Česká zemědělská univerzita v Praze/

Provozně ekonomická fakulta

Katedra informačních technologií (PEF)

Bakalářská práce

Návrh relačního modelu databáze pro provozní deník

Michal Škvařil

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Michal Škvařil

Informatika

Název práce

Návrh relačního modelu databáze pro provozní deník

Název anglicky

Design of a relational database model for an operational log

Cíle práce

Hlavním cílem práce je vypracování relačního modelu pro aplikaci Provozního deníku vybrané firmy za pomoci užití relačního databázového systému Oracle dle specifikací zadavatele.
Mezi vedlejší cíle práce patří zmapování historie databází a jazyka SQL. Dalšími vedlejším cílem je popis relačního databázového modelu. Posledním vedlejším cílem je komparace starého a nového řešení provozního deníku za účelem zjištění přínosu nového systému.

Metodika

Teoretická část bakalářské práce je založena na studiu a analýze odborných informačních zdrojů a rovněž na praktických zkušenostech s databázovým systémem Oracle. V rámci praktické části bude za pomoci metody komparace srovnán nově vypracovaný provozní deník s předchozím způsobem zadávání, tj. zadáváním do evidence v Excelu. Na základě komparace budou vyhodnoceny přínosy nového systému.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

Relační databáze, Oracle, SQL, provozní deník

Doporučené zdroje informací

CONNOLLY, Thomas M., Carolyn E. BEGG a Jennifer WIDOM. Database systems: a practical approach to design, implementation, and management. 5th ed. London: Addison-Wesley, c2010. ISBN 978-0-32-152306-8.

CHLAPEK, D. – ŘEPA, V. – STANOVSKÁ, I. – VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE. FAKULTA INFORMATIKY A STATISTIKY. *Analýza a návrh informačních systémů.* Praha: Oeconomica, 2011. ISBN 978-80-245-1782-7.

ŠILEROVÁ, E. *Informační systémy v podnikové praxi.* ISBN 978-80-87994-78-8.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Mgr. Vladimír Očenášek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 15. 8. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 15. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh relačního modelu databáze pro provozní deník" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.03.2022 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Mgr. Vladimíru Očenáškovi, Ph.D. za vedení, pomoc a vstřícnost při psaní této bakalářské práce. Dále bych chtěl poděkovat firmě IND, s.r.o. za možnost zpracování vlastní práce pod jejich vedením, jejich odborných rad a vstřícného přístupu.

Návrh relačního modelu databáze pro provozní deník

Abstrakt

Bakalářská práce se zabývá tematikou návrhu relačního modelu databáze, za účelem vypracování provozního deníku elektrárny. Teoretická část se zaměřuje na relační databáze, jejich vývoj, popis terminologie a zákonitostí relačních databází.

V rámci praktické části se autor věnuje zabývá problematice sestavení relačního modelu pro potřeby pracovníků dané elektrárny a porovnání nového řešení provozního deníku se starým.

Klíčová slova: Databáze, Provozní deník, Relační databáze, Oracle

Design of a relational database model for an operational log

Abstract

The bachelor's thesis deals with the design of the relational model of the database, in order to develop the operational diary of the power plant. The theoretical part focuses on relational databases, their development, description of terminology and regularities of relational databases.

In the practical part, the author deals with the problem of creating a relational model for the needs of the workers of the given power plant and comparing the new solution of the operational diary with the old one.

Keywords: Database, Operational log, Reational database

Obsah

1 Úvod

Provozní deníky jsou nezbytnou součástí provozu elektrárny. Do deníků jsou z jednotlivých směn přepisována různá data chodu elektrárny, díky kterým mohou pracovníci zajistit její bezpečný chod, předávat informace o chodu následujícím směnám a předcházet tak různým problémům. Provozní deníky byly dlouhou dobu vedeny papírovou formou, později se přesunuly k Excelovým tabulkám. Takovýto přístup má mnoho nevýhod. Mezi ně patří omezení programu Excel, nemožnost současného přístupu více zaměstnanců k datům a bezpečnost uložených dat.

Tato bakalářská práce v teoretické části přibližuje základní principy fungování databází. V praktické části pak ukazuje návrh relačního modelu databáze pomocí systému Oracle a porovnává nové řešení s řešením stávajícím.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je vypracování relačního modelu pro aplikaci Provozního deníku vybrané firmy za pomoci užití relačního databázového systému Oracle dle specifikací zadavatele.

Mezi vedlejší cíle práce patří představení historie databází a jazyka SQL. Dalšími vedlejším cílem je představení relačního databázového modelu. Posledním vedlejším cílem je komparace starého a nového řešení provozního deníku za účelem zjištění přínosu nového systému.

2.2 Metodika

Teoretická část bakalářské práce je založena na studiu a analýze odborných informačních zdrojů a rovněž na praktických zkušenostech s databázovým systémem Oracle. V rámci praktické části bude za pomoci metody komparace srovnán nově vypracovaný provozní deník s předchozím způsobem zadávání, tj. zadáváním do evidence v Excelu. Na základě komparace budou vyhodnoceny přínosy nového systému.

3 Teoretická východiska

3.1 **Databáze**

„Databáze je nyní tak nedílnou součástí našeho každodenního života, že si často ani neuvědomujeme, že nějakou používáme“ (Connolly, 2010, s.4). V základu databáze není nic víc než kolekce informací, které jsou uloženy na dlouhou dobu, nejčastěji v řádu několika let. Velká část organizací a firem by bez databází nebyla schopna fungovat. Při každé návštěvě webové stránky např. Google.com, Seznam.cz, Amazon.com nebo jakékoliv z tisíce jiných stránek se vždy v jejich pozadí nachází databáze, která zpracovává požadavky. Zdaleka ne všechny databáze jsou používány webovými stránkami. Databáze jsou využívány ve všech odvětvích, např. v bankovních systémech, kde jsou uloženy účty uživatelů a jejich účetní zůstatky, v leteckých společnostech, které stejně jako banky musí být schopny zpracovat obrovské množství malých požadavků a zaručit stabilitu a správnost dat, bez jejich ztráty (Garcia-Molina et al. 2014, s1-10).

Databáze se skládá ze Systému řízení báze dat, o kterém se píše v kapitole 3.2, a kolekce dat. Databáze je organizovaná kolekce dat, do které připojujeme, nebo získáváme, data pomocí dotazů. Mohou být hostovány na souborovém sytému, na počítačových clusterech nebo na cloudovém úložišti. V designu databází se používají techniky na modelování dat a efektivní datové struktury pro jejich uchovávání (Connolly 2010, s.4-6).

3.1.1 **Historie Databází**

První databáze vznikaly ze souborových systémů, které byly schopny uchovávat velké množství souborů. Problémem souborového systému byla jeho neschopnost zabránit ztrátě dat a umožnit efektivní přístup k datům v podobě požadavků. Další nedostatek představovalo schéma dat, které bylo limitováno pouze na složky a soubory v systému. Ačkoliv bylo možné mít souběžný přístup k souborům, systém nebyl schopný zvládnout úpravy dvou a více uživatelů (projevily se změny pouze jednoho uživatele, ostatní změny byly ztraceny (Garcia-Molina et al 2014, s.1-10).

Vyjmenované problémy vedly ke vzniku prvních komerčních počítačových databází, které v roce 1960 představil Charles Bachman. První představená databáze byla prezentována jako integrovaný datový systém (zkráceně IDS). Druhá databáze byla představena společností IBM, známá jako systém správy informací. Obě zmíněné databáze jsou předchůdci navigační databáze. Před rokem 1970 byly nejčastěji využívanými databázemi databáze hiearchického typu a síťové databáze, které byly vyvinuty na začátku 60. let 20. století a byly využívány až do roku 1990. Některé z nich jsou vyžívány dodnes. Staly se úspěšnými

komerčními produkty pro nezávislé firmy i velké korporátní firmy jako IBM. Úspěch byl rychle zastíněn už v roce 1970, kdy E.F.Codd představil databázi relačního typu, která je popsána v kapitole 3.5 (Wade, & Chamberlin 2012, s.38-48).

3.1.2 **Systém řízení báze dat**

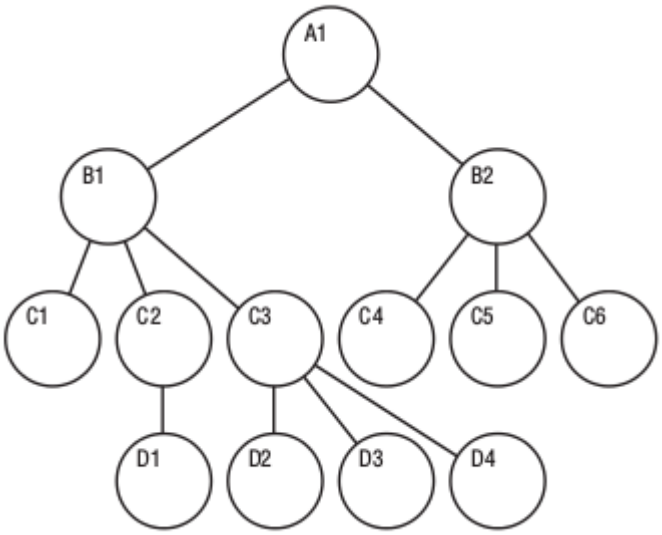
Sytém řízení báze dat (zkráceně SŘBD nebo anglicky DBMS) tvoří rozhraní mezi aplikačními programy a uloženými daty. Společně s databází utváří celek, tzv. databázový systém. Proto, aby mohl být program označený jako DMBS, musí efektivně zpracovávat velké množství dat a musí být schopen řídit a provádět dotazy. DMBS pravidelně synchronizuje data a zajišťuje, že jakákoliv změna dat bude v databázi univerzálně aktualizována (Garcia-Molina et al.2014, s. 5-20)
Z počátku byly systémy na řízení báze dat kombinace obrovských počítačů a velmi drahých softwarů na vývoj. Dnes jsou SŘBD součásti skoro každé aplikace a je možné je využívat i na osobním počítači, jelikož disk dokáže uložit až terabajty paměti (Garcia-Molina et al. 2014, s.5-15).

3.1.3 **Databázové typy**

Od počátku prvních databází až po současnost bylo implementováno několik modelů pro ukládání a správu dat. Z hlediska ukládání dat a vazeb můžeme rozdělit databáze do těchto základních typů:

3.1.3.1 Hierarchické databáze

Hierarchická databáze se využívá ve chvíli, kdy má databáze přesně danou hierarchii, například zaměstnanci ve firmě. Databázová struktura tohoto modelu je jako obrácený strom, ve kterém se nachází jeden kořen, který vede na další větve stromu. Používá se pojmenování vztah dítě – rodič. Každé dítě může mít pouze jednoho rodiče, ale rodič může mít neomezený počet dětí (Jindal & Bali 2012, s.1). Jako výhoda přístupu se uvádí jednoduchost datového modelu. Ten dává uživateli na výběr několik málo příkazů a díky omezené možnosti vazeb lze vytvořit snadnější implementaci určitého modelu. Mezi nevýhody lze zařadit nemožnost N:M vazby a obtížnější vkládání a mazání prvků (Groff & Weinberg 1999, s. 39-40).

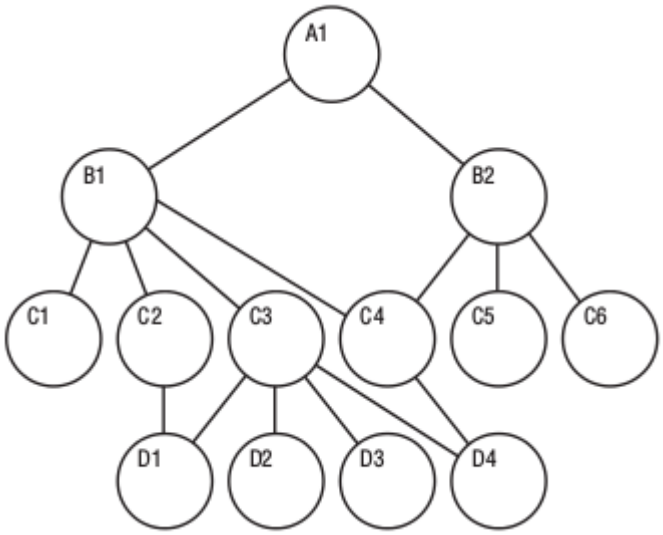


Obrázek 1 Hierarchický datový model, zdroj: mariadb.com

3.1.3.2 Síťové databáze

Síťový databázový model byl navrhnut a vytvořen jako řešení nedostatků hierarchického databázového modelu. Při zachování analogie dítě-rodič, může být v tomto modelu dítě spojeno s více rodiči, což je funkce, která nebyla hierarchickým datovým modelem podporována. Nadřazené uzly se nazývají vlastníci, podřízené se nazývají členové (Groff & Weinberg 1999 s.40, s.41).

Jako nevýhoda síťového modelu se udává větší složitost, než tomu je u hierarchického modelu. Je tedy náročnější na manipulaci i údržbu. Další nevýhodou je, že struktura síťového modelu je poměrně komplikovaná, a proto jí programátor musí dobře rozumět, aby ji mohl implementovat nebo upravit (Groff & Weinberg 1999, s.41).



Obrázek 2 Síťový datový model, zdroj: mariadb.com

3.1.3.3 Relační databáze

Relační databáze je kolekce dat, která jsou organizována pomocí předdefinovaných vztahů a uložena v jedné nebo více tabulkách. Tabulky jsou rozděleny na sloupce a řádky, tzv. entity a záznamy. Toto rozdělení zjednodušuje uživateli schopnost určit, jaký mezi sebou mají datové struktury vztah. Vztahy jsou logická propojení mezi různými tabulkami. V současné době se jedná o nejpoužívanější řešení databázového modelu. Pro určení vztahů jednotlivých entit mezi sebou se používají klíče (Connolly 2010, s. 42-43). Mezi nevýhody tohoto typu databáze se obvykle řadí nutnost velikosti a komplexnosti pro implementování problému (Butuner 2012, s.1).

3.1.3.4 Objektové databáze

Objektově orientované databáze jsou navrženy a sestaveny podle objektově orientovaného paradigmatu, tedy že svá data ukládá formou objektů. Tato funkce podporuje dědičnost, a tedy znuvupoužitelnost podobnou objektově orientovanému programování (Jatana et al., 2012, s.2-4). Tento typ datového modelu pomáhá při řešení složitých datových struktur, například multimediálního obsahu přirozenějším způsobem, a poskytuje bezproblémový přechod od návrhu ke koncepci. Podle manifestu objektově orientovaného databázového systému musí objektově orientovaný databázový systém (OODBMS) splňovat dvě hlavní kritéria. První kritérium říká, že by se mělo jednat o systém pro správu databází (DBMS). Dle prvního kritéria by měl OODBMS poskytovat pět funkcí, které jsou nezbytné pro jakýkoli databázový systém – perzistenci, souběžnost, obnovu dat, správu sekundárního

uložiště a možnost dotazování ad hoc. Druhé kritérium nám říká, že by databázový systém měl podporovat všechny požadované vlastnosti objektově orientovaného systému. Nespornou výhodou tohoto modelu databáze oproti databázím relačním je snazší přenos z analýzy do implementace (Alzahrani 2016, s.2-4).

3.1.3.5 Objektově relační databáze

Objektově relační databáze je typ databáze, která kombinuje objektově orientovaný databázový model a relační databázový model. Podporuje tedy objekty, třídy, dědičnost aj. stejně jako objektově orientované modely. Má podporu pro datové typy, tabulkové struktury aj., stejně jako je tomu u relačního datového modelu (Piattini et al. 2001, s.4).

3.1.4 **Databázová integrita**

Pro zachování konzistence a správnost uložených dat, relační systém řízení báze dat obvykle ukládá jedno nebo více omezení integrity dat. Tato omezení limitují hodnoty dat, které lze vložit do databáze nebo které byly vytvořeny aktualizací databáze. V relačních databázích se běžně vyskytuje několik různých typů omezení integrity dat (Groff & Weinberg 1999, 211).

3.1.4.1 Entitní integrita

Entitní integrita závisí na vytváření primárních klíčů, které jsou unikátní částí dat. Brání tvorbě duplikátních vstupů do databáze. Primární klíč tabulky musí obsahovat jedinečnou nenulovou hodnotu pro každý řádek. Norma ISO podporuje entitní integritu pomocí klauzule PRIMARY KEY v příkazech CREATE a ALTER TABLE (Connolly 2010, 166).

3.1.4.2 Doménová integrita

Doménová integrita pomocí procesů zajišťuje, že se v každému sloupci databáze nachází sada přijatelných hodnot v rámci domény. Například, že do sloupce s datovým typem Boolean (datový typ, který obsahuje pouze True nebo False) není vloženo desetinné číslo (Connolly 2010, 164-165).

3.1.4.3 Referenční integrita

Referenční identita zajišťuje jednotné ukládání a používání dat, pravidla o tom, jak se cizí klíče používají a také kontroluje změny v databázi např. přidávání a smazání dat. Říká nám,

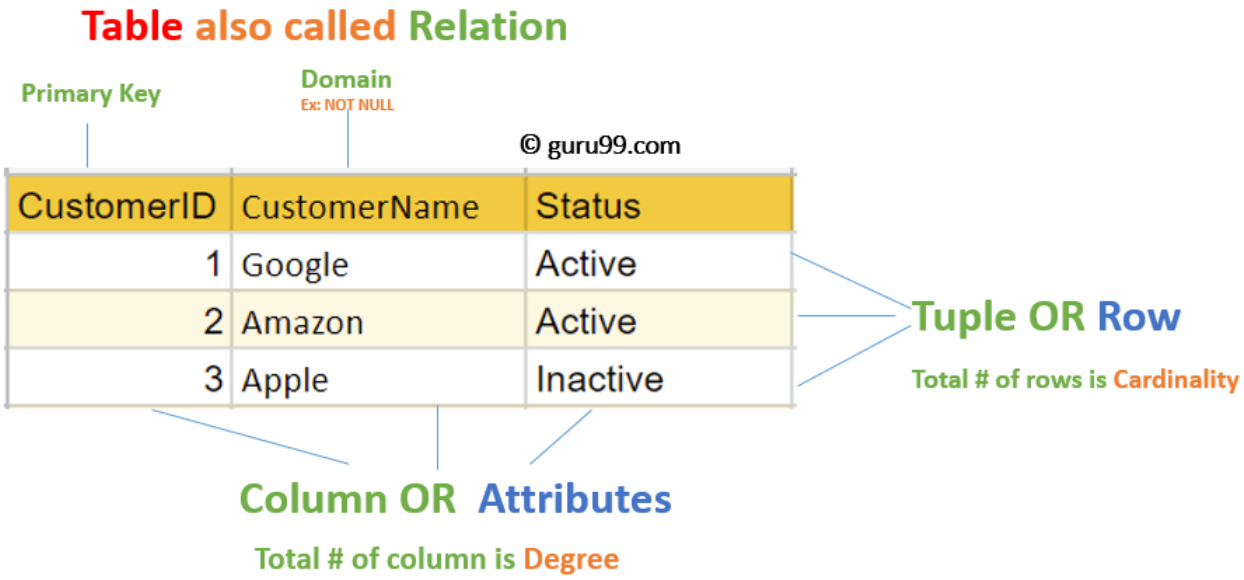
že pokud cizí klíč obsahuje hodnotu, musí tato hodnota odkazovat na platný existující řádek v nadřazené tabulce. Norma ISO podporuje definici cizích klíčů pomocí klauzule FOREIGN KEY v příkazech CREATE a ALTER TABLE (Connolly 2010, 166-167). Cizí klíče jsou popsány v kapitole 3.5.4.

3.1.5 **Relační databáze**

Relační databáze je typ databáze, který je použit při vytváření praktické části bakalářské práce. Obecný popis databáze je již zmíněn v kapitole 2.3.3. V této kapitole budou detailně rozebrány různé klíčové prvky databáze.

3.1.5.1 Relace

Relace je základem relačních databází. V terminologii relačního modelu je pojem relace ekvivalentem slova tabulka. Skládá se ze dvou částí: záhlaví a tělo. Každá relace se skládá z n-tic (záznamů) a atributů (polí). Obrázek 3 ukazuje typickou strukturu relační tabulky. (Hermandez 2013, s.78, s.555).



Obrázek 3 Struktura relační tabulky, zdroj: guru99.com

3.1.5.2 Primární klíče

V dobře navržené relační databázi má každá tabulka určitý sloupec či kombinaci sloupců, jejichž hodnoty jednoznačně identifikují každý řádek v tabulce. Tento sloupec (nebo sloupce) vyjadřují tzv. primární klíč tabulky (Groff & Weinberg 1999, 45). Primární klíč je jednoznačný identifikátor záznamu v tabulce a může to být jeden, či kombinace více sloupců tabulky. Primární klíč musí být vždy jedinečný pro určitou tabulku a musí obsahovat

hodnotu, tzn. pole primárního klíče nemůže být hodnota NULL. Každá tabulka může obsahovat pouze jeden primární klíč (Connolly 2010, 166).

3.1.5.3 Kandidátní klíče

Kandidátní klíč je první typ klíče, který je vytvořen pro tabulku. Jedná se o pole nebo sadu polí, které jednoznačně identifikují jednu instanci tabulky. Každá tabulka musí mít alespoň jeden kandidátní klíč. Z kandidátního klíče se může stát primární klíč (Hermandez 2013, s. 247).

3.1.5.4 Cizí klíče

Cizí klíč je struktura v databázi, která propojuje dvě tabulky mezi sebou. Stejně jako primární klíč tabulky může mít více sloupců a může být také kombinací více sloupců. Cizí klíč bude vždy kombinací více sloupců, pokud odkazuje na tabulku se složeným primárním klíčem (Groff & Weinberg 1999 s.47). Musí tedy vždy odkazovat na primární klíč. Původní tabulka se nazývá rodičovská, nebo odkazovaná tabulka a referencovaná tabulka s cizím klíčem se nazývá podřízená tabulka. Vztah těchto dvou tabulek opět definujeme vztahem rodič – dítě (Connolly 2010, s.166).

3.1.5.5 Index

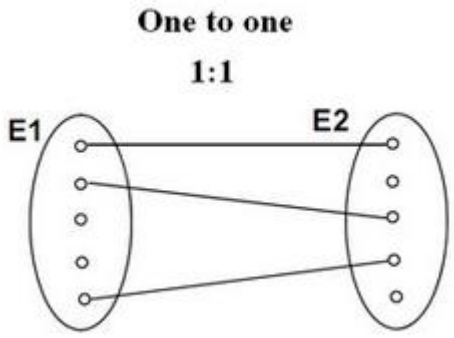
Index je datová struktura, která slouží ke zvýšení rychlosti vyhledávání v relačním sytému řízení báze dat (Hermandez 2013, s. 551). Indexy jsou někdy mylně uváděny jako klíče. Rozdíl mezi indexy a klíči je ten, že klíče jsou logické struktury, které jsou používány k identifikaci záznamů uvnitř tabulky, kdežto indexy jsou fyzické struktury, které jsou používány k optimalizaci dat a jejich zpracování. Nevýhodou indexů je, že vyžadují další místo na disku (Hermandez 2013, s. 86).

3.1.5.6 Typy vazeb

Databázové vazby jsou přidružení mezi tabulkami, která se vytvářejí pomocí příkazů spojení k načtení dat. Maximálnímu počtu vztahů instancí, kterých se entita může účastnit se říká kardinalita vztahu. Podle kardinality vztahu existují tři základní typy vztahů, které můžete setkání: jeden ku jednomu (1:1), jeden ku mnoha (1:M) a mnoho ku mnoha (M:N) (Harrington 2009, s.64).

3.1.5.6.1 Vazba 1:1

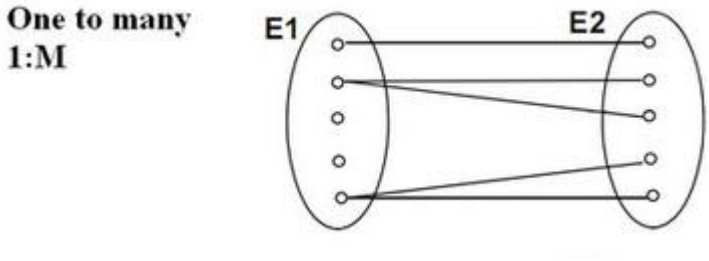
Dvě tabulky spolu mají vztah 1:1, když jeden záznam v jedné tabulce souvisí (má vztah) pouze s jedním záznamem v druhé tabulce a jeden záznam v druhé tabulce souvisí pouze s jedním záznamem v první tabulce. Na obrázku 4 je vidět vizuální znázornění této vazby. V tomto vztahu se jedna tabulka označuje jako rodič (nadřazená) a druhá tabulka jako dítě (podřazená). Tento vztah se dá založit tím, že vezmeme kopii primárního klíče rodičovské tabulky a začleníme jí do podřízené tabulky, kde se stane cizím klíčem (Hermandez 2013 s.87–88).



Obrázek 4 Vazba 1:1, zdroj: datacadamia.com

3.1.5.6.2 Vazba 1:M

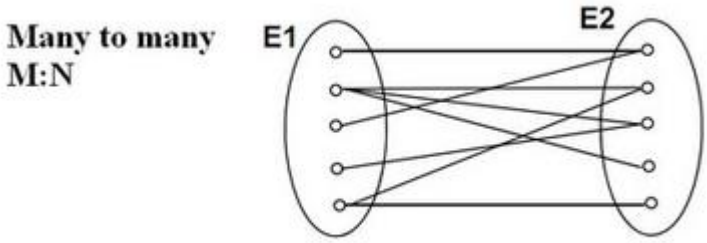
Dvě tabulky definujeme vztahem 1:M, když jeden záznam v jedné tabulce může souviset s více záznamy v druhé tabulce a jeden záznam v druhé tabulce souvisí pouze s jedním záznamem v první tabulce. Vztah těchto dvou tabulek se dá popsat podobně jako u vztahu 1:1, v tomto případě je tabulka se vztahem jedna brána jako rodič a tabulka se vztahem M je brána jako dítě. Na obrázku 5 je vidět vizuální znázornění této vazby. Tento vztah se vytvoří začleněním primárního klíče rodičovské tabulky do tabulky označené jako dítě, kde se stane cizím klíčem. Vztah 1:M je zdaleka nejběžnější vztah, který mezi dvěma tabulkami v databázi existuje. Je zásadní z hlediska integrity dat, protože pomáhá eliminovat duplicitní data a udržet nadbytečná data na minimu (Hermandez 2013, s.88–89).



Obrázek 5 Vazba 1:M, zdroj: datacadamia.com

3.1.5.6.3 Vazba M:N

Dvě tabulky spolu mají vztah M:N, když jeden záznam v jedné tabulce může souviset s více záznamy v druhé tabulce a jeden záznam v druhé tabulce může souviset s více záznamy v první tabulce. Tento vztah se vytvoří pomocí propojovací tabulky. Na obrázku 6 je vidět vizuální znázornění této vazby. Propojovací tabulka se vytvoří tím, že se vezmou z obou tabulek primární klíče, a použijí se k vytvoření struktury nové tabulky (Hermandez 2013 s.88–89).



Obrázek 6 Vazba M:N, zdroj: datacadamia.com

3.1.5.7 Normalizace

Normalizace je technika návrhu databáze, která začíná zkoumáním vztahů tzv. funkčních závislostí mezi atributy (funkční závislost je vysvětlena v kapitole 3.5.8). Atributy popisují některé vlastnosti dat nebo vztahů mezi daty, které jsou pro podnik důležité. Normalizace používá řadu testů (popsaných jako normální formy), které pomáhají identifikovat optimální seskupení pro tyto atributy, aby nakonec identifikovaly sadu vhodných vztahů, které podporují datové požadavky podniku. (Connolly 2010, s.387).

Účelem normalizace je identifikovat vhodnou sadu relací, které podporují datové požadavky podniku. Charakteristiky vhodné sady relací zahrnují:

- minimální počet atributů nezbytných pro podporu datových požadavků podniku
- atributy s blízkým logickým vztahem (popsaným jako funkční závislost) se nacházejí ve stejné relaci (funkční závislost je popsána v kapitole 3.5.8)

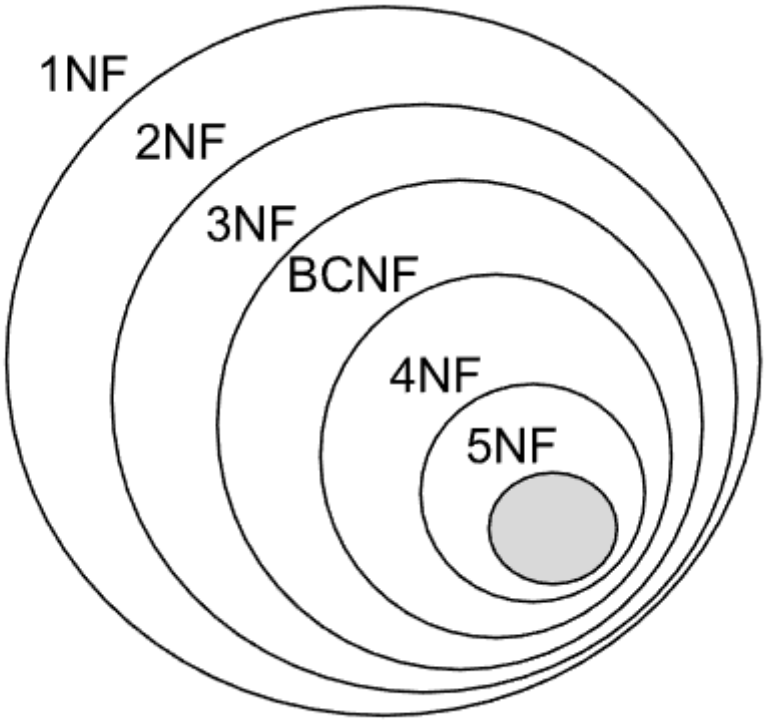
- minimální redundance je s každým atributem zastoupena pouze jednou s výjimkou atributů, které tvoří všechny části cizích klíčů, které jsou nezbytné pro spojení souvisejících relací

(Connolly 2010, s.388).

Výhodou použití databáze, která má vhodnou sadu relací je snazší uživatelská dostupnost k datům a jejich následná správa. Další výhodou je minimální prostor, který databáze zabírá, jelikož se data se v databázi neopakují. (Connolly 2010, s.388-389).

3.1.5.8 Normální formy

Teoretická pravidla, která definují správný návrh relací, jsou znám jako normální formy. Normální formy se navzájem doplňují a každá normální forma představuje stále přísnější seznam pravidel. Obecně platí, že čím vyšší normální forma, tím lépe je design relací navržen. Na obrázku 7 je vidět, že pokud je relace v jedné z vyšších normálních forem, pak je automaticky obsažena i v normálních formách, které jsou pod ní. Pokud lze umístit relace do třetí normální formy (3NF), pak se ve většině případech vyhnete běžným problémům se špatnými návrhy relací. Tři vyšší normální formy – Boyceho – Coddova normální forma, čtvrtá normální forma (4NF) a pátá normální forma (5NF) předchází zvláštním situacím, které mohou občas nastat, tyto situace jsou popsané dále u 4NF a 5NF. (Hermandez 2013 s.105).



Obrázek 7 Normálové formy, zdroj: aksakalli.github.io

3.1.5.8.1 První normální forma (1NF)

Abychom dostali první normálovou formu, každé pole v tabulce musí obsahovat jedinečnou informaci. Nadále pak všechny atributy musí mít jedinou hodnotu. Doména databáze může obsahovat pouze atomické hodnoty a neopakující se skupiny atributů. Za těchto předpokladů je databáze v první normálové formě. Atomické hodnoty jsou hodnoty, které už nelze rozdělit na menší části (Connolly 2010, 403).

3.1.5.8.2 Druhá normální forma (2NF)

K tomu, aby byla relace v druhé normální formě musí být splněno, že se relace nachází v první normální formě a všechny neklíčové atributy jsou funkčně závislé na celém primárním klíči. Funkční závislost je zvláštní vztah mezi atributy. Jedná se o jednosměrný vztah mezi dvěma atributy takový, že v jakémkoliv daném čase, pro každou jedinečnou hodnotu atributu A, je pouze jedna hodnota atributu B, která je s ním spojena prostřednictvím relace (Hermandez 2013 s.111).

Druhá normální forma platí pro vztahy s kompozitem klíče, tedy vztahy s primárním klíčem složeným ze dvou nebo více atributů. Vztah s primárním klíčem s jedním atributem je automaticky minimálně 2NF (Connolly 2010).

3.1.5.8.3 Třetí normální forma (3NF)

Proto, aby relace mohla být ve třetí normální formě, musí být splněny dvě podmínky. Relace je ve druhé normální formě a zároveň v relaci neexistují žádné tranzitivní závislosti (Harrington 2009 s.114). Tranzitivní závislost je vztah mezi třemi atributy A, B a C, kdy platí že A odkazuje na B a B odkazuje na C, tudíž je C tranzitivně závislé na atributu A přes atribut B, za předpokladu, že atribut A není funkčně závislý na atributu B, nebo atributu C (Connolly 2010 s.397).

3.1.5.8.4 Boyceho – Coddova normální forma

Boyceho – Coddova normální forma (BCNF) je založena na funkčních závislostech, které berou v úvahu všechny kandidátní klíče v relaci. BCNF má nicméně také další omezení ve srovnání s 3NF. K tomu, aby relace byla v BCNF, musí platit dvě podmínky. Relace musí být ve třetí normální formě a zároveň každý determinant je kandidátským klíčem. Determinant je atribut nebo skupina atributů, který určuje hodnoty přiřazené dalším atributům ve stejném řádku. Rozdíl v BCNF oproti 3NF je, že pro funkční závislost, kdy atribut A odkazuje na atribut B, umožňuje 3NF tuto závislost ve vztahu: atribut B je primární klíč a atribut A není kandidátní klíč. V BCNF tento vztah není možný a je nutné, any atribut B byl primárním klíčem a atribut B kandidátním klíčem (Harrington 2009, s.116-117; Connolly, 2010, s.419-420).

3.1.5.8.5 Čtvrtá normální forma (4NF)

Přestože BCNF odstraňuje jakékoli anomálie způsobené funkčními závislostmi, další výzkum vedl k identifikaci jiného typu závislosti, tzv. Vícehodnotová závislost (MVD), která může také způsobit redundanci dat. Možnost existence MVD v relaci je způsobena První normální formou, která neumožňuje atributu v n-tice mít sadu hodnot (Connolly 2010, s.429).

MVD představuje závislost například mezi atributy A, B a C v relaci, a to tak, že pro každou hodnotu A existuje sada hodnot pro B a sada hodnot pro C, nicméně sada hodnot pro B a C jsou na sobě nezávislé. Máme-li například dva atributy s více hodnotami ve vztahu, musíme opakovat každou hodnotu jednoho z atributů s každou hodnotou druhého atributu, abychom zajistili, že n-tice vztahu budou konzistentní (Connolly 2010, s.428-429).

K tomu, aby byla relace ve čtvrté normálové formě, musí být relace v BCNF a nesmí obsahovat žádné triviální vícehodnotové závislosti (Harrington 2009, s.118).

3.1.5.8.6 Pátá normální forma (5NF)

Aby byla relace v páté normální formě tak musí být ve 4NF a zároveň originální tabulka musí být znovu sestavena z tabulek, na které byla rozebrána (Harrington 2009, s.122-123).

3.1.5.9 12 Coddových pravidel

V roce 1985 publikoval E.F. Codd sérii dvou článků v týdeníku Computerworld pro počítačový průmysl. První článek stanovil 13 pravidel (pravidlo 0–12), která by měla dodržovat plně relační databáze (Harrington, 2009, s.139).
V průběhu let vyvolala Coddova pravidla velkou kontroverzi. Někteří tvrdí, že tato pravidla jsou pouze akademickým cvičením. Jiní tvrdí, že jejich produkty již splňují většinu, ne-li všechna pravidla. Tato diskuse vyvolala rostoucí povědomí v komunitách uživatelů a dodavatelů o základních vlastnostech skutečného relačního systému řízení báze dat (Connolly, 2010, s.1293).

3.1.5.9.1 Pravidlo 0 - Pravidlo založení

„Pro jakýkoli systém, který je prohlašován za relační systém řízení správy dat, musí být tento systém schopen spravovat databáze zcela prostřednictvím svých relačních schopností.“ Toto pravidlo říká, že DMBS by se neměl uchýlit k žádným nerelačním operacím, aby dosáhl jakékoli ze svých schopností správy dat, kterými jsou například definice dat a manipulace s nimi. Tyto schopnosti správy dat jsou popsány v kapitole 3.6.3 (Connolly 2010, s.1293).

3.1.5.9.2 Pravidlo 1 - Informační pravidlo

„Všechny informace v relační databázi jsou vyjádřeny explicitně na logické úrovni jediným způsobem – hodnotami v tabulkách.“ Toto pravidlo vyžaduje, aby všechny informace, dokonce i metadata uchovávaná v systémovém katalogu, byly uloženy jako vztahy a spravovány stejnými provozními funkcemi, jaké by byly použity k údržbě dat. (Connolly 2010, s.1294).

3.1.5.9.3 Pravidlo 2 - Pravidlo jistoty

„Všechna data v relační databázi jsou zaručeně logicky přístupná kombinací jména tabulky s hodnotami primárního klíče a jménem sloupce.“ (Connolly 2010, s.1295).

3.1.5.9.4 Pravidlo 3 - Systematické zpracování nulových hodnot

„Nulové hodnoty jsou plně podporovány RDBMS pro reprezentaci informace, která není definována, a to nezávisle na datovém typu.“ Toto pravidlo udává, že chybějící informace může být stále zapsána do tabulky s hodnotou NULL (Prázdná) (Connolly 2010, s.1295).

3.1.5.9.5 Pravidlo 4 - Dynamický on-line katalog založený na relačním modelu

„Popis databáze je vyjádřen na logické úrovni stejným způsobem jako zákaznická data, takže autorizovaný uživatel může aplikovat stejný relační jazyk ke svému dotazu jako uživatel při práci s daty.“ Toto pravidlo určuje, že existuje pouze jeden jazyk pro manipulaci s metadaty i daty a také, že existuje pouze jedna logická struktura používaná k ukládání systémových informací (Connolly 2010, s.1296).

3.1.5.9.6 Pravidlo 5 - Obsáhlý datový podjazyk

„Relační systém může podporovat několik jazyků a různých módů použitých při provozu terminálu. Nicméně musí být nejméně jeden příkazový jazyk s dobře definovanou syntaxí, který obsáhle podporuje definici dat, definici pohledů, manipulaci s daty jak interaktivně, tak programem, integritní omezení, autorizovaný přístup k databázi, transakční příkazy apod.“ Standard ISO pro SQL, uveden v kapitole 3.6.2, poskytuje všechny tyto funkce, takže každý jazyk vyhovující tomuto standardu bude automaticky splňovat toto pravidlo. (Connolly 2010, s.1296).

3.1.5.9.7 Pravidlo 6 - Pravidlo vytvoření pohledů

„Všechny pohledy, které jsou teoreticky možné, jsou také systémem vytvořitelné.“ Toto pravidlo říká že pokud je pohled teoreticky vytvořitelný pak by měl být systém schopen ho vytvořit a průběžně aktualizovat (Connolly 2010, s1294-1295).

3.1.5.9.8 Pravidlo 7 - Schopnost vkládání, vytvoření a mazání

„Schopnost zachování relačních pravidel u základních i odvozených relací je zachována nejen při pohledu na data, ale i při operacích průniku, přidání a mazání dat.“ (Connolly 2010, s.1296).

3.1.5.9.9 Pravidlo 8 - Fyzická datová nezávislost

„Aplikační programy jsou nezávislé na fyzické datové struktuře.“ (Connolly 2010, s.1296).

3.1.5.9.10 Pravidlo 9 - Logická datová nezávislost

„Aplikační programy jsou nezávislé na změnách v logické struktuře databázového souboru“ (Connolly 2010, s.1296).

3.1.5.9.11 Pravidlo 10 - Integritní nezávislost

„Integritní omezení se musí dát definovat prostředky relační databáze nebo jejím jazykem a musí být schopna uložení v katalogu, a nikoliv v aplikačním programu.“ Codd zdůrazňuje, že integritní omezení musí být uložena v systémovém katalogu, spíše než zapouzdřena v

aplikačních programech nebo uživatelských rozhraních. Ukládání omezení do katalogu systému má výhodu centralizovaného řízení a vynucování (Connolly 2010, s.1295).

Pravidlo 11 - Nezávislost distribuce

„RDBMS musí být schopny implementace na jiných počítačových architekturách.“ Nezávislost distribuce znamená, že aplikační program, který se připojuje k DBMS na jednom počítači, by měl také fungovat v síťovém prostředí bez úprav, i když se data přesouvají z počítače na počítač (Connolly 2010, s.1297).

Pravidlo 12 - Pravidlo přístupu do databáze

„Jestliže má relační systém jazyk nízké úrovně, pak tato úroveň nemůže být použita k vytváření integritních omezení, a je nutno vyjádřit se v relačním jazyce vyšší úrovně.“ Toto pravidlo vyžaduje, aby veškerý přístup k databázi byl řízen DBMS, aby integrita databáze nemohla být ohrožena bez vědomí uživatele nebo správce databáze. (Connolly 2010, s.1294).

3.2 **SQL**

SQL je nástroj pro organizaci, správu a získávání dat uložených v počítačové databázi. Název "SQL" je zkratka pro Structured Query Language. Z historických důvodů se SQL obvykle vyslovuje jako "SEQUEL", alternativní výslovnost "S.Q.L." je ale také možná. SQL je počítačový jazyk, který je používán k interakci s databází. SQL pracuje s jedním specifickým typem databáze-relační databází (Groff & Weinberg 1999, s.9).

Je-li potřeba načíst data z databáze, je k vytvoření požadavku využit jazyk SQL. DBMS zpracuje požadavek SQL, načte požadovaná data a vrátí je zpět. Tento proces vyžádání dat z databáze a zpětného příjmu výsledků se nazývá databázový dotaz (Groff & Weinberg, 1999, s.9). SQL je v nynější době mnohem víc než jen dotazovací nástroj, ačkoliv to byl jeho původní účel a načítání dat je stále jednou z jeho nejdůležitějších funkcí. SQL se používá k ovládání všech funkcí, které DBMS poskytuje svým uživatelům. Mezi tyto patří např. definice dat, načítání dat, manipulování s daty, kontrola a řízení přístupu, sdílení dat a integrita dat (Groff & Weinberg 1999, s.10).

SQL není úplný počítačový jazyk jako COBOL, C, C++ nebo Java. SQL neobsahuje žádný příkaz IF pro testovací podmínky a žádné příkazy GOTO, DO nebo FOR pro řízení toku programu. Místo toho je SQL databázový podjazyk sestávající asi z čtyřiceti příkazů specializovaných na úlohy správy databází. Tyto SQL příkazy lze vložit do jiného jazyka,

jako je COBOL nebo C, které mohou rozšířit přístup k databázi (Groff & Weinberg 1999, s.10).

3.2.1 **Historie SQL**

Historie jazyka SQL je úzce propojena s vývojem relačních databázi. V roce 1970 Edgar Frank Codd, který v té době pracoval ve společnosti IMB (americká mezinárodní technologická společnost), definoval relační model ve svém článku „A Relational Model of Data for Large Shared Data Banks". V tomto článku Codd popsal matematickou teorii o tom, jak mohou být data uložena a manipulována pomocí tabulkové struktury (Groff & Weinberg, 1999 s.22).

Coddův článek byl spouštěčem pro zahájení výzkumu relačních databází. Jeden z výzkumů byl také projekt přímo od společnosti IBM nazvaný ,,Systém/R''. Cílem projektu bylo prokázat proveditelnost relačního konceptu databází. Mezi lety 1974 a 1975 byl v první fázi projektu Systém/R vyroben první minimální prototyp pro relační databázový systém. Kromě samotného databázového systému zahrnoval projekt System/R také práci na dotazovacím jazyku pro tento typ databází. Jeden z těchto jazyků se jmenoval SEQUEL, což je zkratka pro Structured English Query Language. Název jazyka SEQUEL musel být později z právních důvodů změněn na SQL, jeho výslovnost však zůstala stejná, a to sequel. V letech 1976 a 1977 byl projekt od základů přepsán. Nová implementace podporovala dotazy mezi více tabulkami a umožňovala více uživatelům sdílený přístup k datům. Tuto novou implementaci společnost IBM v letech 1978 a 1979 pro zisk zpětné vazby distribuovala na řadu zákaznických stránek. Zpětná vazba poskytla zkušenosti od skutečných uživatelů s projektem System/R a jeho databázovým jazykem SEQUEL. V roce 1979 byl projekt System/R ukončen a společnost IBM došla k závěru, že relační databáze nejsou jen proveditelné, ale mohou být základem komerčních produktů (Groff & Weinberg 1999, s.24). Narůstající povědomí laické veřejnosti o projektu System/R přitáhla pozornost skupiny inženýrů v Kalifornii, kteří měli vizi, že výzkum IMB předznamenal komerční trh pro relační databáze. V roce 1977 založili společnost Relational Software Inc., aby vytvořili relační databázový systém založený na SQL. Jejich produkt byl nazván Oracle byl vydán v roce 1979 a stal se z něj první komerčně dostupný relační databázový systém. Oracle předběhl první produkt od společnosti IBM o dva roky a běžel na minipočítačích VAX od firmy Digital Equipment Corporation. Společnost Relational Sowtware Inc. V současné době je

společnost známá jako Oracle Corporation a patří mezi přední prodejce systémů pro správu relačních databází s multimiliardovými obraty (Groff & Weinberg 1999, s.24).

V roce 1983 IBM představilo Database 2 (DB2). Jednalo se o další relační DBMS pro systémy sálových počítačů. DB2 fungoval pod operačním systémem IBM MVS (operačním systémem používaným ve velkých datových centrech). První verze DB2 se začala dodávat v roce 1985 a představitelé IBM o ní mluvili jako o strategické součásti softwarových technologie IBM. DB2 se od té doby stala vlajkovou lodí IBM pro RDBMS a jazyk SQL DB2 se sv tu dobu stal standardem databázových jazyků. V roce 1997 IBM posunulo strategii napříč platformami DB2 ještě dále tím, že oznámilo verze DB2 pro počítačové systémy vyrobené společnostmi Sun Microsystems, Hewlett-Packard a dalšími hardwarovými konkurenty IBM (Groff & Weinberg, 1999 s.25).

Během druhé poloviny 80. let byly SQL a relační databáze rychle přijímány jako databázová technologie budoucnosti. Výkony relačních databázových produktů se výrazně zlepšovaly. Zejména produkty společností Ingres a Oracle s každou novou verzí výrazně přeskočili konkurenci s dvojnásobným až trojnásobným výkonem, než měla předchozí verze (Groff & Weinberg, 1999 s.25).

IBM posílila umístění DB2 jako řešení správy dat pro 90. léta. Publikace standardu ANSI/ISO pro SQL v roce 1986 dala SQL oficiální status databázového standardu. SQL se také objevilo jako standard na počítačových systémech založených na Unixu, jehož popularita v 80. letech prudce rostla. Jak se osobní počítače stávaly výkonnějšími a byly propojeny v lokálních sítích, potřebovaly sofistikovanější databázi řízení. Prodejci PC databází přijali SQL jako řešení těchto potřeb a prodejci minipočítačových databází se posunuli „dolů na trh“, aby mohli konkurovat nastupujícím počítačům místní síťový trh. Přes počátek 90. let se neustále zlepšuje SQL implementace a dramatické zlepšení rychlosti procesoru učinily SQL praktickým řešení pro aplikace zpracovávající transakce. SQL stalo klíčovou součástí architektury klient/server, které používaly osobní počítače (Groff & Weinberg 1999, s.25-26).

Koncem 90. let již „správa databáze“ nebyla monolitickým trhem. Specializované databázové systémy se objevily na podporu různých potřeb trhu. Jedním z nejrychleji rostoucích segmentů bylo „skladování dat“, kde byly databáze používány k prohledávání obrovského množství dat k odhalování základních trendů a vzorců. Druhým hlavním trendem bylo začlenění nových datových typů (jako jsou multimediální data) a objektově orientovaných principů do SQL. Třetím důležitým segmentem byly „mobilní databáze“ pro

přenosné osobní počítače, které mohly fungovat, když byly někdy připojeny a někdy odpojeny od centralizovaného databázového systému. Navzdory vzniku podsegmentů databázového trhu zůstal SQL jazykem mezi všemi (Groff & Weinberg 1999, s.26).

3.2.2 **SQL Standardy**

V 80. letech 20. století se relační databázové modely pomalu stávaly databázovým standardem. Z důvodu, že se od sebe začaly produkty od velkých výrobců více a více lišit, bylo nezbytné zavést určité standardy, které musí relační model splňovat (Kriegel & Trukhnov 2008, s.23).

Kolem roku 1978 zadal Výbor pro datové systémy a jazyk (CODASYL) vývoj síťového datového modelu jako prototypu pro jakékoli budoucí implementace databází. Tato práce začala na počátku 70. let 20. století. V roce 1982 tyto snahy vyvrcholily návrhem standardů jazyka pro definici dat (DDL) o které se píše v kapitole 3.6.3 a jazyka pro manipulaci s daty (DML) o které se píše v kapitole 3.6.3. O čtyři roky později se z nich staly standardy schválené organizací tehdy známou jako ANSI Technical Committee X3H2 (Database). Americká standardizační organizace (ANSI) zajišťuje celosvětovou standardizaci jazyka SQL (Kriegel & Trukhnov 2008, s.23).

V letech 1986-87 pracovaly organizace pro standardy ANSI a ISO na standardu SQL, který byl vydán o dva roky později, známý jako SQL-89. Tento standard byl aktualizován v roce 1991 na novou verzi SQL-92 a následně na další verzi SQL-99 (také označovaná jako SQL:1999 a SQL3). Standard SQL se v dělí na tři klíčové komponenty DML, DDL, DCL, o kterých je zmíněno v kapitole 3.6.3 (Darie & Watson 2008, s.3).

3.2.3 **Příkazy jazyka SQL**

Jazyk SQL je podle definice ANSI rozdělen do několika různých sekcí. Příkazy SQL spadají do různých kategorií podle toho, co funkci, kterou vykonávají rozdělení příkazů je vidět na Obrázku 8 (Din 2014, 21).

Data Definition Language neboli DDL (nazývaný Schema Definition Language podle ANSI) se skládá z těch příkazů v SQL, které přímo vytvářejí databázové objekty, jako jsou tabulky, pohledy indexů. Mnoho příkazů SQL také během zpracování vytváří dočasné databázové objekty. Mezi příkazy DDL řadíme příkazy CREATE – pro vytvoření objektu, ALTER – pro změnu objektu, DROP pro odstranění objektu (Din 2014, 21). S DDL příkazy můžeme v databázi provádět:

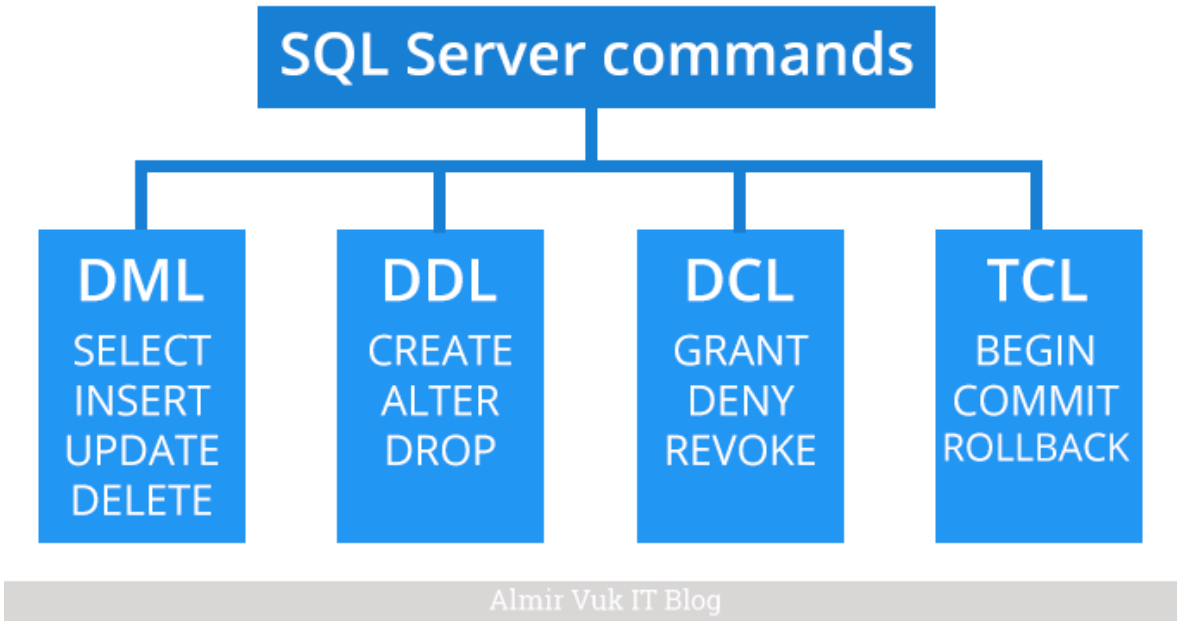
- Definování a vytvoření nové tabulky
- Odstranění tabulky, která již není potřeba
- Změna definice existující tabulky
- Definování virtuální tabulky tzv. pohledu (VIEW)
- Zavedení bezpečnostní kontroly pro databáze
- Vytvoření indexu pro rychlejší přístup k tabulce
- Řídit fyzické ukládání dat pomocí DBMS

Z větší části příkazy DDL izolují od podrobností, jak jsou data fyzicky uložena v databázi. Manipulují s abstraktními databázovými objekty, jako jsou tabulky a sloupce (Groff & Weinberg 1999, s.256).

Data Manipulation Language neboli DML se skládá z těch příkazů, které pracují s daty v databázi. To zahrnuje příkazy, které přidávají data do tabulek, a také příkazy, které se používají k dotazování databáze (Din 2014, 21). Mezi příkazy DML patří příkazy SELECT – pro výběr dat z databáze, INSERT – pro vložení nových dat do databáze, DELETE – pro odstranění dat z databáze, UPDATE – pro změnu dat v databázi (Groff & Weinberg 1999, s.256).

Třetí pododdělení SQL příkazů je Data Control Language nebo DCL. Jedná se o příkazy SQL, které jsou používány pro zabezpečení dat. Tyto příkazy určují, zda uživatel smí provést určitou operaci či nikoli. Standard ANSI seskupuje tyto příkazy jako součást DDL (Din, 2014). Do této kategorie spadají příkazy GRANT – přidělení oprávnění uživatelům k objektům a REVOKE – odebírá práva uživatelům k určitým objektům (Groff & Weinberg, 1999 s.257).

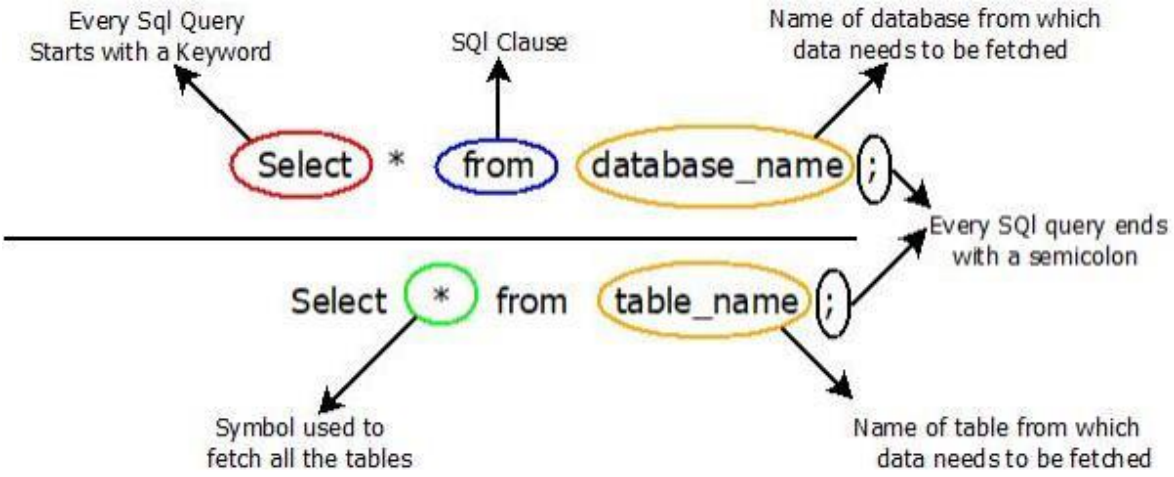
Čtvrtou sekcí příkazů je Transaction Control Language (TCL). Jedná se o příkazy, které řeší transakce v databázi. Do této skupiny patří příkazy COMMIT – potvrzení transakce, ROLLBACK – zrušení transakce, návrat do původního stavu databáze, SAVEPOINT – vytvoření bodu ve skupině transakcí na které se může v případě problému použít příkaz ROLLBACK a SET TRANSACTION – pro specifikaci vlastnosti transakce. (Groff & Weinberg, 1999, s.53).



Obrázek 8 Rozdělení příkazů SQL, zdroj: almirvuk.blogspot.com

3.2.4 **SQL Syntaxe**

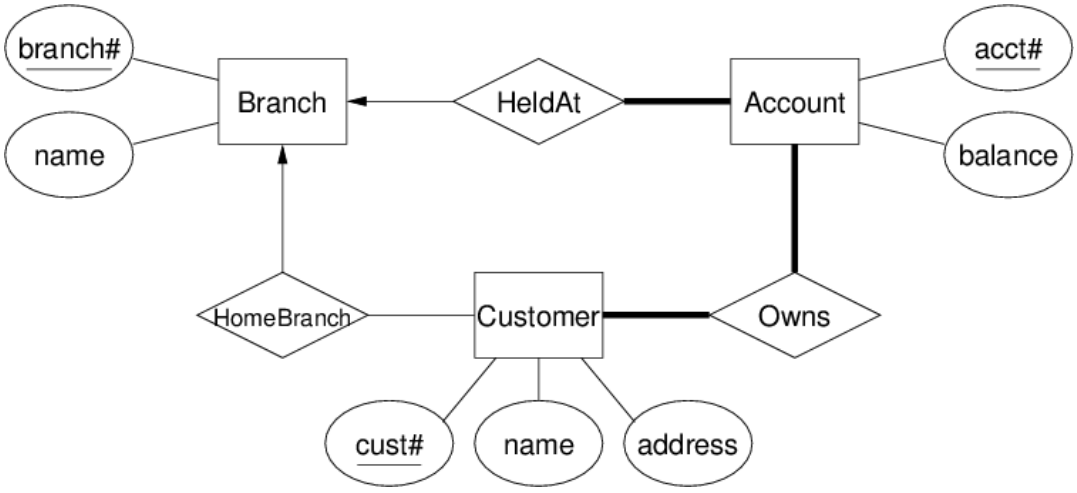
Příkazy jazyka SQL se podobají anglickým větám. Každý příkaz SQL začíná slovesem neboli klíčovým slovem, které popisuje, co daný příkaz dělá. SELECT, CREATE, INSERT, DELETE a COMMIT jsou typická slovesa. Příkaz pokračuje jednou nebo více klauzulemi. Klauzule může specifikovat data, která má příkaz vybrat, nebo poskytnout více podrobností o tom, co má příkaz dělat. Každá klauzule také začíná klíčovým slovem, například WHERE, FROM, INTO a HAVING. Některé věty jsou nepovinné; další jsou vyžadovány. Obrázek 9 ukazuje základní syntaxi jazyka SQL. Konkrétní struktura a obsah se liší od jedné věty k druhé. Mnoho klauzulí obsahuje názvy tabulek nebo sloupců; některé mohou obsahovat další klíčová slova, konstanty nebo výrazy (Groff & Weinberg 1999, s.53).



Obrázek 9 Základní syntaxe jazyka SQL, zdroj: minigranth.in

3.3 ER Diagram

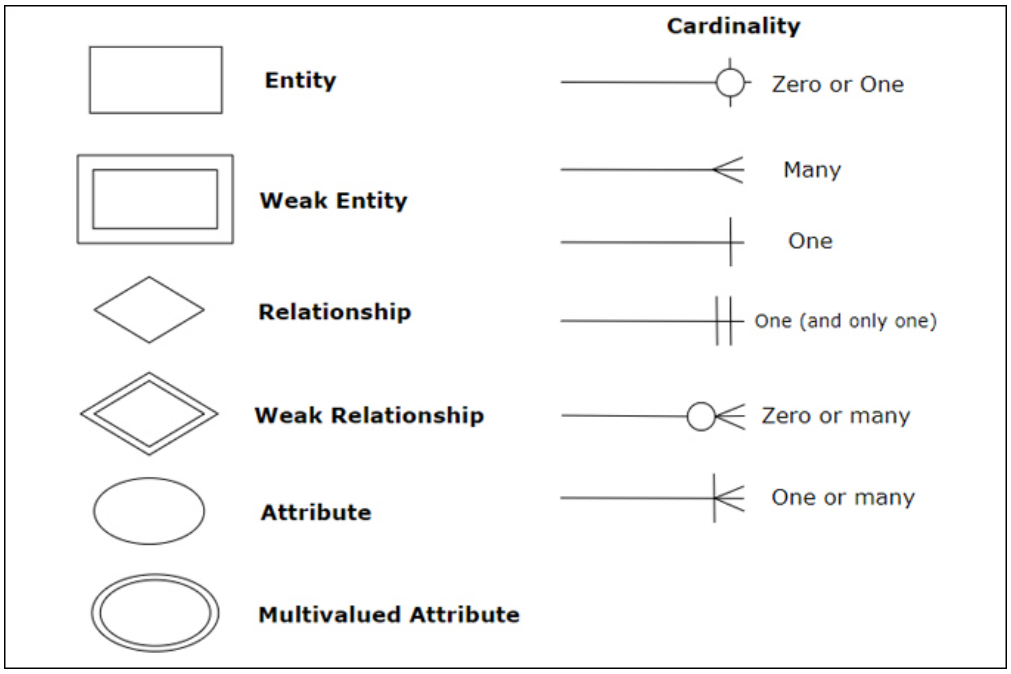
Jedním z nejobtížnějších aspektů návrhu databáze je, že návrháři, programátoři a koncoví uživatelé mají tendenci nahlížet na data a jejich použití různými způsoby. Pokud nebude dosaženo společného porozumění, které odráží, jak podnik funguje, tak design, který vytváříme, nebude splňovat požadavky uživatelů. Jedním z možných prostředků, jak tomuto problému předejít, je použít ER Diagramu, který je ukázán na Obrázku 10. ER Diagram neboli Entity Relationship Diagram (ERD) je vizuální reprezentace dat, která popisuje, jak spolu data souvisí pomocí různých ERD symbolů a notací. ERD pomáhají vysvětlit logickou strukturu databází a jsou vytvořeny na základě tří základních pojmů: entita, atribut a vztah. ERD obsahují tři základní symboly, kterými jsou obdélník, ovál a kosočtverec, které představují vztahy mezi prvky, entitami a atributy. Existuje několik dílčích prvků, které jsou založeny na hlavních prvcích v ER Diagramu Prvky jsou graficky zobrazeny na obrázku 11.



Obrázek 10 ERD Notace, zdroj: gitmind.com

- Obdélník: Tento symbol diagramu vztahů entit představuje typy entit
- Elipsa: Symbol představují atributy
- Diamant: Tento symbol představuje typy vztahů
- Linka: Spojuje atributy s typy entit a typy entit s jinými typy vztahů
- Primární klíč: atributy jsou podtržené
- Dvojitá elipsa: Představují vícehodnotové atributy

(Connolly 2010, s.342).



Obrázek 11 ER Model, zdroj: cgi.cse.unsw.edu.au

4 VLASTNÍ PRÁCE

4.1 Návrh relačního modelu pro provozní deník

Pro relační model je nejprve nutné stanovit zadání, které by měl provozní deník obsahovat. Z požadavků bude následně vypracován konceptuální návrh datového modelu. Po vypracování konceptuálního modelu budou v logickém modelu přiřazeny jednotlivé logické vazby mezi relacemi. V poslední fázi budou ve fyzickém návrhu vazby navrženy přímo na zvolený SŘBD.

4.1.1 Stanovení zadání

Zadáním práce je vypracování relačního datového modelu pro provozní deník. Provozní deník v elektrárně slouží k zachycení událostí v provozu, které se staly během jednotlivých

směn. Do deníku se ukládá řada informací, např. která zařízení jsou v provozu, či se tam nahlašují neobvyklé události během směny. Ke správnému fungování aplikace je zapotřebí navrhnout správný relační model databáze, který zajistí spolehlivost celého sytému. K tomu, aby byl relační model co nejefektivnější, je nutné zajistit co nejmenší opakování dat. Zároveň musí být objekty navrženy tak, aby byl model schopen spolehlivě fungovat s programovacím jazykem Python, ve kterém bude aplikace ,,Provozní deník'' implementována. Do provozního deníku budou uživatelé vkládat prostřednictvím webových formulářů hodnoty, které se následně pomocí aplikace uloží do databáze. Relační model je nutné navrhnout pro SŘBD Oracle, který byl zvolen z toho důvodu, že je na něm postaven datový sklad firmy, odkud budou získávána či ukládána některá data.

4.1.1.1 Požadavky zadavatele

Požadavkem zadavatele je vytvořit webové formuláře, kam budou pracovníci jednotlivých směn vyplňovat hodnoty provozního deníku. Dále je nutné vzít v potaz, že provozní deník se skládá z více směnových deníků, které bude také nezbytné nadefinovat v relačním modelu. Dalším požadavkem je vytvořit možnost vkládání poznámek k jednotlivým deníkům. Je potřeba, aby u každého nově založeného deníku a u každé poznámky bylo vidět, který uživatel akci provedl. Uživatel také musí mít možnost nový deník otevřít, nebo uzavřít. Po uzavření deníku již nebude možno jednotlivé položky upravovat, v případě potřeby znovu otevření deníku bude potřeba kontaktovat správce SŘBD. Data, která budou tímto způsobem získávána, nebo ukládána, jsou popsána v kapitolách 4.2.1 a 4.2.2.

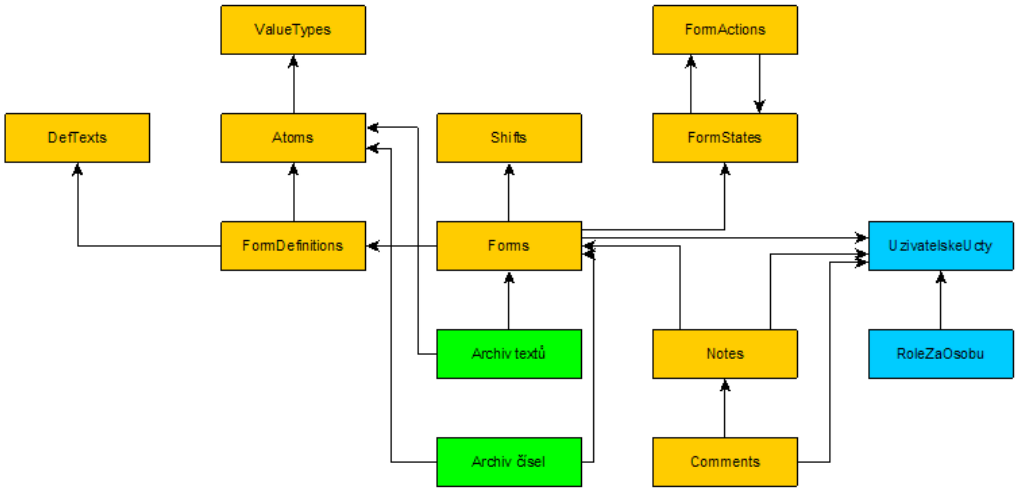
4.1.2 **Datové modelování**

Se stanoveným zadáním a s doplněnými požadavky od zadavatele může být vytvořen správný návrh relačního databázového modelu. Modelování bude prováděno prostřednictvím ER diagramu popsaného v kapitole 3.7. Modelovaní databázového schématu bude rozděleno do tří úrovní návrhu.

4.1.2.1 Konceptuální návrh datového modelu

V konceptuální úrovni je vypracován a popsán zjednodušený návrh modelu. Návrh byl vypracován jednak z požadavků na zadání a jednak tak, aby mohl být celý model napojen na datový sklad firmy, která je pověřena správou provozního deníku.

V návrhu modelu, který je zobrazen na obrázku 12, jsou vypracovány jednotlivé návrhy relací. Jedná se o zjednodušený ER diagram, který ukazuje jednotlivé relace a jejich odkazy na další relace.



Obrázek 12 Konceptuální návrh datového modelu, zdroj: vlastní zpracování

Relace jsou označeny třemi barvami. Oranžová barva znázorňuje relace, které jsou součástí schématu nově vzniklého provozního deníku. Tyto relace jsou nově vytvořené a bude nutné je propojit s datovým skladem firmy. Modrou barvou jsou znázorněny relace, které jsou uloženy v datovém skladu firmy, a model provozního deníku z nich bude získávat potřebná data. Zelenou barvou jsou označeny relace, které získávají data z modelu provozního deníku a ukládají je do datového skladu firmy.

V tabulce 1 je popsán význam jednotlivých objektů, které budou použity pro návrh relačního modelu. Základní relací pro vytvoření modelu je relace Forms (formuláře), která obsahuje obecné informace o formuláři. K relaci Forms jsou nadřazeny relace FormDefinitions (definice formuláře), Shifts (směny), FormStates (stavy formuláře) a UzivatelskeUcty. Relace FormDefinition obsahuje informace o tom, o který ze směnových deníků se jedná. K relaci FormDefinition jsou nadřazeny relace DefTexts, které slouží jako předdefinovaný text do poznámek, a Atoms (Zadávaný údaj), kde bude uložena určitá hodnota z provozního deníku. Relace Atom má jednu nadřazenou relaci ValueTypes, která definuje, o jaký typ hodnoty se jedná. Relace Shifts nám sděluje, která směna v daném formuláři operovala. Relace FormStates dává informaci, v jakém stavu se deník v současné chvíli nachází (Otevřen, Uzavřen). K této relaci je nadřazena relace FormActions, která obsahuje akce, které lze s deníkem udělat (Založit, Uzavřít).

K relaci Forms je podřazena relace Notes, kde se ukládají poznámky z deníků. K této relaci je podřazena relace Comments, kde jsou uloženy komentáře k poznámkám.

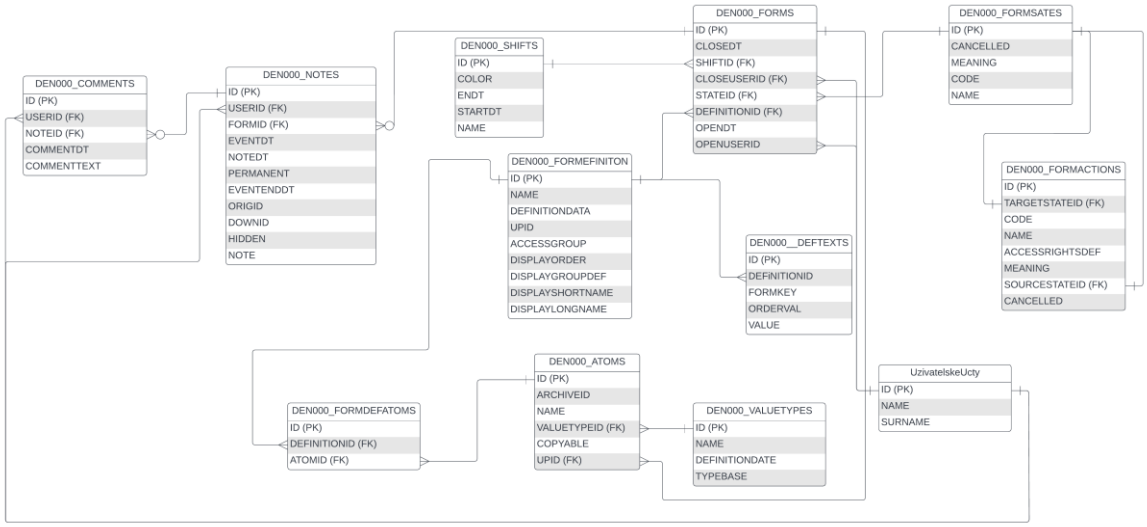
K relacím Forms, Notes a Comments je nadřazena Relace UzivatelskeUcty, která již funguje ve firemním schématu a se kterou bude nově vzniklý model propojen. Tato relace obsahuje informace o tom, o kterého uživatele se jedná.
Relace Forms a Atoms mají odkazované tabulky Archiv textů a Archiv čísel. Tyto tabulky jsou také již ve firemním schématu a slouží k ukládání všech veličin, tedy ne jenom veličin provozního deníku, které firma spravuje.

Název typu objektu	**Význam, popis**	**Barva objektu**
`ValueTypes`	Typ hodnoty zadávaného údaje	Oranžová
`Atoms`	Zadávaný údaj	Oranžová
`DefTexts`	Předdefinovaný text	Oranžová
`FormDefinitions`	Definice deníku	Oranžová
`FormDefAtoms`	Zadávané údaje v definici deníku (vazební)	Oranžová
`Shifts`	Směna	Oranžová
`Forms`	Deník	Oranžová
`Notes`	Poznámky k deníku	Oranžová
`Comments`	Komentáře k poznámkám	Oranžová
`Archiv textů`	Archiv textových hodnot	Zelená
`Archiv čísel`	Archiv číselných hodnot	Zelená
`FormActions`	Akce nad deníkem	Oranžová
`FormStates`	Stavy deníku	Oranžová
`UzivatelskeUcty`	Uživatel	Modrá
`RoleZaOsobu`	Role přidělená uživateli	Modrá

Tabulka 1 Popsané objekty v konceptuálním modelu, zdroj: vlastní zpracování

4.1.2.2 Logický návrh datového modelu

V logickém návrhu datového modelu, který je zobrazen na obrázku 13, jsou popsány jednotlivé vztahy mezi relacemi a přidány atributy jednotlivých relací. Jednotlivé relační atributy modelu jsou popsány níže v této kapitole. Použité značky jsou popsány na Obrázku 11 ER Model. Znaky v závorkách vedle určitých atributů znázorňují, že se jedná o Primární klíč (PK), primární klíče jsou popsány v kapitole 3.5.2, nebo o Cizí klíče (FK), které jsou popsány v kapitole 3.5.4. Jednotlivé atributy, které jsou přiřazeny ke každé relaci, budou popsány dále v této kapitole.



Obrázek 13 Logický návrh datového modelu

Relace ValueTypes, znázorněna v tabulce 2, se skládá z atributů ID, který značí identifikátor hodnoty, tento identifikátor je primárním klíčem této relace; Name, obsahující název typu hodnoty; Typebase, obsahující druh typu hodnoty (bitmap, date, enum, float, text nebo time). Atribut DefinitionData obsahuje definici datového typu ve fromátu JSON.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
Name	Název typu hodnoty
TypeBase	Druh typu hodnoty: • bitmap – n-tice 1 a 0 • date – datum • enum – hodnota z výčtu • float – číselná hodnota • text – textová hodnota • time – čas (bez data)
DefinitionData	Definice datového typu

Tabulka 2 ValueTypes – Typ hodnoty zadávaného údaje, zdroj: vlastní zpracování

Relace Atoms, znázorněna v tabulce 3, se skládá z atributů ID (Primární klíč), jméno údaje, typ hodnoty (cizí klíč) získaného z relace ValueTypes, vztah těchto relací je 1:M, jelikož jeden typ hodnoty může mít více zadávaných údajů. Dále je zde atribut ArchiveID, který obsahuje ID veličiny, jak je veličina nazvána v archivu firmy. Atribut Copyable označuje, jestli se bude zadaný údaj kopírovat do další směny. Atribut UpID odkazuje vztahem 1:M na údaj v nadřízeném deníku z relace Forms.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
Name	Název zadávaného údaje
ValueTypeId (FK)	Odkaz na typ hodnoty
ArchiveId (FK)	Odkaz do archivu schématu firmy
Copyable	Příznak kopírování do následující směny
UpID	Odkaz na údaj v nadřízeném deníku

Tabulka 3 Atoms – Zadávaný údaj, zdroj: vlastní zpracování

V relaci DefTexts, znázorněné v tabulce 4, je primárním klíčem atribut ID. Atribut DefinitionId sděluje, o který druh směnového deníku se jedná, tento atribut je cizí klíč, který odkazuje na relaci FormDefinition vztahem 1:M. Atribut FormKey je klíč určující místo, kde lze tento text použít (odpovídá odkazu v definici deníku). Value je textová hodnota, která se bude ukazovat. Poslední atribut OrderVal sděluje pořadí, ve kterém se text zobrazí (čím vyšší číslo, tím níže v seznamu bude). Předdefinované texty se vkládají do hodnot nebo poznámek v deníku jako text, nikoli jako odkaz.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
DefinitionId (FK)	Odkaz na definici deníku
FormKey	Místo, kde lze text použít
Value	Textová hodnota
OrderVal	Pořadí pro zobrazení

Tabulka 4 DefTexts – Předdefinovaný text, zdroj: vlastní zpracování

Relace FormDefinition, znázorněna v tabulce 5, se skládá z ID (Primární klíč), atributu Name, který nese název druhu deníku. Dále atributů DefinitionData, který nese definici grafického rozložení deníku; UpID nesoucí ID nadřazené definice v této relaci. AccessGroup určuje roli uživatele, DisplayOrder sděluje v jakém pořadí pod sebou se budou jednotlivé typy deníků zobrazovat. DisplayGroupDefinition slouží pro seskupování pro zobrazení. Atributy DisplayShortName a DisplayLongName nesou dlouhý a krátký název jednotlivých typů deníků.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
Name	Název definice (druhu deníku)
DefinitionData	Definice deníku
UpId	Odkaz na nadřízenou definici
AccessGroup	Určení role uživatele
DisplayOrder	Pořadí pro zobrazení
DisplayGroupDef	Seskupování pro zobrazení

Tabulka 5 FormDefinitions – Definice deníku, zdroj: vlastní zpracování

Relace Shifts, znázorněna v tabulce 6, se skládá z ID (Primární klíč), atributu Name, který nese název směny. Atribut Color nese barvu, která se bude u příslušné směny zobrazovat. Atributy StartDT a EndDT nesou čas začátku a konce směny v UTC (Koordinovaném světovém čase). Název a barva směny jsou dány rozpisem používaným v elektrárně, v programu Provozní deník slouží jen ke zobrazení.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
Name	Název směny
Color	Barva pro zobrazení
StartDT	Čas začátku směny (v UTC)
EndDT	Čas konce směny (v UTC)

Tabulka 6 Shifts – Směna – Definice deníku, zdroj: vlastní zpracování

Relace Forms, znázorněna v tabulce 7, se skládá z atributů ID (Primární klíč), DefinitionID (cizí klíč), který odkazuje na relaci FormDefinition ve vztahu 1:M a nese hodnotu druhu deníku. Atribut ShiftID (cizí klíč) odkazuje na relaci Shifts ve vztahu 1:M a nese hodnotu směny, operující v daném formuláři. Atributy OpenDT a CloseDT nesou časy založení a uzavření deníku. Atributy OpenUserID a CloseUserID (cizí klíče) nesou odkaz na uživatele, který deník založil a uživatele, který deník uzavřel. Odkaz se bere ve vztahu 1:M z relace UzivatelskeUcty, která je uložena v relačním schématu firmy (aplikace Provozní deník používá seznam uživatelů společný s ostatními aplikacemi). Atribut StateId (cizí klíč) nese odkaz na relaci FormStates a dává nám informaci o aktuálním stavu životního cyklu deníku, tedy jestli je otevřen, nebo uzavřen.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
DefinitionID (FK)	Odkaz na definici deníku
ShiftID	Odkaz na směnu
OpenDT	Čas založení deníku
CloseDT	Čas uzavření deníku
OpenUserID (FK)	Odkaz na uživatele, který deník založil
CloseUserID (FK)	Odkaz na uživatele, který deník uzavřel
StateId (FK)	Odkaz na aktuální stav

Tabulka 7 Forms – Deník, zdroj: vlastní zpracování

Relace Notes, znázorněna v tabulce 8, se skládá z atributů ID (Primární klíč); FunctionUnitName, který nese název části zařízení, kterého se konkrétní poznámka týká; Note obsahující text poznámky; Permament obsahuje hodnotu toho, zda bude poznámka kopírována i do další směny; Hidden obsahující hodnotu viditelnosti poznámky při vytisknutí; NoteDT nese čas poznámky. Atribut FormID (cizí klíč) odkazuje vztahem 1:M

na relaci Forms a přiřazuje poznámku k určitému deníku. UserID (cizí klíč) přiřazuje z tabulky UzivatelskeUcty vztahem 1:M uživatele, který poznámku napsal. OrigID vznikne, když je poznámka zkopírována z minulé směny a nese ID původní poznámky. DownID vzniká, když je poznámka zkopírována do nadřazeného deníku, a nese odkaz na původní poznámku. Při kopírování OrigID do více směn po sobě atribut odkazuje na původní (nejstarší) z těchto poznámek.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
FormID	Odkaz na deník
EventDT	Čas události
EventEndDT	Čas konce události
FunctionUnitName	Název části zařízení, kterého se poznámka týká
Note	Poznámka
Permanent	Příznak, že se poznámka kopíruje do následující směny
Hidden	Příznak, že se poznámka netiskne
NoteDT	Čas zadání poznámky
UserID (FK)	Odkaz na uživatele, který poznámku zadal
OrigID	Odkaz na původní poznámku při kopírování do následující směny
DownID	Odkaz na původní poznámku při kopírování do nadřízeného deníku

Tabulka 8 Notes – Poznámky k deníku, zdroj: vlastní zpracování

Relace Comments, znázorněna v tabulce 9, se skládá z atributů ID (Primární klíč); CommentText, obsahující text komentáře, a CommentDT, obsahující čas zadání komentáře. Atribut NoteID (cizí klíč) odkazuje na relaci Notes vztahem 1:M, slouží pro přiřazení komentáře k poznámce. UserID (cizí klíč) přiřazuje z tabulky UzivatelskeUcty vztahem 1:M uživatele, který komentář napsal.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
NoteID	Odkaz na poznámku
CommentText	Komentář
CommentDT	Čas zadání komentáře
UserID (FK)	Odkaz na uživatele, který komentář vytvořil

Tabulka 9 Comments – Komentáře k poznámkám, zdroj: vlastní zpracování

Relace FormActions, znázorněna v tabulce 10, se skládá z atributů ID (Primární klíč); Code, označující kód prováděné akce; Name nesoucí jméno akce. Atributy SourceStateID (cizí klíč) a TargetStateID (cizí klíč) odkazují vztahem 1:1 na relaci FormStates určují počáteční a konečný vztah formuláře. Atribut Meaning rozlišuje akci kódu v programu. Atribut Cancelled nese hodnotu, zdali je akce zneplatněna.

Název atributu	Význam
ID (PK)	Identifikátor hodnoty
Code	Kód akce
Name	Název akce
SourceStateID	Odkaz na výchozí stav deníku
TargetStateID	Odkaz na cílový stav deníku
Meaning	Rozlišení akce v kódu programu: • 1 - akce je druhu „založit“ • 3 - akce je druhu „uzavřít“
Cancelled	Akce je zneplatněna

Tabulka 10 FormActions – Akce nad deníkem, zdroj: vlastní zpracování

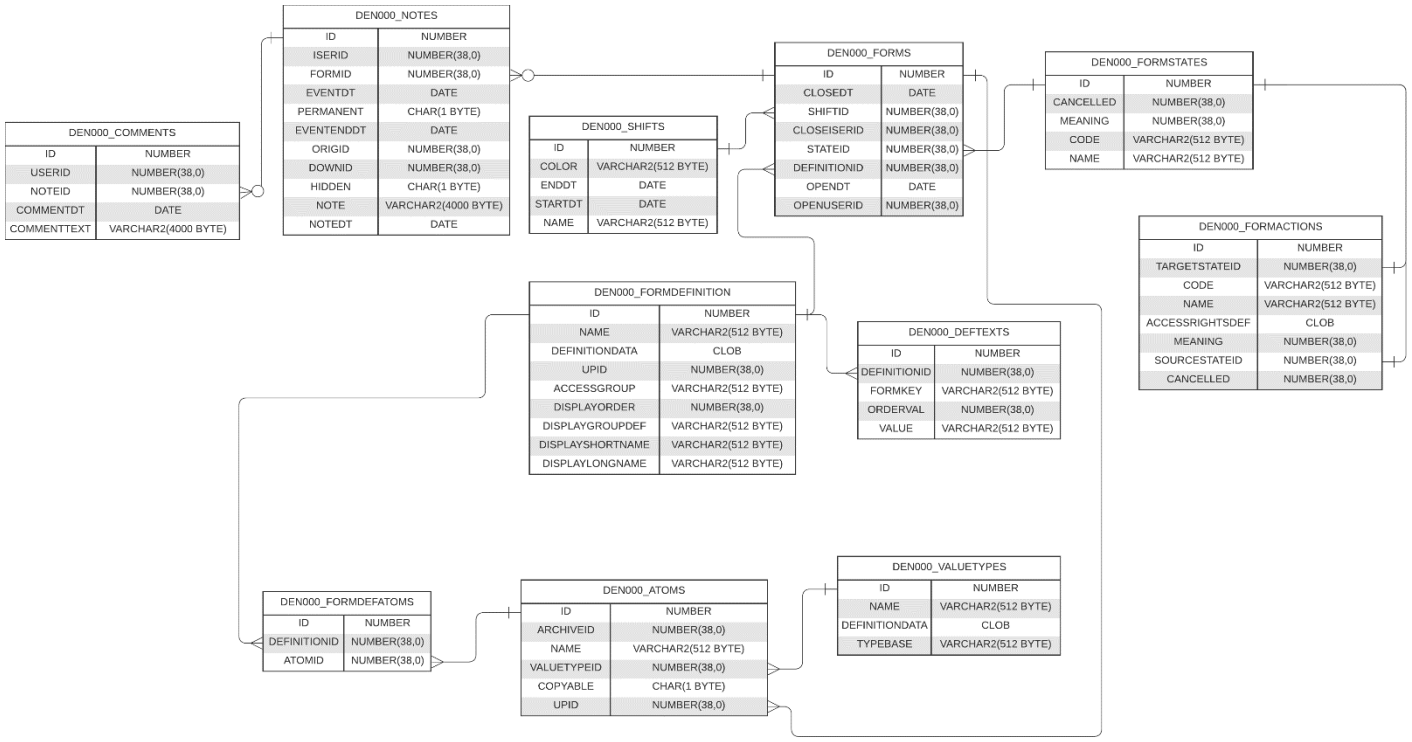
Relace FormStates, znázorněna v tabulce 11, se skládá z atributů ID (Primární klíč); Code, který sděluje, o jaký kód stavu se jedná; Name označující název stavu; Meaning, který rozlišuje stav kódu pro program. Atribut Cancelled nese hodnotu zneplatnění stavu.

Název atributu	Význam
ID	Identifikátor hodnoty
Code	Kód stavu
Name	Název stavu
Meaning	Rozlišení stavu v kódu programu: • 1 - stav je druhu „otevřen“ • 2 - stav je druhu „uzavřen“
Cancelled	Stav je zneplatněn

Tabulka 11 FormStates – Stavy deníku, zdroj: vlastní zpracování

4.1.2.3 Fyzický návrh datového modelu

Ve fyzickém návrhu je přiřazen ke každému atributu datový typ. Datové typy musí odpovídat datovým typům zvoleného SŘBD. V tomto případě musí jít o datové typy Oracle SŘBD. Datový typ Number říká, že daný atribut je číslo, pokud se vyskytuje ve formátu Nuber (X, 0), pak X označuje počet číslic, které může číslo mít. Pokud má atribut datový typ Date, pak se jedná o datum. Datový typ VarChar2(X Byte) slouží pro textové řetězce a X v tomto případě označuje nejvyšší možnou velikost tohoto řetězce. Char (1 Byte) značí, že se jedná o textový řetězec, který je dlouhý právě jeden byte. V tomto modelu je Char použit pro hodnoty, které nabírají T – pravdu, nebo F – Nepravdu. Fyzický návrh s přiřazenými datovými typy je popsán na obrázku 15. Datový typ CLOB (character large object) je použit k uchování velkorozměrových dat.



Obrázek 14 Fyzický návrh datového modelu, zdroj: vlastní zpracování

4.2 Komparace starého a nového řešení Provozní deníku

V této kapitole bude popsáno staré řešení provozního deníku. Vzhledem k tomu, že nebyl udělen souhlas na zveřejnění obrázků ze starého provozního deníku, bude řešení pouze obecně popsáno bez konkrétních dat.

4.2.1 Staré řešení provozního deníku

Původní řešení provozního deníku byl systém založený na sešitech Excelu, pří čemž každý provozní deník měl svůj vlastní sešit Excelu. Všechny informace byly uloženy na sdíleném disku, kde byly uloženy jak všechny sešity Excelu pro jednotlivé provozní deníky, tak data ze všech deníků, která sloužila pro vyplnění hodnot v jednotlivých sešitech Excelu. Pro přístup k jednotlivým sešitům Excelu a datům z deníků sloužil základní sešit, ve kterém si uživatel vybral, jestli chtěl vytvořit nový deník nebo prohlížet archiv deníku.

Po otevření nového deníku mohl uživatel doplňovat data přímo do sešitu v Excelu. Data byla doplňována pomocí prokliku do buňky, otevřelo se nové okno, kde bylo možno data doplnit.

4.2.2 Výhody původního řešení

Původní řešení, které fungovalo na bázi excelovských tabulek, mělo výhodu v tom, že provozní deník byl jednodušší na vývoj. K možnosti vylepšení provozního deníku stačilo mít znalost Excelu, včetně VBA (Programovací jazyk MS Office).

4.2.3 Nevýhody původního řešení

Hlavní nevýhodou starého řešení byla bezpečnost VBA i Excelu, jelikož se nejedná o nepřekonatelné zabezpečení. Další nevýhodou byl přístup pro zápis pouze pro jednoho uživatele a zároveň uživatel, který otevřel sešit pouze pro čtení (nemohl tam tedy zapisovat), neviděl již provedené změny uživatele, který měl v danou chvíli otevřený sešit a vkládal tam data. Každý uživatel, který chce mít přístup k provoznímu deníku, musí mít licenci MS Office, také v každé nové verzi Excelu se může stát, že určitá funkce již nebude podporována.

4.2.4 Výhody nového řešení

Hlavní výhodou nového řešení je simultánní přístup k databázi, umožněný relačním modelem databáze. Další výhodou je kybernetická bezpečnost, databáze mají kontrolu přístupu, mohou být přiřazeny určité role, které říkají, co daný uživatel s databází může udělat. Další výhodou je, bezpečnost a záloha dat, data jsou uložena na serveru, který je navržen tak, aby data byla bezpečně uložena a zálohována. V případě, že by celé diskové pole bylo zničeno, jsou pořád vytvářeny nové zálohy databáze. Přístup k Provoznímu deníku lze pomocí libovolného webového prohlížeče.

4.2.5 Nevýhody nového řešení

Aplikace je rozdělena mezi webový a databázový server, pokud tedy spadne jedna z těchto dvou částí, není přístup k deníku možný.

5 Výsledky a diskuse

V rámci studie byly zanalyzovány potřeby daného podniku ohledně vytvoření provozního deníku. Na základě těchto zjištění byl ve spolupráci s firmou navržen relační databázový model. Pro relační model byl zvolen SŘBD Oracle, jelikož firma má v této SŘBD již napojené další modely, které odkazují na nově vzniklý model pro aplikaci Provozní deník.
Po provedení komparace starého a nového způsobu řešení byly zjištěny výhody a nevýhody nového řešení. Největší výhodou nového řešení je možnost simultánního přístupu do

databáze, to v praxi znamená, že ve webovém prostředí může pomocí formuláře více lidí najednou přidávat, nebo měnit hodnoty. Staré řešení tuto možnost neumožňovalo, původní model byl navržen v prostředí MS Office, kde není simultánní přístup možný. Metodou komparace byla pak jako další nesporná výhoda navrhnutého řešení vyhodnocena výrazně silnější kybernetická bezpečnost. Ta je zajištěna díky přístupovým právům do databáze. V kontextu diskutovaných výhod nového modelu se sluší zmínit také slabou stránku, která byla detekována. Selže-li jedna ze dvou částí aplikace, nebude přístup k datům možný.

6 Závěr

Autor se v rámci práce zabývá problematikou sestavení relačního modelu pro potřeby zaměstnanců dané elektrárny. V úvodu práce byl představen tematický obsah práce, kterým je teorie databází.

Prvním dílčím cíle bylo představit historii vývoje databází a dotazovacího jazyka SQL. Tento cíl byl naplněn v rámci kapitol Historie Databází a Historie SQL. Autor v rámci těchto kapitol popsal zrod a vývoj databází a s nimi spojeného jazyka SQL.

Druhým dílčím cílem bylo představení relačního databázového modelu. Tohoto dílčího cíle bylo naplněno v rámci kapitol Relační databáze. Autor se zde zaměřil na popsání fungování relační databáze a její zákonitosti, popsání vazeb mezi jednotlivými relacemi, jak dosáhnout co nejmenšího opakování dat v relačním modelu a v neposlední řadě, jakými 12 pravidly by se každý relační model měl řídit.

Posledním dílčím cílem bylo provedení metody komparace na stávající a nové řešení provozního deníku. Tento dílčí cíl byl naplněn v kapitole Komparace starého a nového řešení Provozní deníku. Autor zde porovnal staré a nové řešení deníku, jejich výhody a nevýhody.

Hlavního cíle práce Sestavení relačního modelu pro aplikaci Provozní deník bylo dosaženo v kapitole Návrh relačního modelu pro provozní deník. V rámci kapitol Stanovení zadání a Požadavky zadavatele představil autor požadavky a zadání od zadavatele. Dále byl autorem v kapitolách Konceptuální návrh datového modelu Logický návrh datového modelu a Fyzický návrh datového modelu sestaven relační datový model.

Hlavním přínosem předložené bakalářské práce je návrh relačního modelu pro aplikaci Provozní deník. Tento model výrazným způsobem zmodernizuje a zefektivní pracoviště elektrárny. Díky možnosti simultánního přístupu více pracovníků do databáze, bude výrazným způsobem zkrácen čas, který by této činnosti musel každý pracovník věnovat v případě původního modelu. Starý model neumožňoval nahlédnout jednomu pracovníkovi do databáze ve chvíli, kdy do ní jiný pracovník zadával potřebná data. Nejen, že tak mohlo dojít ke zpomalení toku informací, ale ve chvíli, kdy by se jeden pracovník z databáze zapomněl odhlásit, mohlo dojít k zablokování vkládání dat jiným pracovníkem úplně. Přínosem navrhnutého modelu je jistě i možnost dohledání zdroje dat, respektive dohledání pracovníka, který do databáze data dodal. V případě nejasností, nepřesností či řešení problémů je pak snadné dohledat zodpovědného člověka. S tímto pak částečně souvisí i možnost uzavření deníku, který může být znovu otevřen pouze za specifických podmínek.

Je tedy relativně obtížné zpětně svévolně manipulovat s daty. V dnešní nejisté době je jistě žádoucí, chránit elektrárny vůči kybernetickým útokům. Je tedy nesporným přínosem této práce navrhnutí modelu, který kybernetickou ochranu elektráren přímo podporuje. Výhodou předkládaného modelu je tedy vedle modernizace a zefektivnění práce, také vysoká míra transparentnosti, kterou tento model umožňuje. Schopnost ochrany vůči kybernetickým hrozbám je pak vedle zmíněných kvalit další neopominutelnou výhodou tohoto modelu. Dle názoru autora, výhody vysoce převyšují nevýhody vytvořeného modelu. Mezi další přínosy práce pak patří ucelený přehled historie databázových systémů a jazyka SQL, které mohou v daném rozsahu fungovat jako studijní materiály k dané problematice, popřípadě sloužit jako podklad pro výběr vhodného databázového systému pro daný projekt.

Touto bakalářskou prací autorův zájem o diskutovanou problematiku nekončí. Autor bude i nadále v kontaktu s pracovníky elektrárny, se kterými bude diskutovat případné nedostatky a přizpůsobovat relační model požadavkům zadavatele.

Závěrem. Diskutovaný model byl již převeden do praxe, pracovníci elektrárny přechod ze starého na nový model uvítali a snadno si práci s ním osvojili. Tímto se tedy autor domnívá, že zadání zadavatele, a tedy i zadání a cíle této bakalářské práce byly naplněny.

7 Seznam použitých zdrojů

1. CONNOLLY, Thomas M., 2010. Database systems: a practical approach to design, implementation, and management. 5th. London: Addison-Wesley. ISBN 978-0321523068.
2. GARCIA-MOLINA, Hector, Jeffrey D. ULLMAN a Jennifer WIDOM, 2013. Database Systems: The Complete Book. 2nd Edition. London: Pearson. ISBN 978-1292024479.
3. GROFF, James R. a Paul N. WEINBERG, 1999. SQL: The Complete Reference. 2nd Edition. Berkeley: McGraw-Hill Osborne Media. ISBN 0072118458.
4. WADE, Bradford W. a Donald D. CHAMBERLIN, 2012. IBM Relational Database Systems: The Early Years. IEEE Annals of the History of Computing. IEEE, 2012(34), 38–48. ISSN 1934-1547. Dostupné z: doi:10.1109/MAHC.2012.48
5. KRIEGEL, Alex a Boris M. TRUKHNOV, 2008. SQL bible. 2nd ed. Hoboken, N.J.: Wiley. ISBN 978-047-0229-064.
6. DARIE, Christian a Watson KARLI, 2003. The programmer's guide to SQL. New York: APress Media. ISBN 978-1-4302-0800-6.
7. DIN, Akeel I., 2014. Structured Query Language (SQL): a Practical Introduction. Hoboken: Blackwell Pub. ISBN 978-1855543577.
8. HIBATULLAH, Alzahrani, 2016. Evolution of Object-Oriented Database Systems. Global Journal of Computer Science and Technology. Global Journals, 2016(16), 37-40. ISSN 0975-4172.
9. BUTUNER, Hakan, 2012. Advantages Of Object-Oriented Over Relational Databases On Real-Life Applications. A survey and comparison of relational and non-relational database. Industrial Management and Engineering Co., 2012(5), 1-5. ISSN 2045-3345.
10. JINDAL, Guarav a Simmi BALI, 2017. Hierarchical Model Leads To the Evolution of Relational Model. International Journal of Engineering and Management Research (IJEMR). Website Developer, Asiawebnet Pvt, 2012(2), 11-14. ISSN 2250-0758.
11. PIATTINI, Mario, Coral CALERO a Hakim SAHRAOUI, 2001. Object-relational database metrics. Departament of Computer Science University of Castilla-La Mancha Ronda Calatrava, 2001, 1-21.
12. HERMANDEZ, Michael, 2013. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design. 3rd Edition. London: Addison-Wesley Professional. ISBN 978-0321884497.
13. HARRINGTON, Jan L., 2009. Relational Database Design and Implementation: Clearly Explained. 3rd Edition. Burlington: Morgan Kaufmann. ISBN 978-0123747303.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam použitých zkratek

SQL	Structured Query Language
PK	Primary Key
FK	Foreign Key
DBMS	Database Management System
OODMBS	Objected Oriented Database Management System
SŘBD	Systém Řízení Báze Dat
RDBMS	Relational Database Management System

ID	Identifier
DML	Data Manipulation Language
DDL	Data Definition Language
DCL	Data Control Language
TLC	Transaction Control Languages
ER	Entity Relationship
ERD	Entity Relationship Diagram
NF	Normální Forma
BCNF	Boyceho–Coddova Normální Forma
DB2	Database2
ANSI	American National Standards Institute
ISO	International Organization for Standardization