nechť **R(A:D)** je relační schéma a ***X***, ***Y*** jsou jednoduché nebo složené atributy. Atribut ***Y*** je funkčně závislý na atributu ***X***, (***X -> Y***), platí-li pro každou instanci relace R, že pro libovolné dva prvky relace, které se shodují v hodnotě atributu ***X***, platí, že se shodují i v atributu ***Y***.

Atribut ***Y*** je úplně funkčně závislý na složeném atributu ***X***, je-li na ***X*** funkčně závislý a zároveň není funkčně závislý na žádné z jeho složek.

3.3.3 Druhá normální forma relace

Požadavek, aby se relace nacházela v první normální formě, tj. aby každý atribut relace byl definován na jednoduché doméně, nevede vždy k nejžádanějším výsledkům.

Relace R je ve druhé normální formě (2NF), je-li v první normální formě a jestliže pro každý neklíčový atribut platí, že je úplně funkčně závislý na primárním klíči.

Proces redukce relace v nižší normální formě (zde v 1.NF) na relaci ve vyšší normální formě (zde ve 2.NF) se nazývá normalizace relace. Relace v některé z vyšších normálních forem je zároveň relací ve všech vůči ní nižších normálních formách. Je zřejmé, že vyšší normální formu relace lze vytvořit vhodnou dekompozicí relace původní. Požaduje se však, aby výchozí relace byla z výsledných komponent rekonstruovatelná, tj. aby při dekompozici nedošlo ke ztrátě informace.

 **Příklad:**

Relace dodavatelů automobilů

Název	**Stát**	**Město**	**Název auta**	**Počet**
D1	ČR	Mladá Boleslav	Škoda Favorit	67
D2	ČR	Praha	Škoda Favorit	50
D3	SRN	Kolín n.Rýnem	Opel Vectra	9
D4	SRN	Wolfsburg	VW Golf	22
D5	SRN	Mnichov	VW Golf	14
D6	Francie	Paříž	Renault 19	10

Název je PK v relaci $\rightarrow$ relace je aspoň ve 2NF.

Funkční závislosti atributů:

Název $\rightarrow$ Stát

Název $\rightarrow$ Město

Název $\rightarrow$ Název auta

Název $\rightarrow$ Počet

Město $\rightarrow$ Stát

Název $\rightarrow$Město & Město $\rightarrow$ Stát $\Rightarrow$ Název $\rightarrow$ Stát

Stát je tranzitivně závislý na *Názvu.*

Proto relace není v 3NF, ale pouze v 2NF.

 **Příklad:**

Relace stromů v lese:

Číslo	**Typ**	**Druh**	**Výška**

1	smrk	jehličnatý	5
2	jedle	jehličnatý	3
3	smrk	jehličnatý	8
4	dub	listnatý	12
5	dub	listnatý	8

Číslo je PK

Funkční závislosti atributů:

Číslo $\rightarrow$ Typ

Číslo $\rightarrow$ Druh

Číslo $\rightarrow$ Výška

Typ $\rightarrow$ Druh

Číslo $\rightarrow$ Typ $\&$ Typ $\rightarrow$ Druh $\Rightarrow$ Číslo $\rightarrow$ Druh

relace není v 3NF – tranzitivní závislost

Úkoly k zamyšlení:

1. Pokuste se navrhnout relaci, která bude v 1NF a zároveň nebude v 2NF z toho důvodu, že její atributy nebudou úplně funkčně závislé na primárním klíči.
2. Navrhněte relaci, která bude v 2NF.

3.3.4 Třetí normální forma relace

Nechť X, Y, Z jsou atributy (jednoduché nebo složené) daného relačního schématu a nechť mezi dvojicemi atributů platí:

$$X \rightarrow Y \;\&\; Y \rightarrow Z \;\&\; \neg(Y \rightarrow X)$$

Pak je atribut Z tranzitivně závislý na atributu X.

Nutnost podmínky nezávislosti X na Y vyplývá ze skutečnosti, že v opačném případě by v každém relačním schématu se dvěma a více možnými klíči byly všechny neklíčové atributy tranzitivně závislé na kterémkoliv z klíčů.

Relace R je v třetí normální formě (3NF), je-li ve druhé normální formě a platí-li, že žádný neklíčový atribut není tranzitivně závislý na žádném klíči relace R.

Z definice relace ve třetí normální formě vyplývají tyto její vlastnosti :

- Žádný neklíčový atribut relace není ani částečně ani tranzitivně závislý na nějakém klíči relace, jsou tedy úplně závislé na primárním klíči.
- Neklíčové atributy jsou navzájem nezávislé.

Problémy, které mohou u relací v třetí normální formě nastat, mohou však vyplynout ze vzájemně částečně nebo tranzitivně závislých klíčových atributů. Kandidátní klíče, pokud klíče nejsou jednoduché, se totiž mohou částečně překrývat. Dekompozice relací, které vedou ke třetí normální formě, mohou navíc být nejednoznačné a vést k nestejně kvalitním výsledkům.

 Příklad:

Dekompozicí relace v 2NF dostaneme relace v 3NF.

Číslo	Typ	Výška
1	smrk	5
2	jedle	3
3	smrk	8
4	dub	12
5	dub	8

Číslo je PK

Typ	Druh
smrk	jehličnatý
jedle	jehličnatý
dub	listnatý

Typ je PK

Obě relace obsahují pouze závislosti neklíčových atributů na primárním klíči. Jsou odstraněny všechny tranzitivní závislosti. Proto jsou obě relace ve 3NF.

 **Úkoly k zamyšlení:**

1. Navrhněte relaci, ve které bude existovat tranzitivní závislost atributu(ů) na primárním klíči.
2. Navrhněte vhodnou dekompozici.

3.3.5 Boyce-Coddova normální forma relace

Nedostatky plynoucí ze vzájemných závislostí klíčových atributů eliminuje Boyce - Coddova normální forma BCNF, která vylučuje všechny netriviální funkční závislosti atributů kromě závislostí na klíčích relace.

Schéma relace R(A:D) je v Boyce-Coddově normální formě (BCNF), je-li v první normální formě a platí-li pro každou funkční závislost atributu A na atributu X , která není triviální, že X je klíčem v R a A je neklíčový atribut.

Je zřejmé, že každé relační schéma, které je v BCNF je též ve třetí normální formě, nikoliv však naopak. Existují relační schémata ve třetí normální formě, která nejsou v BCNF.

 **Příklad:**

Zaměstnanec (Číslo_zam, RČ, Jméno, Příjmení, Funkce)

Kandidáti PK: Číslo_zam a RČ

Funkční závislosti:

Kromě všech závislostí neklíčových atributů na kandidátech PK, ex. i závislost kandidátů PK navzájem. Proto relace není v BCNF.

Pro převedení relace do BCNF je vhodné jeden z atributů (Číslo_zam, resp. RČ) vypustit a tím zajistit jediný typ úplné funkční závislosti. Totiž závislosti, kdy neklíčový atribut je závislý na klíč.

Relace v BCNF by potom mohla vypadat takto:

Zaměstnanec (Číslo_zam, Jméno, Příjmení, Funkce)

Pokud v databázi potřebujeme oba atributy, pak je vhodné je umístit do různých relací.

3.3.6. Další funkční závislosti

3.3.6.1. Množina funkčních závislostí

Nechť **R** je relační schéma a **A**, **B**, **C** je podmnožina jeho atributů. Dále předpokládejme funkční závislosti $\mathbf{A} \rightarrow \mathbf{B}$ a $\mathbf{B} \rightarrow \mathbf{C}$. Z těchto závislostí lze předpokládat $\mathbf{A} \rightarrow \mathbf{C}$ (tranzitivita). Označme **F** jako množinu funkčních závislostí pro **R (A → B a B → C)** a $\mathbf{X} \rightarrow \mathbf{Y}$ jako libovolnou funkční závislost. Řekneme-li, že **F** logicky implikuje $\mathbf{X} \rightarrow \mathbf{Y}$, pak každý prvek relačního schématu **R**, který splňuje závislosti v **F**, splňuje i závislost $\mathbf{X} \rightarrow \mathbf{Y}$ a zapisujeme

$\mathbf{F} \models \mathbf{X} \rightarrow \mathbf{Y}$

V našem případě relačního schématu **R** pak tuto skutečnost zapíšeme:

$\{\mathbf{A} \rightarrow \mathbf{B}, \mathbf{B} \rightarrow \mathbf{C}\} \models \mathbf{A} \rightarrow \mathbf{C}$

3.3.6.2. Uzávěr množiny funkčních závislostí

$\mathbf{F}^+$ je uzávěrem **F** tehdy, platí-li, že všechny závislosti v $\mathbf{F}^+$ jsou logickými důsledky v **F**. Tuto skutečnost zapisujeme:

$\mathbf{F}^+ = \{\mathbf{X} \rightarrow \mathbf{Y} \mid \mathbf{F} \models \mathbf{X} \rightarrow \mathbf{Y}\}$

3.3.6.3. Kandidáti primárního klíče a funkční závislosti

Mějme schéma $\mathbf{R(A_1,A_2,....,A_n)}$ a funkční závislosti **F**. Nechť **X** je **podmnožina** $\mathbf{\{A_1, A_2, ..., A_n\}}$. Pak o **X** lze říci, že je kandidátem primárního klíče v **R**, jestliže:

1. $\mathbf{X} \rightarrow \mathbf{A_1A_2,....A_n}$ je v $\mathbf{F}^+$

Závislost všech atributů $\mathbf{A_1,A_2,....,A_n}$ na atributu **X** je daná nebo logicky vyplývá.

2. Neexistuje $\mathbf{Y} \subset \mathbf{X}$, pro které by platilo $\mathbf{Y} \rightarrow \mathbf{A_1A_2,....A_n}$ v $\mathbf{F}^+$.

3.3.6.4. Armstrongovy axiomy

1. Reflexivita

Jestliže $\mathbf{Y} \subseteq \mathbf{X} \subseteq \mathbf{U}$, pak závislost $X \rightarrow \mathbf{Y}$ je logicky implikována. Na složeném atributu $\mathbf{A_1A_2,....A_n}$ je funkčně závislý každý atribut $\mathbf{A_i}$, který je jeho složkou.

2. Augmentace

Jestliže platí $\mathbf{X} \rightarrow \mathbf{Y}$ ve schématu **R** a **Z** je podmnožinou atributů **U**, pak taky platí:

$\mathbf{XZ} \rightarrow \mathbf{YZ}$ (**XZ** je zkrácené označení $\mathbf{X} \cup \mathbf{Z}$).

3. Tranzitivita

Jestliže platí **X** $\rightarrow$ **Y** a zároveň **Y** $\rightarrow$ **Z**, pak taky platí **X** $\rightarrow$ **Z**

Příklad:

Mějme schéma **R(A,B,C,D)** s funkčními závislostmi **A** $\rightarrow$**C**, **B** $\rightarrow$ **D**. Zvolme primárním klíčem složený atribut **AB** jako jediný. Dokažte, že **AB** je jediným kandidátem primárního klíče.

1. **A** $\rightarrow$ **C**	daná závislost
2. **AB** $\rightarrow$ **ABC**	augmentace atributy **AB**
3. **B** $\rightarrow$ **D**	daná závislost
4. **ABC** $\rightarrow$ **ABCD**	augmentace atributy **ABC**
5. **AB** $\rightarrow$ **ABCD**	tranzitivita

Všechny atributy relačního schématu **R** jsou závislé na klíči **AB** a přitom nejsou závislé na jeho složkách **A**, **B**.

3.3.6.5. Dodatečná deduktivní pravidla

1. Pravidlo spojení

$\{ X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$

2. Pravidlo pseudotranzitivity

$\{ X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

3. Dekompoziční pravidlo

Jestliže $X \rightarrow Y$ & $Z \subseteq Y$, pak $X \rightarrow Z$

Úkoly k zamyšlení:

Pokuste se s využitím Armstrongových axiomů dokázat výše uvedená tři dodatečná pravidla.

Řešení:

Pravidlo 1

$X \rightarrow Y$	daná závislost
$X \rightarrow XY$	augmentace X
$X \rightarrow Z$	daná závislost
$XY \rightarrow YZ$	augmentace Y

$X \rightarrow XY$ & $X Z \rightarrow YZ \Rightarrow X \rightarrow YZ$

Pravidlo 2

$X \rightarrow Y$	daná závislost
$WX \rightarrow WY$	augmentace W

$WY \rightarrow Z$	daná závislost
$WX \rightarrow Z$	tranzitivita

Pravidlo 3

$Y \rightarrow Z$	vyplývá z reflexivity
$X \rightarrow Y$	daná závislost
$X \rightarrow Z$	tranzitivita

 **Příklad:**

Mějme schéma R(A,B,C) a F = $\{ A \rightarrow B, B \rightarrow C\}$. Určete F^+.

Řešení:

1. Za X dosaďte postupně všechny atributy, obsahující A.

$ABC \rightarrow AB$	dekompoziční pravidlo
$A \rightarrow C$	tranzitivita, vyplývající z F
$ABC \rightarrow BC$	augmentace B

2. Za X dosaďte postupně všechny atributy, které obsahují B, ale neobsahují A.

$BC \rightarrow B$	dekompoziční pravidlo
$B \rightarrow C$	předpoklad
$B \rightarrow 0$	reflexivita

3. Za X dosaďte všechny atributy, které obsahují C, ale neobsahují ani A, ani B.

$C \rightarrow C$	reflexivita
$C \rightarrow 0$	reflexivita
$0 \rightarrow 0$	reflexivita

 **Shrnutí:**

Problematika normalizace relačních schémat je velmi důležitá a není možné se jí vyhnout, chceme-li navrhnout smysluplné schéma databáze. Jejím hlavním úkolem je nalézt zdroje redundance dat, které v konečném důsledku vedou k nekonzistenci databáze. Pro stanovení, ve které normální formě se relace nachází, je třeba nalézt všechny funkční závislosti atributů. Kromě triviální funkční závislosti (atribut je závislý sám na sobě) sledujeme tranzitivní funkční závislost a úplnou funkční závislost. Při každé závislost musíme stanovit, zda se jedná o závislost na primárním klíči, na jeho složce či na neklíčovém atributu. Ideální situace je, pokud všechny navržené relace se vyskytují v BCNF. Někdy je ovšem těžké tohoto stavu docílit a můžeme připustit 3NF. V žádném případě však není možné, aby byly relace v nižší než 3NF. Pro účinné nalezení všech funkčních závislostí (uzávěr množiny funkčních závislostí) lze s úspěchem použít tzv. Armstrongových axiomů a dodatečných deduktivních pravidel.

Kontrolní otázky:

1. Kdy se nachází relace v 1NF, 2NF, 3NF, CBNF?
2. Kdy můžeme říci, že atribut A je funkčně závislý na atributu B?
3. Čeho musíme dbát při dekompozici relačních schémat?
4. Co je to uzávěr množiny funkčních závislostí?
5. K čemu slouží a z čeho jsou odvozena dodatečná deduktivní pravidla?

Pojmy k zapamatování:

Funkční závislost atributů

Úplná funkční závislost

Tranzitivní funkční závislost

1NF, 2NF, 3NF, BCNF

Dekompozice relačních schémat

Uzávěr množiny funkčních závislostí

Armstrongovy axiomy

Dodatečná deduktivní pravidla

Průvodce studiem:

Pro úplnost ještě uvádím, že v problematice normalizace relačních schémat se rovněž vyskytuje pojem 4NF, jehož vysvětlení je nad rámec tohoto textu. Pokud má čtenář zájem se s problematikou seznámit, může sáhnout např. po publikaci [1].

3.4 TRANSFORMACE KONCEPTUÁLNÍCH SCHÉMAT

Cíl:

Jak již z předcházejících kapitol vyplynulo, postup při vytváření schématu databáze je následující:

1. Vytvoření konceptuálního schématu modelované reality.

2. Transformace konceptuálního modelu na model datový (my jsme se zabývali pouze relačním datovým modelem).

3. Vygenerování schématu do příslušného systému řízení báze dat.

Následující kapitola shrnuje, jakými problémy se musí návrhář zabývat, když transformuje konceptuální model na relační datový model. Cílem kapitoly je zopakovat probrané učivo z předcházejících kapitol a systematizovat postup při transformaci konceptuálního modelu na

datový. Čtenář si musí osvojit postup při vytváření schématu databáze a musí se řídit principy, definovanými v této kapitole.

Klíčová slova:

Entita, vztah, atribut, identifikační klíč, relace, primární klíč, alternativní klíč, inverzní klíč, cizí klíč, silné entity, slabé entity, reprezentace entit, reprezentace vztahů.

Průvodce studiem:

Z předcházejících kapitol již bylo zřejmé, že po vytvoření konceptuálního modelu je třeba se rozhodnout, do jakého datového modelu budeme konceptuální model transformovat. Naznačili jsme, že se v dalším textu budeme zabývat pouze datovými modely relačními, i když existují i jiné datové modely, které jsou však v současné době rozšířeny v podstatně menší míře. Pozorný čtenář již získal představu o tom, jak konceptuální model na model relační transformovat. Nicméně mohou vzniknout situace, kdy si nebudete vědět rady. Proto je pro Vás připravena následující kapitola, která diskutuje případy, které při transformaci mohou nastat. Zároveň chápejte tuto kapitolu jako poslední, která se problematice modelování zabývá. Proto ji prostudujte velmi pozorně a pokud Vám bude něco nejasné, vraťte se k předcházejícím kapitolám a znovu si je pozorně pročtěte. Pokud budou pochybnosti či nejasnosti přetrvávat, kontaktujte tutora.

3.4.1 Nejpoužívanější konstruktory E - R modelu

Systematický přístup k transformaci konceptuálního modelu na relační datový model vyžaduje, abychom se zabývali všemi konstrukty, které se v obou typech modelů vyskytují. konceptuální model disponuje dvěma základními konstrukty, kterými jsou entita a vztah. Relační model disponuje pouze jediným konstruktem a tím je databázová relace. Nejdříve si vyjmenujme, jaké entity, vztahy a atributy se mohou vyskytnout v konceptuálním modelu.

ENTITA

- Obecná = bez rozlišení druhu.
- Silná (kmenová, základní, regulární) = existuje nezávisle na jiných entitách.
- Slabá (popisná) = identifikačně závislá na jiných entitách.
- Vazební (asociativní) = realizuje vazbu mezi entitami.
- Generalizace (nadtyp).
- Specializace (podtyp).

VZTAH

- N -ární = mezi více než dvěma entitami.
- Binární.
- Kardinalita (max. a min. počet výskytů zúčastněné entity ve vztahu).
- Výlučnost (exkluzivita) = pro jeden výskyt entity může být realizován právě jeden ze vztahů.
- Existenční závislost (povinné členství).
- Identifikační závislost .

ATTRIBUT

- Jednoduchý (atomický) = dále nedělitelný.

- Složený.
- Základní = který nelze odvodit z jiných atributů.
- Odvoditelný.
- Vícehodnotový.

KLÍČ

- Identifikační.
- Alternativní.
- Cizí.
- Inverzní.

3.4.2 Principy transformace E-R schématu do relačního datového modelu

Základním problémem je skutečnost, že konceptuální model disponuje dvěma základními konstrukty, kterými jsou entita a vztah, kdežto relační datový model disponuje jediným konstruktem, kterým je databázová relace. Je třeba se tedy zabývat, jak konstrukty E-R modelu převést na relace. Entitní a vztahové typy se převádějí na relace. Čtenář relačního schématu, který nezná konceptuální schéma tedy obecně nemusí být schopen rozlišit, která relace odpovídá entitnímu a které vztahovému typu.

3.4.3 Převedení silného entitního typu

- Entita se převede na relaci.
- Identifikační klíč se převede na primární klíč.
- Popisné atributy definují domény relace.

Problém může nastat v souvislosti s normalizací relací. Cílem je, aby výsledná tabulka byla aspoň v 3. NF nebo v BCNF. Dekompozice pak způsobí nárůst relací, které neznají odpovídající entitu v E-R diagramu. Do jisté míry toto lze řešit pohledy.

3.4.4 Reprezentace vztahů

vztahy 1 : 1

a) Mějme entity E1 a E2, které vstupují do vztahu V. Má-li E1 i E2 nepovinné členství ve vztahu V, vzniknou transformací 3 relace E1 - R1 , E2 - R2 , V - R. R bude obsahovat klíče schémat R1 a R2 ,jako cizí klíče. Libovolný z nich může v R být primárním klíčem. Další atributy v R budou odpovídat atributům ve V.

b) Má-li E1 nepovinné a E2 povinné (nebo obráceně), budou se vytvářet pouze relace R1 a R2. Existenční závislost E2 na E1 vyjádříme tak, že k R2 připojíme jako cizí klíč klíčové atributy R1. Atributy vztahu V budou umístěny do R2.

c) Mají-li obě entity povinné členství, můžeme vytvořit jediné schéma relace R, které vznikne z těchto dvou entit. Primárním klíčem bude libovolný z identifikačních klíčů, přidáme atributy V.

vztahy 1 : N

Ve vztahu s kardinalitou 1 : N je entita, vstupující n instancemi do vztahu, nazývána determinantem vztahu.

a) Má-li determinant (E1) povinnou účast ve vztahu V, definujeme 2 relace R1 a R2. Ke schématu R1 připojíme klíčové atributy R2. Primární klíč R1 odpovídá identifikačnímu klíči determinantu E1. K ještě připojíme atributy vztahu V.

b) Má-li determinant (E1) nepovinnou účast ve vztahu V, definujeme tři relace R1, R2, R. R bude obsahovat klíče schémat R1 a R2 jako cizí klíče. Klíč schématu R1 bude v R primárním klíčem. Další atributy budou odpovídat atributům vztahu V.

3.4.5 Reprezentace vícehodnotových atributů

Analytik může sestavit konceptuální schéma tak, že použije vícehodnotový atribut. Tuto skutečnost konceptuální model připouští, nepřipouští ji však relační datový model. Jak převést takovou entitu na relaci (relace)?

1. Některé relační SŘBD připouštějí tabulky s vícehodnotovými sloupci (výjimečný případ).
2. Jestliže se vícehodnotový atribut váže na maximální počet výskytů **m** a databázový model připouští hodnotu **NULL** ve sloupci tabulky, budou vícehodnotové atributy transformovány na **m** atributů a nevyužité atributy budou mít hodnotu NULL.
3. Většinou je třeba provést dekompozici entity na dvě relace.

 **Příklad:**

ZAMĚSTNANEC (OSOBNÍ ČÍSLO, ... RODNÉ_ČÍSLO_DÍTĚTE: Multi)

RODNÉ_ČÍSLO_DÍTĚTE je vícehodnotový atribut. Obecně nevíme, kolik dětí zaměstnanec má. Tuto situaci vyřešíme tak, že provedeme dekompozici na dvě relace:

ZAMĚSTNANEC (OSOBNÍ ČÍSLO, ...)

DATA_NAROZENÍ (OSOBNÍ ČÍSLO, RODNÉ ČÍSLO DÍTĚTE)

Každému dítěti bude odpovídat jedna n -tice v relaci data_narození.

Konstrukce klíče je nejednoznačná:

a) Klíčem může být složený klíč např. OSOBNÍ ČÍSLO, RODNÉ ČÍSLO DÍTĚTE.

b) Klíčem může být pouze vícehodnotový atribut RODNÉ ČÍSLO DÍTĚTE.

3.4.6 Reprezentace skupinových atributů

Skupinové atributy mohou být navrženy např. v případě, kdy u jedné entity jsou atributy více používané a atributy, které více nejsou pominutelné, ale jsou méně používané.

 **Příklad:**

E(KLIČ_E, FREKV_ATR1, FREKV_ATR2,...., LEŽÁKY (L_ATR1, L_ATR2,...))

Tuto situaci musíme převést na 2 relace:

E1 (KLÍČ_E, FREKV_ATR1,...)

LEŽÁKY (KLÍČ_E, L_ATR1,...)

Na úrovni pohledu je pak možno používat schéma E se všemi atributy, na databázové úrovni však jsou dvě relace E1 a LEŽAKY.

3.4.7 Reprezentace slabého entitního typu

Slabý entitní typ je reprezentován relací, která kromě svých atributů má i atributy identifikačního vlastníka. Je třeba dbát na to, aby identifikační vlastník byl pouze jeden. Identifikační klíč je pak tvořen množinou atributů slabého entitního typu + atributy identifikačního vlastníka.

3.4.8 Rozdíl mezi entitním podtypem a slabým entitním typem

Entitní podtyp nemá obecně žádný parciální identifikační klíč. Dědí klíč zdroje ISA hierarchie, který musí být pouze jeden. Definujeme tedy schéma, obsahující atributy odpovídající vlastním atributům entitního podtypu, a k nim přidáme atributy odpovídající identifikačnímu klíči zdroje ISA hierarchie. Primární klíče relací, vzniklých z jedné ISA hierarchie, jsou tedy všechny stejné.

 **Shrnutí:**

Chceme-li vytvořit korektní datové schéma, které jediné je zárukou kvalitní databáze, musíme postupovat následujícím způsobem:

1. Vytvoříme konceptuální model reality. Při tvorbě modelu musíme znát modelovanou realitu a požadavky budoucího uživatele. Jedině tehdy správně navrhneme entity, vztahy a atributy konceptuálního modelu.
2. Vybereme vhodný SŘBD, který bude budovanou bázi dat spravovat. Při výběru se řídíme hlavně tím, jak robustní bude budovaná databáze, jaké nároky budou kladeny na komunikaci, kolik uživatelů bude s bází dat pracovat, jaké budou požadavky na bezpečnost dat, atd. Při rozhodování hraje rovněž roli cena jednotlivých SŘBD.
3. Výběrem SŘBD již jsou dány podmínky pro tvorbu modelu datového. Při transformaci konceptuálního modelu na datový se řídíme zásadami, definovanými v této kapitole.

 **Kontrolní otázky:**

1. Jaké typy entit znáte?
2. Jaké typy vztahů znáte?
3. Jak lze transformovat 2 entity na relace, má-li pouze jedna z nich povinně členství ve vztahu?
4. Jaké máme možnosti při transformaci vícehodnotového atributu?

 **Pojmy k zapamatování:**

Entita

Vztah

Atribut

Identifikační klíč

Relace

Primární klíč

Alternativní klíč

Inverzní klíč

Cizí klíč

Silné entity

Slabé entity

Reprezentace entit

Reprezentace vztahů

3.5 INTEGRITA RELAČNÍCH SCHÉMAT

Cíl:

Cílem této kapitoly je znovu čtenáře přimět k zamyšlení nad nutností správného definování integritních omezení. Bází dat v relačním pojetí je totiž chápána jako množina relací a integritních omezení. Čtenář po prostudování této kapitoly bude vědět, jaká existují integritní omezení, k čemu slouží a kdy a jak je správně definovat. Déle bude umět vyjmenovat, jaké existují způsoby zachování referenční integrity.

Klíčová slova:

Doménová integrita, entitní integrita, referenční integrita, restrict, cascade, set null.

3.5.1 Úvod

Požadavek integrity dat v databázi (jejich souladu se zobrazovaným světem objektů) vede k nutnosti definovat pro domény atributů jednotlivých relačních schémat a jejich vzájemné vztahy jistá omezení, která zabraňují možnostem atributů nabývat nereálných hodnot. Tato omezení se formulují pomocí soustavy uzavřených logických formulí v jazyce predikátové logiky. Pro relační báze dat je kromě skutečnost,že entity i vztahy se reprezentují stejným způsobem, tj. pomocí relací, charakteristickým rysem i to, že splňují tři pravidla integrity.

Prvním z těchto pravidel je pravidlo entitní integrity pro relační báze dat a spočívá v požadavku, aby primární klíč relace měl pro všechny její prvky (n-tice) svou hodnotu.

Druhé pravidlo integrity je doménová integrita, která požaduje, aby hodnoty atributů odpovídaly předem definovaným doménám (množinám hodnot atributů).

Třetí je pravidlo referenční integrity (vztahové), které se týká možností používání "cizích" klíčů relací a spočívá v požadavku, aby ke každé definované hodnotě cizího klíče existoval odpovídající prvek nadřazené (nezávislé) relace.

Jednou z podstatných výhod SŘBD je ta skutečnost, že uživatel se nemusí starat o to, jak jsou integritní omezení kontrolována. Kontrola IO je plně v kompetenci SŘBD. Uživatel (správce databáze) má povinnost integritní omezení správně definovat. Při všech aktualizačních operacích, u kterých by mohlo dojít k narušení integrity, jsou pak tato omezení automaticky kontrolována systémem řízení báze dat (SŘBD)..

3.5.2 Doménové integritní omezení

Doménové integritní omezení je především definováno přiřazením datových typů pro jednotlivé atributy a dále podmínek platnosti, které musí hodnoty atributů splňovat.

 **Příklad:**

Atribut Jméno je datového typu text s délkou max. 15 znaků, přičemž jednotlivé znaky jsou pouze alfabetické. Atribut Věk je datového typu celé číslo, které je kladné a není větší než 130.

3.5.3 Entitní integritní omezení

Entitní integritní omezení je definována primárním klíčem. Primární klíč zajišťuje entitní IO. Proto je naprosto nutné, aby každá relace měla primární klíč.

 **Příklad:**

Relace Pacient, která má atributy Rodné číslo, Jméno, Příjmení, atd. bude mít Rodné číslo definováno jako primární klíč. Tento atribut totiž splňuje podmínky pro PK, že hodnota PK jednoznačně identifikuje instanci (výskyt, n-titu) relace.

Na primární klíč jsou kladeny následující podmínky:

- Hodnota primárního klíče musí být unikátní.
- Entitní typ může mít pouze jeden primární klíč.
- Primární klíč může být i složený klíč, tj. složený z více atributů.
- Každá složka primárního klíče musí být definována jako NOT NULL.

3.5.4 Referenční integritní omezení

K definování referenční integrity slouží *primární klíč(Primary Key)* nezávislé entity a *cizí klíč (Foreign Key)* závislé entity. Nadefinováním cizího klíče jsme již vytvořili referenční IO. SŘBD se bude při aktualizačních operacích starat o to, zda hodnota cizího klíče je správná.

Podmínky kladené na cizí klíč:

- Cizí klíč a primární klíč se musí skládat ze stejného počtu atributů.
- Pokud primární klíč nezávislého entitního typu slouží k definování vztahu, musí být na tento primární klíč vytvořen unikátní index.
- Atributy musí být definovány nad stejnými doménami a jejich pořadí musí být stejné.
- Závislý entitní typ může mít více cizích klíčů a tím i odkazy na více entit.
- Cizí klíč může nabývat hodnoty NULL, pokud definuje existenční typ vztahu.
- Cizí klíč nesmí nabývat hodnoty NULL, pokud definuje identifikační typ vztahu.
- Cizí klíč nabývá hodnoty NULL, pokud aspoň jedna jeho složka nabývá hodnoty NULL.

Pro nezávislé entitní typy je referenční integrita hlídána u operací DELETE a UPDATE. Pro závislé entitní typy je hlídána operace INSERT a UPDATE.

3.5.5 Způsoby zachování referenční integrity

RESTRICT - při mazání a aktualizaci v nezávislém entitním typu se restriktivně vyžaduje, aby v podřízeném entitním typu byla nejdříve vymazána nebo aktualizována svázaná instance. Není-li tomu tak, mazání resp. aktualizace v nadřazeném entitním typu je zakázáno. Zápis nové instance do podřízeného entitního typu je zakázán, nenachází-li se odpovídající instance v nadřazeném entitním typu.

CASCADE - způsobí kaskádovou aktualizaci resp. mazání v podřízeném entitním typu při aktualizaci resp. mazání v entitním typu nadřízeném. Tento způsob může způsobit kaskádovité mazání velkého počtu instancí, proto je třeba jeho použití bedlivě zvážit.

SET NULL - při aktualizačních operacích je možno z důvodu zachování referenční integrity použít nastavení cizího klíče na hodnotu NULL. Tento způsob nelze použít u identifikačního typu vztahu, kde cizí klíč podřízeného entitního typu je součástí primárního klíče. Primární klíč (a to žádná jeho složka, pokud se jedná o složený klíč) nesmí nabývat hodnoty NULL.

 **Příklad:**

Příklad nastavení pravidel pro zachování referenční integrity:

Typ entity/operace	Identifikační vztah	Existenční vztah
Závislá/Delete	Bez omezení	Bez emezení
Závislá/Insert	Restrict	Set Null
Závislá/Update	Restrict	Set Null
Nezávislá/Delete	Cascade	Set Null
Nezávislá/Insert	Bez omezení	Bez omezení
Nezávislá/Update	Cascade	Set Null

 **Shrnutí:**

Hlavním cílem této kapitoly mělo být vysvětlit čtenáři, jak bezpodmínečně nutné je nadefinovat všechna integritní omezení. Jedná se o doménové IO, entitní IO a referenční IO. Po nadefinování všech IO je již v kompetenci SŘBD se starat o to, aby data uvedená IO splňovala. SŘBD tedy při každé aktualizační operaci (vkládání dat, mazání dat a změně dat) kontroluje, zda data splňují podmínky integrity.

Doménová integrita je definována datovými typy jednotlivých atributů a podmínkami platnosti.

Entitní integrita je definována primárním klíčem.

Referenční integrita je definována cizím klíčem a tzv. způsobem zachování referenční integrity (restrict, cascade, set null).

 **Otázka k zamyšlení:**

Pokuste se najít v realitě několik příkladů dvou entit, které spolu vstupují do vztahu a pokuste se definovat integritní omezení pro tento vztah. Hlavně se zamyslete nad tím, která z obou entit je závislá a která nezávislá, kolika instancemi entity vstupují do vztahu, jaký typ vztahu mezi entitami je a jak má být zabráněno narušení referenční integrity.

 **Pojmy k zapamatování:**

Doménová integrita

Entitní integrita

Referenční integrita

Restrict

Cascade

Set Null

3.6 TRANSFORMACE KONCEPTUÁLNÍHO MODELU

Cíl:

Cílem této kapitoly je na řešených příkladech demonstrovat proces transformace konceptuálního modelu na relační datový model. Čtenář by se měl zamýšlet hlavně nad tím, jak:

- *Transformovat entity na relace.*
- *Transformovat vztahy se zvláštním zřetelem na vztah s kardinalitou N:M.*
- *Definovat datové typy jednotlivých atributů.*
- *Definovat alternativní a inverzní klíče, které mají za následek tvorbu pomocných datových struktur, kterými jsou indexy.*
- *Dále je třeba podrobit navržené relace procesu normalizace. Každá navržená relace musí být minimálně v 3NF.*

Klíčová slova:

Relace, dekompozice, datové typy, vyhledávací (alternativní a inverzní) klíče, normalizace.

 **Průvodce studiem:**

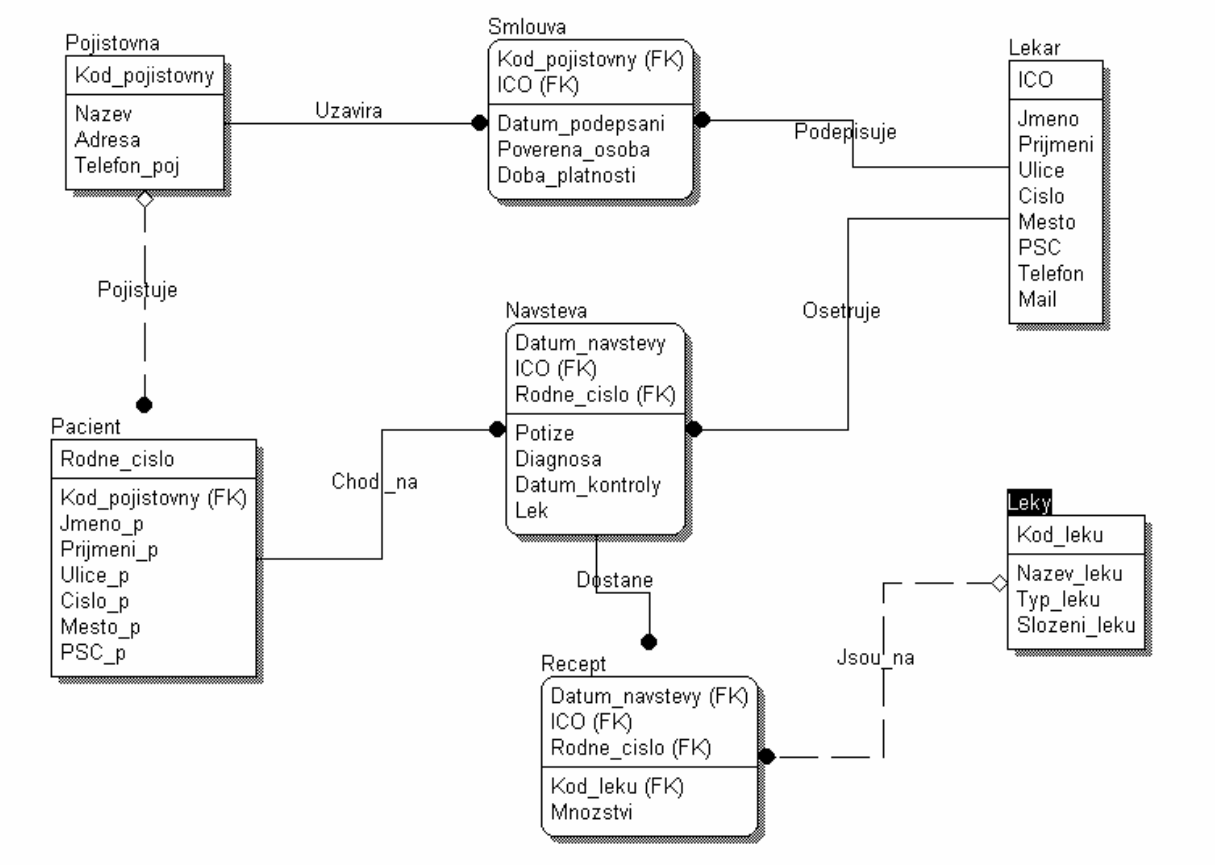
Věnujte, prosím, této kapitole mimořádnou pozornost. Jakmile Vám bude řešení uvedených příkladů jasné, můžete přistoupit ke korespondenčnímu úkolu, jehož zadání najdete v následující kapitole.

 **Příklad:**

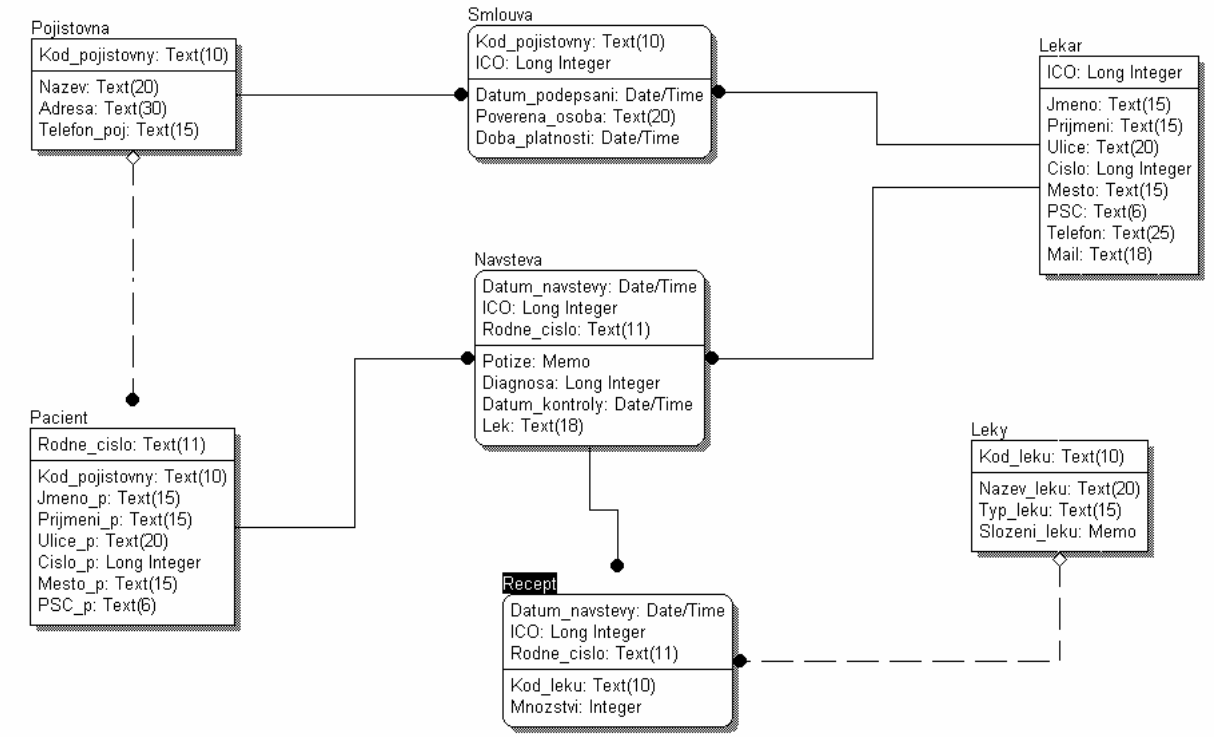
Realita – ordinace u lékaře

Vytvořte model lékařské ordinace, který bude postihovat následující skutečnosti. Lékaři mají smlouvy s pojišťovnami a to tak, že více lékařů může mít smlouvu s toutéž pojišťovnou a zároveň jeden lékař může uzavřít smlouvu s více pojišťovnami. Pacienti jsou pojištěni vždy u jedné pojišťovny. Pacienti chodí na návštěvy k lékařům, na kterých lékaři stanoví diagnózu a předepíší recepty na léky. Atributy a klíče volte dle vlastní zkušenosti. Následuje jedno z možných řešení.

Konceptuální model ordinace



Relační datový model

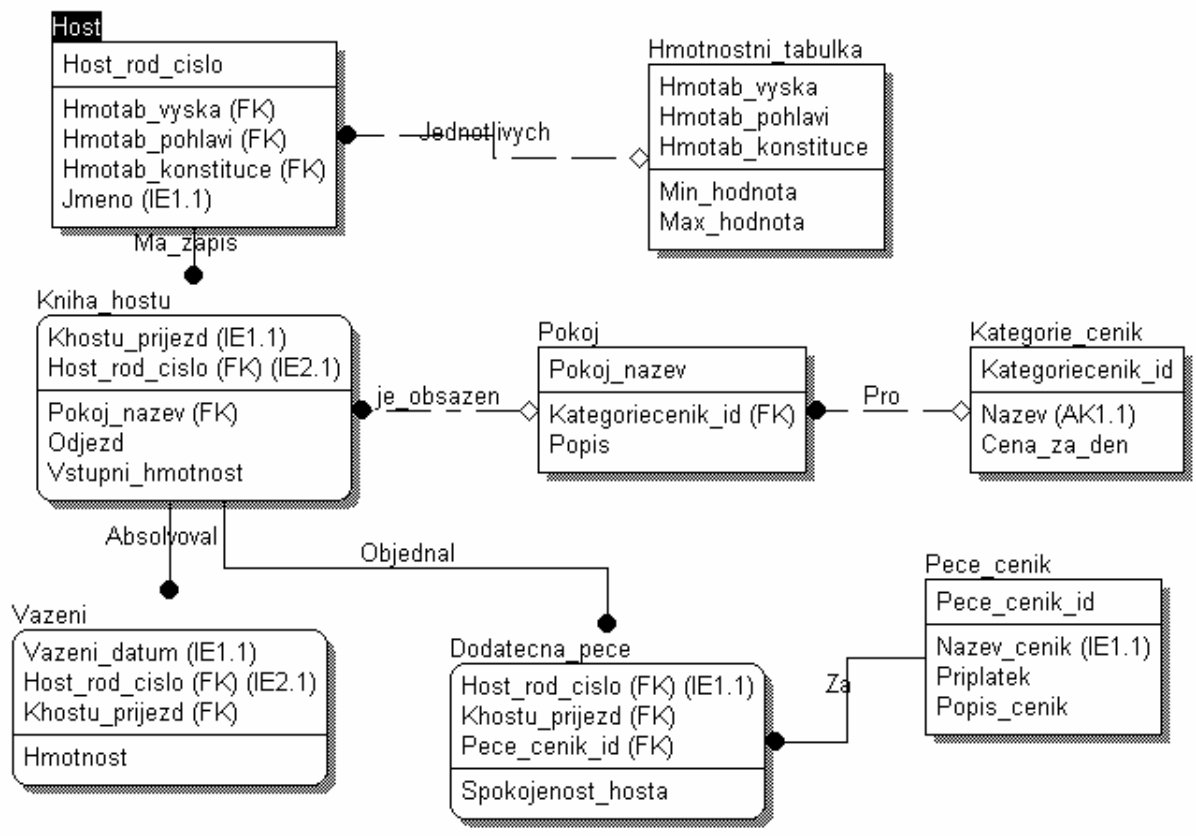


Příklad:

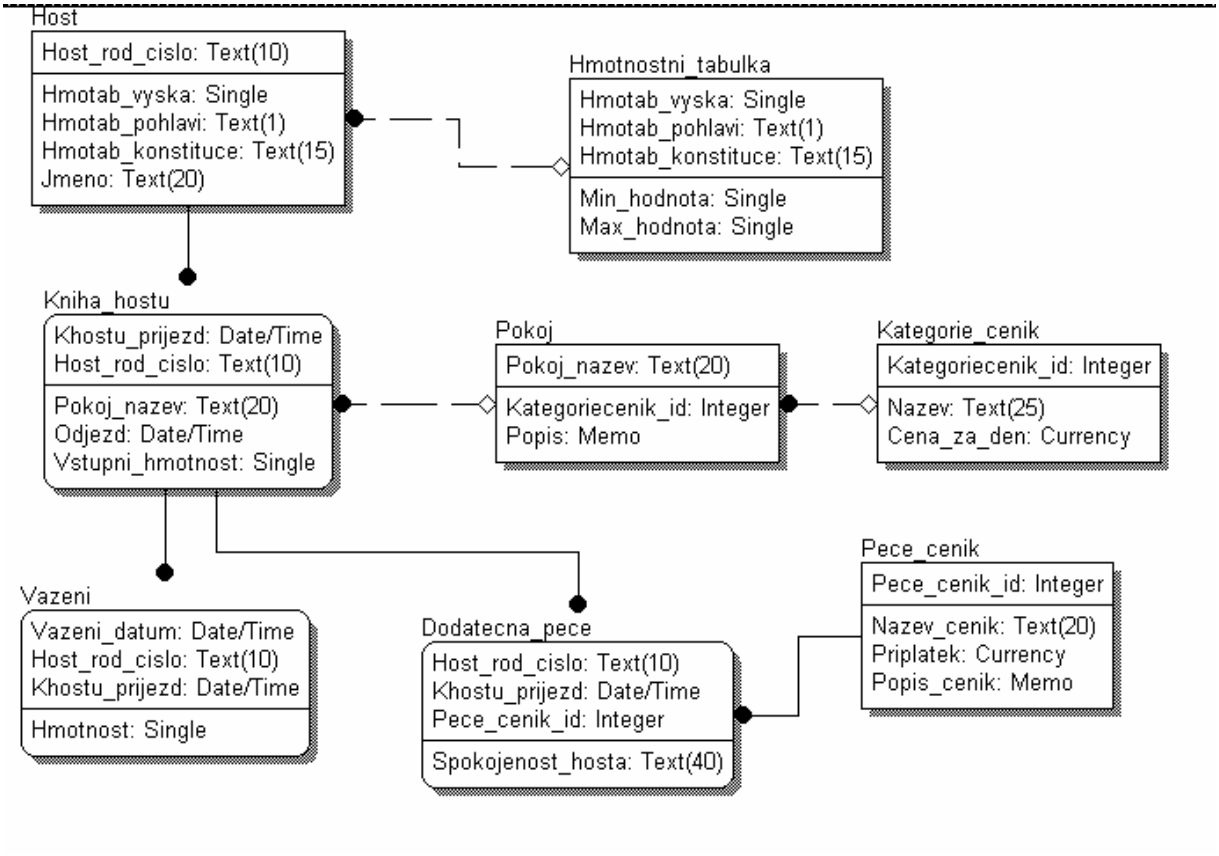
Realita – lázně

Vytvořte model lázeňské péče. Hosté jezdí do lázní a jejich pobyty jsou evidovány v knize hostů. Bude se jednat o lázně, kde bude u pacientů hlavně sledována jejich hmotnost v souvislosti s jejich konstitucí. Informace o různých konstitucích se nachází v hmotnostní tabulce. Hosté budou pravidelně váženi a naměřené hmotnosti budou zapisovány do tabulky vážení. Host si bude moci připlatit dodatečnou péči, která je nad rámec běžného programu. Bude nás zajímat, na kterých pokojích jsou kteří hosté umístěni. Následuje jedno z možných řešení.

Konceptuální model lázní



Relační datový model

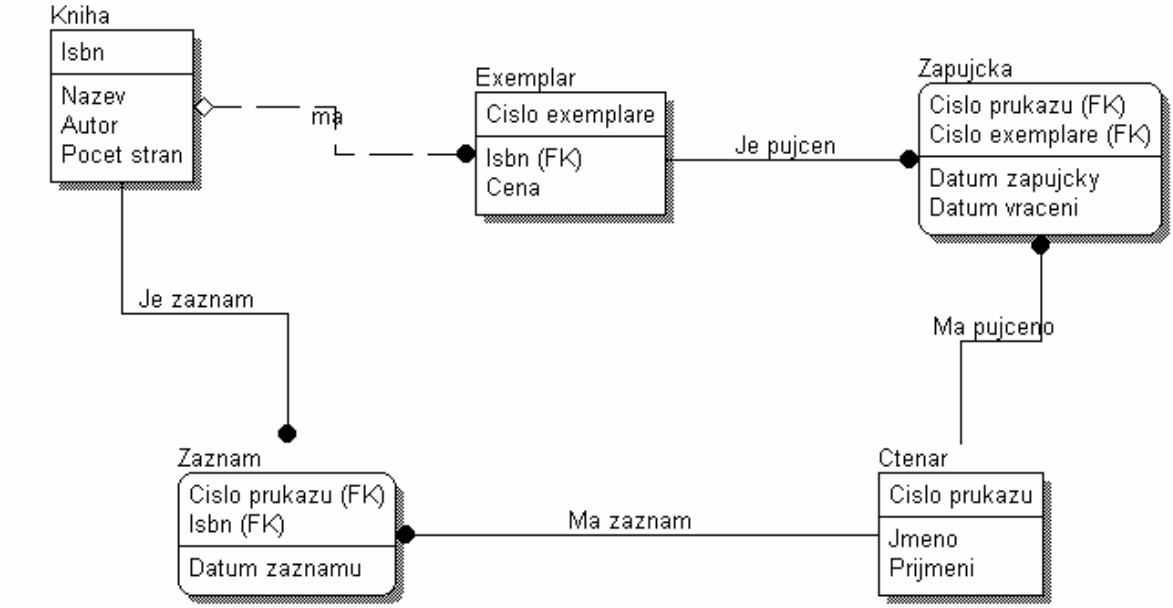


Příklad:

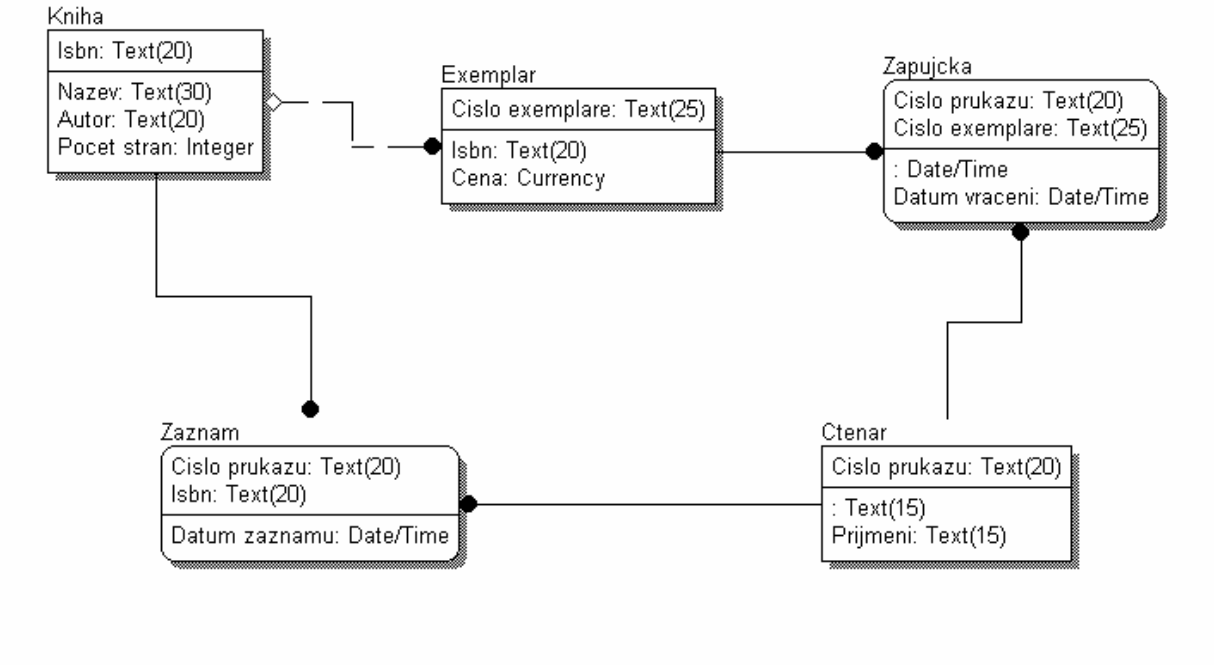
Realita – knihovna

Vytvořte model půjčovny knih v knihovně. Knihovna vlastní exempláře mnoha různých knih. Tyto knihy si mohou půjčovat čtenáři, kteří mají u této knihovny zřízen čtenářský průkaz. Je-li kniha k dispozici, čtenář si půjčí odpovídající exemplář. Pokud kniha momentálně k dispozici není, čtenář si může na tuto knihu vytvořit záznam, pomoci kterého si ji rezervuje, až k dispozici bude. Následuje jedno z možných řešení.

Konceptuální model knihovny



Relační datový model



4 MODUL 4

4.1 ZÁKLADNÍ ORGANIZACE DATOVÉ STRUKTURY SOUBOR

Cíl:

Cílem kapitoly je seznámit čtenáře se základy implementačních technik, tj. technik ukládání dat na vnější paměťová média. Po prostudování této kapitoly bude čtenář schopen vyjmenovat typy souborů a vysvětlit základní rozdíly mezi nimi. Dále bude znát způsob ukládání dat ne vnějších paměťových médiích a způsobem čtení dat do operační paměti.

Klíčová slova:

Primární soubor, index, buffer, sekvenční soubor, indexový soubor, soubor s přímým přístupem.

Průvodce studiem:

Následující kapitola je úplně z jiného soudku, než kapitoly předchozí. Zavádí nás do problematiky úzce spjaté s fyzickým ukládáním dat na paměťová média. V předchozích kapitolách jsme se zabývali databázovou problematikou z naprosto jiného úhlu pohledu (úrovně abstrakce). Zajímala nás realita, o které chceme uchovávat data, zajímalo nás, jak navrhnout struktury, do kterých by se dala data o realitě ukládat. Vysvětlili jsme si, že nejpoužívanějšími strukturami pro ukládání dat jsou databázové relace. Jak ale ve skutečnosti jsou tyto relace uloženy na disku? Následující kapitola se pokusí Vám ozřejmit, jak navržené databázové relace jsou převedeny (mapovány) na soubory dat. Jak již určitě víte, veškerá data, která se na vnějších pamětech ukládají, se ukládají do souborů. Čili naším tématem nyní bude, jak převést (mapovat) relace na soubory. Samozřejmě si musíme nejdříve definovat základní pojmy, pak na příkladech bude ukázáno, jak si lze představit sekvenční soubor, index, atd.

4.1.1 Úvod

Soubor je považován jako základní konstrukt, se kterým je schopen pracovat operační systém. Veškerá data, která jsou uložena na vnějších pamětech počítačů, jsou uložena v souborech. Soubor je logicky organizován jako posloupnost záznamů, které se skládají z položek. Záznamy jednotlivých souborů jsou ukládány (mapovány) do diskových bloků. Na úrovni operačního systému mají bloky konstantní délku (fyzický blok se obvykle rovná 512 B). Ve skutečnosti však jednotlivé výskyty objektů a tedy i instance různých relací mají délku různou.

Existuje několik přístupů, jak tento rozpor vyřešit. Jednou možností je mapovat databázi do několika různých souborů, přičemž jednotlivé soubory budou mít pevnou délku záznamu. Další možností je strukturovat databázový soubor tak, abychom mohli využívat proměnnou délku záznamu. Implementace pevné délky je snadnejší, ovšem většina současných systémů využívá proměnnou délku záznamu z důvodu efektivnějšího přístupu k datům. Při mapování relačních dat do souboru je tedy využíváno dvou technologií. Jedna z nich vytváří pro každou relaci zvláštní soubor s pevnou délkou záznamu. Mnoho rozsáhlých databázových systému však využívá vlastních správců ukládání dat a mapuje celou relační databázi do jednoho souboru. Všechny relace jsou uloženy v jednom souboru. Tento způsob zpracování muže podstatným způsobem urychlit vyhledávání dat v souboru tím, že umožňuje ukládat záznamy z různých relací do stejného bloku souboru.

Na soubor nahlížíme jako na kolekci záznamu. Data mezi vnější a vnitřní pamětí jsou přenášena po logických blocích, které mohou obsahovat různý počet záznamů. Logický blok (cluster, stránka, page) se může skládat z více fyzických bloků. Máme-li soubor vytvořen jako kolekci záznamů s různou délkou, můžeme uchovávat několik záznamů s různou délkou v jednom bloku. Důležité je, že jeden záznam nemůže být uložen ve dvou různých logických blocích.

Výkonnost databázového zpracování je závislá především na minimalizaci počtu přístupů na vnější paměťové médium, tzn. uchovávání co možná největšího počtu bloků v operační paměti. Cílem je maximalizovat šanci, že jsou-li potřebná data, pak jsou přítomna v operační paměti. Paměťový prostor, určený pro ukládání bloků souboru, se nazývá vyrovnávací paměť neboli ***buffer.***

Buffer je část operační paměti, dostupná k ukládání kopií bloků souboru na vnějším paměťovém médiu. Subsystém, zodpovědný za efektivní přidělování paměti, se nazývá ***buffer manager***. Buffer manager interpretuje všechny požadavky na bloky databáze a zjišťuje, je-li požadovaný blok v bufferu, či nikoli. Není-li blok v bufferu, zajistí jeho načtení z disku a umístění do bufferu. Žadateli poskytne adresu bloku v operační paměti. Není-li v bufferu dostatek místa pro načítaný blok, musí být před tímto čtením uvolněn prostor. Obvykle bývá odstraněn ten blok, na něhož nebyl vznesen nejdéle žádný požadavek. Tento blok je zapsán z bufferu do souboru na disk. Cílem celé strategie je minimalizace přístupu na disk. Jednou z nejčastěji používaných částí databáze je katalog (viz. níže). Ten bývá permanentně umístěn v operační paměti.

V klasickém zpracování dat jsou nejpoužívanější sekvenční, indexsekvenční, indexové soubory a soubory s přímým přístupem.

4.1.2 Sekvenční soubor

Je kolekce záznamu, obvykle pevné délky, umísťovaných ve vymezeném prostoru na vnějším paměťovém médiu sekvenčně za sebou. Sekvenční soubor muže být neuspořádaný nebo uspořádaný podle klíče (nejčastěji podle primárního klíče). Vkládání záznamu do uspořádaného sekvenčního souboru se provede zařazením vkládaného záznamu na příslušné místo (dle hodnoty klíče), je-li toto místo volné. Jestliže je místo obsazeno jiným záznamem, vkládaný záznam se umístí do přetokové oblasti. Jestliže je pouze relativně málo záznamu umístěno v přetokové oblasti, rychlost přístupu k záznamům to podstatně neovlivní. Při velkém počtu záznamů v přetokové oblasti je třeba provést reorganizaci souboru, tj. řazení záznamu souboru sekvenčně setříděně podle klíče.

4.1.3 Mapovaní relačních dat do sekvenčního souboru

Některé relační databázové systémy uchovávají každou relaci ve zvláštním souboru. V tomto případě mohou být n-tice relace ukládány do záznamu pevné délky. Tato jednoduchá struktura umožňuje plně využívat služeb operačního systému ke správě datových souborů. Ovšem většina rozsáhlých databázových systémů uchovává mnoho relací v jednom souboru s proměnnou délkou záznamu. Řízení takového souboru je plně v kompetenci systému řízení báze dat.

Příklad:

Následující příklad demonstruje způsob uložení záznamů v sekvenčním souboru. Soubor je setříděn podle hodnoty primárního klíče.

	Adámek	Jan	Dlouhá	1115
	Čížek	Jiří	Krátká	1025
	Konvička	Adam	Poděbradova	15
	Král	Petr	Masarykova	5
	Matějka	Miroslav	Vrchlického	1520
	Neruda	Jan	Malostranská	1
	Žídek	Karel	Vltavská	10

Sekvenční soubor

Následuje soubor po vložení záznamu (Kovář, Pavel, Široká, 50)

	Adámek	Jan	Dlouhá	1115
	Čížek	Jiří	Krátká	1025
	Konvička	Adam	Poděbradova	15
	Král	Petr	Masarykova	5
	Matějka	Miroslav	Vrchlického	1520
	Neruda	Jan	Malostranská	1
	Žídek	Karel	Vltavská	10
	Kovář	Pavel	Široká	50

Sekvenční soubor po vložení záznamu

4.1.4 Ukládání datového slovníku

Kromě dat, která jsou v relacích a ukládají se do souboru, relační databázový systém potřebuje ukládat i data o relacích, která se nazývají metadata (datový slovník, systémový katalog).

Je třeba uchovávat tyto typy informací:

- Jména relací.
- Jména atributu každé relace.
- Domény atributu.

- Jména a definice pohledu na databázi.
- Integritní omezení na všechny relace.
- Jméno indexu.
- Jméno relace, na kterou je vytvořen index.
- Jména atributu, na které je vytvořen index.
- Typ indexu.

Navíc mnoho systému udržuje dodatečné informace:

- Jména autorizovaných uživatelů.
- Počet n-tic v každé relaci.
- Metoda ukládání dat (např. clustered, nonclustered).

4.1.5 Indexování

Je metoda, která umožňuje zrychlit přístup k datům na disku. Kromě primárního souboru, obsahujícího všechna data databáze, který je výsledkem mapování relací, databáze může obsahovat přídavnou datovou strukturu, spojenou vždy s vybranou položkou (položkami) primárního souboru. Tato položka (atribut) se nazývá ***vyhledávací klíč*** a datová struktura se nazývá ***index***.

Existují dva typy indexu:

- **hustý index**

je taková datová struktura, která obsahuje všechny hodnoty vyhledávacího klíče a příslušné odkazy do primárního souboru

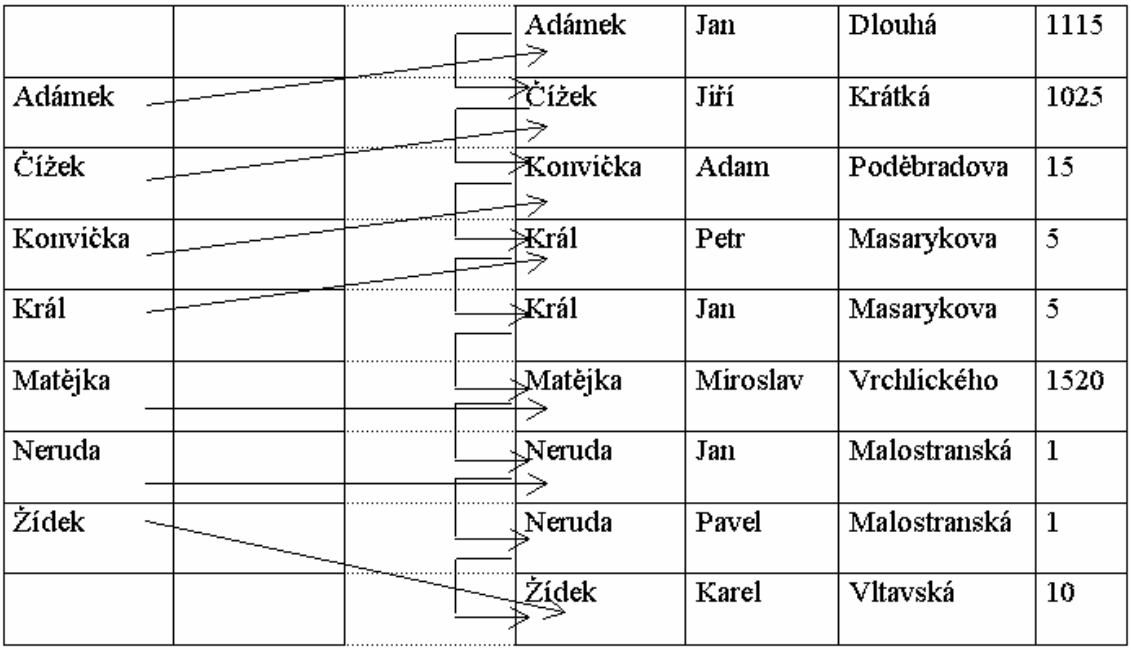
- **řídký**

je vytvořen pouze na některé hodnoty vyhledávacího klíče. Při hledání záznamu s hodnotou vyhledávacího klíče, která se v indexu nenachází, se použije odkaz přirazený nejbližší nižší hodnotě vyhledávacího klíče a v primárním souboru se hledaný záznam dohledá sekvenčním způsobem.

 **Příklad:**

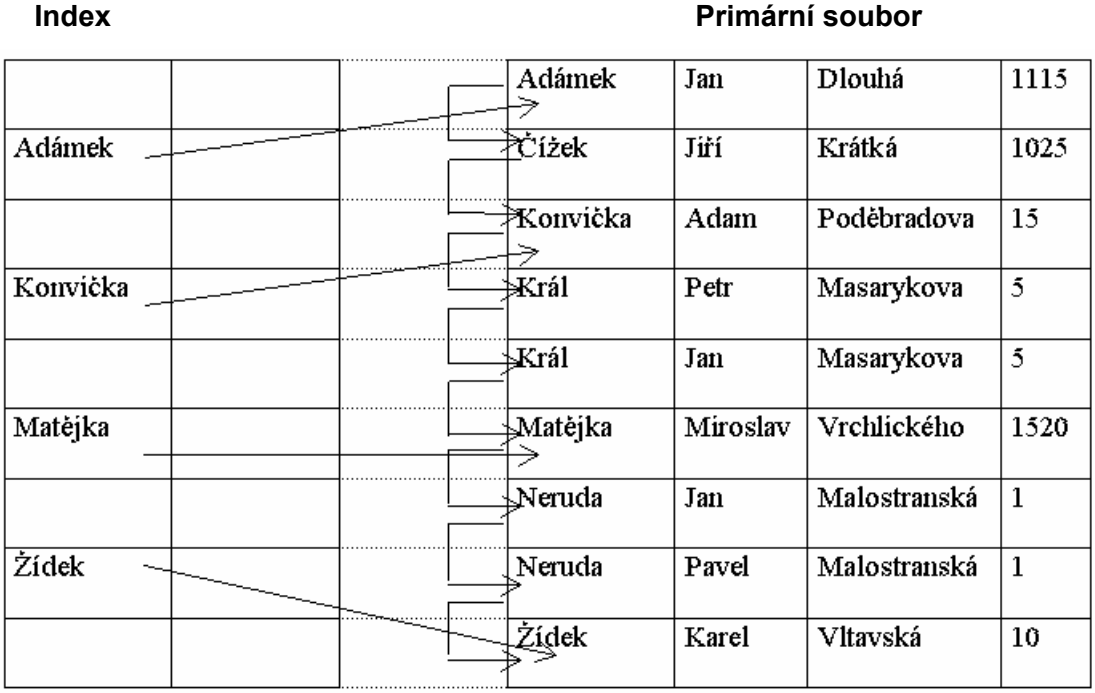
Příklad hustého indexu:

Index **Primární soubor**



Příklad:

Příklad řídkého indexu:



4.1.6 Index-sekvenční soubor

Je soubor tvořen primárním souborem, tj. setříděným sekvenčním souborem (setříděným podle primárního klíče) a přídavnou datovou strukturou nazývanou index. Vyhledávacím klíčem muže být libovolný atribut a lze vytvářet libovolné množství indexů. Protože velikost indexu je závislá na počtu záznamů primárního souboru, bývají indexy rozsáhlých souboru velké, což kromě nároků na paměť způsobuje i zpomalování aktualizačních operací. Aktualizace záznamu v primárním souboru totiž vyžaduje i aktualizace indexu. Aby nebylo nutno v souvislosti s aktualizačními operacemi měnit uložení záznamu v primárním souboru, využívají se přetokové oblasti.

Přetoková oblast je přídavný prostor pro ukládání záznamu do primárního souboru. Je svázána s primárním souborem prostřednictvím směrníků. To vyžaduje, aby fyzické záznamy primárního souboru obsahovaly položku "ukazatel" pro umístění směrníku do přetokové oblasti. Po mnoha aktualizačních operacích (mazání záznamu, vkládání záznamu) s využitím přetokových oblastí se soubor stává příliš komplikovaný. To má za následek zpomalení přístupu k záznamům souboru. Proto je třeba soubor přeorganizovat, tj. uložit nově zařazené záznamy na patřičné místo v primárním souboru. Počet přístupů na disk je možno snížit umístěním indexu do operační paměti, avšak v souvislosti s dostupnou operační pamětí a počtem a rozsahem indexu.

4.1.6.1 Implementace index-sekvenčního souboru metodou kapes (BUCKETS)

Základní charakteristiky:

- Primární soubor je rozmístěn do kapes.
- V kapse je předem definovaný počet záznamů.
- Informace o obsazených pozicích se uchovává v bitové mapě.

- "0" na i-té pozici označuje volné místo pro i-tý záznam.
- Prázdná kapsa je rezervována pro záznamy s klíčem menším než klíč prvního záznamu v primárním souboru.
- Počet záznamů (počet pozic) v kapse se nazývá blokovací faktor.

Příklad:

Mějme soubor indexů:

Hodnoty klíče	**Adresy bloků**
	00
Adámek	10
Král	20

Dále mějme primární soubor rozmístěn do kapes:

Bitová mapa	**Záznam**	**Záznam**	**Záznam**	**Záznam**	**Záznam**	**Ukazatel**
(00)00000						0
(10)11111	Adámek	Dlouhý	Čížek	Frič	Hynek	0
(20)11110	Král	Křivý	Malý	Kyselý		0

Proveďme aktualizační operace: D(Dlouhý), I(Dvořák), I(Březina), I(Novák), I(Adam)

Dostaneme následující soubor indexů:

Hodnoty klíče	**Adresy bloků**
	00
Adámek	10
Král	20

A následující primární soubor:

Bitová mapa	**Záznam**	**Záznam**	**Záznam**	**Záznam**	**Záznam**	**Ukazatel**
(00)10000	Adam					0
(10)11111	Adámek	Dvořák	Čížek	Frič	Hynek	30
(20)11111	Král	Křivý	Malý	Kyselý	Novák	0
(30)10000	Březina					0

4.1.7 Indexový soubor

Je tvořen primárním souborem a indexy pro různé vyhledávací klíče. Indexovány jsou záznamy (hustý index), proto na rozdíl od index-sekvenčního souboru nemusí být primární soubor setříděn a nevyžaduje umístění do souvislé části paměti počítače. Rovněž neuvažuje přetokové oblasti. Je-li index vytvořen na inverzní klíč (tj. takový, který připouští duplicitní hodnoty), může se v indexu opakovat stejná hodnota vícekrát.

4.1.8 Soubor s přímým přístupem

Metoda přímého přístupu umožňuje rychlý způsob vyhledávání záznamů podle hodnoty primárního klíče i jeho aktualizaci. Princip souborů s přímým přístupem spočívá v tom, že při jeho ukládání do paměti se podle určitého algoritmu vypočte z hodnoty primárního klíče adresa bloku (stránky) disku, na níž má být daný výskyt záznamu uložen. Hodnota primárního klíče je vstupem a adresa stránky je výstupem algoritmů, které se nazývají ***hašovací algoritmy.***

Soubor s přímým přístupem má primární záznamy rozptýleny v paměťovém prostoru velikosti **M * R**, kde **M >= N** (**N** … počet záznamu v souboru, **R** … velikost záznamu v bytech) pomocí hašovací funkce **f** definované z **K** do **{0,1, …, M-1}**. Interval **<0, M-1>** se nazývá adresový prostor a množina **K** je množina všech hodnot primárního klíče. Hašovací funkce je výpočetně jednoduchý předpis, který transformuje hodnotu klíče do relativních adres. Absolutní adresy se vypočtou lineární transformací.

Jednou z nejefektivnějších transformací je funkce **f = k mod M'**, kde **M'** je nejbližší prvočíslo menší než **M** a **k** je číselná hodnota klíče. Funkce **f** není obvykle prostá. Tato skutečnost může způsobit kolize, tzn. že několik záznamů má být umístěno na stejné adrese. Kolize lze řešit tzv. otevřeným hašováním, kdy kolidovaný záznam se umístí na nejbližší volné místo s vyšší adresou, rehašováním, kdy kolidovaná adresa se stane vstupem do hašovací funkce a výstupem je rehašovaná adresa, eventuálně lze využít přetokových oblastí, kdy kolidující záznamy jsou zřetězeny pomocí ukazatelů. Je-li hašovací funkce prostá, hovoříme o perfektním hašování.

Při použití statických hašovacích metod musíme vyřešit při zakládání souboru některé volby. Jedná se o výběr hašovací funkce na danou velikost souboru. Toto může způsobit problémy, jakmile se databáze rozroste. Lze sice horní odhad udělat tak, aby nemohlo dojít k zaplnění vyhrazeného prostoru, ale tím se plýtvá místem na disku. Obvykle je velikost prostoru volena **N = 0,8 M**

Nutnost pevného stanovení prostoru pro ukládání záznamu souboru je tedy vážným problémem statických hašovacích technik. Statické hašování vede ke třem třídám voleb a to k výběru hašovací funkce založeném na současné velikosti souboru, výběru hašovací funkce založeném na předpokládané velikosti souboru a na periodické reorganizaci hašovací struktury v důsledku růstu souboru dat. Reorganizace zahrnuje výběr nové hašovací funkce, přepočtení hašovací funkce pro všechny záznamy v souboru a generování nových adres . Tato operace je velmi časově náročná a navíc po dobu reorganizace je zakázán přístup k souboru pro jakékoli operace.

Z praktických důvodů je vhodnější využívat takové struktury, které umožňují pružnou změnu externí paměti potřebné pro uložení databáze. Takovéto techniky jsou nazývány dynamické hašovací funkce.

 **Shrnutí:**

Základní strukturou pro ukládání dat na vnější paměťové médium je soubor. Soubor se skládá z jednotlivých záznamů, které se skládají z položek. Můžeme mít soubor s pevnou délkou nebo soubory s proměnnou délkou. Co se týká typu souborů, je možno používat soubory sekvenční, index-sekvenční, indexové a soubory s přímým přístupem. Data, uložená v souborech se dají zpracovávat pouze v operační paměti. Přesouvání dat z vnější do operační paměti se děje prostřednictvím bufferů (vyrovnávacích pamětí). Jednotkou přenosu

dat je tzv. logický blok (cluster, page, stránka). Základní ideou při fyzické implementaci souborů a databází je minimalizace přístupů na disk při zpracování dat. Přístupy na disk totiž nejvíce zpomalují celý proces zpracování dat. Počet přístupů na disk je samozřejmě závislý na tom, s jak robustní databází pracujeme, záleží ovšem rovněž na tom, jak jsou data organizována, jakým postupem se příslušná data vyhledávají, atd. Nejsnadnější, ovšem nejpomalejší způsob je ukládání dat do sekvenčních souborů. Zrychlení představuje setřídění záznamů podle primárního klíčem vytvoření indexů na různé vyhledávací klíče, vytvoření souboru s přímým přístupem, atd.

 **Kontrolní otázky:**

1. Co je to buffer a k čemu se používá?
2. Jaký je rozdíl mezi fyzickým a logickým blokem?
3. Jak je možno ukládat primární soubor?
4. Co je to index a jaké typy indeků znáte?

 **Pojmy k zapamatování:**

Primární soubor

Index

Buffer

Logický blok (cluster, page, stránka)

Sekvenční soubor

Indexový soubor

Soubor s přímým přístupem.

 **Průvodce studiem:**

Výborně! Zvládli jste to, prokousali jste se tímto textem a tak Vám blahopřeji! Pokud navíc umíte odpovědět na kontrolní otázky, je to úplně skvělé. Určitě si odpočiňte, než začnete studovat následující kapitolu.

4.2 PŘÍSTUPOVÉ TECHNIKY

Cíl:

Cílem kapitoly je rozšířit čtenáři vědomosti o možnostech rychlejšího vyhledání záznamů v souborech. Jsou zde popsány dvě statické organizace, řetězená organizace a indexy. Obě organizace jdou vylepšením klasických sekvenčních souborů. Po prostudování kapitoly by měl čtenář dokázat vysvětlit rozdíl mezi pojmem klíč a index a měl by umět popsat algoritmus tvorby řetězené a invertované organizace (indexy).

Klíčová slova:

Řetězení, hustý index, řídký index, vyhledávací klíč.

4.2.1 Úvod

Jak již bylo uvedeno v kapitole 4.1., problematika uložení záznamů do souborů a způsoby vyhledání příslušných záznamů na vnějším paměťovém médiu podstatně ovlivní celkovou dobu zpracování dat. Sekvenční soubory mohou být vylepšeny tzv. řetězenou organizací, která pro daný vyhledávací klíč vytváří směrníky na příslušné (podle daného klíče setříděné) záznamy. Další zrychlení vyhledávání je možno realizovat vytvořením přídavných datových struktur, tzv. indexů. V této kapitole se budeme věnovat technice řetězení. Indexy, které jsme již uvedli v kapitole předchozí, ještě doplníme. Techniky, které jsou popsány v této kapitole se nazývají technikami statickými. Důvodem je ta skutečnost, že záznam, jednou zapsaný na určitou adresu na vnější paměťové médium, na této adrese zůstává až do okamžiku, kdy je smazán nebo kdy je celý soubor tzv. reorganizován. Reorganizace je operace, která čte jednotlivé záznamy souboru a zapisuje je setříděně podle hodnoty (většinou) primárního klíče.

4.2.2 Řetězení

je spojení záznamů pomocí nasměrování na následující záznam. Řetězení realizují směrníky (poiters). **Směrníky** jsou položky záznamu, které oznamují, kde se nachází následující záznam se stejnou hodnotou alternativního resp. inverzního klíče. Logický směrník je tedy přidanou položkou do záznamu. První záznam s danou hodnotou alternativního klíče se nazývá záhlavím a je označen zvláštním směrníkem. Poslední záznam řetězu má hodnotu směrníku rovnu NULL. K provozování zřetězeného seznamu je třeba zadat vstupní směrník, který ukazuje na záhlaví seznamu a při každém nalezení dalšího záznamu testovat, zda směrník nemá hodnotu NULL.

4.2.2.1 Databázové operace ve zřetězených organizacích

Operace **FETCH** (operace načítání logických bloků do bufferu) znamená u řetězené organizace postupné sledování směrníků a testování, zda-li jde o hledaný záznam.

Operace **UPDATE** (operace změny hodnota databázové položky) se uskuteční tak, že potřebné položky změní svou hodnotu a směrník zůstane stejný.

Operace **DELETE** (operace smazání záznamu) musí dbát na to, aby směrník záznamu ukazujícího na rušený záznam se nahradil směrníkem rušeného záznamu. Záznam určený ke smazání se označí za neplatný. Řetězec se zkrátí o jeden záznam - vznikne prázdný prostor (díra), do kterého je možno operací Insert zapsat záznam, který logicky na toto místo patří.

Operace **INSERT** (operace vložení záznamu) způsobí, že se řetězec prodlužuje o jeden záznam včetně příslušného směrníku. Na vkládaný záznam bude ukazovat směrník dosud posledního záznamu, kde nahradí koncovou hodnotu NULL. Nově vzniklý směrník je adresa prvního volného prostoru (díry).

Příklad:

Mějme relaci R(id_v, jméno_v, stát_v, město_v)

Vyhledávací klíč: stát_v

Vytvořte zřetězení pro výrobce ze SRN

Záhlaví pro stát_v = SRN je V4 - směrník na první záznam, kde stát_v = SRN

Zřetězená organizace je následující:

id_v	jméno_v	stát_v	město_v	směrník
V1	Škoda	ČR	M.Boleslav	
V2	Renault	Francie	Paříž	
V3	Opel	USA	Detroid	
V4	Ford	SRN	Kolín n/R	V5
V5	Volksw.	SRN	Wolfsburg	V27
.				
.				
.				
.				
V27	.....			NULL

4.2.3 Invertované seznamy - indexy

Nejjednodušší způsob indexu je tzv. hustý index, který tvoří tabulka se dvěma sloupci. První sloupec je sloupec hodnot vyhledávacího klíče a druhý sloupec je odkaz na záznam do primárního souboru - adresa záznamu v primárním souboru.

Tato tabulka tvoří pomocnou datovou strukturu a její záznamy jsou setříděny podle hodnoty vyhledávacího klíče. Je možno vytvořit libovolně mnoho indexů na libovolné atributy (vyhledávací klíče). Pokud atribut nebo skupina atributů nepřipouští duplicitu hodnot, jedná se o alternativní klíč a index na alternativní klíč. Pokud atribut nebo skupina atributů duplicitu připouští, jedná se o klíč inverzní.

Nehusté indexování je takové, kdy směrník neukazuje na adresu záznamu, ale na adresu logického bloku, který je zaveden do operační paměti. V tom případě je třeba po zavedení bloku prohledat obsah bloku a nalézt požadovaný záznam. Nehusté indexy jsou zpravidla méně náročné na paměť.

Výše popsané indexy jsou tzv. statickými strukturami a to proto, že jednotlivé záznamy v indexové struktuře nemění své místo po celou dobu existence indexu, resp. po dobu existence příslušného záznamu v indexu. Výjimku tvoří reorganizace souboru, která provede celkové přeorganizování záznamů jak vprimárním souboru, tak ve všech indexech. Operace reorganizace je velmi náročná a po dobu této operace je celá databáze uzamčena pro jakékoli další operace.

Úkoly k zamyšlení:

1. Mějme relaci Student (Osobní_čislo, Jméno, Příjmení, Ročník, Skupina). Navrhněte několik instancí této relace a proveďte mapování relace do sekvenčního souboru s využitím řetězené organizace pro vyhledávací klíč Příjmení.
2. Na výše popsanou relaci vytvořte hustý index na Osobní_číslo.
3. Na výše popsanou relaci vytvořte řídký index na Příjmení.

4.3 DYNAMICKÉ ORGANIZACE

Cíl:

Základním cílem kapitoly je ukázat čtenáři rozdíl mezi statickými a dynamickými strukturami a poukázat na výhody dynamických struktur. Dynamické struktury jsou demonstrovány na indexu a souboru s přímým přístupem. V případě indexu je dynamickou strukturou tzv. B-strom, v případě primárního souboru s přímým přístupem je dynamickou strukturou tzv. dynamické hašování. Po prostudování kapitoly by měl být čtenář schopen vysvětlit rozdíly statické a dynamické struktury, vysvětlit výhody dynamických struktur. Dále by měl být schopen sestrojit B-strom a aktualizovat jej v souvislosti s operacemi Insert a Delete (štěpení a slévání uzlů).

Klíčová slova:

Statická struktura, dynamická struktura, B-strom, neredundantní B-strom, dynamické hašování.

4.3.1 Úvod

Dynamickými organizacemi budeme rozumět takové struktury, které se v čase mění, tzn. že pozice záznamu v takové struktuře není neměnná, ale mění se podle toho, jak probíhají aktualizační operace. Jednou zapsaný záznam na nějaké adrese může tuto adresu opustit z toho důvodu, že na tuto adresu logicky patří jiný záznam. Dynamické organizace proto nemusí využívat přetokových oblastí a časová náročnost vyhledávání záznamů není tudíž zvětšována nutností hledání záznamů v přetokových oblastech. Z dynamických organizací, kterých je samozřejmě více typů, se budeme zabývat pouze B-stromy a krátce se zmíníme o dynamickém hašování.

4.3.2 B – stromy

B-strom je dynamická indexová struktura. Z důvodu efektivity operací INSERT a DELETE je tato struktura nejpoužívanější indexovou strukturou. Struktura je založena na vyváženém stromu, ve kterém každá cesta od kořene k listu má stejnou délku, což znamená, že doba vyhledání libovolného záznamu je stejná. B-strom se chová dynamicky, tzn. dochází při aktualizačních operacích ke štěpení uzlů (bloků záznamů), se kterou souvisí alokace dalších uzlů a ke slévání uzlů. B – stromy jsou víceúrovňové indexy, přičemž jednotlivé úrovně se vytvářejí následujícím způsobem:

1. Primární soubor je indexován podle dané položky (vyhledávací klíč) a tím je vytvořen index *I1.*
2. Vytvořený index *I1* je znovu indexován a tím je vytvořen index *I2.*
3. Druhý krok je možno podle potřeby opakovat.

B – stromy obsahují tři typy údajů:

1. Směrníky na indexy.
2. Hodnoty vyhledávacího klíče, podle něhož je soubor indexován.
3. Směrníky na výskyty záznamu, tj. odkazy do primárního souboru.

Algoritmus vytváření B – stromu začíná od listu, tj. uzlu stromu, který obsahuje pouze odkazy na záznamy primárního souboru. Algoritmus musí zachovávat vyváženost stromu. Mějme B – strom, který bude obsahovat n-1 hodnot vyhledávacího klíče a označme je K_1, K_2, …., K_{n-1} a n ukazatelů P_1, P_2, …,P_n . Hodnoty vyhledávacího klíče jsou setříděné, tzn. je-li i < j, je taky $K_i < K_j$.

Struktura listu je pak následující:

P_1	K_1	P_2	.....	P_{n-1}	K_{n-1}	P_n

Příklad:

Následující příklad Vám ukáže, jak dochází k postupnému vzniku B-stromu. Všimněte si, kdy dochází ke štěpení uzlu a jaký to má dopad na celou B-strukturu. Rovněž si všimněte, kdy dochází ke zvýšení počtu úrovní B-stromu. Označení 0, 1, 2, … jsou odkazy na další uzel B – stromu, označení Z0, Z1, … jsou odkazy do primárního souboru.

Adresa uzlu 0

Konvička	-	-
Z0	-	-

Vložení záznamu s hodnotou vyhledávacího klíče Čížek

Adresa uzlu 0

Čížek	Konvička	-
Z1	Z0	-

Vložení záznamu s hodnotou vyhledávacího klíče Matějka.

Adresa uzlu 2

Konvička	-	-
0	1	-

Adresa uzlu 0

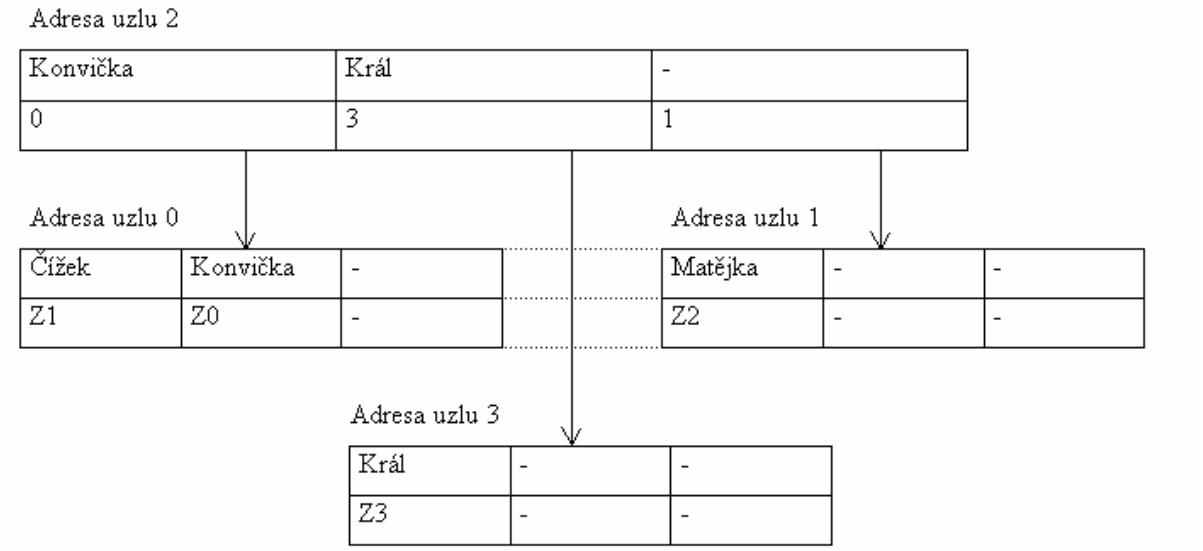
Čížek	Konvička	-
Z1	Z0	

Adresa uzlu 1

Matějka		
Z2		

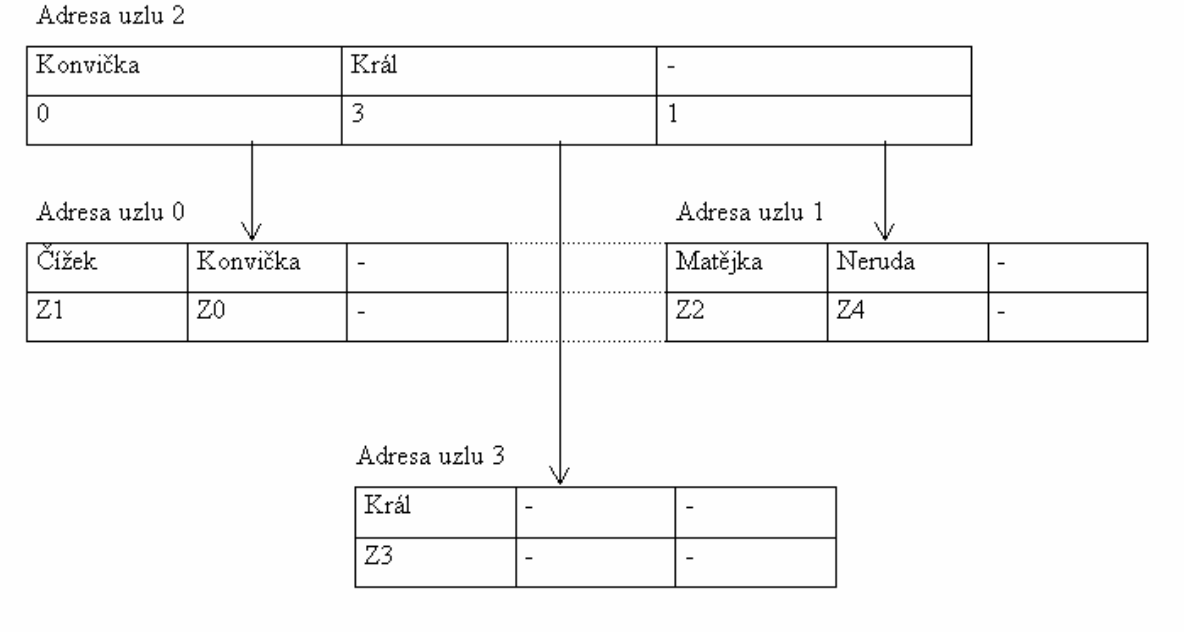
Vložení záznamu s hodnotou vyhledávacího klíče Matějka způsobilo alokaci dalších dvou bloků o adresách 1 a 2.

Vložení záznamu s hodnotou vyhledávacího klíče Král.



Vložení záznamu s hodnotou vyhledávacího klíče Král způsobilo alokaci dalšího bloku o adrese 3.

Vložení záznamů s hodnotami vyhledávacího klíče Neruda .



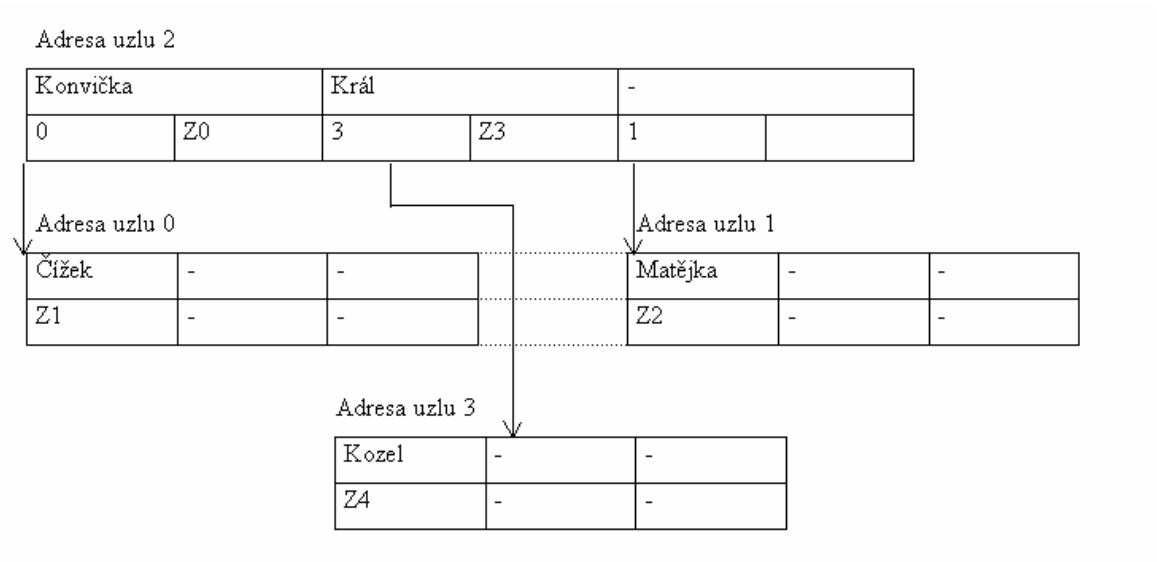
4.3.3 Neredundantní B-stromy

Neredundantní B – stromy jsou modifikací B – stromů. Neredundantní B-strom je sice vyvážený strom, odkaz do primárního souboru však lze najít dříve než v listu stromu. Doba vyhledání záznamu je pak stejná jako u redundantních B-stromů, pouze je-li hodnota vyhledávacího klíče rovna hodnotě uvedené v uzlu, je doba vyhledání kratší. Neredundantní B-strom je tedy vylepšením B-stromu. Vyžaduje však přidání odkazu do primárního souboru do všech uzlů B-stromu, tedy ne jen do listu, jak je tomu u B-stromu. Prohledávání v

neredundantních B – stromech nemusí dospět až do listu. Odkaz na záznam do primárním souboru se může vyskytovat v kterémkoli uzlu.

Příklad:

Následující příklad ukazuje, jak může vypadat neredundantní B-strom. Je-li hodnota vyhledávacího klíče menší než Konvička, postupujeme ve vyhledávání na adrese 0 (další blok B-stromu). Je-li hodnota vyhledávacího klíče větší než Konvička, postupujeme ve vyhledávání na adrese 3 (další blok B-stromu). Je-li hodnota vyhledávacího klíče rovna Konvička, najdeme v uzlu B-stromu přímo odkaz do primárního souboru (adresa Z0).



4.3.4 Dynamické hašování

Cíl

Základním cílem kapitoly je seznámit studenta se základními myšlenkami rozšiřitelného hašování. Po prostudování této kapitoly bude student umět rozlišovat mezi statickými a dynamickými organizacemi souborů a databází, bude umět sestavit algoritmus zápisu záznamů do souboru s využitím rozšiřitelného hašování.

Klíčová slova

Rozšiřitelné hašování, hašovací funkce, adresář, logický blok (kapsa) primárního souboru.

Dynamické hašování je metoda průběžné rekonstrukce (reorganizace) datových struktur. Poprvé popsána v roce 1981.

Jedná se o modifikaci periodické rekonstrukce, která se liší:

V průběhu rekonstrukce se nezastaví zpracování (modifikace databáze, vyhledávací operace, atd.).

Operace se provádějí nad původní strukturou a paralelně probíhá reorganizace (vzniká nová datová struktura).

Po vytvoření nové struktury se stará „zapomene" a operace se provádějí nad touto novou strukturou.

Základní důvody pro přechod na dynamické hašování:

Velikost externí paměti, potřebné pro ukládání dat, by se měla pružně měnit podle stávajícího objemu dat.

Počet přístupů na disk pro většinu požadavků by měl být konstantní.

ROZŠIŘITELNÉ HAŠOVÁNÍ

Zavádí pomocnou strukturu - **adresář**.

Záznamy adresáře obsahují odkazy do primárního souboru. Některé záznamy adresáře mohou obsahovat totožné odkazy (odkazy do totožných logických bloků tzv. kapes).

Hašovací funkce h

Transformuje hodnotu primárního klíče k na adresu, která spadá do intervalu $<0,2^{r+1}-1>$

r+1 Délka reprezentace hodnoty **h(k)** v bitech (počet bitů). Typická hodnota je 32.

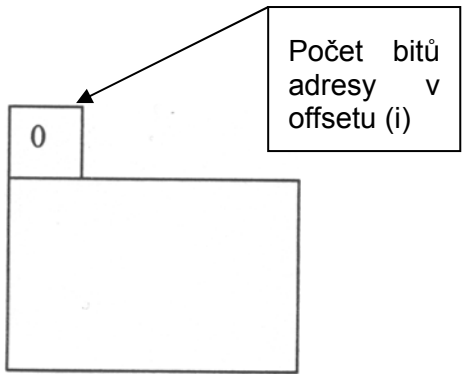
Mějme klíč **k** a hašovací funkci

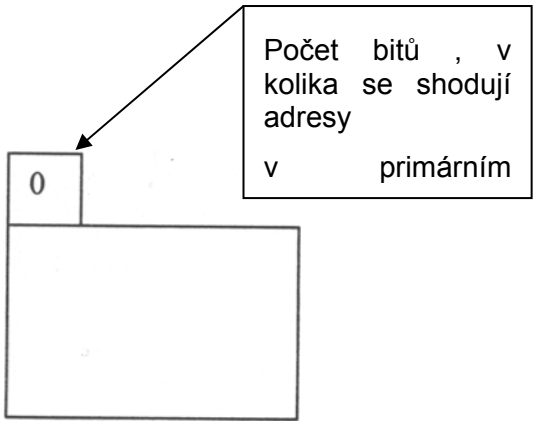
$h(k) = b_r b_{r-1} .... b_o$

Z adresy **$b_r b_{r-1} .... b_o$** vybereme pouze část (**d** bitů) a těchto **d** bitů se jako offset adresy vloží do přidané struktury, adresáře. Velikost **d** se mění v souvislosti s nárůstem záznamů v databázi.

Prázdný soubor

Adresář Logický blok (kapsa) primárního souboru





Je-li **i=1**, pak adresář obsahuje pouze dva záznamy:

0 Odkaz na záznamy, jejichž adresa začíná 0.

1 Odkaz na záznamy, jejichž adresa začíná 1.

Příklad:

Mějme následující záznamy, které se budou zapisovat do databáze pomocí rozšiřitelného hašování.

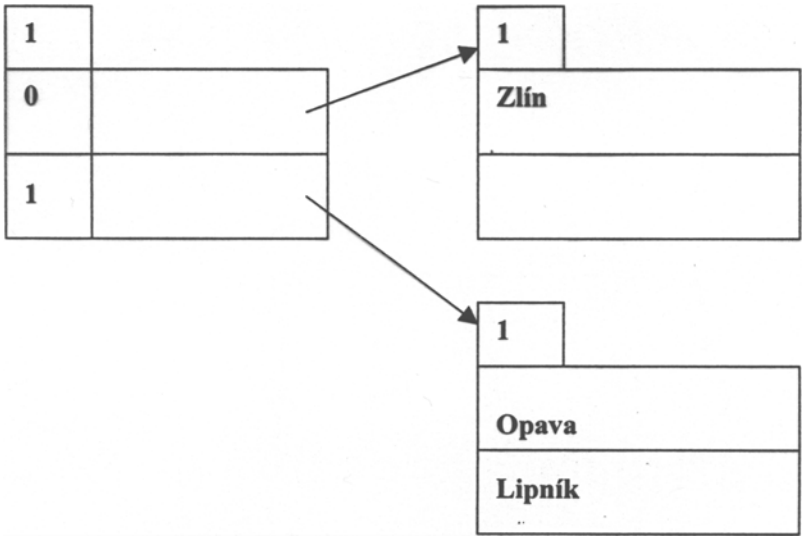
Město	Ulice	Číslo
Brno	Dlouhá	25
Kroměříž	Jánská	100
Lipník	Anglická	142/a
Olomouc	Francouzská	1725
Opava	Lipová	5
Ostrava	Nádražní	128
Zlín	Úzká	8

Předpokládejme, že **primárním klíčem** je atribut **Město** a hašovací funkce **h** generuje následující adresy:

Město	**Adresa**
Brno	00101101111110110010110000110000
Kroměříž	11010101110111100100011010010011
Lipník	10100011101000001100011010011111
Olomouc	10000111111011011011111100111010
Opava	11110001001001001001001101101101
Ostrava	10110101101001101100100111101011
Zlín	01011000001111111001110000000001

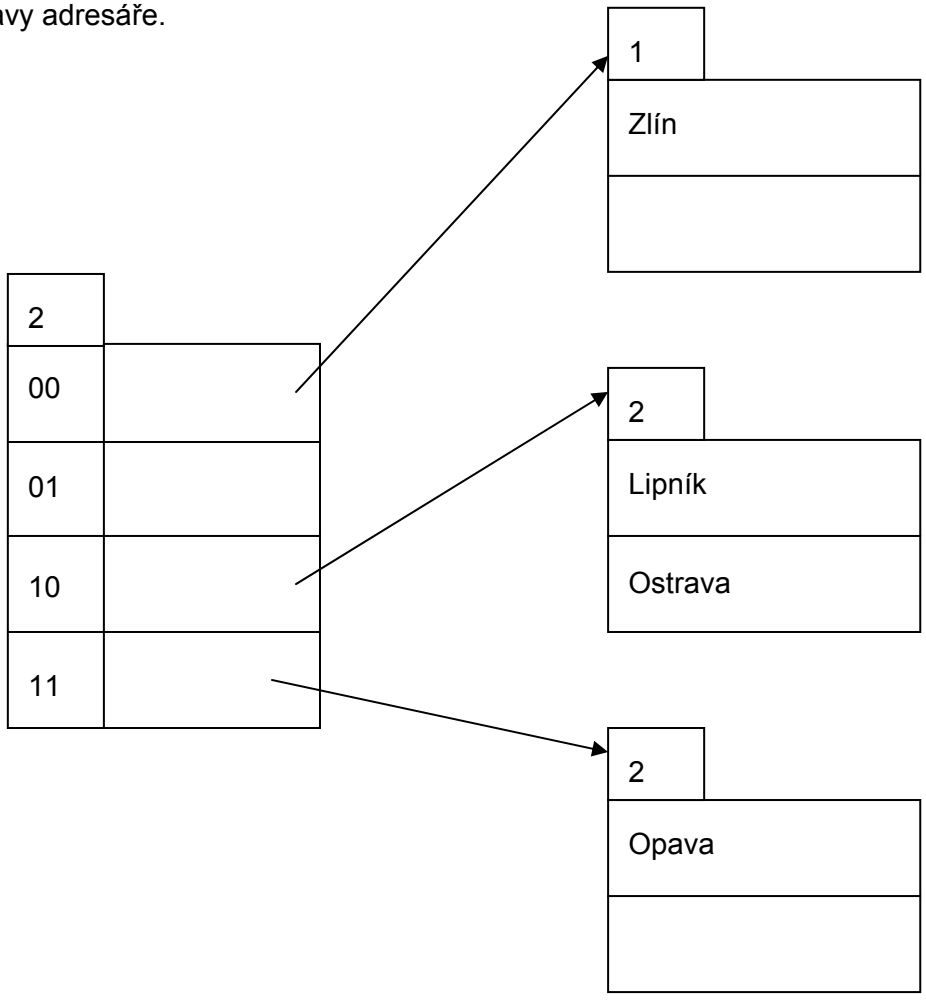
Dále předpokládejme, že každá kapsa smí obsahovat (pro jednoduchost) maximálně 2 záznamy.

Uložte záznamy (Opava, Lipová 5), (Zlín, Úzká 8), (Lipník, Anglická 142/a)

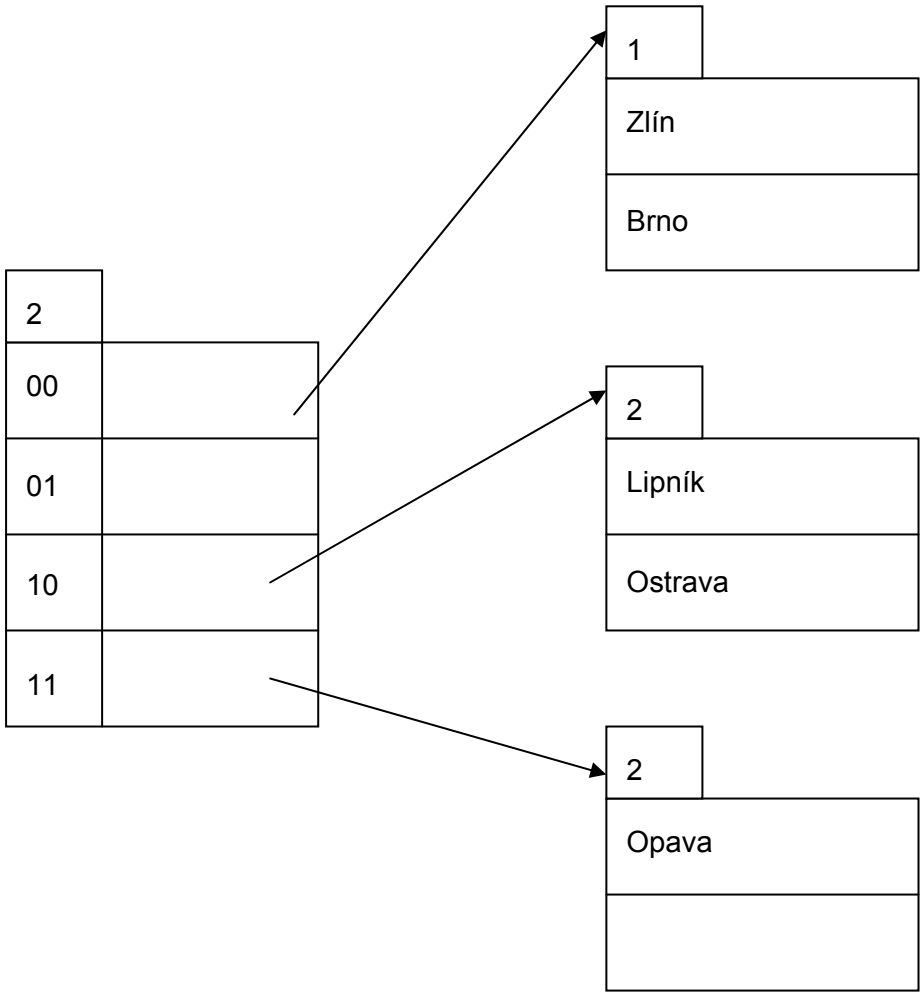


Zatím jsme vystačili s jedním bitem offsetu. Záznamy se ukládaly do dvou různých kapes. V jedné kapse jsou záznamy, jejichž adresa začíná **0**, v druhé kapse záznamy, jejichž adresa začíná **1**(tato kapsa je plná).

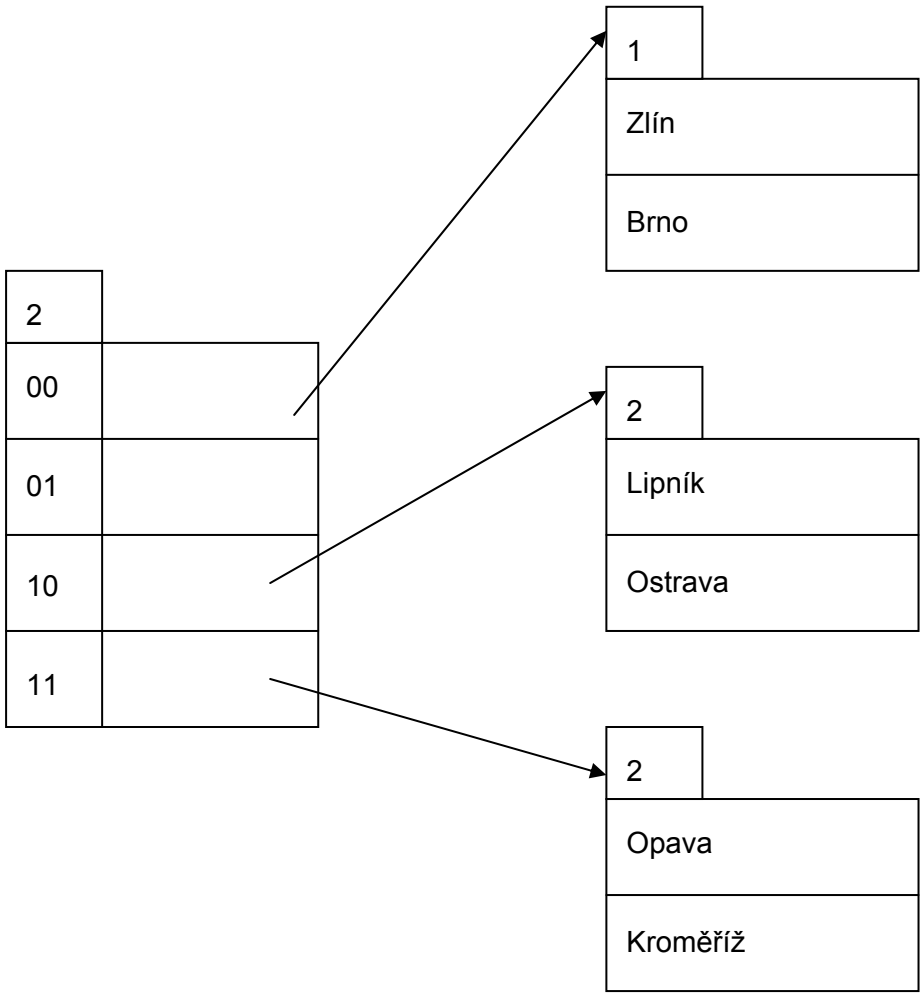
Uložte záznam (Ostrava, Nádražní 128), přičemž kapsa, kde se má záznam ukládat je zaplněna. Je nutno alokovat další kapsu (logický blok) a přeorganizovat záznamy včetně úpravy adresáře.



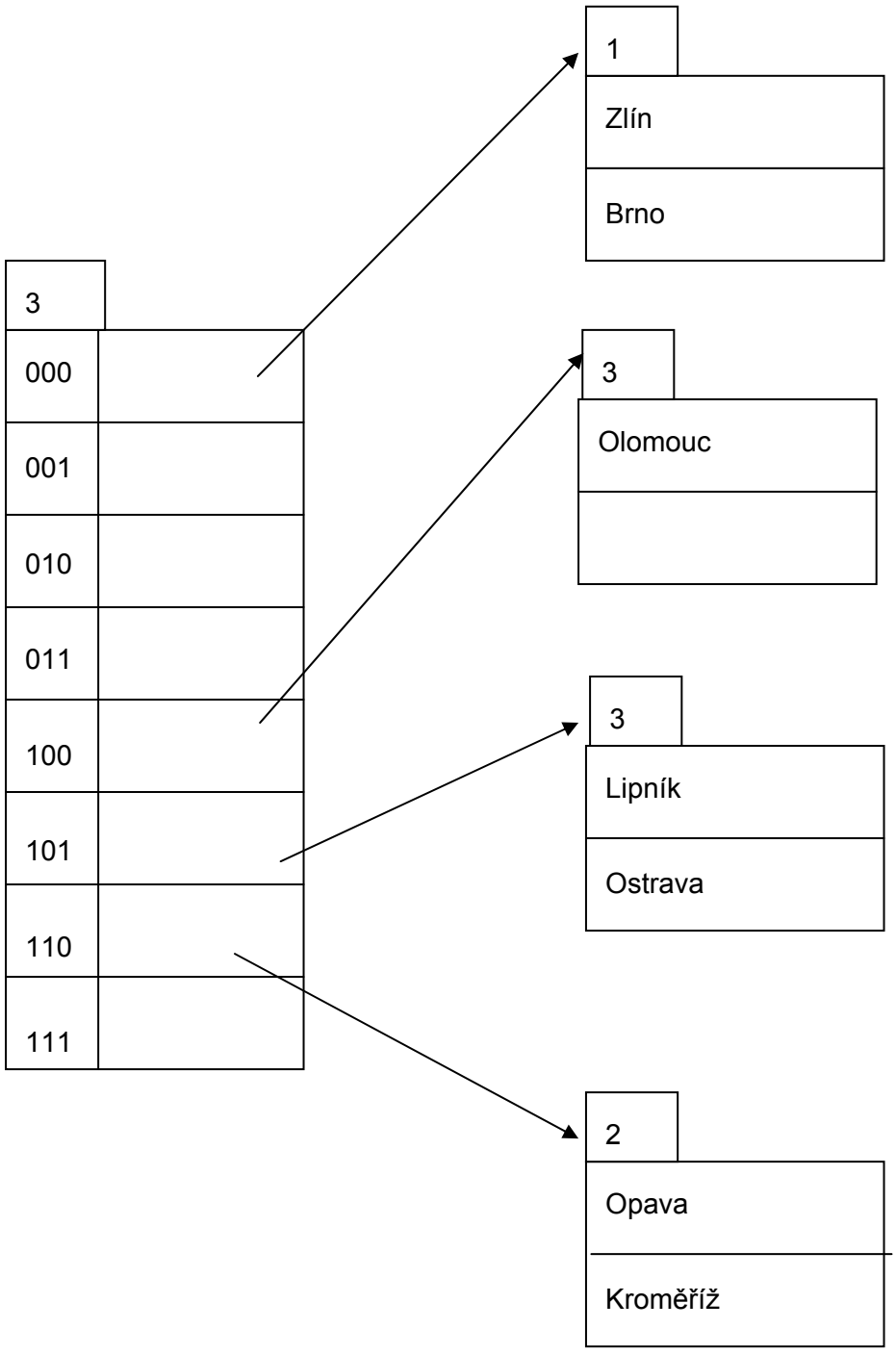
Uložte záznam (Brno, Dlouhá 25)



Uložte záznam (Kroměříž, Jánská 100)

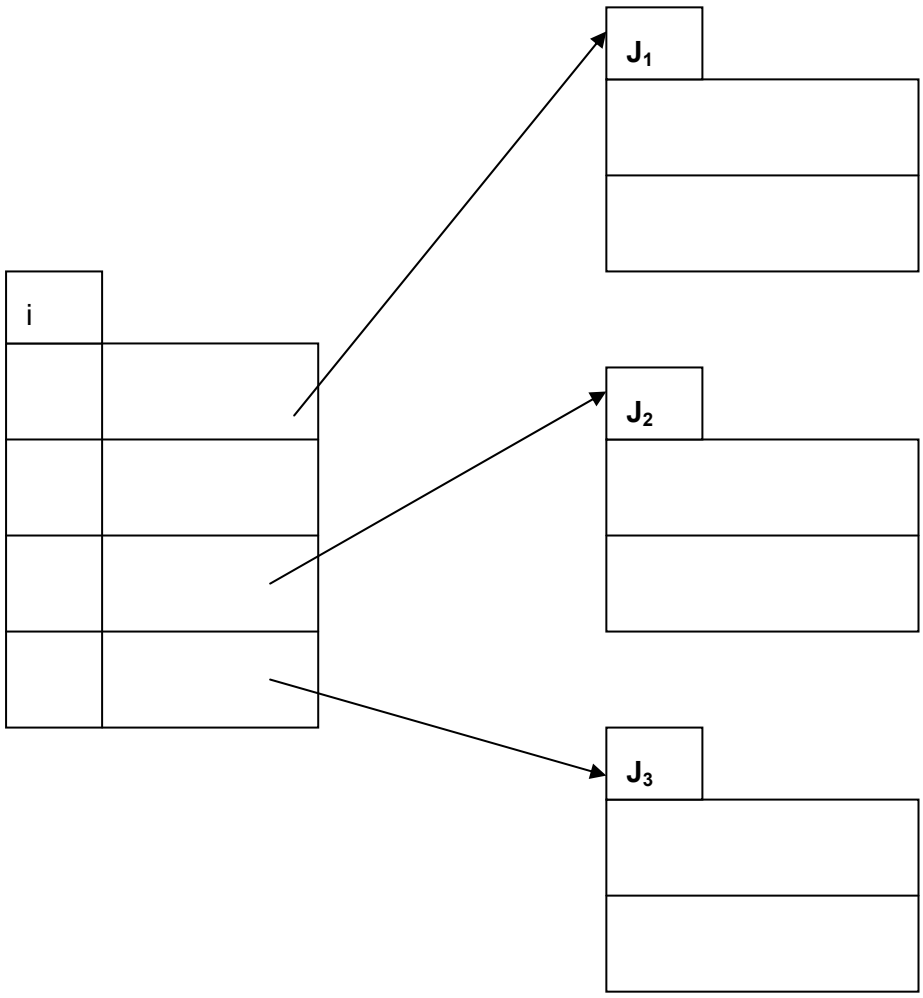


Uložte záznam (Olomouc, Francouzská 1725). Kapsa, do které je nutno umístit záznam, je zaplněna. Proto dojde ke štěpení kapsy a zároveň se zvětší offset.



OBECNÉ SCHÉMA ROZŠIŘITELNÉHO HAŠOVÁNÍ

Adresář Kapsy primárního souboru



i Počet bitů offsetu (adresy, která je umístěna v adresáři).

$\mathbf{j_i}$ Počet bitů, v kolika se shodují adresy záznamů, uložených v i-té kapse.

Je-li $\mathbf{j_i = 1}$, v dané kapse je uložena polovina všech záznamů souboru.

S nárůstem záznamů se zvětšuje zaplněnost kapes. Je-li kapsa zaplněna, musí se alokovat nová kapsa (logický blok). Zároveň se musí zvětšit počet bitů v adresáři o **1** a dynamicky se musí přesunout (reorganizovat) příslušné záznamy v primárním souboru.

Shrnutí

Dynamické struktury jsou v databázové technologii velmi důležité, protože zajišťují konstantní počet přístupů na vnější paměťová média a zajišťují pružnou změnu uložení záznamů v souborech v závislosti na aktualizaci databáze. V současných systémech řízení báze dat převazují dynamické organizace nad organizacemi statickými. K velmi efektivním metodám patří tzv. dynamické hašování. Dynamika spočívá v postupném aktualizaci adresáře a v závislosti na změně adresáře se i mění umístění záznamů souboru do logických bloků. Reorganizace tedy probíhá paralelně s aktualizačními operacemi (delete, insert, update).

Kontrolní otázky

Jaký je rozdíl mezi statickou a dynamickou organizací dat?

Popište algoritmus zápisu záznamů do souboru s využitím rozšiřitelného hašování.

V čem spočívá dynamické hašování?

4.4 REDUKCE DAT

Cíl:

Základním cílem kapitoly je seznámit čtenáře s příčinami redundance dat a základními principy a metodami redukce dat. Po prostudování této kapitoly bude čtenář umět rozlišovat mezi kompresí a kompakcí, bude umět vypočítat délku kódového slova při metodách komprese s pevnou délkou kódového slova a bude umět stanovit kódová slova s využitím Huffmanova kódování a kódování pomocí adaptivního slovníku.

Klíčová slova:

Redundance, redukce, kód, kódové slovo, komprese, kompakce, adaptivní slovník, regál.

4.4.1 Úvod

Díváme-li se na soubory jako na texty, pak jsou tyto texty redundantní. Redundance dat je situace, která v žádném případě nepřispívá ke kvalitě databáze. Redundance ovšem může být způsobena několika příčinami. V předcházejících kapitolách jsem se věnovali redundanci vyplývající z nekorektně navrženého schématu databáze. Pokud se v rámci jedné relace vyskytují atributy, které jsou na sobě závislé a přitom nejsou ve vztahu Primární klíč - neklíčový atribut. Takovému typu redundance je třeba se vyvarovat a provést příslušnou normalizaci relačních schémat. Redundance však může mít i jiné příčiny. Těmito příčinami a jejich řešením se budeme zabývat v kapitole Redukce dat.

4.4.2 Příčiny redundance

Redundance dat také vyplývá z toho, že:

- Některé fráze nebo slova v textu se opakují.
- Existuje závislost mezi po sobě následujícími znaky v textu.

Existuje řada technik, jak redukovat množství ukládaných dat, aniž by došlo ke ztrátě informace. Uvědomme si, že podstatnou vlastností databáze je smysluplné poskytování pravdivých, relevantních informací. Kolik dat k tomu potřebujeme mít explicitně uloženo v databázi, to je již druhá vět. Je jasné, že méně dat, které poskytují totéž kvantum informací, je výhodné.

Výhody redukce

- Snížení velikosti vnějších médií.
- Zkrácení času přístupu k datům.
- Zkrácení času přenosu dat.

Nevýhody redukce

- Přídavná složitost odpovídajících algoritmů.
- Manipulace s proměnnou délkou kódu.
- Manipulace s jednotlivými bity.

4.4.3 Základní pojmy redukce dat

Metody redukce jsou z větší části založeny na kódování.

KÓDOVÉ SLOVO je symbol či posloupnost symbolů, kterými kódujeme zdrojový objekt (jednotku). **KÓD** je množina všech kódových slov. Důležitá vlastnost kódu je jednoznačná dekódovatelnost. Kódy s pevnou délkou jsou snadno dekódovatelné. U kódů s proměnnou délkou se posuzuje bezprostřední rozhodnutelnost.

Kód je bezprostředně rozhodnutelný, jestliže poznáme konec kódového slova bezprostředně po příjmu jeho posledního znaku. Kódy jsou bezprostředně rozhodnutelné, když mají tzv. prefixovou vlastnost. **Prefixová vlastnost** znamená, že žádné kódové slovo kódu není předponou nějakého jiného kódového slova kódu.

Pro měření vhodnosti kompresní techniky slouží kompresní poměr (v %). Kompresní poměr je relativní množství paměti ušetřené po kódování.

D...velikost původního zdrojového řetězce

D´... velikost zakódovaného řetězce

Kompresní poměr **K = (D - D´) / D**

4.4.4 Metody redukce

KOMPRESE je takový způsob redukce dat, kdy vždy existuje možnost dekomprese, při které se neztrácí informace. U **KOMPAKCE** je zachována pouze jistá nutná informace, neexistuje inverzní proces dekódování. Např. zkracování klíčů v souborech indexů, kdy pro rozlišení stačí pouze některá místa klíče.

4.4.5 Kódy s pevnou délkou

Chceme-li zakódovat (**m** výskytu) **n** jednotek, lze to udělat kódovými slovy délky **l** a platí: $\mathbf{l = \log_2 n}$

n ... počet jednotek

Příklad:

Uvažujeme text složený z 10 slov (n = 10), THE, OF, AND, TO, A, IN, THAT, IS, IT, ON

Původní délka textu při délce pro 1 znak = 1 byte, byla 23 bytů, tj. 23x8=184 bitů.

Délka textu po zakódování s pevnou délkou kódového slova:

$l = \log_2 10$

Délka kódového slova (v bitech) je nejbližší celé číslo, vetší než l a to je 4 bity. Celý text je tedy zakódován do délky 4x10=40 bitů.

Známe-li pravděpodobnosti výskytu jednotlivých slov, pak kompresní poměr vypočteme podle vzorce:

$$1 - \frac{\log_2 n}{\sum l_i p_i}$$

$\mathbf{l_i}$ ..... délky slov

$\mathbf{p_i}$ .... pravděpodobnosti výskytu

4.4.6 Huffmanovo kódování

Používá kódy proměnné délky, bylo vyřešeno v r. 1952. Huffmanovo kódování je založeno neorientovaných binárních stromech.

Vstup: **n** jednotek ke kódování a pravděpodobností $\mathbf{p_i}$ jejich výskytu.

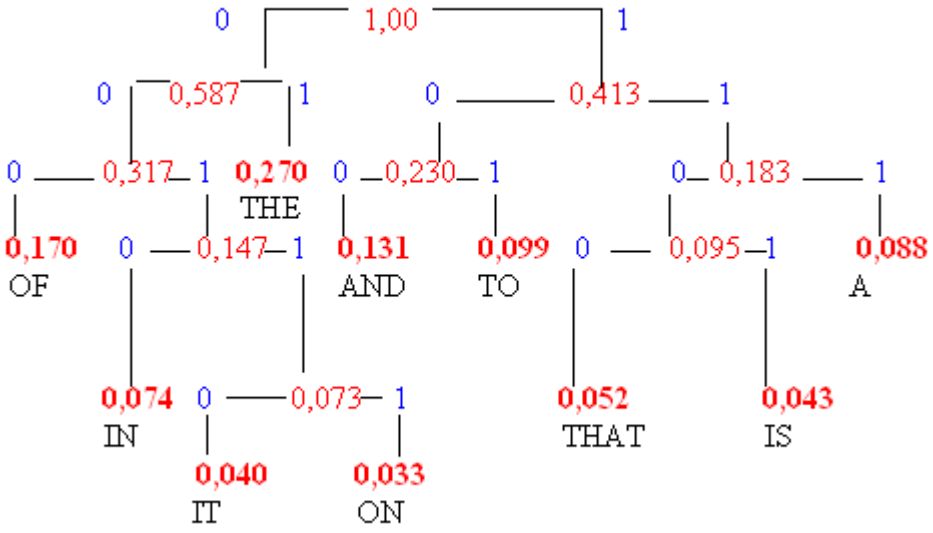
Pro každou jednotku **i** vytvoříme list $\mathbf{o(p_i)}$ binárního stromu, tj. uzel ohodnocený $\mathbf{p_i}$.

Z pravděpodobností se vyberou dvě nejmenší nenulové $\mathbf{p_r}$ a $\mathbf{p_s}$ a vypočte se $\mathbf{q = p_r + p_s}$. Vytvoří se uzel **o** ohodnocený **q** a hrany se ohodnotí **0** resp. **1**.

Tímto způsobem se sestrojí binární strom.

Příklad

Jsou dána slova: THE, OF, AND, TO, A, IN, THAT, IS, IT, ON, která se vyskytují s pravděpodobností (po řadě): 0,270, 0,170, 0,131, 0,099, 0,088, 0,074, 0,052, 0,043, 0,040, 0,033. Sestrojte binární strom a naznačte sestavení kódových slov pro jednotlivá slova textu.



Např. Slovo **of** se zakóduje 00

is 1101

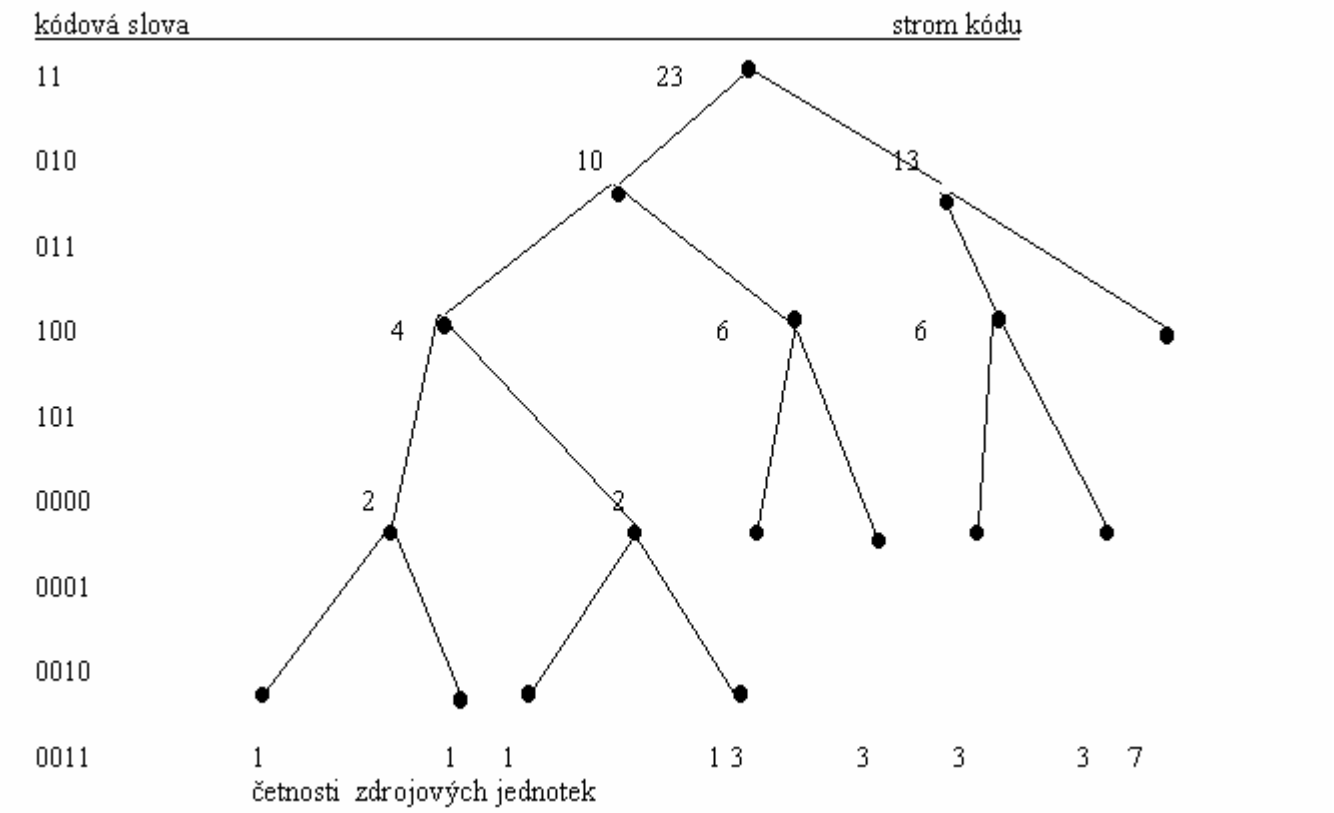
Kód se čte od kořene k listu. Huffmanovo kódování vyžaduj známé a neměnící se $\mathbf{p_i}$.

Příklad

Nechť (setříděné) četnosti slov 9ti znakové abecedy v textu tvoří posloupnost (1,1,1,1,3,3,3,3,7). Sestrojte binární strom a

k němu Huffmanův kód.

Řešení:



4.4.7 Komprese dat adaptivním slovníkem

Techniku vynalezl Jakobsson. Jedná se o techniku komprese obecných textových řetězců pomocí slovníku, který vzniká v paměti souběžně s kódovaným textem. Slovník je po zakódování z paměti odstraněn a při dekódování je opět paralelně generován.

Experimentálně byl zjištěn kompresní poměr 50 %. Podstatou metod je konstrukce datové struktury - lesu. Ke každému znaku použité n-prvkové abecedy je přirazen jeden odpovídající kořen stromu. Stromy jsou n-ární a mají hloubku **h** (parametr metody). Pro daný řetězec **S** se definuje regál slovníku **Fh(S)**, kde každá cesta z kořene odpovídá podřetězci z **S** a naopak. Každý podřetězec délky menší nebo rovné **h** lze v lese nalézt jako cestu z nějakého kořene.

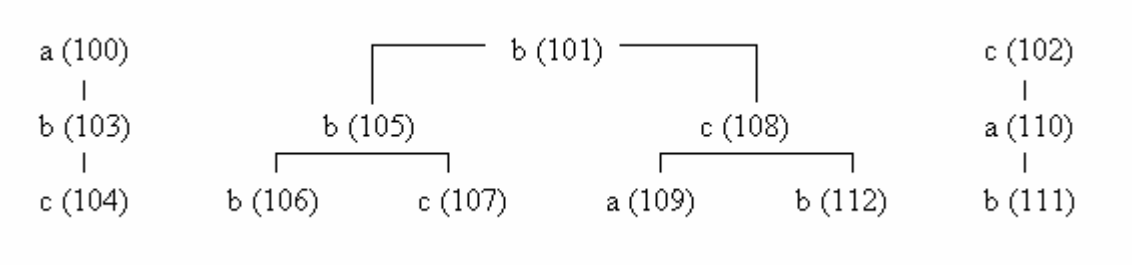
Uzly jsou ohodnoceny znaky abecedy. Kódování řetězce **S** spočívá v náhradě podřetězců adresami, které odpovídají adrese posledního znaku podřetězce, který ještě existuje ve slovníku. Metoda je jednoprůchodová, tj. text se čte sekvenčně znak po znaku pouze jednou.

Příklad

Nechť h = 3, abeceda = (a, b, c)

S = abbbcabbbbbcb

Regál slovníku má tvar:



 **Pojmy k zapamatování:**

Statická struktura

Dynamická struktura

B-strom

Neredundantní B-strom

Dynamické hašování

4.5 ŘEŠENÉ PŘÍKLADY B-STROMŮ

Cíl:

Cílem této kapitoly je na řešených příkladech demonstrovat proces postupného vkládání záznamů do B-stromu a s tím souvisejícího štěpení uzlů. Čtenář by se měl zamýšlet hlavně nad tím, jak:

- *Vypadá struktura B-stromu.*
- *Lze provádět odkazy na další uzly B-stromu.*
- *Lze provádět odkazy na záznamy primárního souboru.*
- *Dochází ke štěpení uzlů.*
- *Zajistit vyváženost B-stromu.*

Klíčová slova:

B-strom, Insert, Delete, štěpení uzlu, alokace uzlu.

 **Příklad:**

Na primární soubor, obsahující 6 záznamů vkládaných v pořadí Z0, Z1, …

Matka	Karla	Poklad	Babička	Pustina	F.L.Věk
Z0	Z1	Z2	Z3	Z4	Z5

vytvořte B-strom, jestliže velikost uzlu = 3. V tabulce jsou uvedeny pouze hodnoty vyhledávacího klíče Příjmení.

Řešení:

1. Nejdříve si setřídíme hodnoty vyhledávacího klíče, tj. Příjmení podle abecedy

Babička	F.L.Věk	Karla	Matka	Poklad	Pustina
Z3	Z5	Z1	Z0	Z2	Z4

2. Dále alokujeme potřebný počet uzlů tak, aby žádný z nich nebyl zaplněn na 100%. Tím vytvoříme listovou úroveň B-stromu. Každému alokovanému uzlu přidělíme adresu.

Adr.0

Babička	F.L.Věk	-
Z3	Z5	-

Adr.1

Karla	Matka	-
Z1	Z0	-

Adr.2

Poklad	Pustina	-
Z2	Z4	-

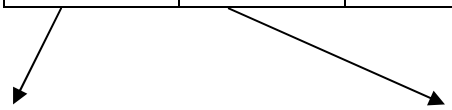
3. Tuto listovou úroveň postupně propojujeme do konstrukce stromu. Do vyšší úrovně produkujeme vždy nejvyšší hodnotu vyhledávacího klíče uzlu nižší úrovně. Každému alokovanému uzlu přidělíme adresu. Všimněte si, že odkaz do primárního souboru se vyskytuje pouze v listové úrovni. Ve všech ostatních úrovních se vyskytují odkazy do nižších úrovní B-stromu. Toto platí pouze pro redundantní B-strom.

Adr.3

Z.L.Věk	Matka	-
0	1	-

Adr.4

Pustina	-	-
2	-	-



Adr.0

Babička	F.L.Věk	-
Z3	Z5	-

Adr.1

Karla	Matka	-
Z1	Z0	-

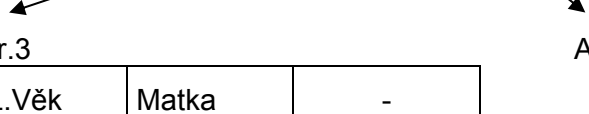
Adr.2

Poklad	Pustina	-
Z2	Z4	-

4. Konstrukce B-stromu končí alokací kořenového uzlu.

Adr.5

Matka	Pustina	-
3	4	-

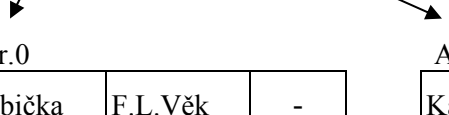


Adr.3

Z.L.Věk	Matka	-
0	1	-

Adr.4

Pustina	-	-
2	-	-



Adr.0

Babička	F.L.Věk	-
Z3	Z5	-

Adr.1

Karla	Matka	-
Z1	Z0	-

Adr.2

Poklad	Pustina	-
Z2	Z4	-

Příklad:

Na primární soubor, obsahující následující záznamy vkládané v pořadí Z0, Z1, …

Aš	Praha	Opava	Krnov	Zlín	Most	Kladno	Brno
Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7

vytvořte B-strom, jestliže velikost uzlu = 3. V tabulce jsou uvedeny pouze hodnoty vyhledávacího klíče Město.

Řešení:

Adr.6

Krnov	Zlín	-
4	5	-

Adr.4

Brno	Krnov	-
0	1	-

Adr.5

Opava	Zlín	-
2	3	-

Adr.0

Aš	Brno	-
Z1	Z8	-

Adr.1

Kladno	Krnov	-
Z7	Z4	-

Adr.2

Most	Opava	-
Z6	Z3	-

Adr.3

Praha	Zlín	-
Z2	Z5	-

Úkoly k zamyšlení:

Pokuste se navrhnout primární sekvenční soubor a zkonstruujte na něj B-strom. Dále se pokuste primární soubor aktualizovat přidáváním a mazáním záznamů a ukažte, jaký vliv tyto operace mají na B-strom.

Průvodce studiem:

Pokud jste pochopili jak teoretickou část modulu o B-stromech, tak všechny uvedené příklady, dá se předpokládat, že nebudete mít potíže s vypracováním korespondenčního úkolu, který obsahuje problematiku B-stromu. Pokud Vám je ještě něco nejasné, prostudujte

příslušné kapitoly ještě jednou, ev. kontaktujte tutora. Určitě však zasloužíte pochvalu za snahu. Věřte, že Vás chápu a dokážu se vžít do Vaší situace. Studium distanční formou určitě není nic snadného.

POUŽITÁ LITERATURA

1. Date, C.J. : An Introduction to Database Systems. Addison - Wesley Publishing Company, USA, 1990
2. Havlát, T., Benešovský, M. : Úvod do databázových systémů. Skripta přírodovědecké, fakulty UJEP, Brno, 1987
3. Korth, H.F., Silberschatz, A. : Database system concepts. McGraw-Hill, Inc., 1991
4. Melichar, B.: Textové informační systémy, ČVUT Praha, 1996
5. Pokorný, J. : Dotazovací jazyky v databázových syst,mech. Skripta MFF UK v Praze, SPN Praha, 1986
6. Pokorný, J. : Databázové technologie. Sborník semináře MOP, MFF UK Praha, 1990
7. Pokorný, J. : Počítačové, databáze. Kancelářské stroje,
8. oborov, informační středisko, Praha, 1991
9. Pokorný, J. : Databázové, systémy a jejich použití v informačních systémech. Academia, Praha, 1992
10. Pokorný, J., Halaška, I. : Databázové, systémy. Vybrané, kapitoly a cvičení. Skripta el. fakulty ČVUT, Praha, 1992
11. Pokorný, J., Benešovský, M., Krejčí F.: Logika a dotazovací jazyky. Sborník celost. Sem. SOFSEM 78, Magura, 1978
12. Pokorný, J.: Dotazovací jazyky, Science, 1994
13. Pokorný, J.: Základy implementace souborů a databází, Karolinum Praha, 1997
14. Telnarová, Z., Lukasová, A., Matula, P.: Úvod do databázové technologie. Skripta OU Ostrava, 1999
15. Telnarová, Z.: Databázové systémy a jejich využití v IS. Skripta OU, Ostrava, 2001
16. Ullman, D.J.: Principles of database and knowledge-base systems, Volume I, Computer Sc. Press, 1988
17. Ullman, D.J.: Principles of database and knowledge-base systems, Volume II, Computer Sc. Press, 1989