VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

OPTIMALIZACE SQL DATABÁZE PRO VÝROBNÍ SPOLEČNOST
OPTIMIZATION OF SQL DATABASE FOR MANUFACTURING COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Lukáš Janča**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jiří Kříž, Ph.D.**
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Janča Lukáš

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Optimalizace SQL databáze pro výrobní společnost

v anglickém jazyce:

Optimization of SQL Database for Manufacturing Company

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK.Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. 1.vyd. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.
KROENKE, David a David J AUER. Databáze. 1. vyd. Brno: Computer Press, 2015. 496 s. ISBN 978-80-251-4352-0.
KOCH, Miloš a Bernard NEUWIRTH. Datové a funkční modelování. 4. vyd. Brno: Akademické nakladatelství CERM, 2010. 142 s. ISBN 978-80-214-4125-5.
KŘÍŽ, Jiří a Petr DOSTÁL. Databázové systémy. 1.vyd. Brno: Akademické nakladatelství CERM, 2005. 111 s. ISBN 80-214-3064-8.

Vedoucí bakalářské práce: Ing. Jiří Kříž, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc. Ředitel ústavu	doc. Ing. et Ing. Stanislav Škapa, Ph.D. Děkan fakulty

V Brně, dne 29.2.2016

Abstrakt

Tato bakalářská práce pojednává o problematice úprav databáze pro výrobní společnost, jejíž stávající databáze je v nevyhovujícím stavu. Práce je rozdělena na 3 části. První část se zabývá teoretickým pozadím problému. Ve druhé části se nachází analýza současného stavu databáze. Ve třetí části je navrženo vlastní řešení problému.

Abstract

This bachelor thesis deals with issue of modification of database for manufacturing company, which is in inadequate state. The thesis is divided into 3 parts. First part is focused on theoretical background of the problem. The second part is to analyze the current situation of database. In the third part is my own solution of this issue.

Klíčová slova

Databáze, SQL, relace, normalizace, datový model, ER diagram

Key words

Database, SQL, relation, normalization, data model, ER diagram

Bibliografická citace

JANČA, L. *Optimalizace SQL databáze pro výrobní společnost.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 49 s. Vedoucí bakalářské práce Ing. Jiří Kříž, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb. o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 30. 5. 2016 …….

Podpis studenta

Poděkování

Rád bych tímto poděkoval vedoucímu bakalářské práce Ing. Jiřímu Křížovi, Ph.D. za pomoc, konzultace a užitečné rady, které mi při psaní této bakalářské práce poskytl.

Obsah

Úvod

Ukládání dat a následná práce s nimi je v dnešní době jednou z výzev, které musí společnost čelit. Každá společnost eviduje různé množství dat a existuje tedy i mnoho způsobů, jak se s tímto problémem vypořádat. Efektivním způsobem je ukládání dat do databází.

Význam této bakalářské práce spočívá v úpravách databáze pro výrobní společnost, jejíž současné řešení již plně neodpovídá ukládanému množství dat a nárokům, které jsou na práci s těmito daty kladeny. Vhodné úpravy a optimalizace této databáze usnadní a zefektivní práci mnoha zaměstnancům společnosti.

První část bakalářské práce bude obsahovat teoretická východiska, vysvětlující základní pojmy z oblasti problematiky databázových systémů, relačních modelů a jazyka SQL.

Další část práce se bude zabývat vlastní analýzou databázového systému společnosti. Tato analýza bude obsahovat nejen informace o databázi, ale i základní informace o společnosti, výrobním procesu, současném přístupu k úpravám databáze a používaných aplikacích.

Ve třetí části bude již samotný návrh řešení. Nejprve je nutné zohlednit analýzu současného stavu a požadavky společnosti. Je také potřebné eliminovat nedostatky předchozího řešení, vypracovat logický a konceptuální návrh, podle kterého navrhnu již samotný fyzický návrh databáze.

Cíle práce

Cílem práce je navrhnout vhodné změny v databázi, do které jsou ukládána data v průběhu celého výrobního procesu. Výsledkem práce tak bude návrh nové databáze pro ukládání výrobních dat. Abych toho docílil, je nutní nejprve vykonat analýzu současného databázového systému a odhalit jeho slabiny a chyby.

Po zpracování analýzy současného stavu navrhnu změny struktury databáze, úpravu tabulek a optimalizaci vztahů mezi nimi tak, aby databáze vyhovovala požadavkům společnosti.

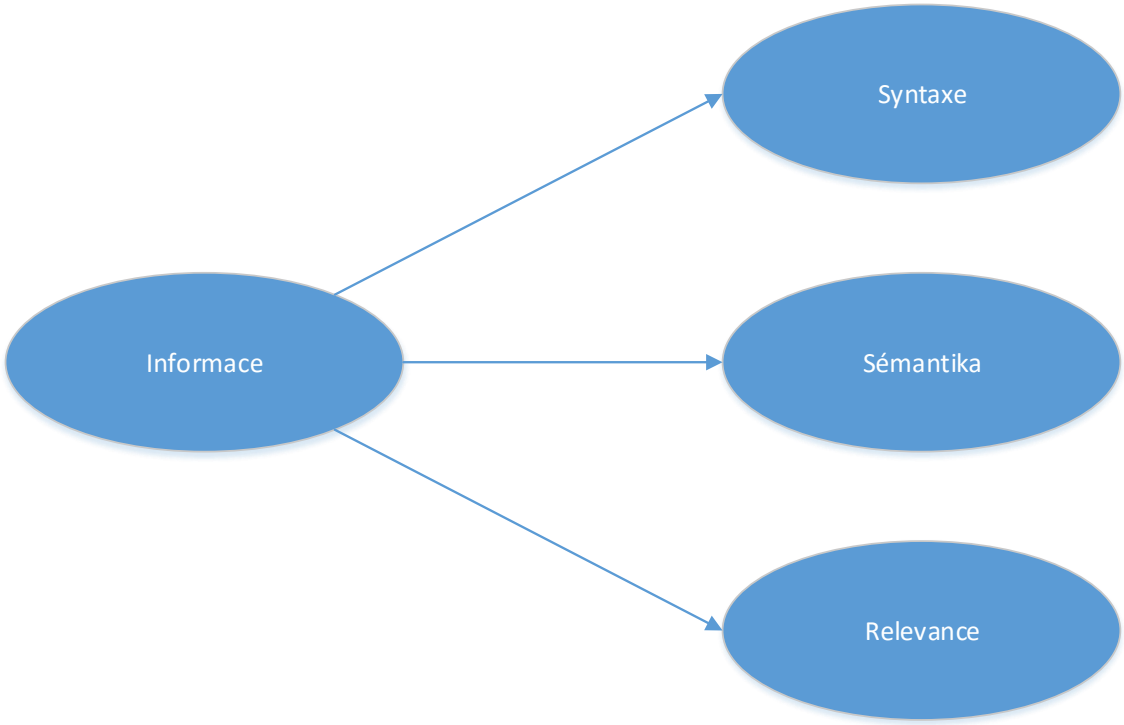
1 Teoretická východiska řešení

V této části se budu věnovat nezbytné teorii, kterou je potřebné znát k dosáhnutí zvoleného cíle.

1.1 Základní pojmy

1.1.1 Informace

Informací se rozumí zpráva nebo vjem, který zároveň musí splňovat 3 požadavky. Tím prvním je syntaktická relevance. Tím se rozumí, že subjekt, který zprávu přijímá, ji musí být schopen detekovat a rozumět jí. Druhým požadavkem je sémantická relevance. Subjekt tedy musí vědět, co daná zpráva znamená a co o něm a jeho okolí vypovídá. Třetím požadavkem je pragmatická relevance, a tedy, že zpráva musí mít pro subjekt význam (1).

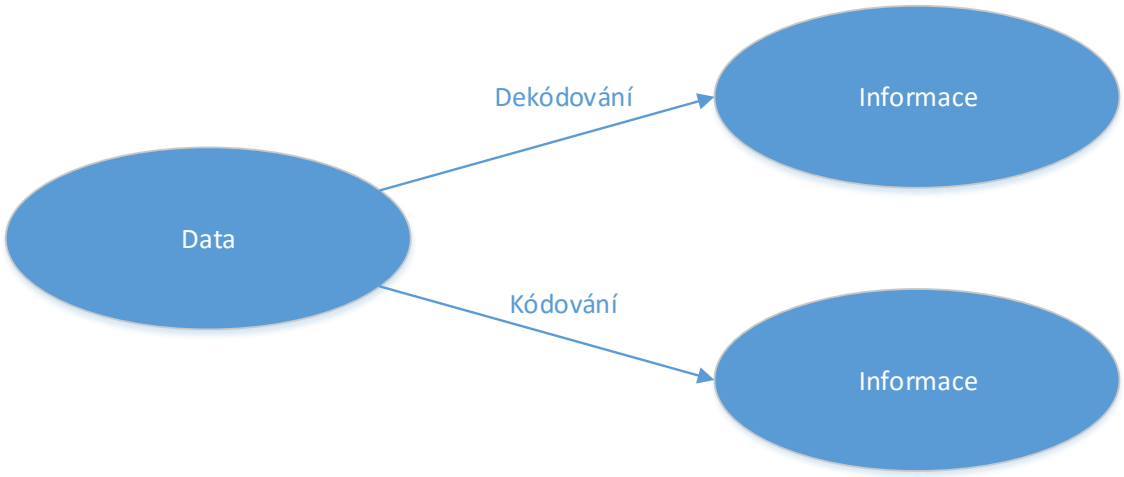


Obrázek 1: Informace (1, str. 4)

1.1.2 Data

Data jsou pro subjekt zprávy, které zachytí a porozumí jim. Ve chvíli, kdy člověk data používá k rozhodování, čímž jim přiřazuje význam a smysl, stávají se pro něj informací. Data tedy jsou potencionálními informacemi.

Data můžeme ukládat pro pozdější zpracování, například je tedy zaznamenat na papír nebo do počítače. Tímto zapsáním na vhodné fyzické medium se z informací stávají data a při jejich přečtení (dekódování) se z nich opět stanou informace pro daného příjemce (1).



Obrázek 2: Data (1, str. 5)

1.2 Databáze

Databáze slouží k evidování a shromažďování informací a v současné době je na nich postavena celá moderní společnost. Setkáváme se s nimi v běžném životě, používají se například pro evidence občanů, ve zdravotnictví, ve školství, nebo i evidence množství produktů na skladě v supermarketu (2).

1.2.1 Relační databáze

Základním pojmem, který je potřebné u relačních databází vysvětlit, je relace. Tu si můžeme představit jako tabulku skládající se ze sloupců a řádků. Jednotlivé sloupce odpovídají vlastnostem (atributům) prvku reálného světa (entity). Řádky představují tyto entity. Pro lepší ilustraci si to můžeme ukázat na následující tabulce (2):

Tabulka 1: Relační databáze (Vlastní)

Cislo_Studenta	Jmeno	Prijmeni	Obor	Rocnik
1	Petr	Stejskal	Manažerská Informatika	2
2	Tomáš	Prchal	Informační Management	1

Tato tabulka zobrazuje základní informace o studentech VUT. Relací je v tomto případě celá tabulka, entitami jsou informace o každém studentovi (řádky) a atributy jsou sloupce tabulky, tedy číslo studenta, jméno, příjmení, obor a ročník.

1.2.2 Pohled

Pohled se chová i vypadá jako tabulka, nicméně v pohledech se neuchovávají žádná data. Jsou v něm uchovávány definice dotazů nad jednou nebo více databázovými tabulkami a zobrazuje data z těchto tabulek. Bývá také často označován jako virtuální tabulka (3).

1.3 Integrita relačního modelu

Integrita relačního modelu je stav, kdy data uložená v modelu odpovídají vlastnostem reálného světa. Tyto integritní omezení dělíme na integritní omezení pro entity (relace) a integritní omezení pro vztahy entit (relační vazby) (1).

Integritní omezení pro entity dále dělíme na:

- Doménová integrita (integrita hodnot)
    - Zajištění, aby uvedený údaj odpovídal množině definovaných přípustných hodnot
- Entitní integrita
    - Jednoznačná identifikace každého řádku relace – použití primárního klíče
- Referenční integrita
    - Hodnoty cizího klíče nemohou nabývat hodnot v rozporu s hodnotami odkazovaného primárního klíče a závisejí na jeho hodnotách, představují tedy záruku konzistence a přesnosti dat v databázích (3), (6).

1.3.1 Kandidátní klíč

Kandidátním klíčem se rozumí pole či kombinace polí, kterou můžeme jednoznačně identifikovat určitý záznam. Jeho hodnota nesmí být nulová a musí být jedinečná (5).

1.3.2 Primární klíč

Primární klíč je typ kandidátního klíče, který byl určen k identifikování jedinečných záznamů v celé databázové struktuře. Po vytvoření primárního klíče se zbývající kandidátní klíče označí jako alternativní klíče (5).

Každá relace musí mít určen primární klíč. Jsou 3 možnosti vytvoření primárního klíče. Můžeme použít existující sloupec v tabulce, který obsahuje jedinečný záznam. Druhou možností je použití více sloupců, jejichž kombinací vznikne jedinečný záznam. Pokud ani to není možné, vytvoříme nový sloupec, který slouží pouze k přesné identifikaci záznamu (3).

1.3.3 Cizí klíč

Cizí klíč je sloupec či skupina sloupců, které se odkazují na primární klíč v jiné tabulce. Společně s tímto primárním klíčem tedy umožňuje vytváření spojení mezi relacemi. Hodnota cizího klíče musí být také identická s hodnotou primárního klíče (1), (6).

1.3.4 Vztah 1:1

Tento vztah nám označuje případ, kdy jedna n-tice relace odpovídá jedné n-tice jiné relace. Příkladem může být vztah mezi tabulkami „člověk“ a „občanský průkaz“. Jeden člověk může mít pouze jeden občanský průkaz, a zároveň jeden občanský průkaz může být vlastněn pouze jednou osobou (1).

Obrázek 3: Vazba 1:1 (Vlastní)

1.3.5 Vztah 1:N

Tento vztah zobrazuje stav, kdy jedné větě (n-tici relace) odpovídá jedna nebo více vět jiné relace. Příkladem je vztah tabulky člověk a tabulky automobil. Jeden člověk může vlastnit několik automobilů, ale jeden automobil může být vlastněn pouze jedním člověkem. Stejným stavem je vztah N:1, pouze je na něj nahlíženo z jiné perspektivy (1).

Obrázek 4: Vazba 1:N (Vlastní)

1.3.6 Vztah N:M

Ve vztahu N:M může odpovídat několik n-tic jedné relace více n-ticím jiné relace. Příkladem je například vztahu mezi tabulkou člověk a činnost. Jeden člověk může vykonávat několik pracovních činností, a každou činnost může zároveň vykonávat více lidí (1).

Obrázek 5: Vazba N:M (Vlastní)

1.4 Jazyk SQL

Jazyk SQL (Structured Query Language) je strukturovaný dotazovací jazyk, který byl původně navržený pro komunikaci se systémy řízení báze dat (SŘBD) založenými na relačním modelu. Byl vyvinut na konci sedmdesátých let dvacátého století v laboratoři společnosti IBM v San Jose v Kalifornii pod názvem SEQUEL pro produkt společnosti IBM s názvem DB2, což je relační databázový systém (2), (3).

Hlavním účelem jazyka SQL je přístup k databázi pomocí interaktivních dotazů. Pomocí jazyka SQL může programátor nebo správce databáze provádět následující:

- Upravovat strukturu databáze
- Měnit nastavení zabezpečení systému
- Přidávat uživatelská oprávnění k databázím či tabulkám
- Dotázat se databáze na nějakou informaci
- Aktualizovat obsah databáze (2), (3)

1.4.1 Základní datové typy

- **INTEGER**
  - Základní datový typ pro celá čísla (rozsah -2 147 483 648 až 2 147 483 647 (-2^{31} až $2^{31}-1$))
- **SMALLINT**
  - Datový typ pro celá čísla s menším rozsahem (-32 768 až 32 767 (-2^{15} až $2^{15}-1$))
- **NUMERIC(m,n)**
  - Slouží k uložení desetinných čísel s určenou přesností (m) a měřítkem (n)
- **FLOAT(n)**
  - Datový typ pro operace s plovoucí desetinnou čárkou, ovlivněný hodnotou parametru (n)
- **CHAR(n)**
  - Slouží k ukládání řetězců s přibližnou konstantní délkou (n)
- **VARCHAR(n)**
  - Na rozdíl od datového typu CHAR zabírá v paměti aktuální velikost podle uložené délky dat
- **DATE**
  - Do datového typu DATE se ukládá datum v rozsahu 0001-01-01 až 9999-12-31 (YYYY-MM-DD)
- **TIME**
  - TIME reprezentuje ukládání času ve 24h formátu v rozsahu 00:00:00.0000000 až 23:59:59.9999999 (hh:mm:ss[.nnnnnnn]) (11)

1.4.2 Příkaz CREATE TABLE

Základním příkazem, který se používá pro vytvoření tabulky je CREATE TABLE. Ten se využívá pro vytváření tabulek v databázi. Za příkazem CREATE TABLE uvedeme jméno tabulky, které by nemělo obsahovat diakritiku nebo mezery. Do závorky za jméno tabulky poté zadáváme jména jednotlivých sloupců, jejich datový typ a integritní omezení (velikost) (2).

```
CREATE TABLE Jmeno_Tabulky (
Jmeno_Sloupce1 Datovy_Typ(Integritni_Omezeni),
Jmeno_Sloupce2 Datovy_Typ(Integritni_Omezeni),
Jmeno_Sloupce3 Datovy_Typ(Integritni_Omezeni),
)
```

1.4.3 Příkaz DROP TABLE

Příkaz k mazání tabulek je jednoduchý, musíme ovšem vždy uvážit, zda danou tabulku a veškerá její data doopravdy chceme smazat. Samotný příkaz poté obsahuje pouze klíčová slova DROP TABLE a název tabulky (2).

```
DROP TABLE Jmeno_Tabulky
```

1.4.4 Příkaz ALTER TABLE

Příkaz ALTER TABLE slouží k úpravám již vytvořené tabulky. Používá se v případě, kdy máme v tabulce již uložena data, a tedy pro nás není jednodušší tabulku smazat a znovu vytvořit. Pomocí tohoto příkazu do tabulky můžeme přidávat sloupec, upravovat vlastnosti sloupce, mazat sloupec nebo přejmenovat sloupec (2).

Přidání sloupců do tabulky

Po klíčových slovech ALTER TABLE a jménu tabulky následuje příkaz ADD, za který se do závorky jména a vlastnosti sloupců se stejnými pravidly, jako když vytváříme tabulku (2).

```
ALTER TABLE Jmeno_Tabulky
ADD (
Jmeno_Sloupce1 Datovy_Typ(Integritni_Omezeni),
Jmeno_Sloupce2 Datovy_Typ(Integritni_Omezeni)
)
```

Úprava vlastností sloupců

V případě úpravy vlastností sloupců se postupuje stejným způsobem, jako při přidávání nových sloupců, jediným rozdílem je nahrazení příkazu ADD příkazem MODIFY. Je nutné si dávat pozor na to, jaká data máme v upravovaných tabulkách, pokud bychom například měnili maximální počet znaků ve sloupci na menší, mohli bychom o některá data přijít (2).

```
ALTER TABLE Jmeno_Tabulky
MODIFY (
Jmeno_Sloupce Datovy_Typ(Integritni_Omezeni)
)
```

Mazání sloupců

Pokud uvážíme, že některý ze sloupců již v tabulce nepotřebujeme, můžeme ho pomocí příkazu ALTER TABLE smazat. Postupujeme stejně jako v předchozích případech, pouze za jménem tabulky použijeme klíčové slovo DROP, za kterým následuje jméno sloupce, který chceme smazat. Pokud chceme smazat i sloupce v jiných tabulkách, které se na tento sloupec odkazovaly, můžeme za jméno sloupce doplnit klíčové slovo CASCADE (2).

```
ALTER TABLE Jmeno_Tabulky
DROP Jmeno_Sloupce CASCADE
```

Přejmenování sloupce

Pro přejmenování sloupce neexistuje v SQL konkrétní příkaz, postupujeme tedy tak, že vytvoříme sloupec s novým názvem, do něj zkopírujeme hodnoty ze starého sloupce a ten poté smažeme (2).

```
ALTER TABLE Jmeno_Tabulky
ADD (
Novy_Sloupec Datovy_Typ(Integritni_Omezeni),
)

UPDATE Jmeno_Tabulky
SET Novy_Sloupec = Stary_Sloupec

ALTER TABLE Jmeno_Tabulky
DROP Stary_Sloupec
```

1.4.5 Příkaz INSERT

Příkaz INSERT INTO se používá pro vkládání dat do tabulek databáze. Tento příkaz slouží pouze pro případ, kdy vkládáme pouze jeden řádek hodnot do tabulky. Za klíčovými slovy INSERT INTO tedy uvedeme jméno již vytvořené tabulky, do které vkládáme hodnoty. Jména sloupců uvádíme do závorky pouze v tom případě, kdy nevkládáme data do všech sloupců, nebo chceme data ukládat v jiném pořadí, než jak je vytvořena tabulka. Poté následuje příkaz VALUES, za který do závorky uvádíme hodnoty, které do tabulky vkládáme (2).

```
INSERT INTO Jmeno_Tabulky(Jmena_Sloupcu)
VALUES (Hodnoty)
```

1.4.6 Příkaz DELETE

Příkaz DELETE slouží k mazání řádků z tabulky. Za příkaz DELETE pouze doplníme název tabulky, poté následuje nepovinná část WHERE, za kterou pokud doplníme nějakou podmínku, smažou se pouze řádky splňující danou podmínku. Pokud tuto část vynecháme, smažou se všechny řádky v uvedené tabulce (2).

```
DELETE FROM Jmeno_Tabulky
WHERE Podminka
```

1.4.7 Příkaz UPDATE

UPDATE se používá pro editaci již existujících záznamů v tabulce. Po příkazu UPDATE a názvu tabulky následuje klíčové slovo SET, poté název upravovaného sloupce a jeho nová hodnota. Pokud vynecháme část WHERE, nastaví se nám na novou hodnotu všechna data v uvedeném sloupci. Při použití části WHERE a vhodné podmínky můžeme upravit pouze některé řádky v tabulce (2).

```
UPDATE Jmeno_Tabulky
SET Sloupec = Hodnota
WHERE Podminka
```

1.4.8 Příkaz SELECT

Příkaz SELECT je základním příkazem v SQL. Slouží k vybírání a zobrazování dat z tabulky, jeho základní struktura je následující. Za příkaz SELECT uvedeme názvy sloupců, které chceme zobrazit. Poté následuje klíčové slovo FROM a za ním název tabulky, ze které zobrazujeme data. Část WHERE je opět nepovinná a používá se pouze v případech, kdy chceme zobrazit data, která splňují určitou podmínku (2).

```
SELECT Sloupec1, Sloupec2
FROM Tabulka
WHERE Podminka
);
```

1.4.9 Příkaz CREATE VIEW

View (pohled) je virtuální tabulka, nejdříve určíme její jméno a poté následuje klasický select (dotaz). Po vytvoření pohledu nad ním můžeme provádět stejné příkazy, jako u tabulek (SELECT, INSERT, UPDATE, DELETE, DROP) (2).

```
CREATE VIEW Nazev_Pohledu AS
SELECT Sloupec1, Sloupec2
FROM Tabulka
WHERE Podminka
);
```

2 Analýza současného stavu

V rámci mé práce jsem se zaměřil na analýzu stavu databáze výrobní společnosti, konkrétně na její pobočku v Brně. Databáze je zde součástí výrobního procesu od příjmu zakázky až po její odeslání k zákazníkovi, je tedy nutné, aby vždy fungovala bezchybně a pro zaměstnance nebylo příliš složité s jejími daty pracovat. Databáze ale v současné chvíli pro tuto práci není dostatečně optimalizovaná.

2.1 Informace o společnosti

Tato společnost, s centrálou na Taiwanu, se ve svém závodě v městské části Brno-Černovice zabývá výrobou serverů. Z celkového počtu více než 60000 zaměstnanců po celém světě jich v této pobočce pracuje přibližně 150. Původně se jednalo o součást společnosti Acer, ale v roce 2000 došlo k oddělení a vytvoření samostatné společnosti. V Brně se dříve zabývala výrobou stolních počítačů, serverů, LCD monitorů a televizorů a zaměstnávala až 2500 lidí, v posledních letech ovšem došlo k omezení výroby a propouštění a v současné době se společnost soustředí na výrobu serverů a datových úložišť pro významné společnosti působící na IT trhu.

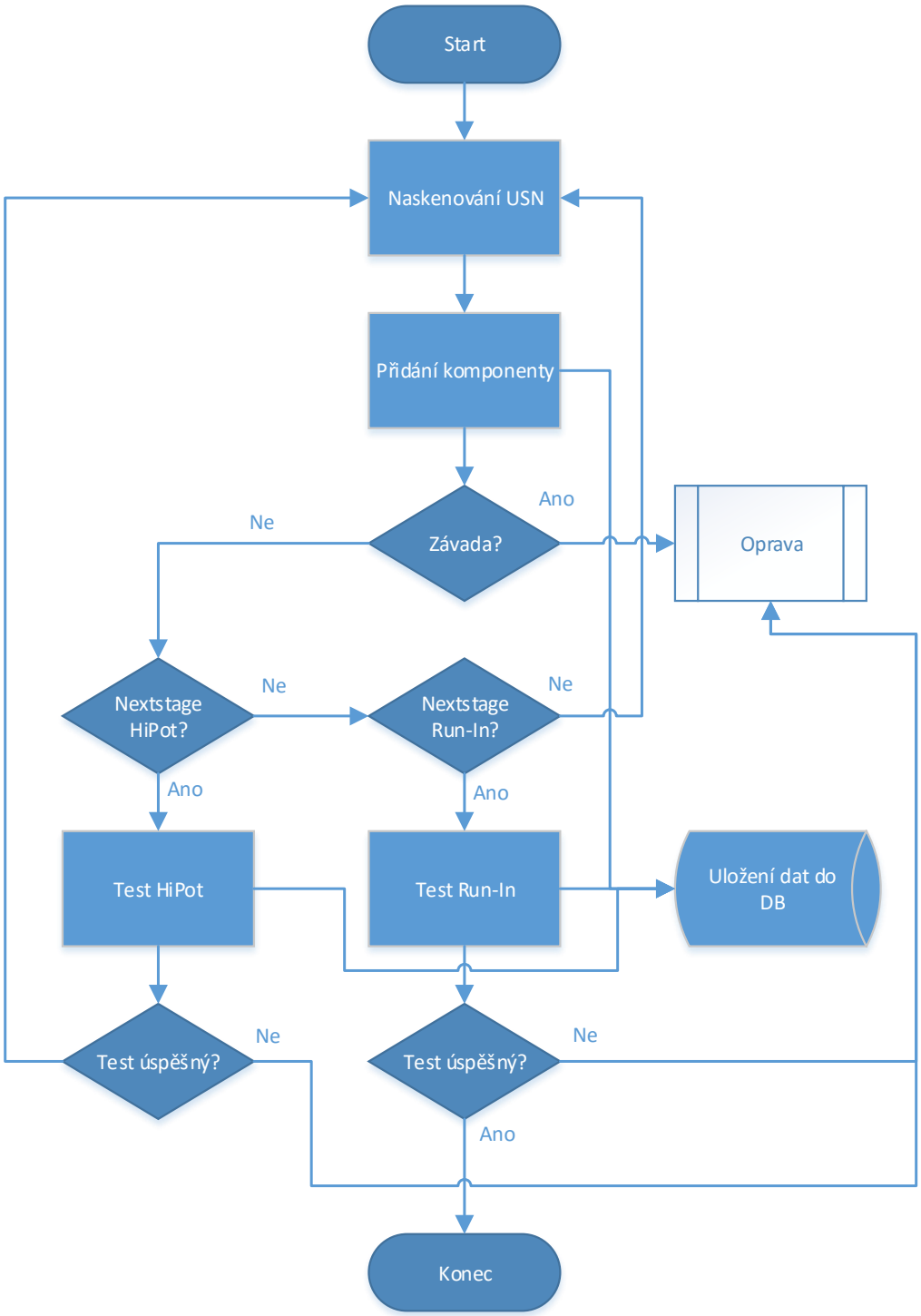
2.2 Výrobní proces

Ve výrobní hale je v současné době jedna hlavní výrobní linka a jedna menší. Na každé z nich probíhá samotné sestavování, testování i balení odděleně. Záznamy ze všech stanic se ukládají do databáze, aby se s nimi mohlo později pracovat, a také pro případnou kontrolu.

Ještě před samotnou výrobou dochází k příjmu objednávky. Objednávky jsou rozděleny do dvou kategorií – BTO (Basic Order – standardní sériově vyráběné modely) a CTO (Custom Order – zákazník si sám určí přesnou specifikaci jednotky). V objednávce je vždy uveden počet kusů, typ objednávky (BTO nebo CTO) a modelová řada (Model Family – MF), nebo v případě individuální objednávky konkrétní osazení komponent a příslušenství. Podle těchto objednávek je ze skladu vychystáván materiál a do systému jsou tyto konfigurace nahrávány.

Na každé výrobní stanici je umístěn počítač, na kterém běží aplikace přizpůsobená pro činnost, která je zde prováděna a operátor pomocí skeneru načítá čárové kódy, které reprezentují sériová čísla vyráběných jednotek nebo používaných komponent. Veškeré záznamy o skenování se ukládají do transakční tabulky.

Dále se zde nachází oddělená část, kde se vyrábí specializovaná datová úložiště. Tato výroba je poměrně nová, odlišná od výroby standardních serverů a teprve se zavádí, proto se touto částí nebudu zabývat.



Obrázek 6: Výrobní proces (Vlastní)

2.2.1 Montáž

Při montáži (Assembly) musí operátor pomocí skeneru načíst sériové číslo vyráběné jednotky a na monitoru se mu zobrazí grafická nápověda s umístěním a číslem komponenty, kterou tam má namontovat. Zde probíhá kontrola, zda operátor používá správnou komponentu. Každá objednávka a každá modelová řada má jinou konfiguraci a dávají se do ní jiné komponenty, proto je tento návod pro operátory a jejich následná kontrola důležitou součástí výroby. Pokud se i přes tuto kontrolu stane chyba a přijde se na ni až dodatečně, tak následující kroky záleží na povaze problému. V případě osazení špatné komponenty putuje jednotka na opravu (Repair), kde technici provedou. Pokud dojde pouze k softwarové chybě, většinou se řeší manuální změnou záznamu v databázi, kterou provádí vždy zaměstnanec IT oddělení.

2.2.2 Testy

Po osazení několika klíčových komponent absolvuje jednotka vysokonapěťový test (Hipot) a pokud tímto testem projde, dokončí se její montáž. Po kompletním dokončení montáže jednotka pokračuje na další testy, kde nejvýznamnějším je Run-In, kde je každá jednotka zapojena do sítě a několik hodin na ní běží dlouhodobé testy, které ji testují při maximálním vytížení.

Před tímto testem musí operátor naskenovat jednotky k vozíkům, na kterých budou po celou dobu testu umístěny. Při dokončení testů tak má v další aplikaci přehled o tom, na kterých vozících jsou umístěny.

Další testy už probíhají pouze u náhodně vybraných produktů a liší se podle modelové řady. U každé modelové řady je potřebné otestovat rozdílné vlastnosti za pomoci jiných testů při různých podmínkách.

V případě neúspěšného provedení testů se testy v případě nejasností ještě jednou zopakují, nebo se jednotky posílají rovnou technikovi na opravu.

2.2.3 Balička

Na baličce (Packing) je proces velmi podobný jako při montáži, na začátku jsou inspekční stanice, kde operátoři kontrolují, zda jednotka při výrobním procesu nebyla nijak poškozena, a zda je zkompletována správně.

Po úspěšné kontrole je k jednotce přidáno veškeré příslušenství a je zabalena do krabice. Krabice se dávají na palety, na kterých jsou převezeny do skladu a informace o nich se posílají do druhé databáze. Pokud kontrola nebyla úspěšná, jednotka se posílá na opravu a o dalším postupu již rozhoduje technik, který danou opravu provádí.

2.2.4 Opravna

V případě, že v některé fázi výroby dojde k poškození, záměně komponent, nebo třeba i nejasnostem při některém z testů, putuje jednotka na opravu (Repair). Tam technik zjistí povahu problému a s dalšími kolegy poté řeší opravení jednotky.

Při příjmu jednotky na opravu ji technik naskenuje do aplikace a vytvoří tím záznam o opravě. Do tohoto záznamu se ukládá povaha problému, a jak jej technik vyřešil. Po opravě ji technik dalším skenováním odhlásí z opravy, uloží se datum ukončení opravy, a jednotka se vrací zpět do výroby.

K nejčastějším problémům patří neopatrnost operátorů při zacházení, a tedy tím vznikající oděrky, praskliny, nebo třeba ohnuté piny při usazování komponent. V takovém případě dochází pouze k výměně poškozené části. Větším problémem je, když dojde k chybě při některém z testů, nicméně tam ve většině případů samotnou příčinu problému rozezná systém a vypíše chybovou hlášku, která obsahuje veškeré potřebné informace.

Podle povahy problému a provedené opravy také technik podle předem určených pravidel rozhoduje, do které části výrobního procesu se jednotka vrátí, v případě výměny některé z komponent je ale nutné, aby jednotka opět prošla všemi testy s kladným výsledkem.

2.3 Informace o databázi

Ve společnosti se používají 4 hlavní databáze – 2 s produkčními daty (jedna databáze je výrobní, druhá je zaměřena na data týkající se informací o jednotlivých objednávkách, jejich odesílání zákazníkovi a finanční záležitosti). Každá z těchto databází má také svoji testovací verzi, ve které jsou fiktivní data a slouží k vývoji a testování nejen nových aplikací, ale i k úpravám databáze, vytváření nových spouštěčů (triggerů) a podobně. V této práci se zaměřím na výrobní databázi, která podle mého názoru potřebuje doznat nejvíce změn a pro proces výroby je naprosto zásadní.

2.4 Práce s databází

Přímý přístup do databáze mají ve společnosti pouze zaměstnanci IT oddělení. Ostatní zaměstnanci přistupují k datům přes webové aplikace, a možnosti jejich zásahů do dat uložených v databázi jsou tak omezeny těmito aplikacemi.

V databázi se nenachází žádné procedury, většina úprav databáze se provádí přes tyto webové aplikace, které ve svém kódu obsahují jednotlivé SQL příkazy.

2.5 Úpravy databáze

Údržbu databáze a jejích dat má na starosti místní IT oddělení. Jejich prací je především kontrola a úpravy dat v případech, kdy dojde v průběhu výroby k lidským chybám, systémovým chybám, nebo úpravám výrobních procesů. V případě úprav výrobních procesů někdy musí dojít i k úpravám samotné struktury tabulek, které je většinou řešeno přidáním nové tabulky.

2.6 Používané aplikace

Ve výrobě se používá množství aplikací, které usnadňují operátorům práci a minimalizují vznik lidských chyb. Tyto aplikace jsou speciálně naprogramované pro potřeby společnosti. Některé z nich jsou naprogramované vývojáři z centrály na Taiwanu, protože se používají ve více pobočkách společnosti, velkou část ale vytvářejí sami zaměstnanci IT oddělení v Brně. Veškeré aplikace pracují s daty ze stejné výrobní

databáze a s přibývajícím počtem aplikací se tak i zvyšuje množství zapsaných dat a celkové nároky na databázi.

Do každé aplikace se musí nejdříve operátor přihlásit pod svým ID číslem a poté provádí činnost, která přísluší stanici, na které se právě nachází.

2.7 Stav databáze

Databáze je pro chod společnosti a správnost celé výroby velmi důležitým a vytěžovaným prvkem.

Stav databáze je v současné době nevyhovující, původní databáze byla převzata z jiného výrobního závodu stejné společnosti, byla částečně optimalizována, ale od té doby probíhají pouze dílčí změny ve smyslu přidávání nových tabulek, pohledů a triggerů. V databázi je tak velké množství tabulek, ve kterých není ani jeden záznam, a naopak jsou zde i tabulky, které obsahují více než 50 milionů záznamů. V těchto tabulkách je tak velice zdlouhavé něco hledat a i jednoduché dotazy trvají příliš dlouhou dobu.

Je zde samozřejmě mnohem víc problémů, které je potřeba řešit, mezi ně patří absence cizích a primárních klíčů, a tedy i neexistující vazby mezi jednotlivými tabulkami.

2.7.1 Množství záznamů v tabulkách

Po celou dobu se do databáze pouze ukládají data, za nejlepší příklad jistě poslouží tabulka transakcí. Do ní se zaznamenávají údaje o celém procesu výroby každé jednotky a denně tak do ní přibývají řádově tisíce údajů. Tuto tabulku také využívá několik aplikací a jakékoliv dotazy nad touto tabulkou pak aplikacím mohou trvat i více než 5 vteřin. To je samozřejmě neúnosná doba a proto se pro některé aplikace využívají pohledy nad touto tabulkou, což je ale pouze dočasné řešení.

2.7.2 Absence primárních klíčů

Primární klíče se v této databázi používají jen ve velmi malé míře a do většiny tabulek tak není žádný problém uložit duplicitní záznamy, které mohou vzniknout jak chybou operátora, tak systémovou chybou některé z aplikací. Použití právě jednoho primárního klíče (jednoduchého či složeného) by zajistilo entitní integritu jednotlivých záznamů a nebylo by tedy potřeba ukládání duplicitních záznamů ošetřovat v každé nově naprogramované aplikaci zvlášť.

2.7.3 Absence cizích klíčů

Většina tabulek (včetně těch základních a nejvytíženějších) také nemá vytvořený cizí klíč, mezi tabulkami tak neexistují žádné vazby.

2.7.4 NULL-ové hodnoty

Kromě primárních a cizích klíčů chybí také v databázi jasně definovány hodnoty, které nemoou nabývat hodnoty NULL (neznámá a nedefinovaná hodnota, čímž se rozumí prázdný sloupec u uloženého záznamu). Je tak umožněno uložit do databáze nekompletní záznamy. I tato skutečnost se ošetřuje v rámci jednotlivých aplikací, efektivnější řešení je ale ošetření tohoto problémů na úrovni databáze.

2.7.5 Nevhodné pojmenování sloupců

V několika případech dochází i k situaci, kdy je sloupec v jedné tabulce pojmenován stejně, jako je pojmenován sloupec v jiné tabulce, ale ukládají se do něj rozdílné informace. Vzhledem k nepoužívání cizích klíčů je tak naprosto nemožné bez znalosti databáze určit, k čemu některé sloupce slouží a o jaká data se konkrétně jedná.

2.7.6 Nepoužívání pohledů

K dalšímu z problémů patří nepoužívání pohledů. Pohledy se vytvářejí pouze pro potřeby jednotlivých aplikací, ale vždy až v případě příliš dlouhé doby provádění dotazů. V databázi však naprosto chybí pohledy, které by zobrazovaly základní data pro usnadnění orientace nebo informování o stavu výroby.

2.8 Shrnutí analýzy

Celkový stav databáze je nevyhovující pro efektivní práci.

Především kvůli neexistenci cizích klíčů, a tedy i vazeb mezi tabulkami, je bez výborné znalosti celého výrobního procesu pro každého nového pracovníka velký problém provádět v databázi jakoukoliv práci a orientovat se v ní. Zaškolení nových pracovníků tak trvá nepřiměřeně dlouho, a ti se pak zabývají činnostmi, které by mohly být prováděny automaticky. Vznikají také problémy a chyby, které by nikdy neměly nastat.

Velkým problémem je rozhodně tabulka transakcí. V této tabulce se uchovávají data stará i několik let, přitom jsou na tuto tabulky navázány aplikace, které využívají pouze aktuální data, a uchovávání starých dat je pouze zpomaluje. Tato tabulka je jednou z nejdůležitějších, které v databázi jsou, protože poskytuje přehled nad vším, co se během výroby děje.

3 Vlastní návrh řešení

Nyní se budu na základě předcházející analýzy věnovat vlastnímu návrhu na zlepšení situace a práce s databází. Navrhnu kompletní změnu struktury celé databáze, ze které by ale šly použít i pouze jednotlivé části, které by i tak usnadnily zaměstnancům práci.

Nový návrh výrobní databáze nabídne zjednodušenou strukturu bez zbytečných (již nepoužívaných) tabulek či sloupců v konkrétních tabulkách.

3.1 Konceptuální návrh databáze

Cílem konceptuálního návrhu databáze je vytvořit ER diagram, ve kterém budou identifikovány entity, relace a atributy. Pro tvorbu ER diagramu je vhodné nejdříve identifikovat základní entity databáze a relace mezi nimi.

3.1.1 Identifikace entit

Tato podkapitola se věnuje určení základních entit databáze. Nejedná se ovšem o všechny entity, pouze o jejich návrh, který nezohledňuje dekompozici a normalizaci.

Tabulka 2: Identifikace entit (Vlastní)

Název entity	Anglický název	Popis
Skupina	Group	Skupiny uživatelů určující jejich oprávnění
Zaměstnanec	Employee	Zaměstnanec společnosti pracující s daty v databázi
Jednotka	USN (Unit Serial Number)	Vyráběná jednotka s unikátním sériovým číslem

Objednávka	MO (Manufacture Order)	Objednávka od zákazníka obsahující informace o požadovaných jednotkách
Modelová řada	Model Family	Konfigurace běžně vyráběných modelových řad
Transakce	Transaction	Záznam o prováděné činnosti
Stanice	Stage	Stanice na výrobní lince
Linka	Line	Výrobní linka
Oprava	Repair	Informace o prováděných opravách
Komponenta	Item	Komponenty instalované do vyráběných jednotek
Test	Test	Informace o prováděných testech

3.1.2 Identifikace relací

Zde si identifikujeme vztahy mezi entitami, které nám později pomohou k provedení dekompozice některých vztahů mezi tabulkami a tvorbě ER diagramu.

Tabulka 3: Identifikace relací (Vlastní)

Entity	Relace	Popis vztahu
Group – Employee	1:N	V jedné skupině může být několik zaměstnanců, ale každý zaměstnanec je pouze v jedné skupině
Employee – Repair	1:N	Jeden zaměstnanec může provádět více oprav, ale jedna oprava může být prováděna pouze jedním zaměstnancem
Employee - Transaction	1:N	Jednu transakci provádí jeden zaměstnanec, ten ale může být uveden u více transakcí
Employee – Old_Transaction	1:N	Jednu transakci provádí jeden zaměstnanec, ten ale může být uveden u více transakcí
USN – Repair	1:N	V rámci jednoho záznamu o opravě je pouze jedna jednotka. Každá jednotka se ale může opravovat vícekrát
USN - Transaction	1:N	Jeden záznam v transakční tabulce se vztahuje k jedné jednotce, jedna jednotka je ale v transakční tabulce uvedená vícekrát
USN – Old_Transaction	1:N	Jeden záznam v transakční tabulce se vztahuje k jedné jednotce, jedna jednotka je ale v transakční tabulce uvedená vícekrát

USN – USN_Trolley	1:1	V tabulce USN_Trolley může být jedna jednotka uvedená pouze jednou
USN - Test	1:N	Jeden záznam o testu je pouze o jedné jednotce, ta ale může mít více záznamů o provedených testech
USN – USN_Item	1:N	V každém záznamu z tabulky USN_Item je pouze jedno USN, to může mít ale více záznamů v tabulce USN_Item
Line – Transaction	1:N	Každá transakce se odehrává pouze na jedné lince, ale na jedné lince se může provádět více transakcí
Line – Old_Transaction	1:N	Každá transakce se odehrává pouze na jedné lince, ale na jedné lince se může provádět více transakcí
Stage - USN	1:N	Jedna jednotka se může nacházet pouze na jedné stanici, zatímco na jedné stanici může být více jednotek
Stage – Transaction	1:N	Jeden záznam o transakci je vždy na jedné stanici, na jedné stanici může být ale uvedeno více záznamů
Stage – Old_Transaction	1:N	Jeden záznam o transakci je vždy na jedné stanici, na jedné stanici může být ale uvedeno více záznamů
MO – USN	1:N	Jedna jednotka může být pouze v jedné objednávce, ale každá objednávka může obsahovat více jednotek

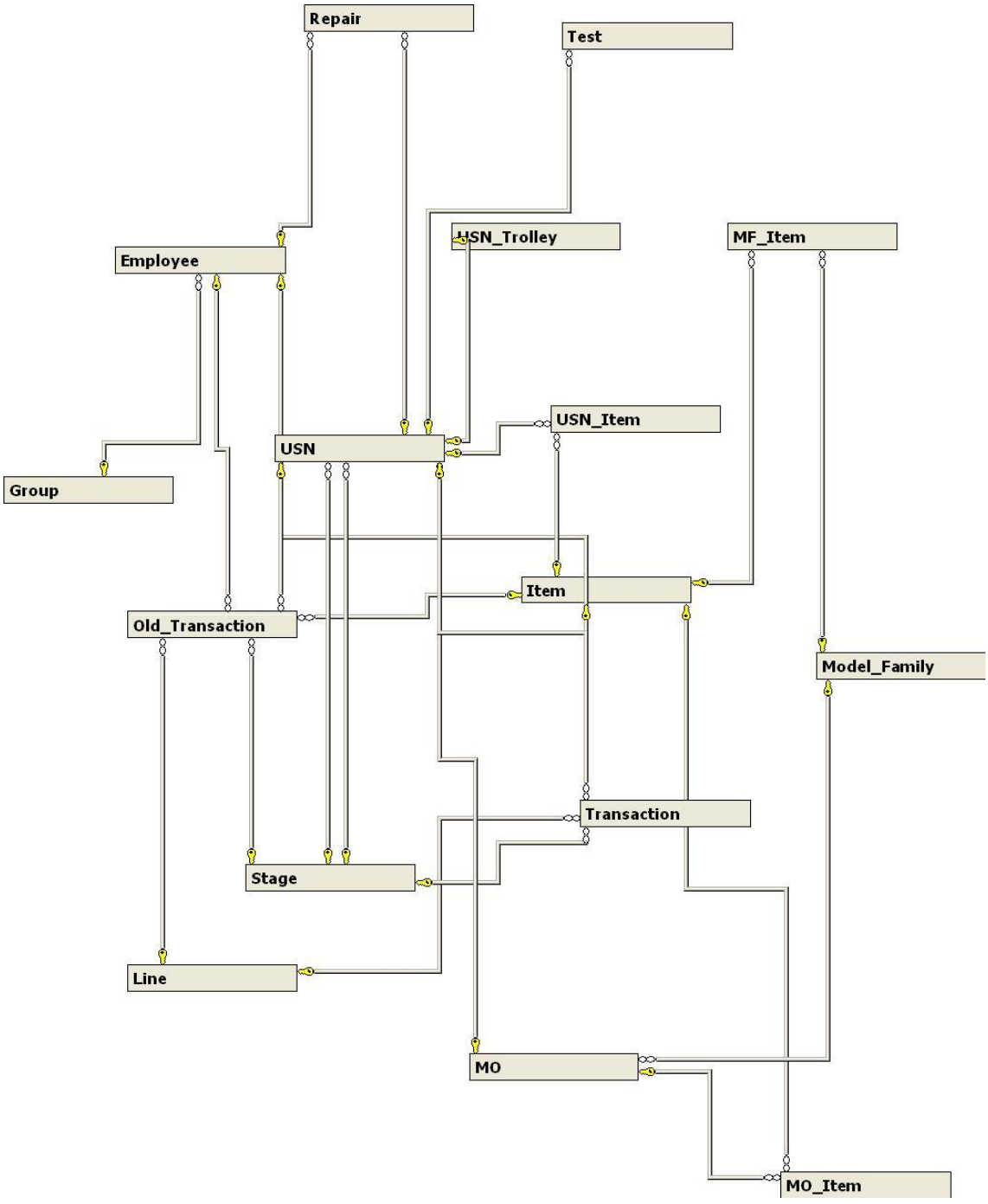
MO – Item	M:N	Jedna objednávka může obsahovat více druhů komponent, každý druh komponenty se může nacházet ve více objednávkách
Model_Family – Item	M:N	Jedna modelová řada může obsahovat více druhů komponent, každý druh komponenty se může nacházet ve více modelových řadách
Item – Transaction	1:N	Jedna transakce obsahuje informace o jednom typu komponenty, tyto typy ovšem můžou být ve více transakcích
Item – Old_Transaction	1:N	Jedna transakce obsahuje informace o jednom typu komponenty, tyto typy ovšem mohou být ve více transakcích

V tabulce je vidět, že se zde nacházejí dvě relace M“N. K optimálnímu fungování databáze je potřebné využít dekompozice těchto vazeb. Pro potřeby dekompozice vztahu tabulek „MO“ a „Item“ přidáme tabulku „MO_Item“, ve které se použijí primární klíče z obou entit a ze vztahu M:N tak vznikne 1:N a N:1.

Stejným způsobem postupujeme také u vztahu tabulek „Model_Family“ a „Item“, kde přidáme tabulku „MO_Item“, použijeme v ní primární klíče obou entit a také nám ze vztahu M:N vznikne 1:N a N:1.

3.1.3 ER Diagram

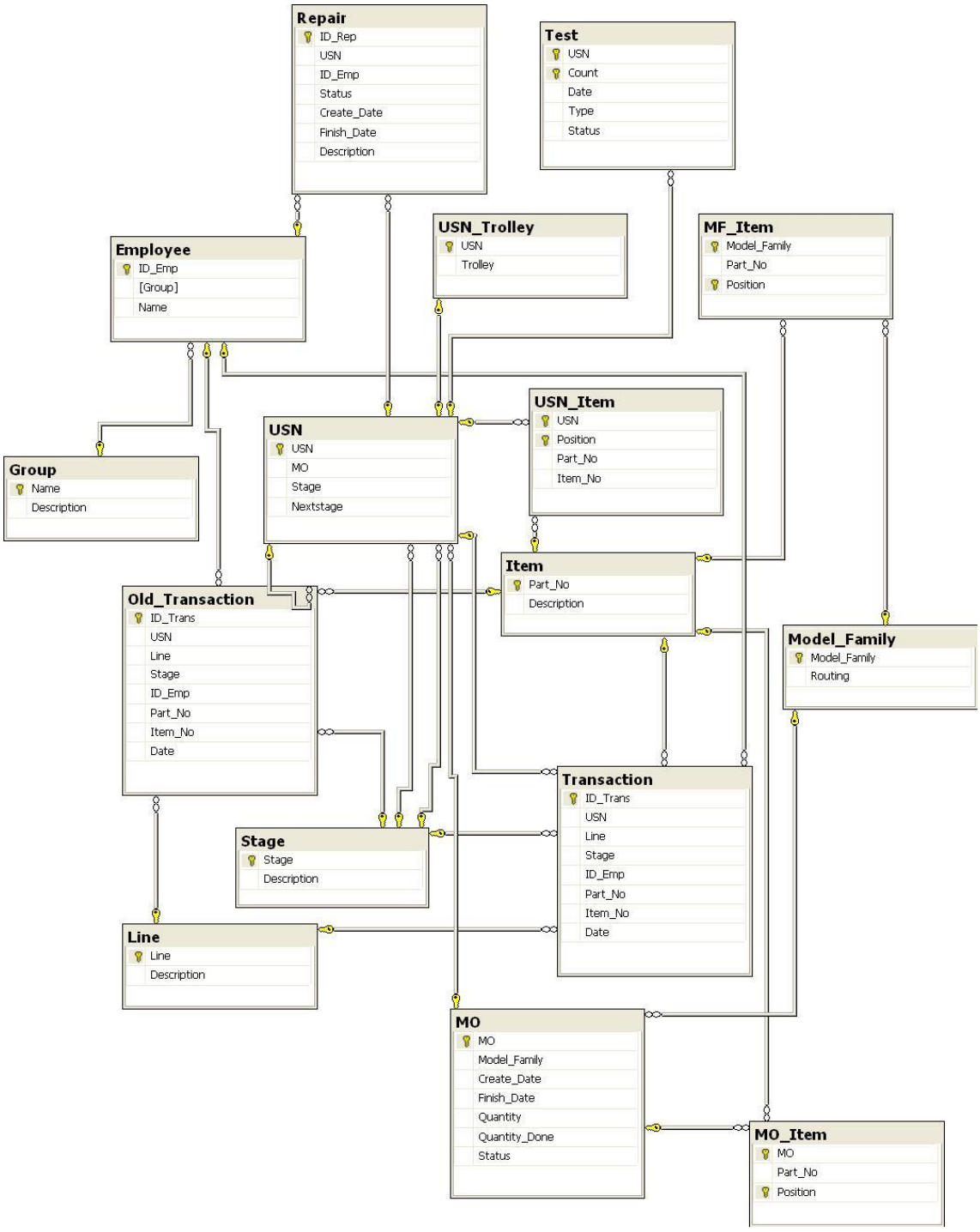
Entito – relační diagram nám umožňuje snadno reprezentovat entity a jejich vztahy. Pomůže nám také při tvorbě logického návrhu.



Obrázek 7: ER Diagram (Vlastní)

3.2 Logický návrh databáze

Logický návrh databáze nám poslouží k vytvoření tabulek. V logickém návrhu jsou rovněž obsaženy všechny tabulky databáze, primární a cizí klíče a vztahy mezi tabulkami.



Obrázek 8: Logický návrh (Vlastní)

3.2.1 Datový slovník

Datový slovník slouží k zobrazení všech entit v databázi a jejich položek. V tabulce jsou rovněž uvedeny datové typy položek, jejich délka, informace zda jsou primárními nebo cizími klíči a také zda má položka nějaké omezení. Tato je jedním z podkladů pro tvorbu SQL databáze.

Tabulka 4: Datový slovník (Vlastní)

Tabulka	Položka	Typ	Délka	Klíč	Omezení
Model_Family	Model_Family	Varchar	20	PK	NOT NULL
	Routing	Varchar	20		NOT NULL
Group	Name	Varchar	20	PK	NOT NULL
	Description	Varchar	50		NOT NULL
Employee	ID_Emp	Varchar	8	PK	NOT NULL
	Group	Varchar	20	FK	NOT NULL
	Name	Varchar	50		NOT NULL
Stage	Stage	Varchar	2	PK	NOT NULL
	Description	Varchar	20		NOT NULL
Line	Line	Varchar	4	PK	NOT NULL
	Description	Varchar	20		NOT NULL
Item	Part_No	Varchar	15	PK	NOT NULL
	Description	Varchar	20		NOT NULL
MO	MO	Varchar	15	PK	NOT NULL
	Model_Family	Varchar	20	FK	NOT NULL
	Create_Date	Datetime			NOT NULL
	Finish_Date	Datetime			NOT NULL

	Quantity	SmallInt			NOT NULL
	Quantity_Done	SmallInt			NOT NULL
	Status	TinyInt			NOT NULL
USN	USN	Varchar	15	PK	NOT NULL
	MO	Varchar	15	FK	NOT NULL
USN_Item	USN	Varchar	15	PK,FK	NOT NULL
	Position	TinyInt		PK	NOT NULL
	Part_No	Varchar	15	FK	NOT NULL
	Item_No	Varchar	15		NOT NULL
Repair	ID_Rep	Int		PK	NOT NULL
	USN	Varchar	15	FK	NOT NULL
	ID_Emp	Varchar	8	FK	NOT NULL
	Status	TinyInt			NOT NULL
	Create_Date	Datetime			NOT NULL
	Finish_Date	Datetime			
	Description	Varchar	100		
MO_Item	MO	Varchar	15	PK,FK	NOT NULL
	Part_No	Varchar	15	FK	NOT NULL
	Position	TinyInt		PK	NOT NULL
MF_Item	Model_Family	Varchar	20	PK,FK	NOT NULL
	Part_No	Varchar	15	FK	NOT NULL
	Position	TinyInt		PK	NOT NULL
USN_Trolley	USN	Varchar	15	PK,FK	NOT NULL
	Trolley	TinyInt			NOT NULL

Test	USN	Varchar	15	PK,FK	NOT NULL
	Count	TinyInt		PK	NOT NULL
	Date	Datetime			NOT NULL
	Type	Varchar	2		NOT NULL
	Status	TinyInt			NOT NULL
Transaction	ID_Trans	Int		PK	NOT NULL
	USN	Varchar	15	FK	NOT NULL
	Stage	Varchar	2	FK	NOT NULL
	Line	Varchar	4	FK	NOT NULL
	ID_Emp	Varchar	8	FK	NOT NULL
	Part_No	Varchar	15	FK	NOT NULL
	Item_No	Varchar	15		NOT NULL
	Date	Datetime			NOT NULL
Old_Transaction	ID_Trans	Int		PK	NOT NULL
	USN	Varchar	15	FK	NOT NULL
	Stage	Varchar	2	FK	NOT NULL
	Line	Varchar	4	FK	NOT NULL
	ID_Emp	Varchar	8	FK	NOT NULL
	Part_No	Varchar	15	FK	NOT NULL
	Item_No	Varchar	15		NOT NULL
	Date	Datetime			NOT NULL

3.3 Popis tabulek

3.3.1 Tabulka *Model_Family*

Tabulka *Model_Family* obsahuje základní informace o běžně vyráběných modelových řadách. Těmi jsou jejich název (*Model_Family*) a informace, na jakých stanicích a v jakém pořadí jsou jednotky vyráběné v rámci této modelové řady vyráběny (*Routing*). Všechny objednávky v rámci jedné modelové řady musejí dodržovat stejný routing.

3.3.2 Tabulka *Group*

V této tabulce se uchovávají informace o skupinách uživatelů. Podle příslušnosti uživatelů k daným skupinám (*Group*) systém rozpozná, na jaké pozici působí a k jakým aplikacím a činnostem má daný uživatel pravomoci.

3.3.3 Tabulka *Employee*

V tabulce *Employee* jsou uloženy informace o zaměstnancích společnosti. Je v ní uvedeno zaměstnancovo osobní číslo (*ID_Emp*), pozici (*Group*) a jeho jméno (*Name*).

3.3.4 Tabulka *Stage*

Výrobní linka je rozdělena na několik částí, k jejich pojmenování slouží tabulka *Stage*, ve které se jsou uloženy zkratky jednotlivých stanic rozmístěných na lince (*Stage*) a také krátký popis, k čemu slouží a kde se fyzicky nachází (*Description*).

3.3.5 Tabulka *Item*

V tabulce *Item* jsou uchovávány typy komponent, které jsou momentálně na skladě a tedy k dispozici pro výrobu. Zaznamenává se zde jejich číselné označení (*Part_No*) a popis (*Description*).

3.3.6 Tabulka *Line*

V tabulce *Line* jsou uloženy základní informace o výrobní lince, a tedy její název (*Line*) a krátký popis (*Description*).

3.3.7 Tabulka *MO*

Do tabulky *MO* se ukládají informace o objednávkách určených do výroby. Ke každé objednávce (*MO*) se zde ukládá datum jejího vytvoření (*Create_Date*), po úspěšném vyrobení všech jednotek také datum uzavření objednávky (*Finish_Date*), celkové množství (*Quantity*), doposud vyrobené množství (*Quantity_Done*) a také číselné označení (*Status*), který indikuje, v jaké fázi výroby se tato objednávka nachází (0 – připraveno k výrobě, 1 – v procesu výroby, 3 – výroba dokončena, 4 – objednávka vyexpedována, 5 – výroba přerušena, 6 – objednávka zrušena).

3.3.8 Tabulka *USN*

Tabulka *USN* obsahuje informace o jednotlivých jednotkách. Je zde záznam o každé vyráběné jednotce (*USN*), k jaké objednávce patří (*MO*), na jaké stanici se nachází (*Stage*) a na jakou stanici má přijít jako další (*Nextstage*).

3.3.9 Tabulka *USN_Item*

Tabulka *USN_Item* slouží k ukládání záznamů o jednotlivých komponentech nainstalovaných jednotek. Ukládá se zde jednotka (*USN*), pozice komponenty (*Position*), číselné označení typu komponenty (*Part_No*) a sériové číslo fyzicky nainstalované komponenty (*Item_No*). Zde je důležité od sebe poslední dva sloupce rozlišit. Zatímco *Part_No* je stejné pro všechny komponenty jednoho typu, *Item_No* je pro každý kus unikátní.

3.3.10 Tabulka *Repair*

Do tabulky *Repair* se ukládají záznamy o provedených opravách. Každá oprava má své identifikační číslo (*ID_Rep*), číslo jednotky (*USN*), číslo zaměstnance, který opravu prováděl (*ID_Emp*), číselný identifikátor, zda byla oprava úspěšná (*Status*), datum začátku opravy (*Create_Date*), datum ukončení opravy (*Finish_Date*) a popis, co vše bylo na jednotce opravováno (*Description*).

3.3.11 Tabulka *MO_Item*

Tabulka *MO_Item* slouží k uložení informací o komponentech, které budou do jednotek v dané objednávce instalovány. Ukládá se do ní číslo objednávky (*MO*), číselné označení pozice, na kterou daná komponenta patří (*Position*) a identifikační číslo této komponenty (*Part_No*).

3.3.12 Tabulka *MF_Item*

V tabulce *MF_Item* jsou uloženy informace o tom, jaké komponenty (*Part_No*) se běžně používají v modelových řadách (*Model_Family*) a na jakou pozici (*Position*) ve vyráběných jednotkách patří. Pro tvorbu objednávek však na rozdíl od routingu nejsou informace o obsažených komponentech tak striktní, v konkrétních objednávkách je možné komponenty měnit.

3.3.13 Tabulka *USN_Trolley*

Tabulka *USN_Trolley* slouží k párování jednotky (*USN*) a vozíku (*Trolley*), na kterém je jednotka při provádění Run-In testu umístěna.

3.3.14 Tabulka *Transaction*

Transaction tabulka, neboli tabulka transakcí, je jednou z nejdůležitějších a nejvíce využívaných tabulek v databázi. Ukládají se do ní data o činnostech prováděných s jednotkou. Každý záznam obsahuje identifikační číslo transakce (*ID_Trans*), číslo jednotky (*USN*), příslušnou stanici (*Stage*), linku, na které se stanice nachází (*Line*), identifikační číslo zaměstnance, který danou činnost prováděl (*ID_Emp*), číselné označení typu komponenty (*Part_No*), sériové číslo komponenty (*Item_No*) a datum, kdy byl záznam uložen (*Date*).

3.3.15 Tabulka *Old_Transaction*

Vzhledem k obrovskému množství dat v tabulce *Transaction* (řádově miliony záznamů) je vhodné pro efektivní práci vytvořit záložní tabulku, do které se budou přesouvat data starší než týden. Tyto záznamy je vhodné nadále uchovávat, ale přistupuje se k nim pouze ve výjimečných případech. V novějších záznamech se informace dohledávají velmi často a množství. Dat v této tabulce práci pouze zpomaluje.

Tabulka *Old_Transaction* má stejnou koncepci jako tabulka *Transaction,* pouze v identifikačním číslu tabulky (*ID_Trans*) není použit automatický přírůstek.

3.4 Zhodnocení stavu a přínosy společnosti

Cílem této práce bylo zanalyzovat a upravit produkční databázi ve výrobní společnosti. V doposud používané databázi bylo velké množství nepoužívaných tabulek, které nová databáze neobsahuje. Rovněž bylo vhodným identifikováním entit a vztahů mezi relacemi navrhnuto vhodné databázové schéme, které již bude plně vyhovovat nárokům společnosti.

Přínosy pro společnost jsou zejména zjednodušení orientace v databázi a zrychlení práce s daty. To povede nejen ke zefektivnění práce zaměstnanců, ale také k rychlejšímu zaučení nových pracovníků zejména do IT oddělení.

Závěr

Cílem této bakalářské práce bylo zanalyzovat produkční databázi, která slouží k ukládání informací o vyráběných serverech. Analýza této databáze ukázala, že tato databáze trpí řadou nedostatků, které způsobují pomalou a neefektivní práci a v krajních případech i chyby při práci s daty.

První část bakalářské práce obsahuje teoretická východiska, vysvětlující základní pojmy z oblasti problematiky databázových systémů, relačních modelů a jazyka SQL.

Druhá část se zabývá vlastní analýzou databázového systému společnosti. Tato analýza obsahuje nejen informace o databázi, ale i základní informace o společnosti, výrobním procesu, současném přístupu k úpravám databáze a používaných aplikacích.

Ve třetí části je již samotný návrh řešení. Je v něm zohledněna analýza současného stavu a požadavky společnosti. Podařilo se mi také eliminovat nedostatky předchozího řešení, vypracovat logický a konceptuální návrh, podle kterého jsem navrhl samotný fyzický návrh databáze. Tento návrh jsem poté implementoval pomocí jazyka SQL v prostředí SQL Server.

Navržená databáze by měla zjednodušit orientaci v databázi, snížit hardwarovou náročnost, minimalizovat chyby a celkově zefektivnit práci zaměstnanců společnosti.

Podle výše uvedených informací tak tato bakalářská práce splnila svůj cíl.

Seznam použité literatury

(1) KOCH, Miloš a Bernard NEUWIRTH. *Datové a funkční modelování*. Vyd. 4., rozš. Brno: Akademické nakladatelství CERM, 2010. ISBN 978-80-214-4125-5.

(2) KŘÍŽ, Jiří a Petr DOSTÁL. *Databázové systémy*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2005. ISBN 80-214-3064-8.

(3) STEPHENS, Ryan K, Ronald R PLEW a Arie JONES. *Naučte se SQL za 28 dní*. Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.

(4) POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy*. 1. vyd. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

(5) GILFILLAN, Ian. *Myslíme v MySQL 4*. Praha: Grada, 2003. Knihovna programátora (Grada). ISBN 80-247-0661-X.

(6) LACKO, Ľuboslav. *1001 tipů a triků pro SQL*. Brno: Computer Press, 2011. ISBN 978-80-251-3010-0.

(7) CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK.Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. 1.vyd. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.

(8) KROENKE, David a David J AUER. Databáze. 1. vyd. Brno: Computer Press, 2015. 496 s. ISBN 978-80-251-4352-0.

(9) KŘÍŽ, Jiří. *Databázové systémy - DBS* [online]. Vysoké učení technické v Brně, [cit. 2016-28-05]. Dostupný z: http://luhan.comlu.com/DBS/doc/P/04/04.pdf

(10) OPPEL, Andrew J. *SQL bez předchozích znalostí: [průvodce pro samouky]*. Vyd. 1. Brno: Computer Press, 2008, 240 s. Učební texty vysokých škol. ISBN 978-80-251-1707-1.

(11) SQL Server 2008 - datové typy. *Programujte.com* [online]. [cit. 2016-05-30]. Dostupné z: http://programujte.com/clanek/2010022200-sql-server-2008-datove-typy/

Seznam použitých obrázků

Seznam použitých tabulek

Přílohy

Příloha 1: Skript pro vytvoření databáze

```
CREATE TABLE Model_Family (
Model_Family VARCHAR(20) PRIMARY KEY,
Routing VARCHAR(20) NOT NULL
);

CREATE TABLE "Group" (
"Name" VARCHAR(20) PRIMARY KEY,
"Description" VARCHAR(50) NOT NULL
);

CREATE TABLE Employee (
ID_Emp VARCHAR(8) PRIMARY KEY,
"Group" VARCHAR(20) REFERENCES "Group"("Name"),
"Name" VARCHAR(50) NOT NULL
);

CREATE TABLE Stage (
Stage VARCHAR(2) PRIMARY KEY,
"Description" VARCHAR(20) NOT NULL
);

CREATE TABLE Item (
Part_No VARCHAR (15) PRIMARY KEY,
"Description" VARCHAR(20) NOT NULL
);

CREATE TABLE Line (
Line VARCHAR (4) PRIMARY KEY,
"Description" VARCHAR(20) NOT NULL
);

CREATE TABLE MO (
MO VARCHAR(15) PRIMARY KEY,
Model_Family VARCHAR(20) REFERENCES Model_Family(Model_Family),
Create_Date Datetime NOT NULL,
```

```
Finish_Date Datetime,
Quantity SMALLINT NOT NULL,
Quantity_Done SMALLINT NOT NULL,
"Status" TinyInt NOT NULL
);

CREATE TABLE USN (
USN VARCHAR(15) PRIMARY KEY,
MO VARCHAR(15) REFERENCES MO(MO),
Stage VARCHAR(2) REFERENCES Stage(Stage),
Nextstage VARCHAR(2) REFERENCES Stage(Stage)
);

CREATE TABLE USN_Item (
USN VARCHAR(15) REFERENCES USN(USN),
POSITION TinyInt NOT NULL,
Part_No VARCHAR(15) REFERENCES Item(Part_No),
Item_No VARCHAR(15) NOT NULL,
CONSTRAINT PK_USN_Item PRIMARY KEY (USN, POSITION)
);

CREATE TABLE Repair (
ID_Rep INT IDENTITY(1,1) PRIMARY KEY,
USN VARCHAR(15) REFERENCES USN(USN),
ID_Emp VARCHAR(8) REFERENCES Employee(ID_Emp),
"Status" TinyInt NOT NULL,
Create_Date Datetime NOT NULL,
Finish_Date Datetime,
"Description" VARCHAR(100)
);

CREATE TABLE MO_Item (
MO VARCHAR(15) REFERENCES MO(MO),
Part_No VARCHAR(15) REFERENCES Item(Part_No),
POSITION TinyInt NOT NULL,
CONSTRAINT PK_MO_Item PRIMARY KEY (MO, POSITION)
);
```

```
CREATE TABLE MF_Item (
Model_Family VARCHAR(20) REFERENCES Model_Family(Model_Family),
Part_No VARCHAR(15) REFERENCES Item(Part_No),
POSITION TinyInt NOT NULL,
CONSTRAINT PK_MF_Item PRIMARY KEY (Model_Family, POSITION)
);

CREATE TABLE USN_Trolley (
USN VARCHAR(15) PRIMARY KEY REFERENCES USN(USN),
Trolley TinyInt NOT NULL
);

CREATE TABLE Test (
USN VARCHAR(15) REFERENCES USN(USN),
"Count" TinyInt NOT NULL,
DATE Datetime NOT NULL,
"Type" VARCHAR(2) NOT NULL,
"Status" TinyInt NOT NULL,
CONSTRAINT PK_Test PRIMARY KEY (USN, "Count")
);

CREATE TABLE "Transaction" (
ID_Trans INT IDENTITY(1,1) PRIMARY KEY,
USN VARCHAR(15) REFERENCES USN(USN),
Line VARCHAR(4) REFERENCES Line(Line),
Stage VARCHAR(2) REFERENCES Stage(Stage),
ID_Emp VARCHAR(8) REFERENCES Employee(ID_Emp),
Part_No VARCHAR(15) REFERENCES Item(Part_No),
Item_No VARCHAR(15) NOT NULL,
DATE Datetime NOT NULL,
);

CREATE TABLE Old_Transaction (
ID_Trans INT PRIMARY KEY,
USN VARCHAR(15) REFERENCES USN(USN),
Line VARCHAR(4) REFERENCES Line(Line),
```

```
Stage VARCHAR(2) REFERENCES Stage(Stage),
ID_Emp VARCHAR(8) REFERENCES Employee(ID_Emp),
Part_No VARCHAR(15) REFERENCES Item(Part_No),
Item_No VARCHAR(15) NOT NULL,
DATE Datetime NOT NULL,
);
```