VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFROMATIKY

FACULTY OF BUSINESS AND MANAGEMENT

NÁVRH SQL DATABÁZE PRO VÝROBNÍ SPOLEČNOST

PROPOSAL OF SQL DATABASE FOR MANUFACTURING COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — JIŘÍ KREISINGER
AUTHOR

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

BRNO 2014

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2013/2014
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Kreisinger Jiří

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:

Návrh SQL databáze pro výrobní společnost

v anglickém jazyce:

Proposal 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:

LACKO, L. ORACLE Správa, programování a použití databázového systému. Vyd. 2. Brno: Computer Press, 2007, 576 s. ISBN 978 80-251-1490-2.
Oracle. docs.oracle.com Oracle [online] Dostupný na http://docs.oracle.com/
ŠIMŮNEK, M. SQL Kompletní kapesní průvodce. Praha: GRADA, 1999. ISBN 80-7169-692-7.

Techonthenet. techonthenet.com SQL [online] Dostupný na http://www.techonthenet.com/oracle/index.php

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

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2013/2014.

L.S.

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

V Brně, dne 17.05.2014

Abstrakt

Tato bakalářská práce se zabývá návrhem databáze pro výrobní společnost, která se rozhodla nahradit svoji stávající databázi, jež je v nevyhovujícím stavu. V první části je uvedeno teoretické pozadí problému. Dále pak analýza procesů při výrobě a z nich vycházejících požadavků na databázi. Na konci je uvedeno vlastní řešení databáze.

Abstract

This bachelor's thesis describes the proposal of a database for a manufacturing company. That decided to replace its existing database which is in inadequate state. In the first part is the theoretical background of the problem. Next part is analysis of manufacturing processes which leads to requirements on database. At the end is my own database solution.

Klíčová slova

SQL, databáze, Oracle, pohled, procedura, index

Keywords

SQL, database, Oracle, view, procedure, index

Bibliografická citace práce

KREISINGER, J. *Návrh SQL databáze pro výrobní společnost.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 62 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 ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 31. května 2014 ……

podpis studenta

Poděkování

Tímto bych chtěl poděkovat panu Ing. Jiřímu Křížovi, Ph.D. za odborné vedení bakalářské práce a čas, který mně věnoval.

OBSAH

ÚVOD

V dnešní době má v podstatě každá firma nějakou databázi, kde zaznamenává informace nezbytné pro její provoz. Může se jednat o jednoduchou evidenci zaměstnanců v tabulkovém editoru jako je například MS Excel přes malé databáze v MS Access až po rozsáhlé databáze, které můžou obsahovat miliony záznamů. Všechny tyto databáze se snaží zachytit nějaký problém z reálného světa a pomocí vhodně navržené struktury o něm uložit data.

Každá společnost, by si pak měla zvážit, jaká data potřebuje zaznamenávat a podle toho vhodně vytvořit svoji databázi. K tomu by měla provést analýzu, která data skutečně potřebuje zaznamenávat, protože ukládání nepotřebných dat pak stojí peníze za zbytečně využívaný výpočetní a úložný prostor a v neposlední řadě také čas který je použit na jejich vyplňování. Poté následuje rozhodnutí, na jaké platformě budu tuto svoji databázi provozovat.

Ve své bakalářské práci se budu zabývat návrhem databáze pro výrobní společnost, která se rozhodla nahradit svoji stávající databázi, která je v nevyhovujícím stavu. První část je věnována popisu teoretického pozadí řešeného problému.

V druhé části se budu věnovat popisu současného stavu databáze a rozepíší její zásadní nedostatky a následky které mají na chod výroby ve společnosti. Poté provedu analýzu a rozdělení procesů ve výrobě, ze kterých poté budu vycházet při návrhu samotné databáze.

V poslední části se budu zabývat návrhem databáze. Jednotlivé procesy zde budou pro lepší názornost zachyceny vývojovými diagramy. Poté bude následovat konceptuální, logický a nakonec fyzický návrh databáze.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Vymezení problému

Ve společnosti, pro kterou databázi navrhuji, se provádí montáž LCD televizí. Ty jsou sestavovány na výrobních linkách, kde se následně testují a poté zabalí. V průběhu výroby je potřeba detailně zachytit celou cestu televize po výrobní lince. Je tedy potřeba evidovat jaké komponenty byly do televize namontovány, jakými testy a s jakými výsledky televize testem prošla, kdo se podílel na její výrobě a jaké příslušenství k ní bylo zabaleno. V případě kdy dojde během výroby k závadě je potřeba zachytit i tuto informaci a následně i opravy které se na televizi provedly.

Cíle práce

Cílem práce je navrhnout pro tuto společnost databázi, která bude schopná zachytit její výrobní proces a všechny ostatní požadavky, které jsou na ní kladeny. Výsledkem by měla být databáze, která tyto požadavky splní a zároveň bude dostatečně flexibilní, aby mohla reagovat na změny, které při výrobě nastávají. Nesmí se u ní také vyskytovat nedostatky, které obsahuje původní databáze.

1. TEORETICKÁ VÝCHODISKA PRÁCE

1.1 Databázové systémy

Databázový systém se skládá z dat a nástrojů pro práci s daty. Součástí každého databázového systému musí být nástroje pro:

- Vytvoření, vyhledávání, aktualizaci a mazání uživatelských dat
- Definici struktury dat
- Definici a zajištění integrity dat
- Zajištění fyzické a logické nezávislosti dat

1.1.1 Historie databázových systémů

Počátky databází sahají až do 60. let kdy firma IBM přišla s prvním databázovým systémem založeným na hierarchickém modelu. Tento model počítal se stejným hierarchickým modelem jako organizační struktura organizace. Datová základna byla tvořena stromy, které měly mezi nadřízeným a podřízeným uzlem vztah 1:n.
V 70. letech se začaly stromy v hierarchickém modelu propojovat, což umožnilo popisovat i složitější struktury, než jen hierarchické vazby, ale pořád to nestačilo na zachycení reálných dat. V polovině 70. let tak přicházejí relační databáze a v 80. letech již zcela převládnou. Relační databáze přicházejí s jazykem SQL, který se postupně stal jediným prostředkem pro práci s těmito databázemi. [1]

1.2. Oracle

Společnost Oracle byla založena v roce 1977 Lawewncem J. Ellisonem, Robertem N. Minerem a Edwardem Oatesem. Původní jméno společnosti bylo SDL, později se přejmenovala na své nynější jméno, které vzniklo podle jednoho jejich projektu pro CIA. Společnost postupně vyvíjela databázové platformy, v roce 1978 vznikla Oracle verze 1.0 a do roku 1993 již bylo vytvořeno postupně sedm verzí.

1.2.1 Oracle 8i a 9i

Verze s podporou jazyka Java a XML. Obsahuje také mnohá vylepšení pro zlepšení výkonu a zdokonalenou optimalizaci pro PL/SQL. Písmeno i zde znamená orientaci na internet.

1.2.2 Oracle 10g

Tato verze byla vydána v roce 2003 a později v roce 2005 vyšla její aktualizace v podobě 10g Release 2. Písmeno g v tomto případě znamení grid. To znamená, že Oracle umí efektivněji využít hardware. Díky tomu, že v gridu přerozděluje nevyužitou výpočetní kapacitu procesům, které ji zrovna potřebují. [2]

1.3 Relační datový model

Relační datový model popsal v roce 1970 Edgar Frank Codd a je stále nejrozšířenějším datovým modelem, na kterém je založen návrh a tvorba databází.
Základní ideje relačního modelu

- RMD důsledně odděluje data, která jsou brána jako relace, od jejich implementace.
- Přístup k datům je symetrický, tj. při manipulaci s daty se nezajímáme o přístupové mechanizmy k datům.
- Pro manipulaci s daty je k dispozici relační kalkul a relační algebra.

Dle relační teorie lze pomocí základních operací (sjednocení, kartézský součin, rozdíl, selekce, projekce a spojení) uskutečnit veškeré operace s daty a ostatní operace jsou již jen kombinacemi těchto šesti [3].

1.4 Klíče relace

1.4.1 Primární klíč

Jedná se jedinečný identifikátor řádku v tabulce. Může být složen z jednoho či více sloupců, ale musí být vždy minimální, tzn. nelze z něj žádný sloupec vypustit, aniž by byla narušena jednoznačnost primárního klíče.

1.4.2 Kandidátní klíč

Z množiny kandidátních klíčů vybíráme primární klíč, tudíž pro něj platí i stejná omezení jako pro výše popsaný primární klíč.

1.4.3 Cizí klíč

Slouží ke spojení dvou tabulek. Cizí klíč se odkazuje na primární klíč v druhé tabulce. Slouží tím k zajištění referenční integrity [4].

1.5 Druhy relací

1.5.1 Relace 1:1

Tento příklad vazby jde nejjednodušeji vysvětlit, máme-li například tabulku člověk a tabulku DNA, pak mezi člověkem a DNA bude právě vazba 1:1

1.5.2 Relace 1:N

Tyto vazby jsou obvykle nejčastější, příkladem může být zaměstnanec a oddělení, na kterém pracuje, na jednom oddělení může pracovat více zaměstnanců, ale jeden zaměstnanec pracuje právě na jednom oddělení.

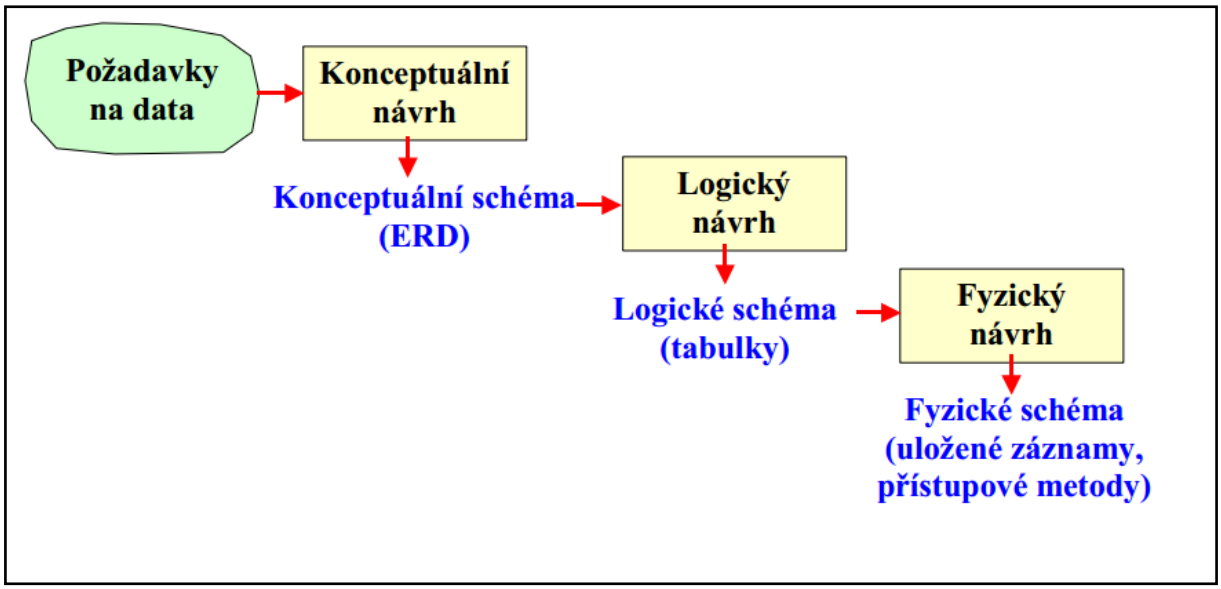
1.5.3 Relace N:M

Příklad této vazby lze vysvětlit, na situaci kdy autopůjčovna vede historii o půjčení svých aut, jeden klient si mohl půjčit více aut a jedno auto mohlo být vypůjčeno více klienty. Při implementaci se pak tato vazba rozpadne na tři tabulky s vazbami 1:N.

1.5.4 Bez relace

Mezi tabulkami není žádný vztah [5].

1.6 Návrh databáze



Obr. 1: Postup návrhu databáze (Zdroj [6])

1.6.1 Konceptuální návrh

Konceptuální modelování se skládá z těchto částí

- Datové fáze
- Objektové analýzy

ER Model

„ER model je založen na chápání světa jako množiny základních objektů - entit (Entity) a vztahů (Relationship) mezi nimi. Popisuje data "v klidu", neukazuje, jaké operace s daty budou probíhat. Někdy se označuje také jako ERA – třetím základním prvkem modelu jsou atributy (attributes).“[6,s.6]

Postup vytvoření ERD

- Identifikace entit
- Identifikace relací
- Identifikace a spojení atributů s entitami nebo relacemi
- Určení domén atributů

- Určení atributů, které budou kandidátními, primárními a alternativními klíči
- Specializace/generalizace entit (volitelný krok)
- Kontrola redundance v modelu
- Kontrola, zda model podporuje uživatelské transakce
- Posouzení konceptuálního návrhu databáze s uživateli

1.6.2 Logický návrh

Přenesení ERD modelu do tabulek

- Vytvoření tabulek
- Kontrola tabulek pomocí normalizace
- Kontrola, zda tabulky podporují uživatelské transakce
- Kontrola integritních omezení
- Posouzení logického návrhu databáze s uživateli

1.6.3 Fyzický návrh

V této části si již volíme konkrétní databázovou platformu, na které budeme návrh realizovat. Zvolíme organizaci souborů a indexů, dále navrhneme uživatelské pohledy a bezpečnostní mechanizmy [6].

1.7 Normalizace

Jedná se o proces, ve kterém se snažíme odstranit redundance v databázi. Tedy v čím vyšší normální formě máme tabulky, tím menší by měla být nadbytečnost dat. V praxi se obvykle používají první dvě až tři normální formy a v případě tabulek určených pro OLAP analýzu se od normalizace zpravidla ustupuje zcela.

1.7.1 Nultá normální forma

"Relace je v nulté normální formě, existuje-li takové její pole, které obsahuje více než jednu hodnotu." [7]
Například máme-li pole a v něm uloženou adresu, ve které máme město, ulici a PSČ.

1.7.2 První normální forma

Tabulka je v první normální formě, pokud jsou všechny atributy atomické, tedy déle nedělitelné.
Budeme-li tedy chtít do tabulky ukládat adresu skládající se z města, ulice a PSČ pak ji rozdělíme do třech sloupců.

1.7.3 Druhá normální forma

„Tabulka splňuje 2NF, právě když splňuje 1NF a navíc každý atribut, který není primárním klíčem je na primárním klíči úplně závislý. To znamená, že se nesmí v řádku tabulky objevit položka, která by byla závislá jen na části primárního klíče. Z definice vyplývá, že problém 2NF se týká jenom tabulek, kde volíme za primární klíč více položek než jednu.“ [8]

1.7.4 Třetí normální forma

Tabulka je ve třetí normální formě, když je i ve druhé normální formě a zároveň neexistují závislosti neklíčových sloupců tabulky.

1.7.5 Čtvrtá normální forma

Pokud tabulka splňuje třetí normální formu a obsahuje jen jednu souvislost nebo fakt pak je ve čtvrté normální formě.

1.7.6 Pátá normální forma

Pokud tabulka splňuje čtvrtou normální formu a je nemožné do ní vložit jeden či více sloupců aniž by se tabulka rozpadla do několika menších, pak je v páté normální formě.

1.8 Jazyk SQL

V průběhu 70. let byl firmou IBM vyvinut jazyk SEQUEL (Structured English Query Language). Tento jazyk měl být svojí syntaxí co nejbližší anglickému jazyku. Později se jeho vývojem začali zabývat i další firmy a na počátku 80. let se jazyk přejmenoval na SQL. V roce 1986 přijímá ANSI jazyk SQL za standard pro relační databáze pod označením SQL86, který je později aktualizován v roce 1992 jako SQL-92, zkráceně jen SQL2 a obsahuje navíc prvky pro integritu dat a odstranění některých nedostatků původního standardu. V roce 1999 je pak přijat standard SQL, zaměřený na objektově organizované databáze. [1]
Základní prvky jazyka SQL jsou shodné pro všechny databázové systémy, ale v některých detailech se může syntaxe jednotlivých příkazů lišit, pro zde přehled příkazů a jejich syntaxe pro databázi od Oraclu.

1.8.1 Data Definition Language (DDL)

Tyto příkazy slouží k definování, vytváření, měnění a mazání datových struktur, kterými mohou být databáze, tabulky, pohledy, indexy, triggery a uložené procedury.

Create database

Příkaz sloužící k vytvoření databáze

Syntaxe: CREATE DATABASE Jmeno_Databaze;

Create table

Příkaz sloužící k vytvoření tabulky

Syntaxe: CREATE TABLE Nazev_Tabulky (Nazev_Sloupce Datovy_typ(velikost);

Alter table

Změna sloupců v tabulce (přidání/odebrání/změna datového typu)

Syntaxe:

ALTER TABLE Nazev_Tabulky ADD/DROP/MODIFY Nazev_Sloupce Datovy_typ;

Drop table

Příkaz pro smazání tabulky

Syntaxe: DROP TABLE Nazev_Tabulky;

Create index

Příkaz pro vytvoření indexu typu B-Strom nebo bitmapového

Syntaxe: CREATE (BITMAP) INDEX Nazev_indexu

Drop index

Smazání indexu

Syntaxe: DROP INDEX index_name

Create view

Vytvoření pohledu

Syntaxe: CREATE VIEW view_name AS (Select ...)

Alter view

Změna pohledu

Drop view

Smazání pohledu

Create sequence

Vytvoření autoincrementu, neboli automatického číslování.

Alter sequence/Drop sequence

Zmněna/smazání autoincrementu

Create procedure

Vytvoření procedury

Drop procedure

Smazání procedury

Create trigger

Vytvoření triggeru

Drop trigger

Smazání triggeru [9]

1.8.2 Data Manipulation Langueage (DML)

Příkazy pro manipulaci s daty. Slouží pro zobrazení, vložení, změnu nebo smazání dat.

Select

Základní příkaz jazyka SQL pro zobrazení dat

Syntaxe: Select seznam_sloupcu from seznam_tabulek

Insert

Vkládání dat do databáze

Syntaxe: INSERT INTO Nazev_Tabulky (Sloupec1, Sloupec2, Sloupec3,...)

VALUES (hodnota1, hodnota2, hodnota3,...);

V případě že zadáváme hodnoty všech sloupců, můžeme jejich výpis v příkazu vynechat

Update

Změna dat v databázi

Syntaxe:

UPDATE Nazev_Tabulky

SET Sloupec1= hodnota1, Sloupec2= hodnota2,...

WHERE Sloupec=hodnota;

Podmínka where není povinná, ale při jejím vynechání se změní hodnoty ve všech řádcích

Delete

Syntaxe:

DELETE FROM Nazev_Tabulky

WHERE Sloupec=hodnota;

Podmínka where opět není povinná, ale při jejím vynechání se smažou hodnoty ve všech řádcích. Pokud bychom chtěli vymazat celou tabulku, můžeme k tomu použít také příkaz TRUNCATE TABLE.[10]

1.8.3 Data Control Language (DCL)

Slouží k řízení a údržbě databáze. Umožňuje

Grant

Revoke create user

Alter user

Drop user

Grant

Revoke

Transaction Control Commands (TCC)

Set transaction

Commit

Rollback

Save Point [11]

1.8.4 Datové typy v ORACLE 10g

Textové datové typy

CHAR (délka) [BYTE | CHAR]

Slouží pro uložení řetězce o pevné délce, daným povinným parametrem délka, který má rozsah <1;2000>. Pomocí volitelného parametru určujeme, jestli délka znamená počet bajtů nebo znaků.

NCHAR (délka)

Obdobný datový typ jako CHAR. Slouží pro určený datový typ národní znakové sady.

VARCHAR2 (délka) [BYTE | CHAR]

Jedná se o datový typ určený pro uložení řetězce proměnné délky. Parametr délka je povinný a má rozsah <1;4000>. Na rozdíl od typu CHAR, nevyužité znaky nezabírají místo v paměti.

NVARCHAR2 (délka)

Opět stejný případ jako NCHAR akorát s proměnlivou délkou.

LONG

Datový typ pro textové řetězce o maximální velikosti až 2 GB. Tento datový typ ale nelze používat v klauzuli WHERE!

Číselné datové typy

NUMBER (počet číslic, z toho číslic za desetinou čárkou)

Jedná se o jediní numerický datový typ v Oracle. Ostatní numerické typy jako například INTEGER, SMALLINT , REAL atd. lze také definovat, ale převádějí se na základní typ number.

Datové typy pro vyjádření data, času a časového intervalu

Date

Rozsah od 1. ledna 4712 před naším letopočtem do 31. prosince 9999 našeho letopočtu. Umožňuje zadávat datum s přesností na sekundy.

Timestamp

Obdobné jak formát Date, s tím rozdílem že umožňuje uložit čas s přesností na miliontinu sekundy.

Timestamp with time zone

Timestamp with local time zone

Interval year [(rozsah_roků)] to month

Interval day [(rozsah_dní)] to second [přesnost_sekund]

Datové typy pro uložení objemných dat

BLOB

Binární objekt o velikosti až 4GB

CLOB

Binární objekt o velikosti až 4GB

NCLOB

Binární objekt o velikosti až 4GB, obsahující znaky UNICODE [2]

2. ANALÝZA PROBLÉMU A SOUČASNÉHO STAVU

2.1 Informace o společnosti

Jedná se výrobní společnost s přibližně 30 000 zaměstnanci s centrálou na Taiwanu. Hlavní náplní práce společnosti je outsourcová výroba elektroniky pro své zákazníky, jimiž jsou velké firmy, které poskytují svoji značku, avšak vlastní výrobu outsourcují. Jednou z částí společnosti, kterou se budu ve své práci zabývat, je výrobní závod LCD televizí, který se nachází v průmyslové zóně v městské části Brno-Slatina. Tento závod má přibližně 800 zaměstnanců a 8 montážních linek.

2.2 Popis současné situace

Ve výrobní hale se nachází 8 montážních linek, na kterých se LCD televize napřed montuje, poté prochází testy, aby se zjistilo, jestli televize správně funguje a na konci je balící část linky kde se přidá příslušenství, televize se vloží do polystyrenové výplně a vše se následně vloží do krabice. Krabice se pak seskupí do palety a ta je připravena pro transport do skladu. Pro sběr informací a správný chod výroby, je zde databáze, do které se informace zaznamenávají a zároveň načítají nejrůznějšími aplikacemi, která potřebují kontrolovat postup při výrobě.

2.2.1 IT oddělení ve společnosti

Ve společnosti má IT oddělení silné postavení a mnoho programů a systému si navrhuje samo, hlavně těch co se týkají výroby. Centrem toho vývoje je IT na Taiwanu, odkud pochází mnoho aplikací a systémů. Ty se pak posílají jednotlivým divizím po celém světě a ty si je pak mohou lokálně uzpůsobovat. Politika společnosti je v tomto poměrně volná a jednotlivé pobočky mají v tomto směru možnost si aplikace uzpůsobovat, případně napsat samy pokud je potřebují a mají na to potřebný personál. Tato roztříštěnost v IT vyplývá z celkové koncepce, kde jednotlivé divize společnosti po světě jsou celkem samostatně řízené a ne všechny mají stejné postavení, které se pak projevuje například rozdílným vybavením závodů a tím pak i jinými výrobními procesy.

2.2.2 Současný stav databáze

Pro provoz současné databáze je použita databázová platforma Oracle 10g. Jedná se o ideální volbu z důvodu velkého množství ukládaných dat a počtu transakcí, které se provádějí. Z těchto důvodů je volba spolehlivé a výkonné platformy v podobě Oracle správná volba a není potřeba ji měnit.
V závodě tedy existuje databáze, do které se zaznamenávají podstatné informace z výrobního procesu, ale ta trpí řadou problémů. Tím nejpodstatnějším je, že se jedná o databázi, která byla navržena při vzniku závodu a částečně vycházela z návrhu databáze v jiném závodě, neboť během vzniku závodu ve Slatině se musel najmout kompletně nový personál, který se teprve musel zaučit, což měli na starosti školitelé z Taiwanu. Za tu dobu, co probíhá výroba v současném závodě, se mnoho změnilo. Upravování databáze pak probíhalo ve stylu, který by se dal nazvat „princip sněhové koule“, kdy potřebné úpravy databáze probíhaly nesystematickým přidáváním dalších a dalších tabulek. Databázi tedy chyběla koncepce a tím se velmi rychle stala nepřehlednou, a mnohé informace byly duplikovány a byla narušena také normalizace databáze. Další nedostatky současné databáze jsou:

Absence cizích klíčů
v databázi nejsou použity téměř žádné cizí klíče, tudíž veškeré vazby v databázi a propojení tabulek nelze z databáze vyčíst a v praxi se při práci s databází a psaní SQL dotazů musí vycházet ze znalosti toho, jak jednotlivé procesy ve firmě fungují a jaké jsou reálné návaznosti s ohledem na to, co tabulky reprezentují. Z toho pak vyplývá, že kdo se neorientuje v tom, jak celý proces výroby funguje, nemá téměř žádnou šanci s databází pracovat a to vše je ještě umocněno jejím současným rozsahem. Tudíž zaškolení někoho nového na IT oddělení trvá velmi dlouho a pracovníci se znalostí databáze jsou pak velmi vytížení, protože musí dělat i tu práci, kterou by ve správně navržené databázi zvládli i jiná oddělení.

Absence primárních klíčů

Některé tabulky přidané do databáze v průběhu jejího používání neobsahují primární klíče, a tudíž může docházet k porušení základního pravidla, že řádky v tabulce se neopakují. Takovýchto tabulek sice není v databázi příliš mnoho, ale vyskytují se tam. Nejčastěji z důvodu jejich nesystémového přidávání, kdy bylo potřeba rychle vyřešit nastalý problém, např. zavádění nové aplikace pro jiné oddělení (takových to požadavků bylo celkem hodně), kde tato aplikace potřebovala určitá data a tudíž se narychlo vytvořila nová tabulka. Často se pak stávalo, že po tom co se aplikace přestala používat a zavedla se jiná, tak tabulka v databázi zůstala a zcela zbytečně zabírala místo.

Nepoužívání pohledů

Jak již bylo zmíněno, do databáze byly ad-hoc přidávány tabulky které často duplikovali informace, a to jen proto, že databáze byla příliš nepřehledná a některé informace nebylo možné snadno vyčíst. A mnohdy se vytvářeli fyzicky další tabulky i v případech kdy by stačilo vytvořit view. V některých případech je vytvoření fyzické tabulky opodstatněné, například když provádí agregační funkce nad velkým množstvím dat, a spouštění view by zpomalovalo databázi a tím i celý výrobní proces. V tomto případě je žádoucí vytvořit fyzickou tabulku a informace do ní, aktualizovat a přidávat v noci když neprobíhá výroba, ale z tabulek které byly do databáze přidány, je takových jen velmi málo.

Špatně nastavené nebo zastaralé indexy

Indexy mohou při správném použití zrychlit chod celé databáze, naopak při špatném mohou mít nulový, případně ještě záporný efekt. V databázi je použito několik složených indexů, u kterých záleží na pořadí sloupců v indexu, a pokud není první sloupec při dotazu vyplněn, je velká pravděpodobnost, že bude index při dotazu nad databází ignorován. U některých indexů bylo toto pořadí nepříliš ideálně zvoleno, a proto se u mnoha dotazů vůbec nepoužijí. Další chybou bylo použití B-stromových indexů na sloupci s nízkou kardinalitou, kde by naopak bylo vhodné použít spíše bitmapové indexy.

Dlouhá doba ukládání transakčních záznamů

Po celou dobu existence databáze se do ní záznamy pouze ukládaly a nikdy nebyly mazány nebo přesunuty. Došlo tedy ke stavu kdy v transakční databázi, jejímž primárním úkolem je podporovat chod výroby, se stalo jakési skladiště dat, která již ve výrobě dávno nepotřebují. Uložení záznamů, které jsou staré více než rok a patří televizím, které jsou už dávno prodané, transakční databázi zbytečně zpomaluje. Prodlužuje se tím doba odezvy na dotazy pokládané databázi, a jelikož je databáze provázána s aplikacemi na výrobní lince, dochází tím i ke zpomalování výrobního procesu.

Shrnutí stavu databáze

Výsledkem výše popsaných problémů se stalo to, že databáze obsahuje více jak stovku tabulek, kde mnohé jsou zcela zbytečné. Takovýto stav má pak za následek zvýšené náklady na datová úložiště pro databázi (což je ještě zesíleno tím, že data jsou zálohována a ukládána na server, který používá RAID). Dalším důsledkem ještě horším než náklady na ukládání je zpomalování celé databáze, v důsledku toho že databáze má delší dobu odezvy při ukládání a načítaní dat aplikacemi na výrobních linkách, což vede ke zpomalení celého výrobního procesu a tím prodražení celé výroby. Tyto problémy se s postupem času dále stupňovaly, ale na IT oddělení nebyla příliš velká ochota s tímto stavem něco dělat, neboť hlavní bylo, že databáze nějak fungovala, a v době nejvyššího vytížení výroby nebyl ani čas tento problém řešit. To se změnilo až s tím jak se problémy s databází stupňovali a také s poklesem produkce a tudíž se snížením vytížení většiny oddělení v závodu. V této době tedy začala různá oddělení řešit problémy a práci, která se v důsledku vysokého vytížení odkládala. IT oddělení není v tom výjimkou, a tudíž si dalo za úkol návrh nové databáze, která by neobsahovala výše vypsané nedostatky.

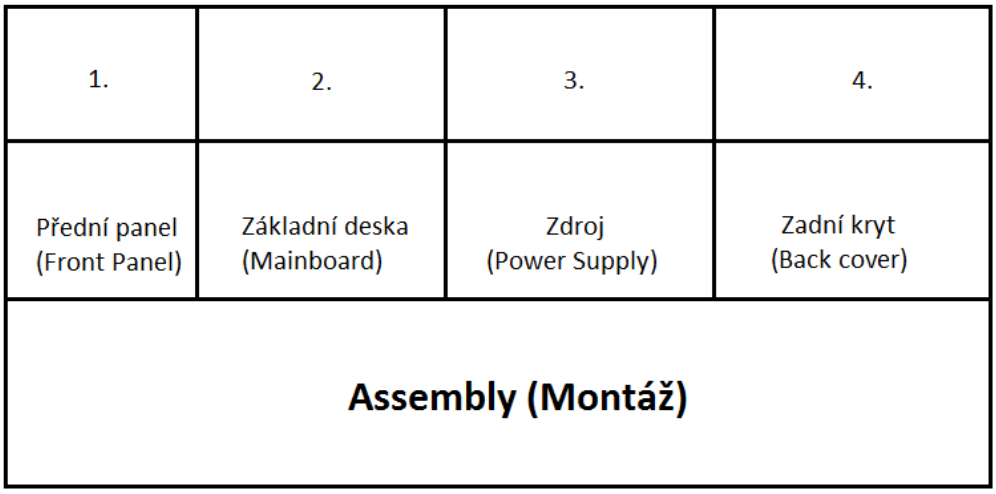
2.3 Analýza problému

2.3.1 Výrobní linka

Výrobní linka sestává ze tří základních částí, z první kde se LCD televize sestavuje, druhé kde probíhají testy, jestli televize správně funguje a poslední kde se přidá příslušenství a celá televize se zabalí.
Linka jako taková je výrobní pás, po kterém projíždí televize. Na lince jsou rozmístěny takzvané stanice, což je pracovní pozice, u které stojí operátor výroby a provádí určitou činnost, která mu byla přidělena. Rozvržení stanic na lince určuje oddělení, které má na starosti výrobní proces. Ne každá stanice na lince je podstatná z hlediska databáze, zpravidla ty kde se do televize nic nepřidává nebo nekontroluje. Ty stanice, ze kterých se data do databáze ukládají, případně načítají, jsou obvykle vybavena osobním počítačem, případně terminálem. Na počítači je pak spuštěna aplikace, která operátorovi pomáhá při výrobě a slouží také ke kontrole, neboť je napojena na databázi. Operátor se do stanice loguje pomocí svého identifikačního čísla, aby bylo dohledatelné, kdo stanici obsluhoval.
Pro identifikaci televize při výrobě slouží štítek s jedinečným identifikačním číslem. Tato čísla jsou vygenerována speciální aplikací, poté uložena do databáze a následně vytištěna na štítky, na nichž je kromě číselné podoby tato informace také v podobě čárového kódu. Tento štítek je pak přiložen k sestavené televizi a slouží pro její identifikaci. K načítání čárových kódu mají pak operátoři ruční scannery, které jsou pak napojeny do PC nebo terminálu na dané stanici. Scannery se také používají pro logování operátorů do stanice, neboť každý operátor má svoji identifikační kartu, kde je jeho ID také v podobě čárového kódu.
Jednotlivé komponenty a součástky pro výrobu jsou přiděleny ze skladu a o jejich přiděleném počtu rozhoduje oddělení které má na starosti plánování výroby. Před tím než jsou komponenty přivezeny k lince, tak prochází vstupní kontrolou, aby se zamezilo montování televize z poškozených dílů, a tím se pak následně prodlužoval výrobní proces v důsledku oprav a výměny vadných komponent.

2.3.2 Montáž (Assembly)

V této části linky se televize kompletně smontuje. Sestaví se tady ze čtyř základních částí, předního panelu, základní desky, zdroje a zadního krytu. Tyto čtyři hlavní komponenty na sobě mají čárový kód s jedinečným identifikátorem a je potřeba je zaznamenat a přiřadit konkrétní televizi aby je v případě potřeby bylo možné dohledat, kde se nacházejí. Z hlediska databáze se tedy na této části linky nacházejí čtyři stanice, odkud jsou posílány a zároveň načítána data do databáze. Fyzicky se zde nacházejí ještě další stanice, kde operátoři televizi sestavují, zapojují kabely, šroubují jednotlivé části televize, ale nic se zde už do databáze nezaznamenává.



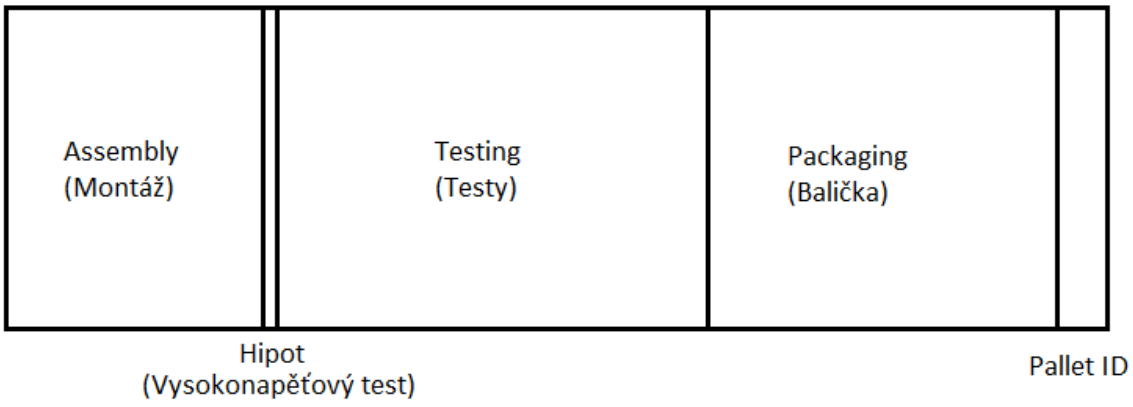
Obr. 2: Montážní část linky (Zdroj: vlastní zpracování)

2.3.3 Testování (Testing)

Na této části linky se televize kompletně otestuje, jestli správně funguje. Prvním testem je test vysokým napětím (hipot) a dále pak série různých testů, které se můžou pro jednotlivé modely lišit. Výsledky každého testu jsou zaznamenány a zároveň je zaznamenáno, jestli televize testem úspěšně prošla.

2.3.4 Balička (Packaging)

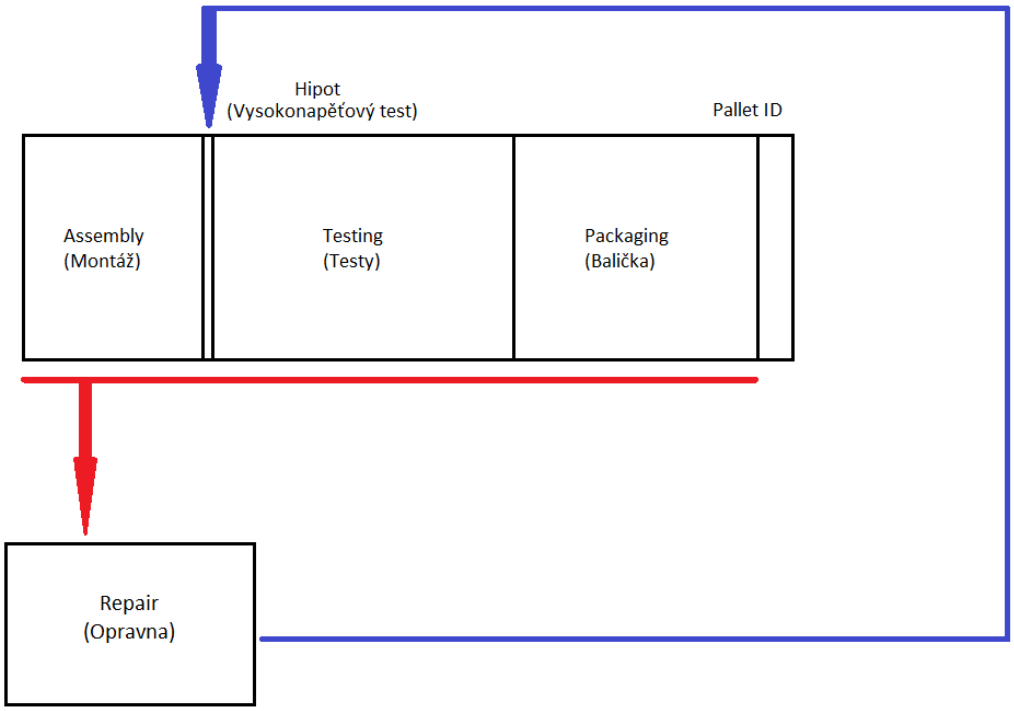
Tady se již ke kompletně smontované a otestované televizi přidá příslušenství, které si zákazník pro danou objednávku určil. Jednotlivé položky příslušenství se zde skenují a taktéž přiřazují ke konkrétní televizi ale na rozdíl, od hlavních komponent nemají jedinečný identifikátor, ale jen kód pro danou skupinu, jelikož účelem zde není dohledatelnost konkrétního příslušenství, ale kontrola že veškeré objednané příslušenství bylo přiloženo a že se na nic nezapomnělo. Televize se pak zabalí do ochranné polystyrenové výplně a spolu s příslušenstvím pak vloží do krabice. Ta se pak přidá na paletu a ta je následně připravena na převoz do skladu.



Obr. 3: Výrobní linka (Zdroj: vlastní zpracování)

2.3.5 Opravy (Repair)

Při výrobě televize může dojít k mnoha různým problémům, které si vyžádají, aby byla televize odebrána z linky a následně opravena. Závada se může vyskytnout již při montáži, pokud například bude některá z komponent montovaných do televize poškozena, nejčastějším případem je poškrábaný přední panel nebo zadní kryt, který nedopatřením prošel přes vstupní kontrolu kvality. Nebo se mohlo stát, že se televize poškodila v důsledku nehody nebo špatné manipulace přímo na lince. Je tedy potřeba aby měl operátor na lince, který si toho všimne možnost tento kus označit a následně odebrat z linky. Nejčastější závadou je pak to, že televize neprojde některým z testů v testovací části linky. O jakékoliv chybě musí existovat záznam. Vadná televize se pak přemístí do opravny, která se nachází ve výrobní hale, kde opravář vadu opraví, případně vymění neopravitelnou komponentu. O opravě se pak také musí provést záznam. Následně se televize vrátí zpátky na linky, pokud byla otevřena tak se vždy posílá na testy od začátku přes test vysokým napětím.



Obr. 4: Opravy na lince (Zdroj: vlastní zpracování)

2.3.6 Shrnutí

Úkolem výroby je tedy zhotovit televizi, zkontrolovat její funkčnost a spolu s příslušenstvím zabalit. Celý proces výroby musí být zaznamenán v databázi, aby v případě potřeby bylo možné dohledat veškeré informace o každé televizi, která se zde vyrobila. Musí být tedy dohledatelné jaké komponenty v televizi jsou, který operátor je tam dal, kdy jednotlivé úkony proběhly, jaké byly výsledky testů a také to jestli televize prošla nějakou opravou, případně výměnou komponentů. Na druhou stranu už se při výrobě nemusí řešit problémy, které má na starosti sklad a tím je evidence komponent a jejich množství pro výrobu. Komponenty jsou tedy vydávány ze skladu dle plánu sestaveného oddělením zabývajícím se plánováním výroby a zodpovědnost za jejich dostatečné množství a evidenci už je na skladu. Koncovým bodem pro záznam v databázi je přiřazení televize na paletu. Hotové palety se potom odvezou do skladu a následně jsou převzaty zákazníkem.

3. Vlastní návrhy řešení, přínos návrhů řešení

V této kapitole se budu věnovat návrhu databáze, tak aby splnila nároky, které jsem popsal v analýze problému a popisu současného stavu.

3.1 Rozdělení základních procesů

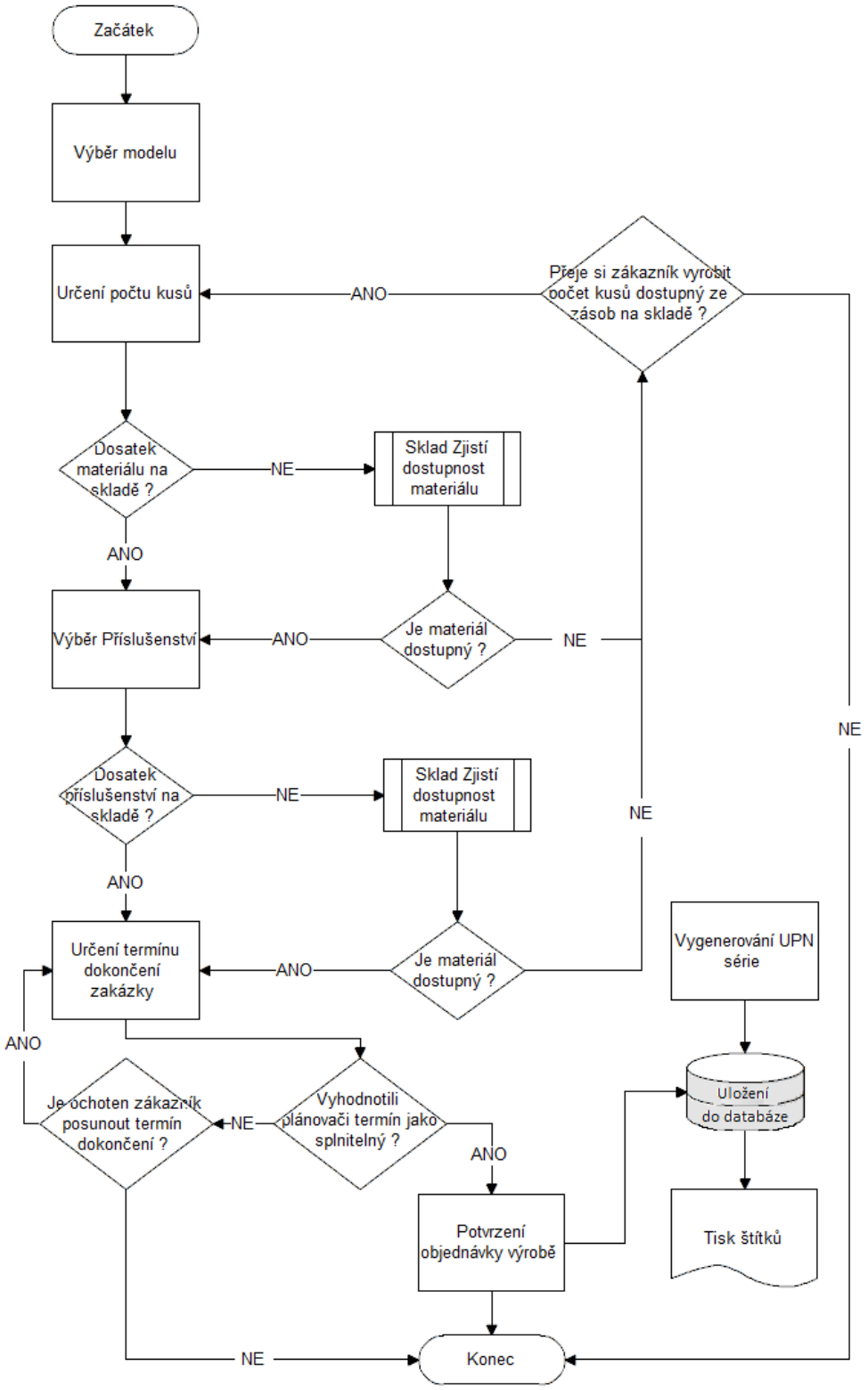
Celá výroba z hlediska zaznamenávání dat do databáze se dá rozdělit na čtyři základní procesy, které jsou sestavení výrobní objednávky MO (manufacturing order), dále pak rozložení pracovních stanic na lince a jejich přiřazení výrobní objednávce, třetím procesem je samotná výroba a posledním je opravný proces v případě že při výrobě televize dojde k zjištění závady.

3.1.1 Proces sestavení výrobní objednávky

Výrobní objednávka je požadavek zákazníka na výrobu určitého počtu kusů daného modelu televize, který si vybere. Určí si také, jaké příslušenství chce, aby bylo v balení obsaženo, a do kdy má být objednávka dokončena. V databázi tedy musí být veden seznam modelů televizí, které se v závodě vyrábějí. Seznam modelů pochází od oddělení vývoje a o zavedení do výroby se pak stará oddělení, které má na starosti výrobní proces. V databázi tedy musí být veden seznam modelů, a určité informace pro jejich výrobu, přičemž tím hlavním je velikost úhlopříčky aby nedošlo k tomu, linka bude sestavena nevhodně a příliš velká televize jí neprojde.
Pro jednotlivé modely se můžou lišit komponenty, ze kterých se televize sestavuje, případně i pro stejný model ale jinou objednávku pokud si to zákazník vyžádá a pokud existují zaměnitelné komponenty. Každá, ze čtyř hlavních komponent, má na sobě štítek s čárovým kódem. Ten má na sobě jedinečné identifikační číslo, které zároveň obsahuje vzor, podle kterého se dá určit typ komponenty. V databázi tedy musí být tento seznam veden, aby se zabránilo tomu, že bude televize sestavena ze špatných součástí.
Jak již bylo řečeno, zákazník si vybírá také příslušenství, které bude součástí balení, musí tedy existovat seznam veškerého příslušenství, ze kterého zákazník potencionálně

vybírá. Později ve výrobě, než se televize zabalí, bude také potřeba zkontrolovat, že televize obsahuje veškeré příslušenství a že nedošlo k záměně, jak se již v minulosti díky nedostatkům v databázi stalo, například v podobě špatného napájecího kabelu nepoužitelného v zemi, kam se televize dovezly.

Tím posledním, co je s výrobní objednávkou spojeno, je vygenerování jedinečných identifikačních čísel pro každou televizi, aby se ještě před započetím výroby mohly, vytisknou identifikační štítky. Někdy se stane, že se některý ze štítků ztratí, tudíž je potřeba aby bylo pomocí databáze možné vyhledat, ke kterému štítku nebyly přiřazeny ještě žádné komponenty, a tento ztracený štítek se pak vytiskne znovu.

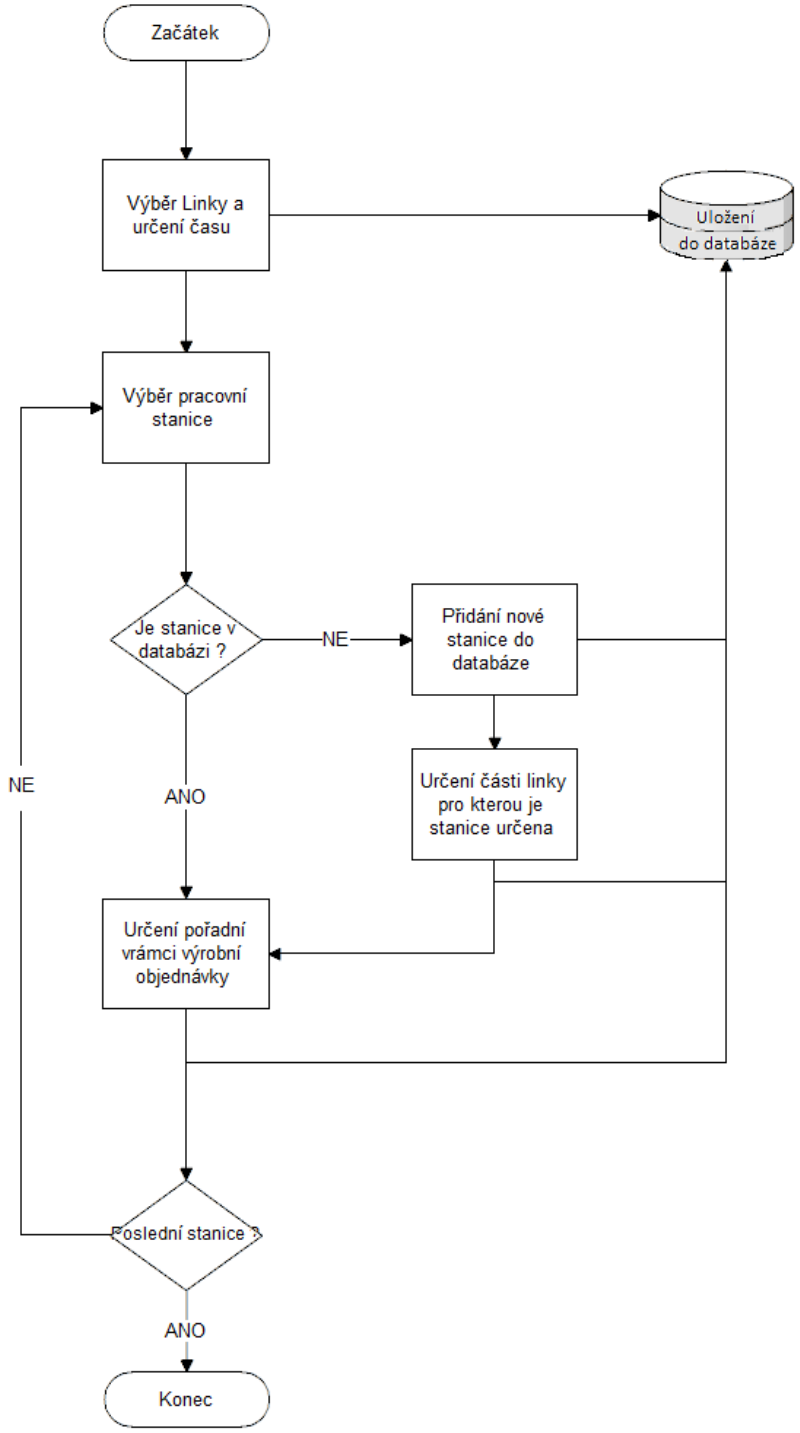


Obr. 5: Vývojový diagram procesu sestavení výrobní objednávky

(Zdroj: vlastní zpracování)

3.1.2 Proces rozložení pracovních stanic na lince a jejich přiřazení výrobní objednávce

V momentě kdy je výrobní objednávka schválena pro výrobu, tak je jí přiřazena linka, na které se výroba provede. V současné době je k dispozici 8 linek, ze kterých se vybírá. Linka se přidělí na určitou dobu, za kterou by mněly být všechny televize zhotoveny. Výběr linky a času, po který bude objednávce přiřazena, určují plánovači. Ti jsou zodpovědní za to, aby se výrobní kapacita továrny využívala co nejefektivněji a objednávky byly dokončovány v co nejkratších časech.
Po přidělení linky je potřeba určit rozmístění pracovních stanic. To se dělá pro každou výrobní objednávku. Na to je zde oddělení, které se zabývá výrobním procesem a snaží se, aby výroba probíhala co nejoptimálněji. Proto i pokud se vyrábí stejný model televize, tak se může rozmístění stanic na lince lišit v důsledku pokusu o lepší optimalizaci výroby.
Je tedy potřeba evidovat všechny možné pracovní stanice, ze kterých se vybírá při sestavování linky. U stanic je pak popsáno jaká činnost se na ní provádí a do jaké ze tří základních částí (sestavení, testování, balička) linky patří. I stanice jsou občas v rámci inovace výrobního procesu vytvořeny nové, a je potřeba je pak zaevidovat do databáze. Nejčastěji se jedná o stanice v testovací fázi a na baličce.
Požadavkem na databázi je tedy, aby umožňovala pružné sestavení rozložení linky a nikoliv pouze jedno fixní uspořádání.

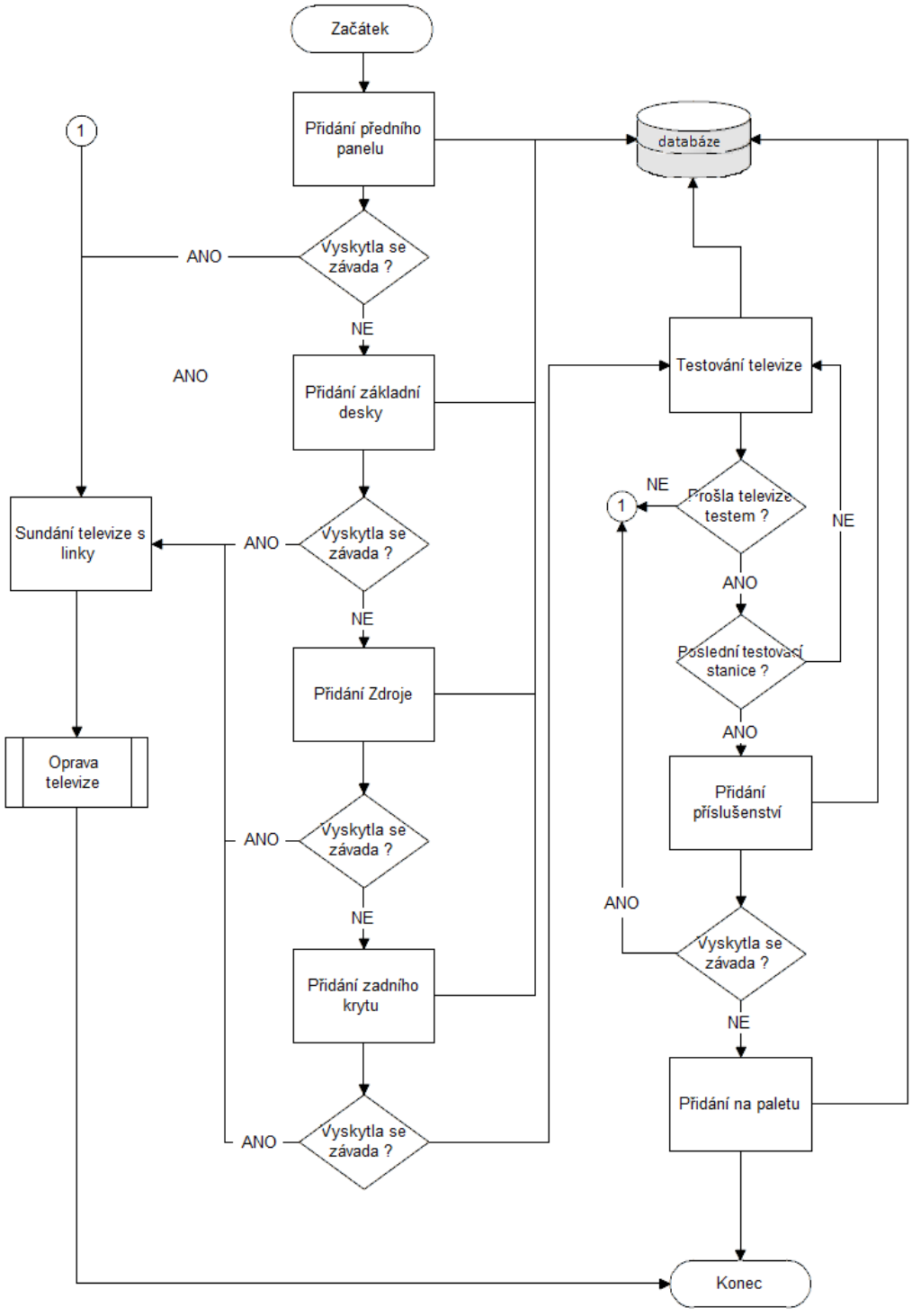


Obr. 6: Vývojový diagram procesu připravení výrobní linky

(Zdroj: vlastní zpracování)

3.1.3 Proces výroby

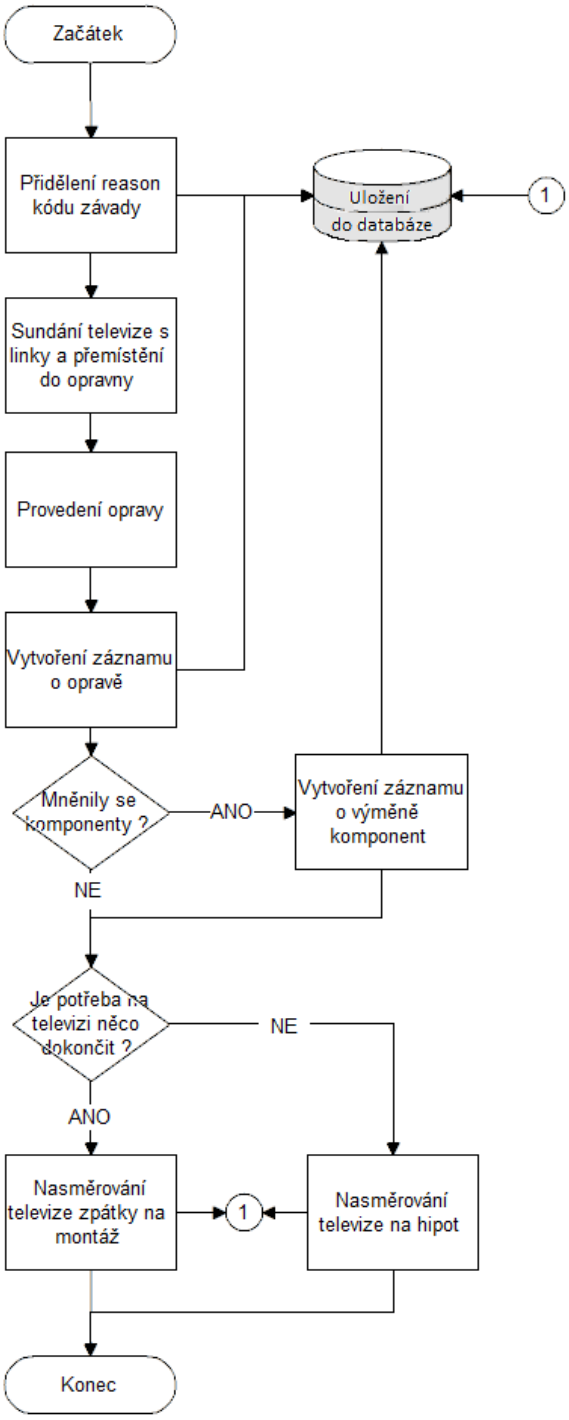
Po připravení výrobní linky je spuštěna výroba. Prvním dílem, který se na linku dostane, je přední panel televize a po jeho propojení s identifikačním štítkem pak pokračuje výroba televize skrz celou linku. Z montážní části linky je potřeba uložit informace o načtených komponentách a propojit je s identifikačním číslem televize.
Z části linky pro testy je potřeba uložit hodnoty naměřených testů a zároveň zajistit aby aplikace, co provádí testy, měly pokud je to potřeba, k dispozici seznam povolených, případně zakázaných hodnot testů. Tyto hodnoty se poté porovnají s naměřenými a následně se vyhodnotí, jestli televize prošla testem.
Po dokončení testů jde televize na baličku, kde se přidají komponenty, vše se zabalí a následně přidá na paletu.



Obr. 7: Vývojový diagram procesu výroby (Zdroj: vlastní zpracování)

3.1.4 Proces oprav

Proces opravy zahrnuje sundání televize s linky, její opravu, vytvoření záznamu a následné vrácení zpátky na linku.



Obr. 8: Vývojový diagram procesu oprav (Zdroj: vlastní zpracování)

3.2 Konceptuální návrh

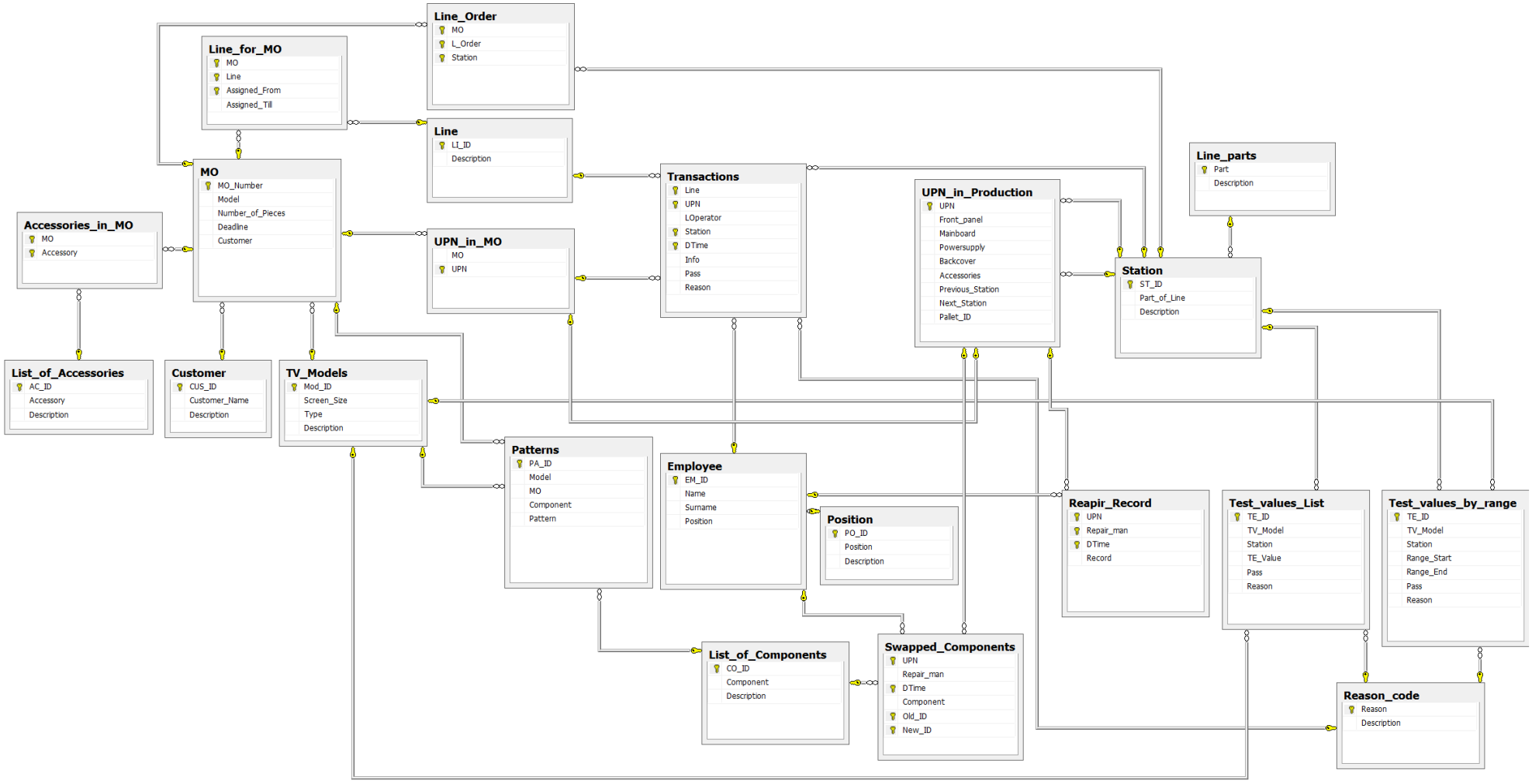
V konceptuální části návrhu je účelem navrhnout ERD diagram, v něm identifikujeme entity, relace, atributy a atributy které se stanou primárními klíči. Nakonec zkontrolujeme redundanci a posoudíme celkový návrh.
Na základě výše popsaných procesů a rozkreslení ve vývojových diagramech jsem identifikoval následující entity. Jelikož ve výsledné databázi budou názvy tabulek anglicky, jsou v tabulce uvedeny i překlady a případně zkratky, pokud jsou používány.

Tab. 1: Zachycené entity (Zdroj: vlastní zpracování**)**

Český název	**Anglický překlad**	**Anglická zkratka**
Linka	Line	
Části linky	Line parts	
Postup po lince	Line order	
(Pracovní) stanice	Station	
Výrobní objednávka	Manufacturing order	MO
Model televize	TV model	
Příslušenství	Accessories	
Komponenta	Component	
Vzor komponenty	Pattern	
Televize	Unit production number	UPN
Test televize	Test	
Důvod závady	Reason kód	
Zaměstnanec	Employee	
Oprava	Repair	
Vyměněné komponenty	Swapped components	
Záznam o opravě	Repair record	

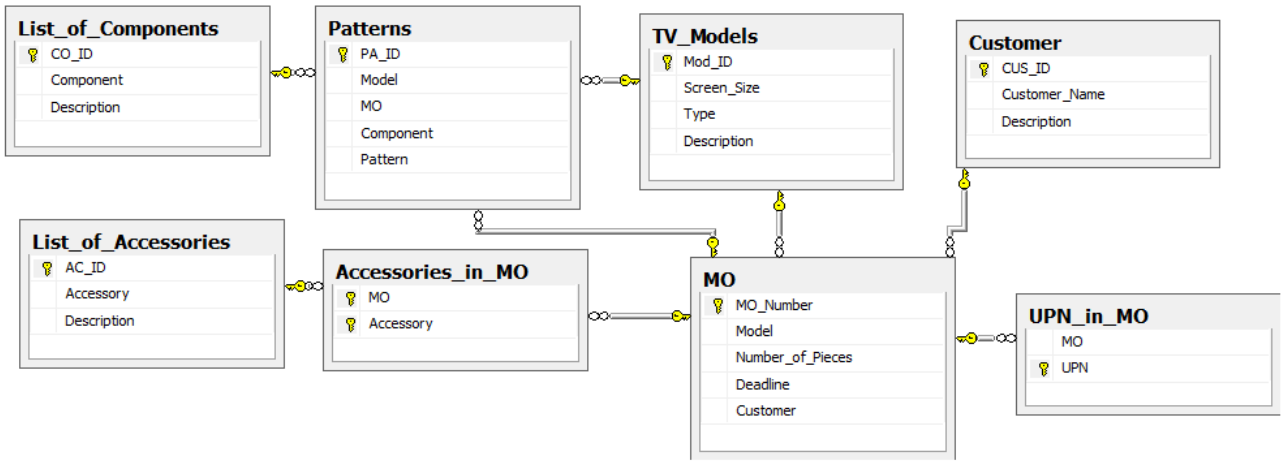
3.3 Logický návrh

V logické fázi návrhu transformuje konceptuální návrh na množinu relačních tabulek. Strukturu návrhu ověříme za pomoci normalizace sloužící k zamezení redundance dat. Dále pak formulujeme integritní omezení.



Obr. 9: Logický návrh databáze (Zdroj: vlastní zpracování)

3.3.1 Sestavení výrobní objednávky



Obr. 10: Sestavení výrobní objednávky (Zdroj: vlastní zpracování)

Tabulka MO

Výrobní objednávka je zachycena v tabulce MO, má jako svůj primární klíč MO_Number, což je jedinečné identifikační číslo objednávky. Dále obsahuje sloupec model, který reprezentuje model televize který, si zákazník vybral. Poté je zde počet kusů (Number_of_Pieces), které se mají vyrobit, datum požadovaného dokončení (Deadline) a nakonec identifikace zákazníka.

Tabulka Customer

V této tabulce je seznam zákazníku, přestože je v současné době výroba pouze pro jednoho zákazníka umožňuje to rozšíření později hladký průběh objednávek pro další potencionální zákazníky, a navíc je zaručena referenční integrita aby nedocházelo k chybám a překlepům při vyplňování této položky v tabulce MO.

Tabulka List_of_Accessories

V této tabulce je uvedeno veškeré možné příslušenství, které je možné k televizi vybrat.

Tabulka Accessories_in_MO

Zde je uveden seznam příslušenství, které si zákazník pro danou objednávku vybral.

Tabulka TV_Models

Tato tabulka obsahuje seznam vyráběných modelů. Obsahuje položku Mod_ID, což je její primární klíč, který zároveň slouží jako odkaz pro cizí klíče v ostatních tabulkách. Důležitou položkou se zde úhlopříčka (Screen_Size), aby ve výrobě věděli, jak velkou televizi budou vyrábět, a nedošlo k problémům.

Tabulka Patterns

V této tabulce jsou vzory pro identifikaci komponent. Jedná se v podstatě o vstupní masku, podle které se ověří, že skenovaná komponenta, kterou operátor montuje do televize, skutečně náleží pro daný model, případně objednávku.

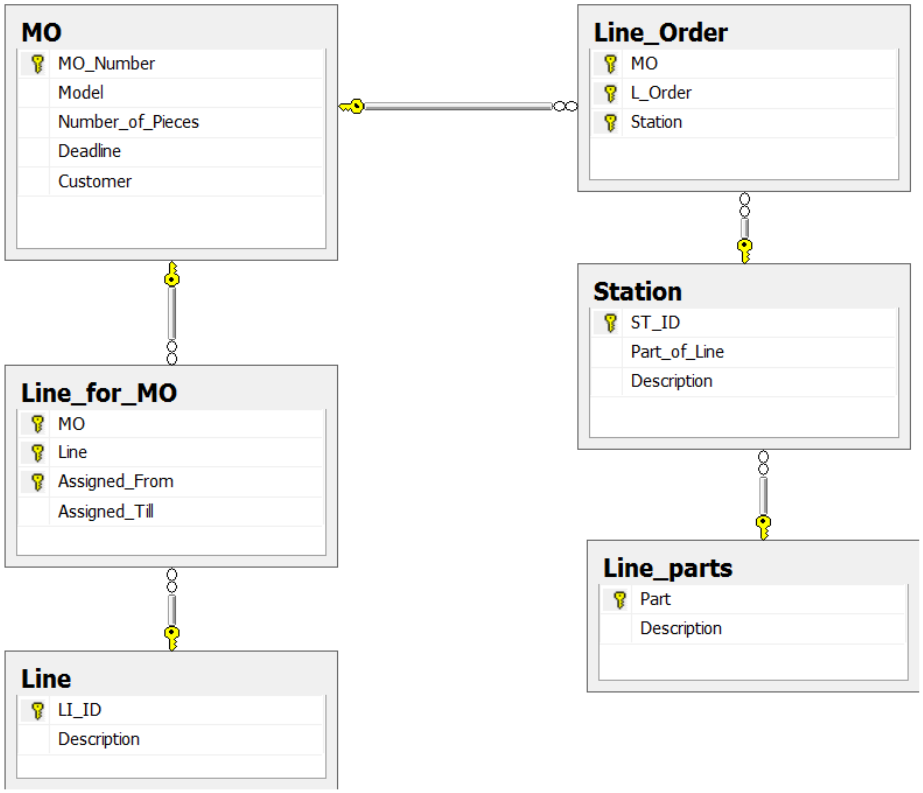
Tabulka List_of_Components

Zde je uložen seznam základních typů komponent, které se do televize montují.

Tabulka UPN_in_MO

Tato tabulka slouží pro uložení seznamu vygenerovaných sériových čísel televizí (UPN) pro danou objednávku. Čísla jsou vygenerována aplikací k tomu určenou a po jejich uložení do databáze se vytisknou v podobě štítků.

3.3.2 Příprava a rozložení linky



Obr. 11: Příprava a rozložení linky (Zdroj: vlastní zpracování)

Tabulka Line_parts

Zde je zachycené rozložení linky na tři základní části, a navíc jsou zde také uvedeny části, které se nenachází přímo na lince jako například opravna.

Tabulka Station

Tady je zachycen seznam všech pracovních stanic, které se nacházejí na lince případně mimo ni. Sloupec Part_of_line slouží jako cizí klíč, aby bylo jasné, pro kterou část linky je stanice určena.

Tabulka Line_Order

V této tabulce je zachycený postup televize po lince pro každou výrobní objednávku. Sloupec Station určuje stanici a sloupec L_Order její pořadní v rámci postupu po lince. Sloupec MO určuje pro jakou objednávku je tento postup určen.

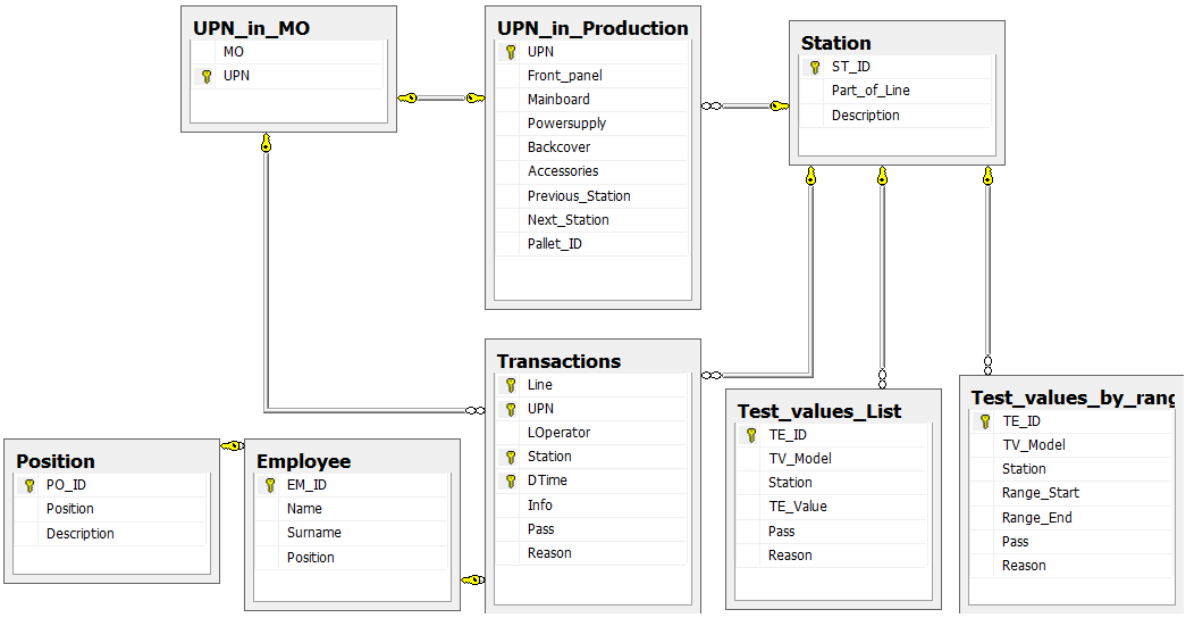
Tabulka Line

Tato tabulka obsahuje seznam linek, na kterých probíhá výroba.

Tabulka Line_for_MO

Zde je určeno přidělení linky pro výrobní objednávku. Sloupec Line určuje, která linka byla přidělena, a sloupec Assigned_From a Assigned_Till určují od kdy do kdy, byla linka přidělena. Sloupec MO pak, pro kterou výrobní linku je připravena.

3.3.3 Výroba



Obr. 12: Výroba (Zdroj: vlastní zpracování)

Tabulka UPN_in_Production

Tato tabulka obsahuje seznam UPN, které se dostaly do produkce. Dále obsahuje čtyři sloupce (Front_Panel, Mainboard, Powersupply, Backcover), tedy čtyři základní komponenty co se do televize montují. Do nich se ukládají identifikační čísla daných komponent. Tyto údaje jsou sice redundantní, protože se dají vyčíst z tabulky transakcí, ale jelikož ta obsahuje nejvíc dat a komponenty se můžou měnit a tím by při

vyhledávání docházelo k nutnosti použít agregační funkce. Další položkou je sloupec Accesories kam se potvrdí, že jednotka obsahuje všechno příslušenství. Další dva sloupce jsou Previous_Station a Next_Station které určují předchozí a následující stanici, což je podstatné pokud je výroba přerušena a televize sundána z linky nebo pokud se její trasa oproti předpokládané změní v důsledku opravy. Posledním sloupcem je číslo palety, na které televize skončila.

Tabulka Employee

Tato tabulka obsahuje seznam zaměstnanců ve výrobě. Každý zaměstnanec je identifikován svým číslem, které má zároveň na své identifikační kartě. Dále je zde uvedeno jeho jméno, příjmení a pozice. Více informací není potřeba, neboť se nejedná o primární seznam zaměstnanců.

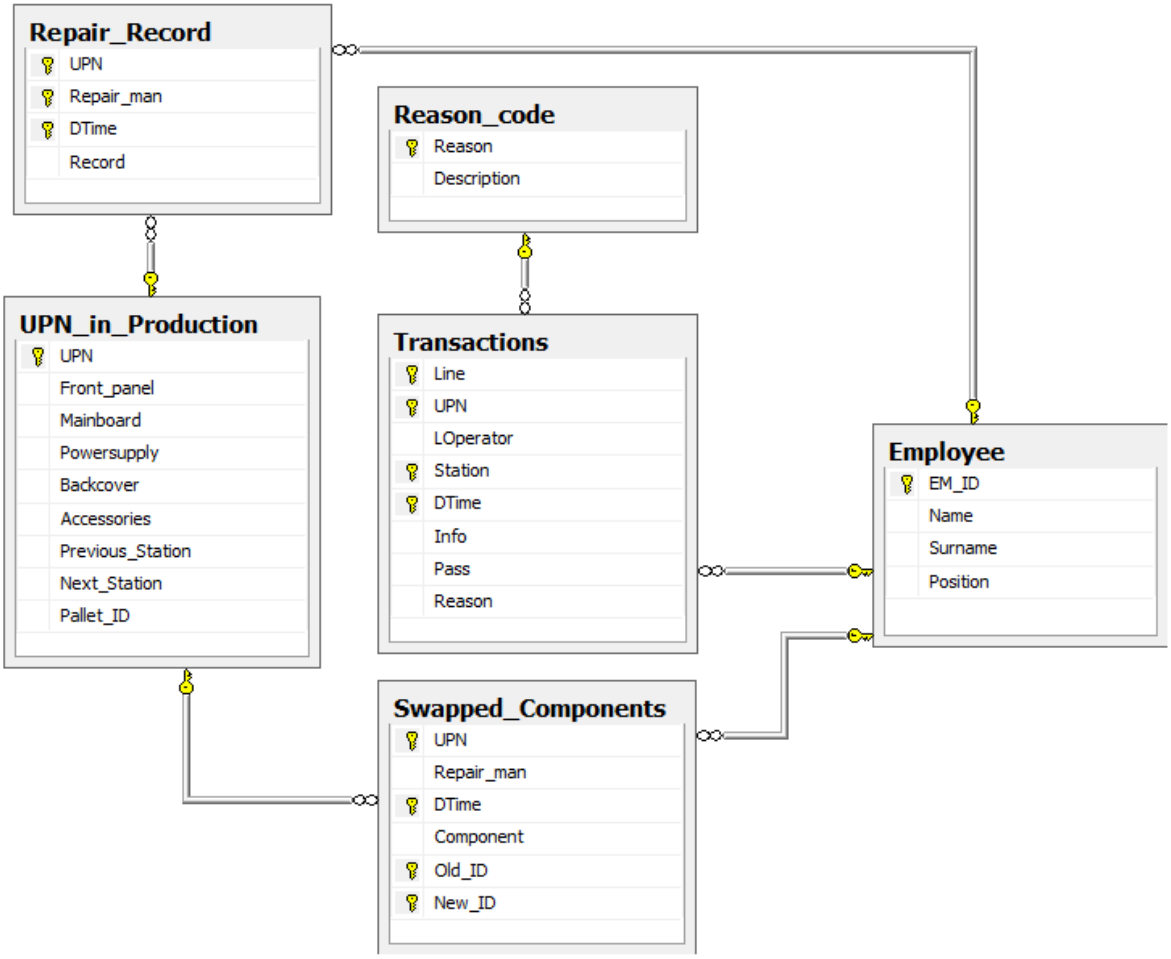
Tabulka Position
Zde je uveden seznam všech pozic, na kterých můžou být zaměstnanci ve výrobě.

Tabulka Test_values_List
Zde je seznam přípustných, případně nepřípustných hodnot pro výsledek testu na určité stanici pro daný model televize.

Tabulka Test_values_by_range
V této tabulce je seznam přípustných, případně nepřípustných rozsahů hodnot pro výsledek testu na určité stanici pro daný model televize.

Tabulka Transactions
Celý průběh výroby televize je zachycen v této tabulce. Obsahuje sloupec pro určení linky po které se tabulka pohybovala, operátora který se na výrobě podílel, stanice přes které televize prošla, čas a datum kdy která operace proběhla, to jestli televize danou stanicí bez závady prošla a pokud ne tak obsahuje důvod selhání.

3.3.4 Opravy



Obr. 13: Opravy (Zdroj: vlastní zpracování)

Tabulka Reason_code

Obsahuje seznam možných závad, které při výrobě mohou nastat.

Tabulka Repair_Record

Pokud dojde k opravě televize je o tom proveden záznam. V něm se uloží UPN televize, dále pak identifikační číslo opraváře který opravu provedl, čas kdy byla oprava provedena a nakonec záznam toho co bylo opravováno.

Tabulka Swapped_Components

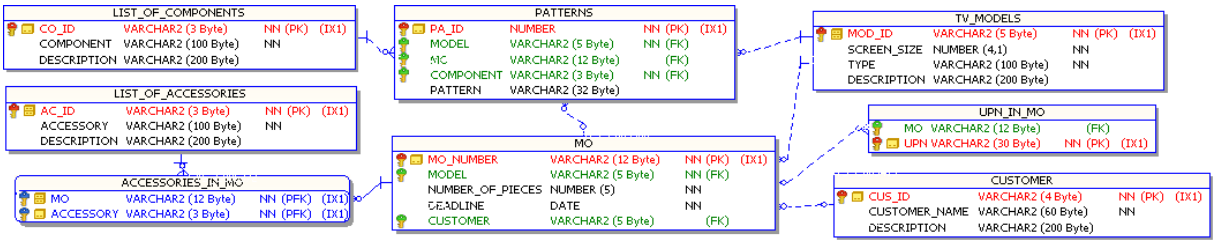
Pokud nešlo některou komponentu opravit tak ji opravář vymění za novou. Do záznamu se uloží UPN televize, dále pak identifikační číslo opraváře který opravu provedl, čas kdy byla oprava provedena, typ komponenty která se měnil a nakonec ID staré a nové komponenty.

3.4 Fyzický návrh

V této části bude logický návrh převeden na fyzický. Pro jednotlivé tabulky budou určeny datové typy. Dále pak budou určeny indexy, pohledy a procedury.
Převedení logického schématu do fyzického na platformě Oracle 10g je zachyceno na následujících čtyřech schématech, které reprezentují čtyři procesy popsané výše. Celý skript pro vytvoření databáze je v příloze č. 1, skript pro vygenerování cvičných dat v příloze č. 2 a datový slovník pak v příloze č. 3.

3.4.1 Sestavení výrobní objednávky

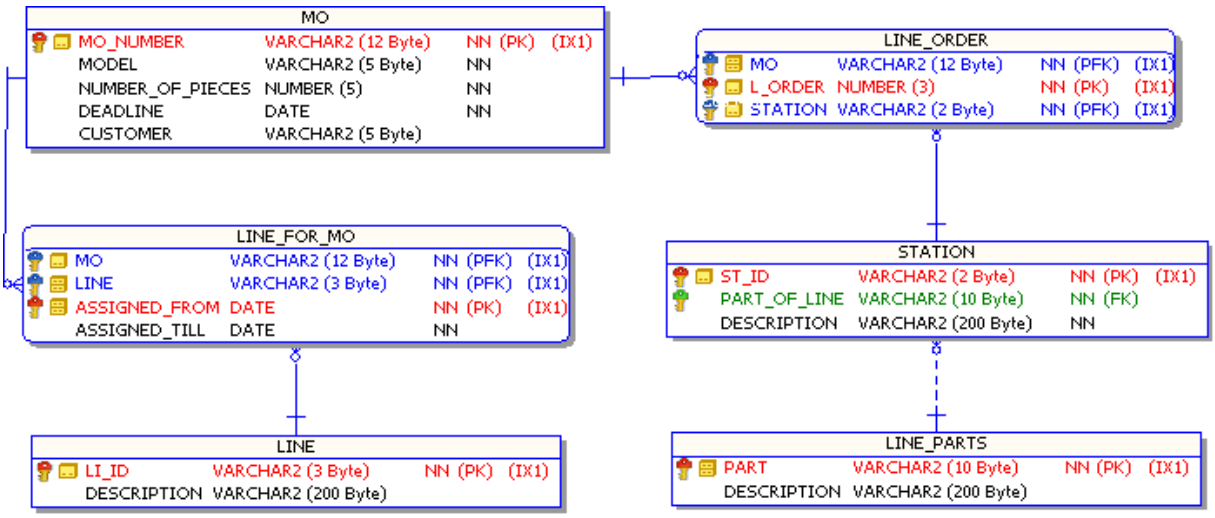
V tabulce MO byl zvolen datový typ varchar2 o délce 12 znaků pro identifikaci výrobní objednávky. Délka 12 byla zvolena proto, že pro identifikaci objednávky se používá kód složený z identifikace zákazníka, data přijetí a sériového čísla. Ostatní datové typy jsou popsány v obrázku číslo 14.



Obr. 14: Fyzický návrh pro výrobní objednávku (Zdroj: vlastní zpracování)

3.4.2 Sestavení linky

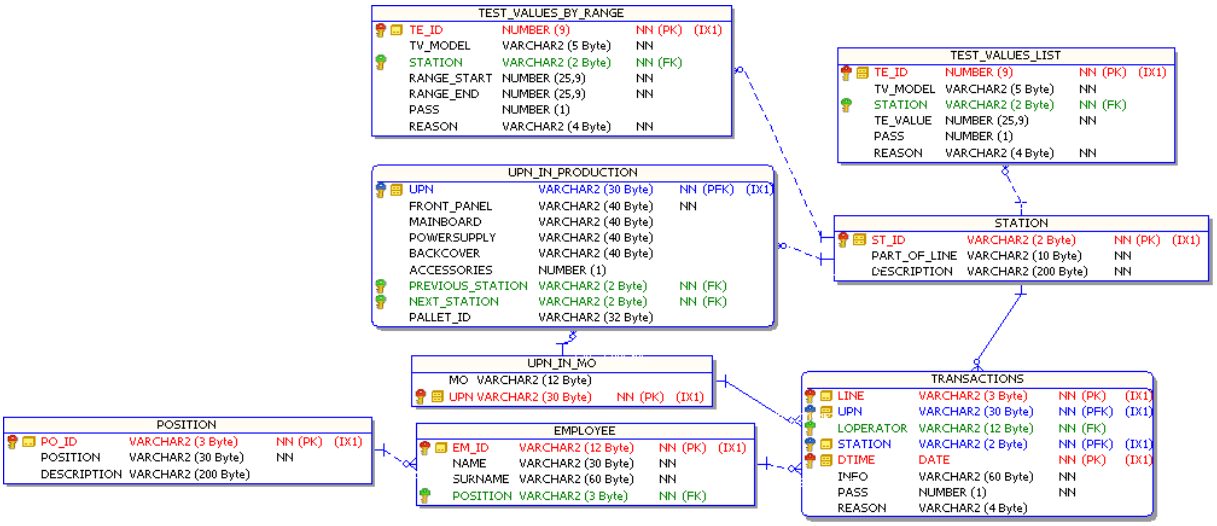
V tabulce station je identifikace stanice určena typem varchar2 o dvou znacích, kde obvykle první slouží pro označení části linky, do které stanice patří a druhý je už jen abecední pořadí. Pro počet kusů ve výrobní objednávce je použit číselný typ number o délce 5 což umožňuje rozsah od 1 do 99 999, což pokrývá obvyklý počet kusů v objednávce, který se pohybuje od několika set do pár tisíc jednotek. Ostatní zvolené datové typy jsou vidět na obrázku číslo 15.



Obr. 15: Fyzický návrh sestavení linky (Zdroj: vlastní zpracování)

3.4.3 Výroba

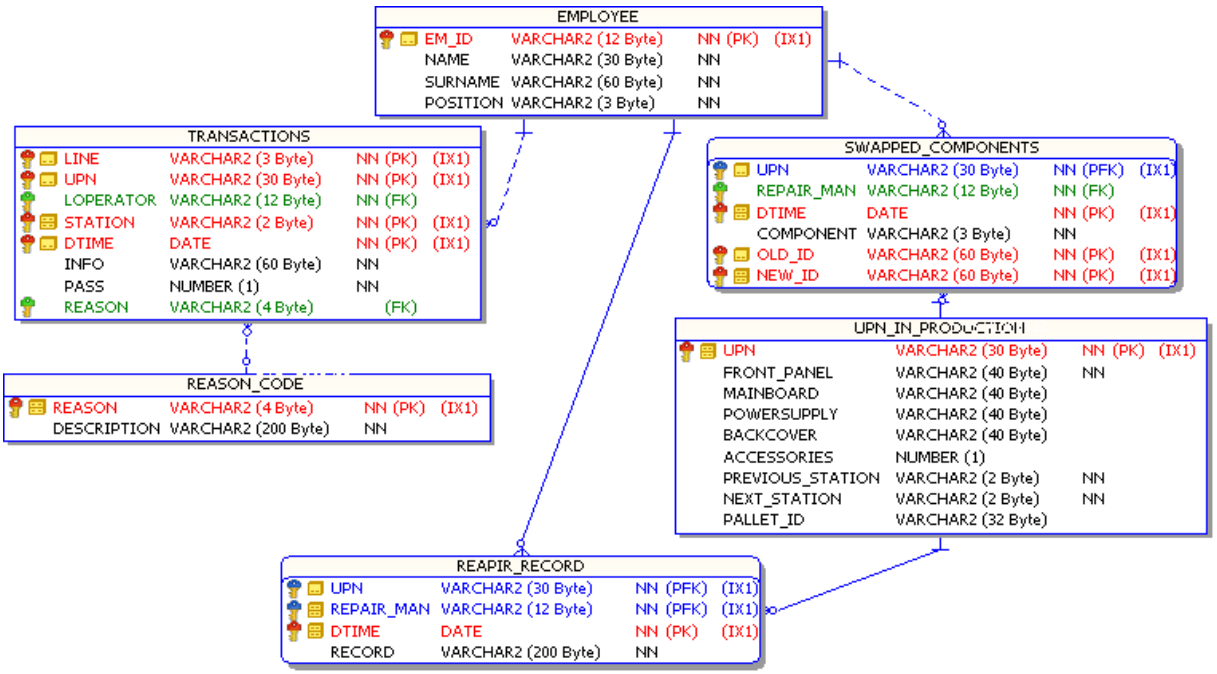
Pro identifikační číslo televize UPN v tabulce UPN_in_Production je opět zvolen datový typ varchar2 tentokrát o délce 30 znaků. Pro identifikaci komponent je zvolena délka 40 znaků a to z důvodu, že délka identifikačního čísla se může podle výrobců někdy lišit, proto byla zvolena jistá rezerva. Pro zjištění že bylo přidáno všechno příslušenství je použit číselný typ number o délce jedna, z důvodu že Oracle nezná typ Boolean. Ostatní datové typy jsou zachyceny na obrázku číslo 16.



Obr. 16: Fyzický návrh pro výrobu (Zdroj: vlastní zpracování)

3.4.4 Opravy

Poslední částí návrhu jsou opravy. Zde opravář zapíše popis závady. Pro tento popis slouží pole o délce 200 znaků, kam opravář napíše krátký report toho, co na televizi opravil. Ostatní datové typy jsou zachyceny na obrázku číslo 17.



Obr. 17: Fyzický návrh pro opravy (Zdroj: vlastní zpracování)

3.4.5 Pohledy

Tento pohled slouží k zobrazení aktuální produkce na lince, seskupené podle linky, výrobní objednávky a poté obsahuje i celkové součty. Obsahuje také počet závad, které se na lince vyskytly a jejich procentuální podíl na dané výrobě. Tento pohled primárně slouží vedoucím pracovníkům ve výrobě ke kontrole dodržování výrobního plánu. Zároveň se promítá pomocí televizí rozmístěných kolem výrobní linky, aby i pracovníci na lince viděli, jak jsou na tom s produkcí.

```
Create View Production as (
-----------------------------------------------------------------------------------------------
select line , mo, sum (case when station = 'AD' then 1 else 0 end ) AA ,
sum (case when station = 'AD' then 1 else 0 end ) AD ,
sum (case when station = 'PP' then 1 else 0 end ) PP ,
sum(case when pass = 1 then 1 else 0 end) Failed,
sum(case when pass = 1 then 1 else 0 end) / sum(case when station = 'AD' then 1 else 0
end )*100 "Percentage fail rate"
from transactions t join upn_in_mo um on t.upn = um.upn
where dtime between sysdate - 1/24 and sysdate
group by line, mo
-----------------------------------------------------------------------------------------------
union
select line , 'ALL', sum (case when station = 'AD' then 1 else 0 end ) AA ,
sum (case when station = 'AD' then 1 else 0 end ) AD ,
sum (case when station = 'PP' then 1 else 0 end ) PP ,
sum(case when pass = 1 then 1 else 0 end) Failed,
sum(case when pass = 1 then 1 else 0 end) / sum(case when station = 'AD' then 1 else 0
end )*100 "Percentage fail rate"
from transactions t join upn_in_mo um on t.upn = um.upn
where dtime between sysdate - 1/24 and sysdate
group by line
-----------------------------------------------------------------------------------------------
union
select 'ALL' , 'ALL', sum (case when station = 'AD' then 1 else 0 end ) AA ,
sum (case when station = 'AD' then 1 else 0 end ) AD ,
sum (case when station = 'PP' then 1 else 0 end ) PP ,
sum(case when pass = 1 then 1 else 0 end) Failed,
sum(case when pass = 1 then 1 else 0 end) / sum(case when station = 'AD' then 1 else 0
end )*100 "Percentage fail rate"
from transactions t join upn_in_mo um on t.upn = um.upn
where dtime between sysdate - 1/24 and sysdate )
-----------------------------------------------------------------------------------------------
```

Ukázku výsledku tohoto pohledu představuje tabulka číslo 2.

Tab. 2: Ukázku výsledku pohledu Production (Zdroj: vlastní zpracování**)**

Line	MO	AA	AD	PP	Failed	Percentage fail rate
ALL	ALL	216	212	210	6	2,78
L01	10245SON0001	78	76	75	3	3,85
L01	10245SON0002	69	68	68	1	1,45
L01	ALL	147	144	143	4	2,72
L02	10245SON0003	69	68	67	2	2,90
L02	ALL	69	68	67	2	2,90

Další pohled slouží pro získání seznamu chybějícího příslušenství pro jednotlivé televize.

```
Create view Missing_Components as
with CTE as (select UPN, Info
from transactions
where station = 'PA'
union
select UPN, Info
from transactions
where station = 'PB'
) , TAB as (
SELECT UPN,TRIM(REGEXP_SUBSTR( Info, '[^;]+', 1, LEVEL)) Accessory
FROM CTE
CONNECT BY LEVEL <= LENGTH(REGEXP_REPLACE(Info, '[^;]+')) + 1
AND PRIOR UPN        = UPN and  PRIOR Info        = Info
AND PRIOR DBMS_RANDOM.VALUE IS NOT NULL )
,TAB1 as(
select MO,tab.UPN,Accessory
from tab join UPN_in_MO UM on tab.UPN = UM.UPN)

select AM.MO , UM.UPN, ACCESSORY
from ACCESSORIES_IN_MO AM, UPN_in_MO UM
where  AM.MO = UM.MO
minus
select * from tab1
```

3.4.6 Procedury

Následující procedura vychází z předchozího pohledu a vypisuje seznam chybějících komponent pro zadanou televizi.

```
CREATE PROCEDURE Missing_Components_For
 ( SUPN IN varchar2 )
IS
type radek is record (
RUPN upn_in_mo.upn%type,
RMO varchar(50),
RACCESSORY varchar(50));

type tabulkaradku is table of radek ;

tabulka tabulkaradku;

BEGIN

select MO , UPN , ACCESSORY
bulk collect into tabulka
from missing_components
where upn = SUPN;

dbms_output.put_line('Následující jednotce ' || SUPN ||' chybý tyto komponenty:');

for i in tabulka.first .. tabulka.last
loop
dbms_output.put_line(tabulka(i).RACCESSORY);
end loop;

EXCEPTION
   WHEN OTHERS THEN dbms_output.put_line('Nastala chyba, prosím zkontrolujte
zadávané UPN');

END Missing_Components_For;

execute Missing_Components_For('UPN10245SON0001A0006');
```

3.4.7 Indexy

Správné vytvoření indexů je klíčem k optimálnímu fungování databáze. Index se automaticky vytvoří, pokud tabulka obsahuje primární klíč nad sloupcem případně sloupci, ze kterých se primární klíč skládá. Proto všechny tabulky současné databáze index obsahují, ale v některých bude ještě dobré index přidat a tím zrychlit případné vyhledávání.

Tabulka Transactions je co se týká objemu dat největší a je nad ní prováděna řada dotazů, proto je klíčové u ní správně zvolit indexy.

Primární klíč je u ní složen z následujících sloupců UPN, DTime, Station, Line. Oracle umožňuje využít index, i pokud nejsou při dotazu využity všechny sloupce, podmínkou je zde ovšem aby byl vyplněn první sloupec UPN který má vysokou kardinalitu. Poté je možné při vyhledávání přidávat další sloupce v pořadí z leva doprava a index bude plně využit.

Dále se pak přidají bitmapové indexy nad sloupci s nízkou kardinalitou. Jelikož bitmapové indexy je možné na rozdíl od těch B-stromových možné při vyhledávání spojovat.

```
CREATE bitmap INDEX In_Station
ON transactions (station)

CREATE bitmap INDEX In_Line
ON transactions (line)

CREATE bitmap INDEX In_Reason
ON transactions (reason)
```

Nyní je tedy možné použít indexy i při dotazech které nezahrnují konkrétní sloupec s identifikačním číslem televize, jedná se o případ kdy je potřena zjistit například produkci na daných linkách přes jednotlivé stanice.

3.5 Doba ukládání záznamů

Aby se předešlo tomu, že se databáze opět zaplní starými záznamy jako v případě předešlé databáze, tak bude vytvořena identická kopie této databáze, kam se budou převádět všechny záznamy o televizích, které byly vyrobeny a jsou starší jednoho měsíce. Tento převod bude probíhat v noci kdy je k dispozici volný výkon databázového serveru. Současně bude zachována i stará databáze, pro případ kdy by se společnost někdy v budoucnu rozhodla zavést datový sklad a tyto data potřebovala. Zároveň by se ze staré databáze do nové převedla část dat nezbytných pro plynulé dokončení rozdělané výroby.

3.6 Přínosy návrhu pro společnost

Nově navržená databáze splňuje všechny nároky, které na ní byly kladeny z hlediska zachycení dat z výroby. Představuje flexibilní a komplexní řešení, které umožňuje zaznamenávat data z měnícího se procesu výroby, aniž by bylo nutné ji dále fyzicky upravovat nebo rozšiřovat. Oproti stávající databázi má následující výhody:

- Přehlednější
- Menší počet tabulek
- Žádné zbytečné fyzické tabulky
- Nižší náklady na ukládání
- Cizí klíče zajišťující referenční integrity a viditelnost vazeb mezi tabulkami
- Každá tabulka má primární klíč
- Zavedení pohledů místo vytváření fyzických tabulek
- Správně nastavené indexy
- Vyřešenou problematiky starších záznamů
- Celkový koncept umožňující efektivní údržbu databáze

ZÁVĚR

Cílem této bakalářské práce bylo vytvoření návrhu databáze pro výrobní společnost zabývající se výrobou LCD televizí. Databáze měla být schopna zachytit celý výrobní proces a uložit z něj všechna požadovaná data. Měla se také vyvarovat nedostatků, kterými trpěla předešlá databáze. Toho bylo dosaženo na základě analýzy současné situace a rozkreslení jednotlivých procesů, které ve výrobě probíhají. A poté postupným přechodem od konceptuálního návrhu, přes logický až k fyzickému návrhu.

Tato bakalářská práce tedy splnila svůj cíl a navržená databáze splňuje všechny požadavky, které na ní byly kladeny jak z hlediska zachycení výrobního procesu, tak z hlediska efektivnosti a přehlednosti návrhu.

SEZNAM POUŽITÉ LITERATURY

[1] ŠIMŮNEK, M. *SQL Kompletní kapesní průvodce*. Praha: GRADA, 1999. ISBN 80-7169-692-7.

[2] LACKO, L. ORACLE *Správa, programování a použití databázového systému*. Vyd. 2. Brno: Computer Press, 2007, 576 s. ISBN 978 80-251-1490-2.

[3] TELNEROVÁ, Z. Relační Databáze [online]. Ostravská univerzita, Přírodovědecká fakulta: 2006 [cit. 2014-03-18]. Dostupný na http://fakulty.osu.cz/prf/rsk/uploaded/1942_XRELA1.pdf

[4] OČENÁŠEK, V. Klíče a relace [online]. Česká zemědělská univerzita v Praze, Katedra informačních technologií: 2008 [cit. 2014-03-18]. Dostupný na http://www.kubenka.org/PEF/1_rocnik/Internetove_technologie/tema_2_opory_a_soubory/cviceni_2_2_klice_a_relace.pdf

[5] Spsepn. spsepn.edu.sk Základy práce s SQL databázemi [online] Dostupný na http://www.spsepn.edu.sk/skola/pk_info/studium/ucebtext/ele/rozne/sql_man/mysql_cz1/mysql_cz1.htm

[6] ZENDULKA, J. Databázové systémy a návrh databází [online]. Vyskoké učení technické v Brně, Fakulta informačních technologií: 2010 [cit. 2014-03-18]. Dostupný nahttp://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_kmod.pdf

[7] Programujte. Normalizace relačních databází. *programujte.com* [online] Dostupný na WWW:http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/

[8] ZONER software. Databáze a jazyk SQL. *interval.cz* [online] Dostupný na WWW: http://interval.cz/clanky/databaze-a-jazyk-sql/

[9] W3Schools. w3schools.com SQL [online] Dostupný na http://www.w3schools.com/sql/

[10] Oracle. docs.oracle.com Oracle [online] Dostupný na http://docs.oracle.com/

[11] Techonthenet. techonthenet.com SQL [online] Dostupný na http://www.techonthenet.com/oracle/index.php

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

PŘÍLOHY

Příloha č. 1: Skript pro vytvoření databáze

Příloha č. 2: Skript pro naplnění databáze cvičnými daty

Příloha č. 3: Datový slovník

Příloha č. 1: Skript pro vytvoření databáze

```
create table List_of_Accessories (
AC_ID varchar2(3) Primary key,
Accessory varchar2(100) Not null,
Description varchar2(200)  );
---------------------------------------------
create table TV_Models (
Mod_ID varchar2(5) Primary key,
Screen_Size number(4,1) not null,
Type varchar2(100) not null,
Description varchar2(200) );
---------------------------------------------
Create table Customer (
CUS_ID varchar2(4) primary key,
Customer_Name varchar2(60) not null,
Description varchar2(200) );
---------------------------------------------
create table MO (
MO_Number Varchar2(12) Primary key,
Model varchar2(5) REFERENCES TV_Models(Mod_ID) not null,
Number_of_Pieces number(5) not null,
Deadline date not null,
Customer varchar2(4) REFERENCES Customer(CUS_ID) not null);
---------------------------------------------
create table Accessories_in_MO (
MO varchar2(12) references MO(MO_Number),
Accessory varchar2(3) references List_of_Accessories(AC_ID) not null,
Primary key (MO,Accessory) );
---------------------------------------------
create table List_of_Components(
CO_ID varchar2(3) Primary key,
Component varchar (100) not null,
Description varchar2(200)  );
---------------------------------------------
create table Patterns (
PA_ID number primary key,
Model varchar2(5) REFERENCES TV_Models(Mod_ID) not null,
MO Varchar2(12) references MO(MO_Number) not null,
Component varchar2(3) references List_of_Components(CO_ID) not null,
Pattern varchar2(32) );
---------------------------------------------
create table Line (
LI_ID varchar(3) primary key,
Description varchar2(200)  );
---------------------------------------------
create table Line_for_MO (
MO varchar2(12) references MO(MO_Number) not null,
Line varchar2(3) references Line(LI_ID) not null,
Assigned_From date not null,
Assigned_Till date not null,
primary key (MO,Line,Assigned_From) );
---------------------------------------------
create table Position (
PO_ID varchar2(3) primary key,
```

```
Position varchar2(30) not null,
Description varchar2(200) );
---------------------------------------------
create table Employee (
EM_ID varchar2(12) primary key,
Name varchar2(30) not null,
Surname varchar2(60) not null,
Position varchar2(3) references position(PO_ID) not null );
---------------------------------------------
create table Reason_code (
Reason varchar(4) primary key,
Description varchar(200) not null );
---------------------------------------------
create table Line_parts (
Part varchar2(10) primary key,
Description varchar2(200) );
---------------------------------------------
create table Station (
ST_ID varchar2(2) primary key,
Part_of_Line varchar2(10) references Line_parts(part) not null,
Description varchar(200) not null );
---------------------------------------------
create table Line_Order (
MO varchar2(12) references MO(MO_Number),
L_Order number(3) not null,
Station varchar(2) references Station(ST_ID) not null,
primary key(MO,L_Order,Station) ) ;
---------------------------------------------
create table UPN_in_MO (
MO varchar2(12) references MO(MO_Number),
UPN varchar2(30) primary key )  ;
---------------------------------------------
create table Test_values_by_range (
TE_ID number(9) primary key,
TV_Model varchar2(5) REFERENCES TV_Models(Mod_ID) not null,
Station varchar(2) references Station(ST_ID) not null,
Range_Start number(25,9) not null,
Range_End number(25,9) not null,
Pass number(1),
Reason varchar(4) references Reason_code(Reason) not null );
---------------------------------------------
create table Test_values_List (
TE_ID number(9) primary key,
TV_Model varchar2(5) REFERENCES TV_Models(Mod_ID) not null,
Station varchar(2) references Station(ST_ID) not null,
TE_Value number(25,9) not null,
Pass number(1),
Reason varchar(4) references Reason_code(Reason) not null );
---------------------------------------------
create table UPN_in_Production (
UPN varchar2(30) references UPN_in_MO(UPN) primary key,
Front_panel varchar2(40) not null,
Mainboard varchar2(40) ,
Powersupply varchar2(40) ,
Backcover  varchar2(40) ,
```

```
Accessories number(1) ,
Previous_Station varchar2(2) references station(ST_ID) not null,
Next_Station varchar2(2) references station(ST_ID) not null,
Pallet_ID varchar2(32)  );
 ---------------------------------------------
create table Transactions (
Line varchar2(3) references Line(LI_ID) not null,
UPN varchar2(30) references UPN_in_MO(UPN) not null,
LOperator  varchar2(12)  references Employee (EM_ID) not null,
Station varchar2(2) references station(ST_ID) not null,
DTime date not null,
Info varchar2(60) not null,
Pass number(1) not null,
Reason varchar(4) references Reason_code(Reason) ,
primary key(UPN,DTime,Station,Line) );
 ---------------------------------------------
create table Swapped_Components (
UPN varchar2(30) references UPN_in_Production(UPN) not null,
Repair_man  varchar2(12)  references Employee (EM_ID) not null,
DTime date not null,
Component varchar2(3) references List_of_Components(CO_ID) not null,
Old_ID varchar2(60) not null,
New_ID varchar2(60) not null,
primary key (UPN,Dtime,Old_ID,New_ID) );
 ---------------------------------------------
Create table Repair_Record (
UPN varchar2(30) references UPN_in_Production(UPN) not null,
Repair_man  varchar2(12)  references Employee (EM_ID) not null,
DTime date not null,
Record varchar2(200) not null,
primary key (UPN,Repair_man,DTime) );
 ---------------------------------------------
```

Příloha č. 2: Skript pro naplnění databáze cvičnými daty

```
Insert into Line values( 'L01', 'Line in the main building');
Insert into Line values( 'L02', 'Line in the main building');
Insert into Line values( 'L03', 'Line in the main building');
Insert into Line values( 'L04', 'Line in the main building');
Insert into Line values( 'L05', 'Line in the main building');
Insert into Line values( 'L06', 'Line in the main building');
Insert into Line values( 'L07', 'Line in the main building');
Insert into Line values( 'L08', 'Line in the main building');

Insert into LIST_OF_ACCESSORIES values( 'BAT', 'AAA Battery', 'Batery for remote control');
Insert into LIST_OF_ACCESSORIES values( 'RMC', 'Remote Control', 'Universal remote control');
Insert into LIST_OF_ACCESSORIES values( 'MAN', 'Manual', 'Universal manual');
Insert into LIST_OF_ACCESSORIES values( 'PC1', 'EU Power Cord', 'Power cord for Europe');
Insert into LIST_OF_ACCESSORIES values( 'PC2', 'UK Power Cord', 'Power cord for  United
Kingdom');
Insert into LIST_OF_ACCESSORIES values( 'PB1',  'Packaging box, size 1', 'Paper packaging
box, size 80x60x15');
Insert into LIST_OF_ACCESSORIES values( 'PB2',  'Packaging box, size 2', 'Paper packaging
box, size 120x90x20');

Insert into TV_MODELS values( '10245', 90, 'LCD90WZ10245', 'LCD TV, screen size 90 cm' );
Insert into TV_MODELS values( '10223', 102, 'LCD102WZ10223', 'LCD TV, screen size 102 cm'
);

insert into Customer values('SON' , 'SONNY' , '');

Insert into MO values( '10245SON0001', '10245',500, to_date('10.5.2014 10:00', 'DD.MM.YYYY
hh24:MI'), 'SON');

Insert into ACCESSORIES_IN_MO values( '10245SON0001', 'BAT');
Insert into ACCESSORIES_IN_MO values( '10245SON0001', 'RMC');
Insert into ACCESSORIES_IN_MO values( '10245SON0001', 'MAN');
Insert into ACCESSORIES_IN_MO values( '10245SON0001', 'PC1');
Insert into ACCESSORIES_IN_MO values( '10245SON0001', 'PB1');

--------------------------------------------------------------------------------
declare
i  number;
cis varchar2(3);
begin
cis:= '';
FOR i IN 1..500
LOOP
cis :=  i;
while length(cis) <> 3
loop
cis :=  '0' || cis;
end loop;
insert into UPN_IN_MO values ('10245SON0001', 'UPN10245SON0001A0' || cis);
END LOOP;
end;
--------------------------------------------------------------------------------
```

```
Insert into LINE_FOR_MO values( '10245SON0001', 'L01', to_date('8.5.2014 10:00',
'DD.MM.YYYY hh24:MI'), to_date('10.5.2014 10:00', 'DD.MM.YYYY hh24:MI'));

Insert into LINE_PARTS values( 'Assembly', 'Part fo assmebling');
Insert into LINE_PARTS values( 'Testing', 'Part for testing');
Insert into LINE_PARTS values( 'Packaging', 'Part for packaging');
Insert into LINE_PARTS values( 'Repair', 'Independent station out of main line');
Insert into LINE_PARTS values( 'Finished', 'Virtual part for finished units');

Insert into STATION values( 'AA', 'Assembly', 'Assembly, Front panel ');
Insert into STATION values( 'AB', 'Assembly', 'Assembly, Mainboard  ');
Insert into STATION values( 'AC', 'Assembly', 'Assembly, Powersupply  ');
Insert into STATION values( 'AD', 'Assembly', 'Assembly, Backcover');
Insert into STATION values( 'HI',  'Testing', 'High voltage testing');
Insert into STATION values( 'TA', 'Testing', 'Graphic testing');
Insert into STATION values( 'TB', 'Testing', 'Graphic testing');
Insert into STATION values( 'TC',  'Testing', 'Graphic testing');
Insert into STATION values( 'PA', 'Packaging', 'Packaging, accessories');
Insert into STATION values( 'PB', 'Packaging', 'Packaging, Paper packaging box');
Insert into STATION values( 'PP', 'Packaging', 'Packaging, Pallet number');
Insert into STATION values( 'RR', 'Repair', 'Repair station');
Insert into STATION values( 'ZZ', 'Finished', 'Virtual station for finished units');

Insert into LINE_ORDER values( '10245SON0001',1, 'AA');
Insert into LINE_ORDER values( '10245SON0001',2, 'AB');
Insert into LINE_ORDER values( '10245SON0001',3, 'AC');
Insert into LINE_ORDER values( '10245SON0001',4, 'AD');
Insert into LINE_ORDER values( '10245SON0001',5, 'HI');
Insert into LINE_ORDER values( '10245SON0001',6, 'TA');
Insert into LINE_ORDER values( '10245SON0001',7, 'TB');
Insert into LINE_ORDER values( '10245SON0001',8, 'TC');
Insert into LINE_ORDER values( '10245SON0001',9, 'PA');
Insert into LINE_ORDER values( '10245SON0001',10, 'PB');
Insert into LINE_ORDER values( '10245SON0001',11, 'PP');

Insert into REASON_CODE values( 'TOUR', 'Test value out of range');
Insert into REASON_CODE values( 'SCAR', 'Scratched');
Insert into REASON_CODE values( 'OTHE', 'Other problem');
Insert into REASON_CODE values( 'OK', 'Passed');

Insert into TEST_VALUES_BY_RANGE values(1, '10245', 'TA',5,20,0, 'TOUR');

Insert into POSITION values( 'OPE', 'Operator', 'Line oprator');
Insert into POSITION values( 'LLE', 'Line leader', '-');
Insert into POSITION values( 'REP', 'Repairman', '-');

Insert into EMPLOYEE values( 'WCZ100000528', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000529', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000428', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000328', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000228', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000128', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000628', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000728', 'Name', 'Surname', 'OPE');
Insert into EMPLOYEE values( 'WCZ100000828', 'Name', 'Surname', 'REP');
```

```
Insert into TEST_VALUES_LIST values(1, '10245', 'TB',5,0, 'TOUR');
Insert into TEST_VALUES_LIST values(2, '10245', 'TC',8,1, 'TOUR');

Insert into LIST_OF_COMPONENTS values( 'FRP', 'Front panel',  '-');
Insert into LIST_OF_COMPONENTS values( 'MAB', 'Mainboard',  '-');
Insert into LIST_OF_COMPONENTS values( 'PWS', 'Power supply',  '-');
Insert into LIST_OF_COMPONENTS values( 'BCV', 'Backcover',   '-');

Insert into PATTERNS values(1, '10245', '10245SON0001', 'FRP', 'FP6A+++@@');
Insert into PATTERNS values(2, '10245', '', 'MAB', 'MB7QW++');
Insert into PATTERNS values(3, '10245', '', 'PWS', 'PW4@@+@');
Insert into PATTERNS values(4, '10245', '', 'BCV', 'B++@@');
--------------------------------------------------------------------------------
declare
i  number;
cis varchar2(3);
upn varchar(40);
cas date:= sysdate;
cas1 date;
cas2 date;
begin
cis := '';
FOR i IN 1..500
LOOP
cis :=  i;
while length(cis) <> 3
loop
cis :=  '0' || cis;
end loop;
upn:= 'UPN10245SON0001A0' || cis;
cas := cas + dbms_random.value(5,10)/24/3600;
Insert into TRANSACTIONS  values( 'L01', upn, 'WCZ100000528', 'AA', cas,'panel' , 0,'' );
cas1 := cas + dbms_random.value(5,10)/24/3600;
Insert into TRANSACTIONS  values( 'L01', upn, 'WCZ100000528', 'AB', cas1,'panel' , 0,'' );
cas2 := cas1 + dbms_random.value(5,10)/24/3600;
Insert into TRANSACTIONS  values( 'L01', upn, 'WCZ100000528', 'AC', cas2,'panel' , 0,'' );
END LOOP;
end;



declare
cas dbms_sql.date_table;
Info dbms_sql.varchar2_table;
i  number;
cis varchar2(3);
Gupn varchar(40);
begin
cas(1) := sysdate;
cis := '';
FOR i IN 1..500
LOOP
cis :=  i;
```

```
while length(cis) <> 3
loop
cis :=  '0' || cis;
end loop;
Gupn:= 'UPN10245SON0001A0' || cis;

---------(1)(Assembly)--------------------------AA---------------------------------------

cas(1) := cas(1) + dbms_random.value(15,25)/24/3600;
Info(1) := 'FP6A' || round(dbms_random.value(100, 999))  || dbms_random.string('U', 2)
|| dbms_random.string('X', 20) ;
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000528', 'AA', cas(1),Info(1) , 0,'' );

insert into UPN_in_production (UPN,  FRONT_PANEL , PREVIOUS_STATION, NEXT_STATION)
values(GUPN,info(1) , 'AA' , 'AB' );
---------(2)(Assembly)--------------------------AB---------------------------------------

cas(2) := cas(1) + dbms_random.value(20,30)/24/3600;
Info(2) :=  'MB7QW' || round(dbms_random.value(10, 99))  || dbms_random.string('X',
22);
Insert into TRANSACTIONS  values( 'L01', Gupn,'WCZ100000529', 'AB' ,cas(2) ,Info(2)  , 0,'' );

update UPN_in_production set MAINBOARD = Info(2) , NEXT_STATION = 'AC' where upn =
Gupn;
---------(3)(Assembly)--------------------------AC---------------------------------------

cas(3) := cas(2) + dbms_random.value(20,30)/24/3600;
Info(3) := 'PW4' ||dbms_random.string('U', 2) || round(dbms_random.value(1, 9))  ||
dbms_random.string('U', 1) || dbms_random.string('X', 20);
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000428', 'AC', cas(3),  Info(3) , 0,'' );

update UPN_in_production set POWERSUPPLY = Info(3) ,PREVIOUS_STATION = 'AB'
,NEXT_STATION = 'AD' where upn = Gupn;
---------(4)(Assembly)--------------------------AD---------------------------------------

cas(4) := cas(3) + dbms_random.value(15,25)/24/3600;
Info(4) := 'B' ||round(dbms_random.value(10, 99))  || dbms_random.string('U', 2) ||
dbms_random.string('X', 20);
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000328', 'AD', cas(4),  Info(4) , 0,'' );

update UPN_in_production set BACKCOVER = Info(4) ,PREVIOUS_STATION = 'AC'
,NEXT_STATION = 'HI' where upn = Gupn;
---------(5)(Testing)-----------------------------HI---------------------------------------

cas(5) := cas(4) + dbms_random.value(25,30)/24/3600;
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000228', 'HI', cas(5),  'Passed' , 0,'' );

update UPN_in_production set PREVIOUS_STATION = 'AD' ,NEXT_STATION = 'TA' where upn
= Gupn;
---------(6)(Testing)-----------------------------TA---------------------------------------

cas(6) := cas(5) + dbms_random.value(30,40)/24/3600;
info(6) := round(dbms_random.value(5,20));
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000128', 'TA', cas(6),  info(6) , 0,'' );
```

```
update UPN_in_production set PREVIOUS_STATION = 'HI' ,NEXT_STATION = 'TB' where upn =
Gupn;
---------(7)(Testing)----------------------------TB---------------------------------------

cas(7) := cas(6) + dbms_random.value(30,40)/24/3600;
info(7):= 5;
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000628', 'TB', cas(7),  info(7) , 0,'' );

update UPN_in_production set PREVIOUS_STATION = 'TB' ,NEXT_STATION = 'TC' where upn
= Gupn;
---------(8)(Testing)----------------------------TC---------------------------------------

cas(8) := cas(7) + dbms_random.value(30,40)/24/3600;
info(8):= 7;
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000728', 'TC', cas(8),  info(8) , 0,'' );

update UPN_in_production set PREVIOUS_STATION = 'TB' ,NEXT_STATION = 'PA' where upn
= Gupn;
---------(9)(Packaging)--------------------------PA---------------------------------------

cas(9) := cas(8) + dbms_random.value(30,40)/24/3600;
info(9):= 'BAT' || ';MAN' || ';PC1' || ';RMC';
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000728', 'PA', cas(9),  info(9) , 0,'' );

update UPN_in_production set PREVIOUS_STATION = 'TC' ,NEXT_STATION = 'PB' where upn
= Gupn;
---------(10)(Packaging)-------------------------PB---------------------------------------

cas(10) := cas(9) + dbms_random.value(30,40)/24/3600;
info(10):= 'PB1';
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000728', 'PB', cas(10),  info(10) , 0,''
);

update UPN_in_production set PREVIOUS_STATION = 'PA' ,NEXT_STATION = 'PP' ,
ACCESSORIES = 1 where upn = Gupn;
---------(11)(Packaging)-------------------------PP---------------------------------------
cas(11) := cas(10) + dbms_random.value(30,40)/24/3600;
info(11):= 'PACZ' ||to_char(sysdate,'DDMMYYYYHH24')  ||cis ;
Insert into TRANSACTIONS  values( 'L01', Gupn, 'WCZ100000728', 'PP', cas(11),  info(11) , 0,''
);

update UPN_in_production set PREVIOUS_STATION = 'PB' ,NEXT_STATION = 'ZZ' , Pallet_id =
info(11) where upn = Gupn;
---------(11)(Finished)-------------------------ZZ---------------------------------------
END LOOP;
end;
```

Příloha č. 3: Datový slovník

Table	Key	Column Name	Datatype	Not Null
UPN_IN_PRODUCTION	PFK	UPN	VARCHAR2 (30 Byte)	True
		FRONT_PANEL	VARCHAR2 (40 Byte)	True
		MAINBOARD	VARCHAR2 (40 Byte)	False
		POWERSUPPLY	VARCHAR2 (40 Byte)	False
		BACKCOVER	VARCHAR2 (40 Byte)	False
		ACCESSORIES	NUMBER (1)	False
	FK	PREVIOUS_STATION	VARCHAR2 (2 Byte)	True

Table	Key	Column Name	Datatype	Not Null
LIST_OF_ACCESSORIES	PK	AC_ID	VARCHAR2 (3 Byte)	True
		ACCESSORY	VARCHAR2 (100 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
MO	PK	MO_NUMBER	VARCHAR2 (12 Byte)	True
	FK	MODEL	VARCHAR2 (5 Byte)	True
		NUMBER_OF_PIECES	NUMBER (5)	True
		DEADLINE	DATE	True
	FK	CUSTOMER	VARCHAR2 (5 Byte)	False

Table	Key	Column Name	Datatype	Not Null
LINE	PK	LI_ID	VARCHAR2 (3 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
LINE_ORDER	PFK	MO	VARCHAR2 (12 Byte)	True
	PK	L_ORDER	NUMBER (3)	True
	PFK	STATION	VARCHAR2 (2 Byte)	True

Table	Key	Column Name	Datatype	Not Null
LIST_OF_COMPONENTS	PK	CO_ID	VARCHAR2 (3 Byte)	True
		COMPONENT	VARCHAR2 (100 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
SWAPPED_COMPONENTS	PFK	UPN	VARCHAR2 (30 Byte)	True
	FK	REPAIR_MAN	VARCHAR2 (12 Byte)	True
	PK	DTIME	DATE	True
	FK	COMPONENT	VARCHAR2 (3 Byte)	True
	PK	OLD_ID	VARCHAR2 (60 Byte)	True
	PK	NEW_ID	VARCHAR2 (60 Byte)	True

Table	Key	Column Name	Datatype	Not Null
CUSTOMER	PK	CUS_ID	VARCHAR2 (4 Byte)	True
		CUSTOMER_NAME	VARCHAR2 (60 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
ACCESSORIES_IN_MO	PFK	MO	VARCHAR2 (12 Byte)	True
	PFK	ACCESSORY	VARCHAR2 (3 Byte)	True

Table	Key	Column Name	Datatype	Not Null
PATTERNS	PK	PA_ID	NUMBER	True
	FK	MODEL	VARCHAR2 (5 Byte)	True
	FK	MO	VARCHAR2 (12 Byte)	False
	FK	COMPONENT	VARCHAR2 (3 Byte)	True
		PATTERN	VARCHAR2 (32 Byte)	False

Table	Key	Column Name	Datatype	Not Null
REAPIR_RECORD	PFK	UPN	VARCHAR2 (30 Byte)	True
	PFK	REPAIR_MAN	VARCHAR2 (12 Byte)	True
	PK	DTIME	DATE	True
		RECORD	VARCHAR2 (200 Byte)	True

Table	Key	Column Name	Datatype	Not Null
STATION	PK	ST_ID	VARCHAR2 (2 Byte)	True
	FK	PART_OF_LINE	VARCHAR2 (10 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	True

Table	Key	Column Name	Datatype	Not Null
TRANSACTIONS	PFK	LINE	VARCHAR2 (3 Byte)	True
	PFK	UPN	VARCHAR2 (30 Byte)	True
	FK	LOPERATOR	VARCHAR2 (12 Byte)	True
	PFK	STATION	VARCHAR2 (2 Byte)	True
	PK	DTIME	DATE	True
		INFO	VARCHAR2 (60 Byte)	True
		PASS	NUMBER (1)	True
	FK	REASON	VARCHAR2 (4 Byte)	False

Table	Key	Column Name	Datatype	Not Null
UPN_IN_MO	FK	MO	VARCHAR2 (12 Byte)	False
	PK	UPN	VARCHAR2 (30 Byte)	True

Table	Key	Column Name	Datatype	Not Null
LINE_FOR_MO	PFK	MO	VARCHAR2 (12 Byte)	True
	PFK	LINE	VARCHAR2 (3 Byte)	True
	PK	ASSIGNED_FROM	DATE	True
		ASSIGNED_TILL	DATE	True

Table	Key	Column Name	Datatype	Not Null
TEST_VALUES_LIST	PK	TE_ID	NUMBER (9)	True
	FK	TV_MODEL	VARCHAR2 (5 Byte)	True
	FK	STATION	VARCHAR2 (2 Byte)	True
		TE_VALUE	NUMBER (25,9)	True
		PASS	NUMBER (1)	False
	FK	REASON	VARCHAR2 (4 Byte)	True

Table	Key	Column Name	Datatype	Not Null
TV_MODELS	PK	MOD_ID	VARCHAR2 (5 Byte)	True
		SCREEN_SIZE	NUMBER (4,1)	True
		TYPE	VARCHAR2 (100 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
UPN_IN_PRODUCTION	PFK	UPN	VARCHAR2 (30 Byte)	True
		FRONT_PANEL	VARCHAR2 (40 Byte)	True
		MAINBOARD	VARCHAR2 (40 Byte)	False
		POWERSUPPLY	VARCHAR2 (40 Byte)	False
		BACKCOVER	VARCHAR2 (40 Byte)	False
		ACCESSORIES	NUMBER (1)	False

Table	Key	Column Name	Datatype	Not Null
LIST_OF_ACCESSORIES	PK	AC_ID	VARCHAR2 (3 Byte)	True
		ACCESSORY	VARCHAR2 (100 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
POSITION	PK	PO_ID	VARCHAR2 (3 Byte)	True
		POSITION	VARCHAR2 (30 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	False

Table	Key	Column Name	Datatype	Not Null
EMPLOYEE	PK	EM_ID	VARCHAR2 (12 Byte)	True
		NAME	VARCHAR2 (30 Byte)	True
		SURNAME	VARCHAR2 (60 Byte)	True
	FK	POSITION	VARCHAR2 (3 Byte)	True

Table	Key	Column Name	Datatype	Not Null
REASON_CODE	PK	REASON	VARCHAR2 (4 Byte)	True
		DESCRIPTION	VARCHAR2 (200 Byte)	True

Table	Key	Column Name	Datatype	Not Null
TEST_VALUES_BY_RANGE	PK	TE_ID	NUMBER (9)	True
	FK	TV_MODEL	VARCHAR2 (5 Byte)	True
	FK	STATION	VARCHAR2 (2 Byte)	True
		RANGE_START	NUMBER (25,9)	True
		RANGE_END	NUMBER (25,9)	True
		PASS	NUMBER (1)	False