ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA EKONOMICKÁ

Bakalářská práce

Porovnání databázových modelů

Comparison of Database Models

Plzeň 2020

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta ekonomická

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš FÜRBACHER**
Osobní číslo:	**K18B0003P**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Informační management**
Téma práce:	**Porovnání databázových modelů**
Zadávající katedra:	**Katedra ekonomie a kvantitativních metod**

Zásady pro vypracování

1. Objasněte principy relačních databází.
2. Popište principy ERA modelu.
3. Představte společnost Diebold Nixdorf.
4. Navrhněte metodiku, pomocí které lze identifikovat rozdíly mezi dvěma databázovými modely.
5. Aplikujte navržený postup na vybrané databáze a diskutujte výsledky srovnání.

Rozsah bakalářské práce: **40 – 60 stran**
Rozsah grafických prací: **neuveden**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

- KROENKE, David a David J. AUER. *Databáze*. Brno: Computer Press, 2015. ISBN 978-80-251-4352-0.
- POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy*. ČVUT Praha, 2013. ISBN 978-80-01-05212-9.
- STANEK, William R. *Microsoft SQL Server 2012: kapesní rádce administrátora*. Brno: Computer Press, 2013. ISBN 978-80-251-3797-0.
- TEOREY, Toby J. *Database modeling and design: logical design*. MA: Morgan Kaufmann, 2011. ISBN 978-0123820204.

Vedoucí bakalářské práce: **Doc. RNDr. Mikuláš Gangur, Ph.D.**
Katedra ekonomie a kvantitativních metod

Datum zadání bakalářské práce: **22. října 2019**
Termín odevzdání bakalářské práce: **22. dubna 2020**

L.S.

Doc. Ing. Michaela Krechovská, Ph.D.
děkanka

Ing. Mgr. Milan Svoboda, Ph.D.
vedoucí katedry

V Plzni dne 22. října 2019

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma

„*Porovnání databázových modelů“*

vypracoval/a samostatně pod odborným dohledem vedoucí/vedoucího bakalářské práce za použití pramenů uvedených v přiložené bibliografii.

Plzeň dne 3. 5. 2020 .....

podpis autora/autorky

Poděkování

Rád bych poděkoval panu doc. RNDr. Mikuláši Gangurovi, Ph.D. za vedení mé bakalářské práce a za pomoc při jejím zpracování.

Dále bych chtěl poděkovat panu Radku Votípkovi ze společnosti Diebold Nixdorf za poskytnutí tématu k práci a za konzultační činnost.

Obsah

Úvod **9**

1 Relační databáze **12**

1.1 Struktura relační databáze 12

1.1.1 Tabulka 12

1.1.2 Sloupce 13

1.1.3 Řádky 13

1.1.4 Klíče 13

1.1.5 Relace 14

1.2 ERA model 15

1.2.1 Entita 15

1.2.2 Relace 16

1.2.3 Atributy 16

1.2.4. ERA model v MySQL Workbench 16

1.3 Normální formy 17

1.3.1 První normální forma 17

1.3.2 Druhá normální forma 18

1.3.3 Třetí normální forma 18

1.4 Integritní omezení 19

1.4.1 Entitní omezení 19

1.4.2 Doménové omezení 19

1.4.3 Referenční omezení 19

1.5 Dotazovací jazyk SQL 20

1.5.1 Data Definition Language (DDL) 20

1.5.2 Data Manipulation Language (DML) 21

1.5.3 T-SQL 26

1.6 Databázové objekty 26

1.6.1 Tabulka 27

1.6.2 Pohled 27

1.6.3 Klíče 27

1.6.4 Funkce 27

1.6.5 Procedura 28

1.6.6 Index ... 28
1.6.7 Trigger ... 28
2 Společnost Diebold Nixdorf ... 29
2.1 Historie ... 29
2.2 Produkt ... 30
2.3 Plzeňská pobočka společnosti Diebold Nixdorf ... 31
2.3.1 Organizační struktura ... 32
3 Řešení ... 33
3.1 Liquibase ... 33
3.1.1 Příkaz diff ... 34
3.1.2 Příkaz diffChangeLog ... 37
3.1.3 Shrnutí Liquibase ... 37
3.2 SQL Server Data Tools (SSDT) ... 37
3.2.1 Shrnutí SSDT ... 39
3.3 SQL Examiner ... 40
3.3.1 Shrnutí SQL Examiner ... 42
3.4 Vyhodnocení nástrojů ... 42
3.4.1 Finální výběr ... 43
4 Postup pro porovnání objektů dvou databázových modelů ... 44
4.1 Zprovoznění nástroje VirtualBox ... 44
4.1.1 Zprovoznění internetové sítě ... 45
4.2 Zprovoznění nástroje Microsoft SQL Server Management Studio (SSMS) ... 47
4.3 Spuštění Visual Studio a SSDT ... 48
4.4 Popis postupu ... 49
4.4.1 Rozdíly mezi databázemi v SQL Server Management Studio ... 49
4.4.2 Práce v SQL Server Data Tools ... 49
4.4.3 Manipulace s databázovými objekty ... 53
4.4.4 Update ... 56
Závěr ... 57
Seznam použité literatury ... 58
Seznam obrázků ... 60
Seznam použitých zkratek a značek ... 61

Úvod

V dnešní době disponuje každá firma velkým množstvím dat. Ať už jde o data zákaznická, zaměstnanecká, či jakákoli jiná, firmy musejí tato data nějakým způsobem ukládat. Vzhledem ke kvantitě dat se v poslední době osvědčily databáze. Velké množství firem využívá databáze relační, neboť svojí strukturou jsou poměrně intuitivní a dokáží dobře vyjádřit skutečné vztahy mezi daty. V relačních databázích jsou jednotlivé záznamy uloženy přehledně formou tabulek. Databáze se také snadno zálohují a vytváří se jejich kopie.

Tohoto faktu se dá využít například k obnově, pokud se nějaká data ztratí nebo poškodí. Rovněž ale může sloužit pro kontrolu databáze v různých časových okamžicích. Taková kontrola je užitečná v momentě, kdy je očekávána nějaká změna v databázi, ale také v okamžiku, kdy chce firma zjistit, zda s databází někdo nemanipuloval a nezměnil v ní tak nějaké zásadní prvky.

Ať už je databáze měněna úmyslně, nebo nechtěně, je dobré mít přehled o tom, v jakém stavu zůstala databáze po kontaktu s uživatelem. Často jsou však rozdíly uživateli dobře skryty a nejsou viditelné na první pohled. Rovněž nelze vzhledem k objemnosti databází kontrolovat každý jeden záznam nebo databázový objekt ručně. Podstatou práce je tedy nalézt způsob, pomocí kterého lze zjišťovat, zda se dvě databáze (konkrétně jejich databázové modely) nějakým způsobem liší. Vzhledem ke skutečnosti, že s databázemi mnohdy pracují stovky až tisíce uživatelů, může velice snadno dojít k nežádoucí změně. Může se jednat o pouhou chybu, či o úmyslné poškození, ale v obou případech je vhodné o databázi mít nějakým způsobem přehled a pravidelně ji kontrolovat.

Hlavním cílem práce je využití vhodného nástroje a s pomocí něj navržení postupu, díky kterému bude možné porovnat dva databázové modely a zjistit, zda se v něčem liší. Tento postup bude následně použit na dvě testovací databáze a výsledek bude vyhodnocen.

V rámci dalších dílčích cílů budou rovněž představeny principy relačních databází.

Práce bude zaměřena konkrétně na databázový systém Microsoft SQL Server, který je používán firmou Diebold Nixdorf, ve které je tato práce vypracovávána.

Dílčí cíle, které povedou k naplnění cíle hlavního, jsou definovány následujícím způsobem:

- Objasnění principů relačních databází.
- Popis principů ERA modelu.
- Představení společnosti Diebold Nixdorf.
- Uvedení a popsání jednotlivých softwarových nástrojů, které budou využity.
- Navrhnutí metodiky, pomocí které lze identifikovat rozdíly mezi databázovými modely.
- Aplikování navrženého postupu na vybrané databáze a diskuze nad výsledky srovnání.

V první části práce se tedy, jak vychází ze seznamu dílčích cílů, budeme věnovat popisu relačních databází. V první kapitole tohoto tématu bude popsána samotná struktura relačních databází a budou blíže vysvětleny jednotlivé typy relačních vztahů. Teoretický popis bude doplněn o podpůrné praktické ukázky z prostředí phpMyAdmin. Druhá kapitola první části je vyhrazena objasnění principů ERA modelů – základního kamene popisu všech relačních databází. Stejně jako v předchozí kapitole bude i v rámci této rovněž uvedeno několik praktických příkladů. Tentokrát k tomuto účelu poslouží softwarové prostředí MySQL Workbench. V následující kapitole se podíváme na téma normálních forem, tedy jakýchsi regulí pro vytváření efektivních databází. Na toto téma navazuje kapitola čtvrtá, zabývající se integritními omezeními, které s normálními formami úzce souvisí a do jisté míry je definují. Pátá kapitola, která se dělí na tři části, vysvětluje principy databázového dotazovacího jazyka SQL. Všechny tři části se věnují jednotlivým skupinám příkazů, které spadají pod SQL jazyk. Jedná se o Data Definition Language (DDL), Data Manipulation Language (DML) a T-SQL. Třetí jmenovaná část jazyka SQL je pouze okrajově zmíněna, neboť její obsáhlejší popis je nad rámec této práce. Stejně jako některé předchozí části bude i tato kapitola doplněna o praktické ukázky. V tomto případě se jedná opět o prostředí phpMyAdmin. Kromě ukázek ze zmíněného prostředí bude praktická část tohoto tématu obohacena rovněž o fragmenty kódu pro lepší představu, jakým způsobem jsou samotné příkazy jazyka strukturovány. Na kapitolu o SQL bude navazovat kapitola o stěžejním tématu pro tuto práci – databázové objekty. Budou zde stručně představeny vybrané objekty, se kterými se

bude v dalších kapitolách pracovat. Kapitola věnovaná databázovým objektům uzavírá první část práce.

Druhá část práce má za úkol představení firmy Diebold Nixdorf. V jednotlivých kapitolách bude postupně představena historie firmy a její produkt. Poslední téma druhé části se bude věnovat samotné pobočce společnosti, která sídlí v Plzni, včetně nastínění její organizační struktury.

Třetí část práce se bude věnovat existujícím nástrojům, které disponují funkcionalitami potřebnými k vyřešení zadaného problému. Budou představeny základní vlastnosti, výhody a nevýhody jednotlivých možností, na základě kterých dojde k jejich vyhodnocení a výběru jednoho z nich. Ve vybraném nástroji bude následně provedeno porovnání databázových modelů.

Čtvrtý úsek bude již zaměřen na samotný proces tvorby a aplikace vybraného popisu, který slouží k odhalení odlišností v objektech dvou databázových modelů, včetně představení veškerého softwarového vybavení, které bude v tomto procesu použito. V rámci představení těchto softwarových produktů bude také popsáno, jakým způsobem je možné je zprovoznit. V samém závěru této části dojde k interpretaci výstupu a souhrnnému zhodnocení.

1 Relační databáze

Databázové technologie získávají v dnešní době na popularitě. Těší se jí v široké škále působnosti, ať už jde o odvětví elektronického podnikání, či jakýchkoli dalších webových aplikací. Tvoří jádro aplikací sloužících k řízení podniku a podpoře rozhodování. K databázím navíc obecně přistupuje obrovské množství uživatelů po celém světě. (Kroenke, 2015)

Relační databáze je jeden z modelů databáze, který je dnes velice hojně používán.

V takových databázích dochází k ukládání a přistupování k datovým bodům, mezi kterými existuje nějaká vzájemná souvislost. Jádrem relačních databází je relační model, který umožňuje reprezentovat data formou tabulek. Tato forma reprezentace umožňuje přehledné a intuitivní uložení dat. (Oracle, 2019)

1.1 Struktura relační databáze

Základem každé relační databáze jsou tyto prvky (Gangur, 2014):

- tabulka
- sloupce
- řádky
- hodnoty
- klíče
- relace

1.1.1 Tabulka

Jak již bylo řečeno, principem relačních databází je reprezentace dat pomocí tabulek. Tabulka je tedy základním prvkem takového typu databází.

V tabulkách se nachází data o entitě, která je touto tabulkou reprezentována. (Gangur, 2014)

Obrázek 1: Databázová tabulka

id	band	genre	country
1	Sticx	Ska punk	Česká republika
2	Budulínek	Ska punk	Česká republika
3	To the Rats and Wolves	Metalcore	Německo
4	Asking Alexandria	Metalcore	UK
5	Glad For Today	Post-hardcore	Česká republika
6	The Paranoid	Alternative metal	Slovensko
7	The Snuff	Alternative metal	Česká republika
8	Parkway Drive	Metalcore	Austrálie
9	Kurtizány z 25. avenue	Alternative rock	Česká republika
10	Rammstein	Industrial metal	Německo
11	Papa Roach	Nu metal	USA
12	Bad Omens	Metalcore	USA
13	Slaves	Post-hardcore	USA
14	Eskimo Callboy	Metalcore	Německo
15	Cocotte Minute	Nu metal	Česká republika
16	Trautenberk	Alternative metal	Česká republika
17	Up!Great	Nu metal	Česká republika
18	Infected Rain	Metalcore	Moldávie
19	Jinjer	Progressive metal	Ukrajina
20	Tribulation	Gothic metal	Švédsko
21	Wintersun	Power metal	Finsko
22	Arch Enemy	Melodic death metal	Švédsko
23	Set To Stun	Post-hardcore	USA
24	Escape The Fate	Post-hardcore	USA
25	Klogr	Alternative metal	Itálie

Zdroj: phpMyAdmin (2019), zpracováno autorem

Na obrázku 1 je vidět příklad tabulky, která reprezentuje hudební skupiny.

1.1.2 Sloupce

Sloupce tabulky obsahují jednotlivé atributy, které charakterizují entitu. (Oracle, 2019)

Sloupce mají unikátní pojmenování a musí mít přiřazený datový typ, který se liší s ohledem na konkrétní systém řízení báze dat (SŘBD). (Gangur, 2014)

Na obrázku 1 jsou těmito atributy *id*, *band*, *genre*, *country*, které charakterizují entitu *hudební skupiny*.

1.1.3 Řádky

Řádky v tabulce obsahují jednotlivé záznamy. Nachází se v nich tedy konkrétní hodnoty atributů. (Gangur, 2014)

1.1.4 Klíče

Klíče slouží k identifikaci jednotlivých záznamů v tabulce. Klíč, který jednoznačně určuje záznam a je tedy unikátní, se nazývá primární klíč. Každá tabulka takový klíč

obsahuje. Může být reprezentován jedním nebo více atributy. V obou případech však platí, že musí disponovat vlastností unikátnosti. Často takový atribut, který by byl unikátní, v tabulce ani nenalezneme, a proto, aby byla tato podmínka splněna, se standardně volí způsob vytvoření umělého atributu „id“. (Gangur, 2014)

Tento atribut je vidět také na obrázku 1.

1.1.5 Relace

Relací je označováno spojení mezi tabulkami pomocí klíčů. Primární klíč první tabulky se stává cizím klíčem druhé tabulky, se kterou je první tabulka propojena. V relačních databázích se rozlišují tři typy relací, přičemž toto označení reprezentuje kardinalitu vazeb (Gangur, 2014):

- 1:1
- 1:N
- M:N

Obrázek 2: Databázová tabulka s cizím klíčem

id	name	country	genre
1	Sticx	Česká republika	1
2	Budulínek	Česká republika	1
3	To the Rats and Wolves	Německo	2
4	Asking Alexandria	UK	2
5	Glad For Today	Česká republika	4
6	The Paranoid	Slovensko	3
7	The Snuff	Česká republika	3
8	Parkway Drive	Austrálie	2
9	Kurtizány z 25. avenue	Česká republika	5
10	Rammstein	Německo	6
11	Papa Roach	USA	7
12	Bad Omens	USA	2
13	Slaves	USA	4
14	Eskimo Callboy	Německo	2
15	Cocotte Minute	Česká republika	7
16	Trautenberk	Česká republika	3
17	Up!Great	Česká republika	7
18	Infected Rain	Moldávie	2
19	Jinjer	Ukrajina	8
20	Tribulation	Švédsko	9
21	Wintersun	Finsko	10
22	Arch Enemy	Švédsko	11
23	Set To Stun	USA	4
24	Escape The Fate	USA	4
25	Klogr	Itálie	3

Zdroj: phpMyAdmin (2019), zpracováno autorem

Na obrázku 2 je vidět, že ve sloupci *genre* jsou nyní místo konkrétních názvů jen čísla, která jsou primárním klíčem v tabulce reprezentující hudební žánry. V tabulce na obrázku 2, která reprezentuje hudební kapely, je tento atribut jako cizí klíč.

1.1.5.1 Relace 1:1

V takové relaci se na vazbě podílí maximálně jeden záznam z každé tabulky. (Gangur, 2014)

1.1.5.2 Relace 1:N

V případě relace 1:N se vazby účastní z jedné tabulky maximálně jeden záznam a z druhé tabulky záznamů více. (Gangur, 2014)

V některých specifických situacích může být N konstanta. Tedy je pevně dané, kolik záznamů z druhé tabulky se podílí na vazbě. Příkladem může být základní sestava fotbalového zápasu, ve které je dané, že ji bude tvořit právě 11 hráčů. N je tedy v tomto případě rovno 11. (Teorey, 2011)

1.1.5.3 Relace M:N

Ve vazbě M:N náleží maximálně jednomu záznamu z první tabulky více záznamů z druhé tabulky a naopak maximálně jednomu záznamu z druhé tabulky náleží více záznamů z první tabulky. Tato vazba se realizuje pomocí spojovací tabulky skrz dvě vazby 1:N. (Gangur, 2014)

1.2 ERA model

„Databázový model je kolekce pojmů sloužících k modelování. Přesněji řečeno, databázový model může být formální či matematický aparát. Výsledkem databázového modelování je databázové schéma, tj. vlastně jistý popis struktury dat.“ (Pokorný, 2013, s. 11)

ERA model je grafické zobrazení jednotlivých reálných prvků (entit), které jsou vzájemně propojeny vztahy (relacemi) a obsahují atributy. Akronym ERA tedy pochází od zkratek těchto slov: entity, relationship, attribute. (Gangur, 2014)

1.2.1 Entita

Entity tvoří hlavní datové objekty, o nichž jsou informace ukládány. Konkrétní záznam entity se nazývá instance. (Teorey, 2011)

Entity jsou poté reprezentovány formou tabulek. (Gangur, 2014)

1.2.2 Relace

Relace v ERA modelu fungují stejným způsobem, jak bylo již popsáno výše.

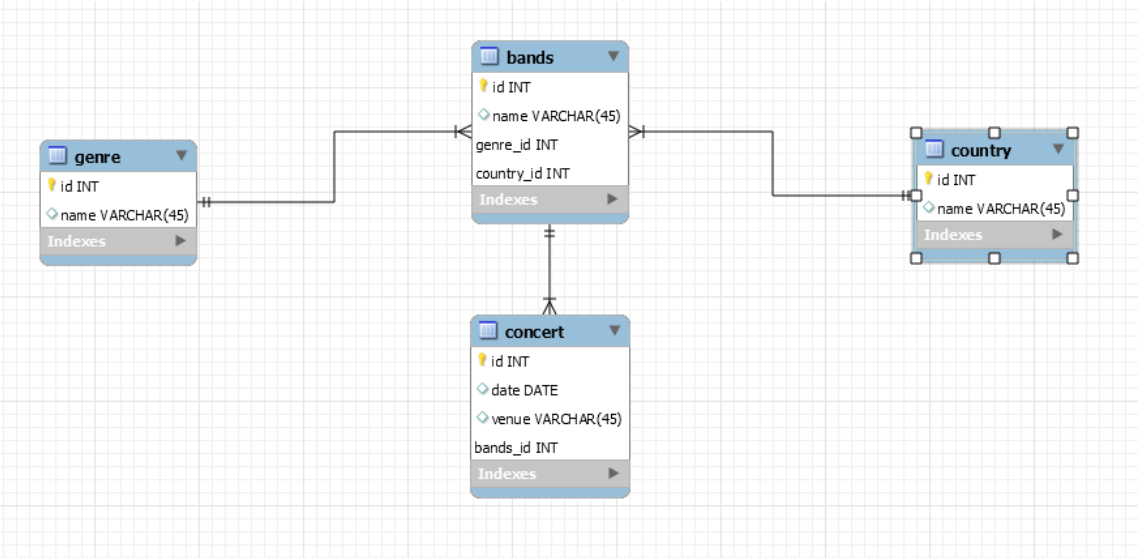
1.2.3 Atributy

Atributy charakterizují vlastnosti entit, ke kterým se vážou. Slouží k podrobnějšímu popisu těchto entit. (Teorey, 2011)

V tabulce jsou atributy zobrazeny pomocí sloupců.

1.2.4. ERA model v MySQL Workbench

Obrázek 3: ERA model

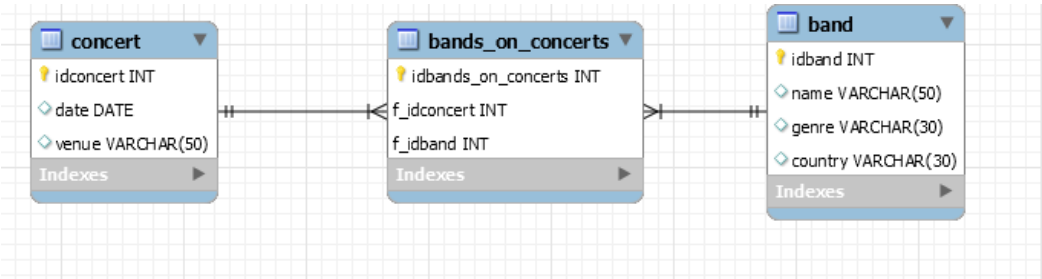


Zdroj: MySQL Workbench (2019), zpracováno autorem

Na obrázku 3 je zobrazen příklad ERA modelu, který je vytvořen v nástroji MySQL Workbench. Tento model obsahuje čtyři entity (*genre*, *country*, *bands*, *concert*), přičemž každá obsahuje svůj primární klíč. Entita *bands* dále obsahuje cizí klíče z tabulek *genre* (jakého je hudební skupina žánru) a *country* (z jaké země hudební skupina pochází). Následně entita *concert* obsahuje kromě primárního klíče a atributů *date* a *venue* také cizí klíč entity *bands*, který indikuje hudební skupiny, které na koncertě vystupovaly. Tento model je vytvořen tak, že ke každému koncertu lze přiřadit více kapel. S čím však už nepočítá, je to, že rovněž každá kapela mohla vystupovat na více koncertech.

V předešlých kapitolách bylo již zmíněno, že existuje vazba M:N. Tato vazba se v ERA modelu realizuje pomocí vytvoření spojovací tabulky, která je propojena dvěma vazbami typu 1:N.

Obrázek 4: ERA model s vazbou M:N



Zdroj: MySQL Workbench (2019), zpracováno autorem

Na obrázku 4 je pak vidět, jak takový rozklad vazby M:N vypadá v prostředí MySQL Workbench. Je zde zobrazena situace, kdy v jedné tabulce jsou uloženy hudební skupiny a v druhé tabulce jsou uloženy koncerty. V obou případech je v tabulkách obsaženo kromě primárních klíčů také několik dalších atributů charakterizujících objekt. Veprostřed se nachází tabulka *bands_on_concerts*, ve které jsou záznamy o tom, které hudební kapely se účastnily kterého koncertu. Právě tato tabulka je tabulkou spojovací, která byla popsána výše. Tabulka obsahuje kromě svého vlastního primárního klíče také dva cizí klíče získané z tabulek *concert* a *band*. Tímto způsobem je vytvořen záznam o účasti kapely na koncertě. Tento model také řeší problém předešlého příkladu, kdy bylo možné každou kapelu přiřadit maximálně k jednomu koncertu.

1.3 Normální formy

Relační databáze mohou často trpět různými nedokonalostmi, jako je například nízký výkon, porušení integrity nebo problémy s udržitelností. K předejití těchto problémů slouží převedení databáze do normálních forem. Jsou to tedy jakési stavy, ve kterých se databáze nachází a které splňují určité požadavky. (Teorey, 2011)

1.3.1 První normální forma

„Tabulka se nachází v první normální formě (1NF) právě tehdy, když všechny sloupce obsahují pouze atomické hodnoty – to znamená, že každý sloupec může obsahovat pouze jednu hodnotu pro každý řádek v tabulce.“ (Teorey, 2011, s. 111)

Tato definice znamená, že všechny komponenty v relačním modelu dat musí být atomického charakteru (například číslo, textový řetězec) a nesmí se tedy jednat o relace. (Rychlík, 2016)

1.3.2 Druhá normální forma

„Tabulka se nachází v druhé normální formě (2NF) právě tehdy, když se nachází v 1NF a každý neklíčový atribut je silně závislý na primárním klíči, pokud se nachází na pravé straně funkční závislosti, ve které je na pravé straně samotný primární klíč, nebo něco, co lze odvodit od primárního klíče použitím transitivity funkční závislosti.“ (Teorey, 2011, s. 114)

Výše uvedená definice říká, že každý atribut, který není součástí klíče, je silně funkčně závislý na klíči. Druhá normální forma je automaticky splněna, pokud je primárním klíčem jednoduchý atribut. (Rychlík, 2016)

K lepšímu pochopení si zde uvedeme definici funkční závislosti: „Mějme tabulku R, množina atributů B je funkčně závislá na jiné množině atributů. A pokud je v jakémkoli časovém okamžiku každá hodnota množiny A spojena právě s jednou hodnotou množiny B.“ (Teorey, 2011, s. 113)

1.3.3 Třetí normální forma

O tabulce řekneme, že se nachází ve třetí normální formě, nachází-li se ve druhé normální formě a neexistuje funkční závislost mezi neklíčovými atributy. (Rychlík, 2016)

Máme-li tedy tabulku, která reprezentuje oddělení na Západočeské univerzitě a jejímž primárním klíčem je id a dalšími atributy název oddělení a název budovy, ve které se oddělení nachází, není tato tabulka ve třetí normální formě. Existuje zde totiž funkční závislost mezi dvěma neklíčovými atributy, kterými jsou název oddělení a název budovy, ve které se oddělení nachází. Oddělení se totiž bude nacházet vždy ve stejné budově. (Rychlík, 2016)

Existují také čtvrtá a pátá normální forma, které dále počítají s takzvanou multizávislostí. Nicméně dnešním standardem pro běžné komerční databáze se stala třetí normální forma. (Teorey, 2011)

1.4 Integritní omezení

S předešlou kapitolou, která se věnovala normálním formám, poměrně úzce souvisí také integritní omezení. Stejně jako u normálních forem je i u integritních omezení zapotřebí splňovat určité podmínky. Obecně jsou základními integritními omezení tato tři (Rychlík, 2016):

- entitní
- doménové
- referenční

Integritní omezení může tvořit v podstatě jakékoli omezení, které musí daná databáze splňovat. Tato omezení se budou v každé databázi lišit s ohledem na její požadavky a především na požadavky na data, která se v databázi ukládají. (Teorey, 2011)

V každém případě však platí, že integritními omezeními jsou popisovány podmínky, které musí data splnit, aby databáze, ve které jsou ukládána, mohla co nejlépe odpovídat reálnému světu. (Rychlík, 2016)

1.4.1 Entitní omezení

Entitní omezení znamená, že lze jednoznačně určit entitu. Toto úzce souvisí s vytvářením primárních klíčů. (Rychlík, 2016)

1.4.2 Doménové omezení

Vzhledem k tomu, že doménou je označována množina přípustných hodnot, kterých mohou data v databázi nabývat, pak doménové omezení spočívá v obsahu této množiny. Kromě výčtu ji lze také definovat určením vlastností dat. (Rychlík, 2016)

1.4.3 Referenční omezení

Referenční omezení odkazuje na vztah mezi dvěma tabulkami a popisuje způsob jejich propojení pomocí relace kardinality 1:N. Popisuje také povinnost výskytu této vazby. (Rychlík, 2016)

1.5 Dotazovací jazyk SQL

V předešlých kapitolách je popsáno, jak relační databáze vypadá a funguje. S takovou databází se ale dále pracuje a je potřeba zpracovávat data, která se v ní nacházejí. K tomuto účelu slouží mimo jiné dotazovací jazyk SQL.

SQL (Structured Query Language) je standardní jazyk, který slouží pro přistupování k databázi a k její následné manipulaci. (W3Schools, 2019)

Syntax se může trochu lišit v závislosti na výběru konkrétního SŘBD, ale v každém případě se v základu SQL dělí na dva podjazyky – DDL a DML. (Teorey, 2011)

1.5.1 Data Definition Language (DDL)

DDL slouží k samotné definici databázových objektů a jeho základní příkazy jsou:

- CREATE TABLE – slouží k definování tabulky včetně jejích atributů
- DELETE TABLE – slouží ke smazání vytvořené tabulky
- ALTER TABLE – slouží k vytvoření nových sloupců, případně k upravení nebo smazání sloupců, které již byly vytvořeny
- CREATE VIEW – slouží k vytvoření databázového pohledu (view)
- DROP VIEW – slouží ke smazání databázového dotazu (view)

Některé verze SQL mohou disponovat také příkazy CREATE INDEX a DROP INDEX, které slouží k vytvoření a smazání indexu, v závislosti na tom, zda zvolené SŘBD indexy využívá. (Teorey, 2011)

1.5.1.1 Syntax DDL

Obrázek 5: DDL – příkaz CREATE TABLE

```
CREATE TABLE bands(
     id INT
     name VARCHAR (50)
     genre VARCHAR (30)
     country VARCHAR (30)
     PRIMARY KEY (id)
);
```

Zdroj: vlastní zpracování, 2019

Na obrázku 5 je znázorněno definování tabulky pomocí DDL. Tabulka nese název *bands* a obsahuje 5 atributů. Prvním atributem je *id*, který je datového typu INT (tedy celé číslo). Dalšími atributy jsou *name*, *genre* a *country*, které jsou typu VARCHAR (tedy textový řetězec) a v závorce je uvedena jejich délka ve znacích. Na posledním řádku je uvedeno, že primárním klíčem je atribut id.

Obrázek 6: DDL – příkaz ALTER TABLE

```
ALTER TABLE bands
     ADD note VARCHAR (255);
ALTER TABLE bands
     DROP COLUMN country;
```

Zdroj: vlastní zpracování, 2019

Na obrázku 6 je použit příkaz ALTER TABLE nejprve pro přidání nového sloupce do tabulky *bands*. Tento sloupec nese název *note* a je datového typu VARCHAR o velikosti 255. V dalším příkazu je ALTER TABLE použit pro smazání sloupce *country*.

1.5.2 Data Manipulation Language (DML)

Jakmile je databáze nadefinována a naplněna, přichází na řadu manipulace s daty, která jsou uvnitř ní uložena. Právě k těmto účelům slouží Data Manipulation Language.

Základními příkazy DML jsou:

- SELECT – ačkoli by tento příkaz měl být správně zařazen do kategorie Data Query Language (DQL), při jeho přirozeném využití – tedy ve spojení s klíčovými slovy FROM a WHERE – již dochází k manipulaci s daty a to z něj dělá zásadní příkaz pro datovou manipulaci, a proto byl zařazen do kategorie DML, tvoří jádro databázového dotazu
- INSERT – slouží k přidání hodnot do tabulky
- UPDATE – slouží k aktualizaci hodnot v tabulce
- DELETE – slouží ke smazání hodnot z tabulky

1.5.2.1 Syntax DML - SELECT

Obrázek 7: Tabulka interpret před aplikováním příkazu SELECT

id_interpret	nazev	zeme
1	Papa Roach	USA
2	Whitechapel	USA
3	Tegan and Sara	Kanada
4	Doga	Česká republika
5	The Paranoid	Slovensko
6	Zoči Voči	Slovensko
7	Escape the Fate	USA
8	Motionless In White	USA
9	Asking Alexandria	Spojené království
10	Stitched Up Heart	USA
11	Rise of the Northstar	Francie
12	Hatebreed	USA
13	Def Leppard	Spojené království
14	Jinjer	Ukrajina
15	Pale Waves	Spojené království
16	Eskimo Callboy	Německo
17	Infected Rain	Moldávie
18	Falling In Reverse	USA
19	Butcher Babies	USA
20	Emmure	USA

Zdroj: phpMyAdmin (2019), zpracováno autorem

Na obrázku 7 je vidět tabulka reprezentující hudební skupiny, která obsahuje tři sloupce (*id_interpret, nazev, zeme*) a 20 záznamů. Na této tabulce bude nyní ukázáno, jak funguje příkaz SELECT.

Obrázek 8: DML – příkaz SELECT

```
CREATE VIEW americke_kapely AS
SELECT DISTINCT id_interpret, nazev, zeme
FROM interpret
WHERE zeme = "USA"
ORDER BY nazev DESC
```

Zdroj: vlastní zpracování, 2019

Takto vypadá příklad dotazu vytvořeného pomocí SQL jazyka. V prvním řádku kódu je příkaz CREATE VIEW, který byl již zmíněn v přechozí části o DDL. Jak už také bylo uvedeno, tento příkaz slouží k vytvoření databázového pohledu, což je vlastně jen

pojmenovaný a uložený dotaz. Spuštěním dotazu bez tohoto příkazu se totiž pouze zobrazí vybraná data, avšak nikam se neuloží. (Teorey, 2011)

Následuje samotný příkaz SELECT, pomocí kterého se uvádí, které sloupce chce uživatel vybrat (zde *id_interpret, nazev, zeme*). V případě, který je uveden na obrázku 7, je možné nahradit vypsané sloupce znakem „*“, který značí, že uživatel požaduje výběr všech sloupců tabulky. Klíčové slovo DISTINCT slouží k odstranění duplicit, tedy záznamů, které se shodují. V tomto případě nemá žádný vliv, neboť se v tabulce žádné duplicity nenachází, avšak v praxi je hojně využíváno. Slovo FROM vybírá tabulku, ze které se mají dané záznamy extrahovat. Dále v pořadí je slovo WHERE, po kterém přichází podmínka, kterou musí splňovat záznamy, které chce uživatel zobrazit. V tomto případě je uvedena podmínka, že sloupec *zeme* musí mít hodnotu *USA* – tedy vybírají se interpreti, kteří pocházejí z USA. Poslední řádek kódu slouží k řazení vybraných záznamů. V tomto případě je tabulka řazena podle sloupce *nazev* sestupně (to značí slovo DESC).

Obrázek 9: Tabulka interpret po aplikování příkazu SELECT

id_interpret	nazev ▼ 1	zeme
2	Whitechapel	USA
10	Stitched Up Heart	USA
1	Papa Roach	USA
8	Motionless In White	USA
12	Hatebreed	USA
18	Falling In Reverse	USA
7	Escape the Fate	USA
20	Emmure	USA
19	Butcher Babies	USA

Zdroj: phpMyAdmin (2019), zpracováno autorem

Na obrázku 9 je vidět tabulka, která vznikne po aplikování výše uvedeného SQL dotazu na vybranou tabulku.

Jak již bylo zmíněno, tabulky se dají navzájem propojovat pomocí cizích klíčů. Tato skutečnost umožňuje extrahovat data z několika tabulek zároveň.

Obrázek 10: Tabulka pisen před aplikováním příkazu SELECT

id_pisen	nazev	f_id_interpret
1	Dva svety	5
2	Unos	5
3	Bring Me Home	2
4	Gravitace	4
6	Goodbye, goodbye	3
8	570	8
9	If You Cant Ride Two Horses At Once You Should Get...	9
10	A Prophecy	9
11	A Candlelit Dinner With Inamorta	9
12	A Lesson Never Learned	9
14	Monster	10
15	Finally Free	10
16	I Will Be Heard	12
17	Destroy Everthing	12
18	Again and Again	11
19	Samurai Spirit	11

Zdroj: phpMyAdmin (2019), zpracováno autorem

Na obrázku 10 je zobrazena tabulka *pisen*, která obsahuje primární klíč *id*, sloupec *nazev* a sloupec *f_id_interpret*, který je cizím klíčem z výše uvedené tabulky *interpret*.

Obrázek 11: DML – příkaz SELECT s více tabulkami

```
SELECT i.nazev AS interpret, p.nazev AS pisen
FROM interpret i, pisen p
WHERE i.id_interpret = p.f_id_interpret AND i.zeme = "USA"
ORDER BY p.nazev
```

Zdroj: vlastní zpracování, 2019

Na obrázku 11 je příklad SQL kódu, který spojuje dvě tabulky. V tomto případě spojuje tabulku na obrázku 7 s tabulkou na obrázku 10. V příkazu SELECT přibyly dva nové prvky. Vybírané sloupce mají před názvem uveden také alias tabulky, ze které jsou extrahovány. Důvodem tohoto přístupu je shoda názvů sloupců v různých tabulkách (což je i případ tohoto příkladu). Doplněním aliasu je jednoznačně identifikována tabulka, které sloupec náleží. Aliasy jsou definovány v příkazu FROM ve druhém řádku dotazu. Druhou novinkou je nové pojmenování sloupců za klíčovým slovem AS, které rovněž slouží k vypořádání se s nejasnostmi duplicitních názvů. Následuje klauzule WHERE, ve které se nyní, kromě již uvedeného požadavku na hodnotu ve sloupci *zeme*,

nachází propojení tabulek pomocí primárního a cizího klíče. Podmínkou tedy je, že primární klíč v tabulce *interpret* se musí rovnat cizímu klíči v tabulce *pisen*. Je zde rovněž nově použito slovo AND, které značí, že podmínky musí platit současně. Kromě slova AND lze na této pozici použít i slovo OR, kdy musí platit alespoň jedna z podmínek. Na konec je opět uveden řádek pro řazení, tentokrát však bez slova DESC, takže bude tabulka řazena abecedně. V takovém případě je možné také použít slovo ASC, nicméně se běžně vynechává, neboť záznamy budou i tak automaticky seřazeny vzestupně.

Obrázek 12: Tabulka pisen po aplikování příkazu SELECT

interpret	pisen
Motionless In White	570
Whitechapel	Bring Me Home
Hatebreed	Destroy Everthing
Stitched Up Heart	Finally Free
Hatebreed	I Will Be Heard
Stitched Up Heart	Monster

Zdroj: phpMyAdmin (2019), zpracováno autorem

Na obrázku 12 je zobrazena tabulka, která vznikne po vykonání výše uvedeného dotazu. Jak je vidět, výsledek obsahuje data z obou tabulek, které byly propojeny – sloupec *interpret* z tabulky *interpret* a sloupec *pisen* z tabulky *pisen*.

Takových dotazů je možné vytvořit velké množství a mohou obsahovat různá další specifika – za zmínku stojí agregátní funkce (například MAX, MIN, AVG) nebo klauzule GROUP BY, nicméně výše uvedené příklady stačí jako základ pro uvedení do problematiky vytváření dotazů pomocí příkazu SELECT.

1.5.2.2 Syntax DML – INSERT, UPDATE, DELETE

Obrázek 13: DML – příkazy INSERT a DELETE

```
INSERT INTO interpret (nazev, zeme)
VALUES ("Bring Me The Horizon", "Spojené království");

DELETE FROM interpret WHERE zeme = "USA";
```

Zdroj: vlastní zpracování, 2019

V prvním řádku kódu uvedeného na obrázku 13 je definováno, do které tabulky se budou data vkládat. V závorce jsou uvedeny konkrétní sloupce. Následně jsou za

slovem VALUES v závorce vypsány hodnoty, které do těchto sloupců budou vloženy. Třetí řádek uvádí příklad použití příkazu DELETE. Dojde k vymazání všech záznamů, které splňují podmínku uvedenou v klauzuli WHERE – v tomto případě se musí hodnota ve sloupci *zeme* rovnat *USA*.

Obrázek 14: DML – příkaz UPDATE

```
UPDATE interpret
SET nazev = "kapela_z_usa"
WHERE zeme = "USA";
```

Zdroj: vlastní zpracování, 2019

Kód uvedený výše na obrázku 14 ukazuje použití příkazu UPDATE. V prvním řádku je specifikována tabulka, ve které bude docházet ke změně záznamů. Na druhém řádku je definována samotná změna – v tomto případě dojde k dosazení hodnoty *kapela_z_usa* do sloupce *nazev*. Tato změna se projeví u záznamů, které splňují podmínku uvedenou v posledním řádku kódu – v tomto případě musí být hodnota ve sloupci *zeme USA*.

Kromě dvou základních kategorií – DDL a DML – uvedených v předchozích kapitolách je možné setkat se také s již zmíněným DQL a čtvrtou kategorií DCL (Data Control Language). Pro účely této práce jsou nicméně poslední dvě zmíněné kategorie nepodstatné.

1.5.3 T-SQL

Kromě základního použití jazyka SQL, které je uvedeno v předešlých kapitolách, existují také jeho rozšíření. Tato rozšíření mají různá označení vzhledem k tomu, kdo je výrobcem a v jakých systémech se používá. Vzhledem k zaměření této práce je nejrelevantnější volbou verze T-SQL, které je používáno systémem Microsoft SQL Server. Kromě T-SQL stojí dále za zmínku PL/SQL vytvořený firmou Oracle. Stejně jako pojmenování, tak rovněž jeho syntax se opět může lišit. Nicméně v základu slouží tato rozšíření k možnosti procedurálního programování. (Zhang, 2018)

1.6 Databázové objekty

Vzhledem ke skutečnosti, že se tato práce poměrně významně opírá o pojem „databázové objekty“, některé z nich si tu představíme. Takových databázových objektů existuje velké množství, proto zde bude uvedeno jen několik základních. Mnohé z nich

existují jako součást dalších databázových objektů. Databázové objekty se mohou stejně jako mnohé další aspekty lišit vzhledem k tomu, jaký databázový systém uživatel využívá. Vzhledem k tomu, že předmětem této práce je systém MS SQL Server, budou zde představeny objekty, se kterými pracuje právě tento systém.

1.6.1 Tabulka

Ačkoli se databázové objekty mohou, jak bylo zmíněno, často lišit s ohledem na databázový systém, tabulka je základním objektem každého z nich.

Tabulka slouží k ukládání dat a je tvořena dvěma dimenzemi – sloupci a řádky. Každý sloupec je navíc dále definován datovým typem, délkou, názvem a dalšími atributy. Datové typy vlastně slouží rovněž jako další databázový objekt. (CareerRide, 2020)

1.6.2 Pohled

Pohled (často také anglicky view) slouží k získání dat z jedné či více tabulek (nebo také jiných pohledů) a není to nic jiného než uložený kus SQL kódu typu SELECT, jehož výsledkem je opět nová tabulka. (CareerRide, 2020)

1.6.3 Klíče

Databázovými objekty jsou rovněž různé klíče, které se v databázi vyskytují. Kromě primárního a cizího klíče, jejichž funkce byla již vysvětlena v jedné z předešlých kapitol práce, to mohou být také další omezení v podobě klíčů – unikátnost nebo kontrolní klíč pro zakázání nechtěných dat. Kromě omezení reprezentovaných nějakým klíčem jsou za databázový objekt považována i jakákoli další definovaná databázová omezení. (CareerRide, 2020)

1.6.4 Funkce

Funkce je podobně jako pohled tvořena SQL kódem. Liší se však v tom, že funkce není na rozdíl od pohledu tvořena jednoduchým SELECT příkazem, ale kódem transakčního SQL, které bylo zmíněno na konci kapitoly o jazyku SQL. Její návratovou hodnotou může být skalární datová hodnota či celá tabulka. (CareerRide, 2020)

1.6.5 Procedura

Procedura je podobně jako výše popsaná funkce také tvořena kódem transakčního SQL. Na rozdíl od funkce však disponuje jednou zásadní nevýhodou – není možné ji použít v příkazech SQL jazyka. (CareerRide, 2020)

1.6.6 Index

Dalším významným databázovým objektem je index. Tento objekt využívá sloupcové indexové klíče a ukazatele k lokalizování záznamu. Účelem indexu je maximalizovat rychlost výkonu pohledů. Stejně jako klíčů, tak i indexů je několik druhů. (CareerRide, 2020)

1.6.7 Trigger

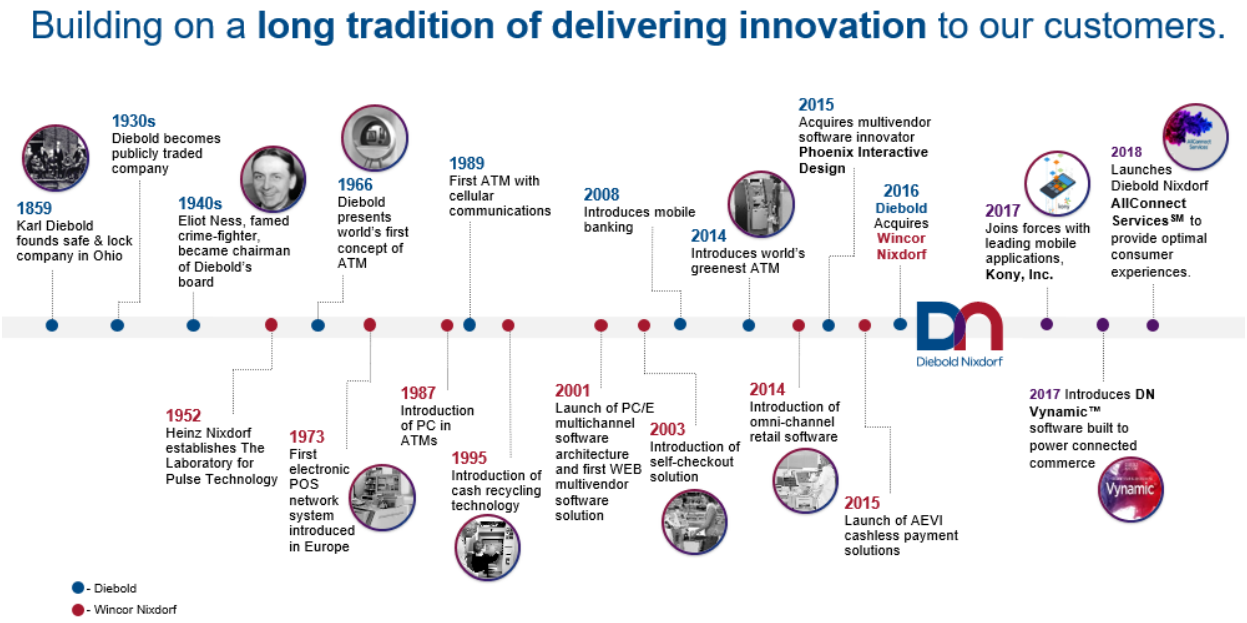
Trigger je speciální typ události, který je využíván v rámci procedur, které je schopen zavolat. Může být spuštěn pomocí příkazů INSERT, DELETE či UPDATE. Obecně trigger slouží ke kontrole referenční integrity. (CareerRide, 2020)

2 Společnost Diebold Nixdorf

Diebold Nixdorf s.r.o. je mezinárodní firma se sídlem ve Spojených státech amerických. Kromě tohoto sídla disponuje také velkým množstvím poboček, které jsou rozprostřeny různě po světě. Kromě mnoha evropských států se tyto pobočky nacházejí také v takových exotických destinacích, jako je Vietnam, Srí Lanka, Nepál na území Asie, Nigérie, Maroko, Alžírsko na území Afriky či Brazílie v Jižní Americe. (Diebold Nixdorf, 2019a)

2.1 Historie

Obrázek 15: Historie společnosti Diebold Nixdorf



Zdroj: Diebold Nixdorf (2019b)

Na obrázku 15 je vyobrazen stručný přehled historických milníků firmy Diebold Nixdorf. Kořeny této společnosti sahají až do roku 1859, kdy byla v americkém státě Ohio založena společnost, která se zabývala výrobou sejfů. Jejím zakladatelem byl Karl Diebold. O necelých sto let později byla Heinzem Nixdorfem založena Laboratoř pro pulzní technologii. V roce 1966 byl firmou Diebold představen první koncept ATM, kterému se následně obě společnosti věnovaly po dalších několik desetiletí. V roce 2016 došlo k významné události – společnosti Diebold a Wincor Nixdorf se spojily a započaly tak novou fázi, ve které již existují společně pod jménem Diebold Nixdorf a zabývají se produkty pro bankovnictví a retail. (Diebold Nixdorf, 2019b)

2.2 Produkt

Diebold Nixdorf je společnost, která poskytuje služby, software a hardware pro finanční a retailové odvětví v průběhu celého procesu. (Diebold Nixdorf, 2019a)

V oblasti bankovnictví i retailu se firma zaměřuje na digitální integraci, což vyžaduje vysoký stupeň začlenění samotného zákazníka do procesu. Zákazníci v dnešní době požadují rychlost, jednoduchost a vysoký bezpečnostní standard. Trendem této doby je také personalizace – tedy přizpůsobování požadavkům zákazníka. Správné řešení digitální integrace vede ke zvýšení efektivnosti, zvýšení výnosů a lepšímu řízení rizik. (Diebold Nixdorf, 2019a)

Firma Diebold Nixdorf nabízí několik druhů produktů (Diebold Nixdorf, 2019a):

- DN Vynamic software – softwarové produkty, které mají za účel především vytvoření kompaktní sítě, ve které budou zákazníci propojeni způsobem, který povede k výše popsaným efektům
- DN systémy – systémy firmy Diebold Nixdorf, které jsou navrženy pro začlenění do digitální éry
- DN AllConnect služby – služby poskytované firmou, které jsou zaměřeny především na digitalizaci, tyto služby propojují machine learning a pokročilé analytické metody s prediktivními a proaktivními přístupy takovým způsobem, aby bylo dosaženo optimálního řešení pro zákazníky, mezi takové služby patří například údržba či implementace

Řešení firmy Diebold Nixdorf jsou stavěna na třech základních pilířích (Diebold Nixdrof, 2019a):

1) Firma poskytuje produkt, který se vyvíjí již více než 150 let, a to ve více než 130 zemích. Poskytuje podporu ve velkém množství jazyků a disponuje zhruba 23 000 zaměstnanci po celém světě.
2) Firma se zaměřuje na technologie a inovace, a to především propojení fyzického světa s tím digitálním. Disponuje více než 3000 patenty a investuje okolo 156 milionů do oblasti výzkumu a rozvoje.
3) Firma disponuje infrastrukturou a prostředky k vývoji logistického řetězce, který se stává čím dál důležitějším s tím, jak se mění preference zákazníků. Ročně je

vyřešeno přes 10 milionů žádostí skrze telefonní podporu a každý den je zprostředkováno kolem 1,6 milionu transakcí platební kartou.

2.3 Plzeňská pobočka společnosti Diebold Nixdorf

Global Center Pilsen je strategická pobočka společnosti Diebold Nixdorf, která se nachází v Plzni. Pobočka je jedním ze šesti globálních vývojových center a zabývá se především odvětvím bankovnictví. Zaměřuje se na softwarové produkty, které řídí bankovní transakce a bankomaty na širokém poli působnosti – jak už bylo zmíněno, společnost Diebold Nixdorf působí ve více než 130 zemích po celém světě, a tak je schopna fungovat opravdu globálně. (Diebold Nixdorf, 2019c).

Vznik plzeňského globálního centra se datuje do roku 2015. V tomto roce začalo v centru pracovat okolo desíti lidí, přičemž dnes se tento počet rozšířil na číslo 170. Do tohoto počtu spadají kromě externistů a zaměstnanců, kteří spadají do firemních týmů, také zaměstnanci, kteří působí v zahraničí. Pobočka si během několika let dokázala vypracovat velice silnou pozici a stala se nedílnou součástí globálních projektů a produktů, které jsou v rámci těchto projektů vyvíjeny. Pobočka byla rovněž pověřena vývojem dohledového bankovního systému Vynamic View. Plzeňské globální centrum usiluje o neustálý růst a zisk nových strategických projektů, které budou z jejich pohledu perspektivní. (Diebold Nixdorf, 2019c)

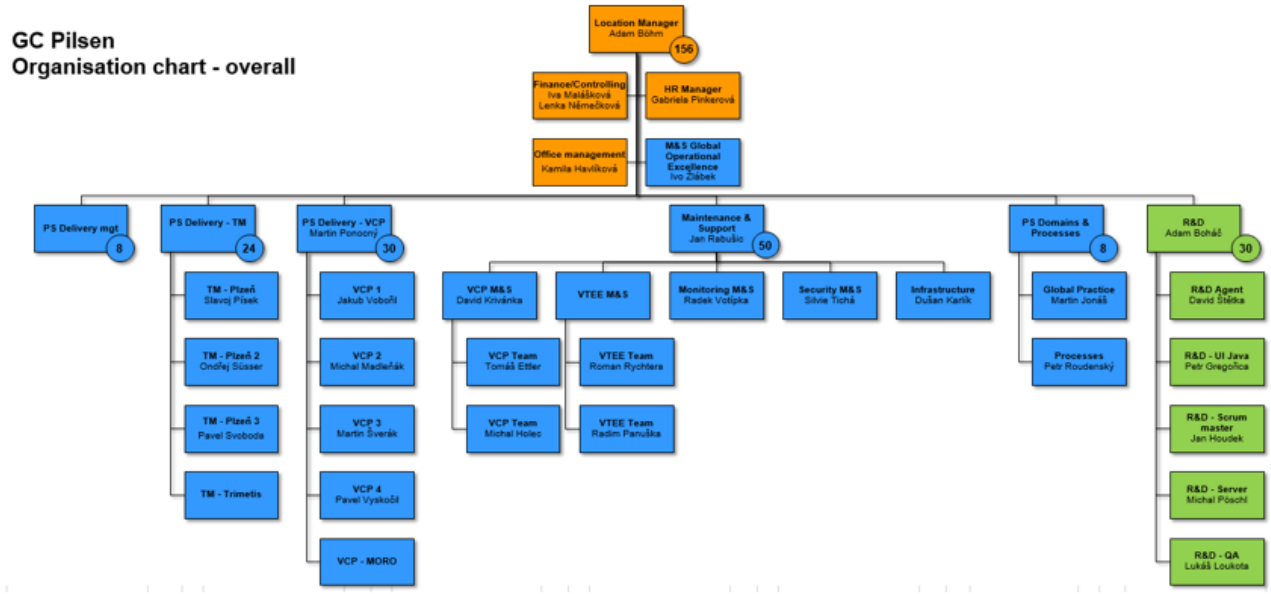
V plzeňské pobočce Global Center Pilsen operují tři týmy (Diebold Nixdorf, 2019c):

- Research & Development
- Delivery
- Maintenance & Support

V rámci těchto tří týmů je zpracováván komplexní produkt, jenž poté firma poskytuje svým zákazníkům, kterými jsou banky. V centru dochází kromě vývoje standardní verze produktu také k následnému přizpůsobování tohoto produktu pro specifické potřeby každého zákazníka. Po vytvoření produktu na míru přichází na řadu úsek Maintenance & Support, který má za úkol zajišťování vysoké kvality a bezproblémového chodu aplikací, které byly vytvořeny. (Diebold Nixdorf, 2019c)

2.3.1 Organizační struktura

Obrázek 16: Organizační struktura společnosti Diebold Nixdorf



Zdroj: Diebold Nixdorf (2019b)

Na obrázku 16 je zobrazena organizační struktura společnosti Diebold Nixdorf. Struktura je funkcionálního charakteru a na jejím vrcholu se nachází lokální manažer Adam Böhm. Následně je struktura organizována do funkčních celků, které korespondují s výše popsanými operujícími týmy (Delivery, Maintenance & Support, Research & Development), a celku, který se zabývá firemními procesy. Tyto celky jsou dále členěny na další menší týmy. (Diebold Nixdorf, 2019b)

3 Řešení

Pro dosažení hlavního cíle této práce existuje několik možných volně dostupných nástrojů a metod. V této části práce bude několik těchto možností představeno a porovnáno z různých perspektiv. Následně bude toto srovnání vyhodnoceno a bude vybrána jedna z možností, kterou dále použijeme pro naplnění hlavního cíle práce.

Na úvod by autor rád podotknul, že za jednu z možností by se teoreticky dala považovat také manuální kontrola každého jednoho objektu v databázích. Tento postup by mohl být aplikovatelný pro případy, kdy databáze obsahují několik málo objektů a proces by se neopakoval příliš často. Reálně je tím pádem tato metoda krajně nepraktická a ve většině případů nepoužitelná.

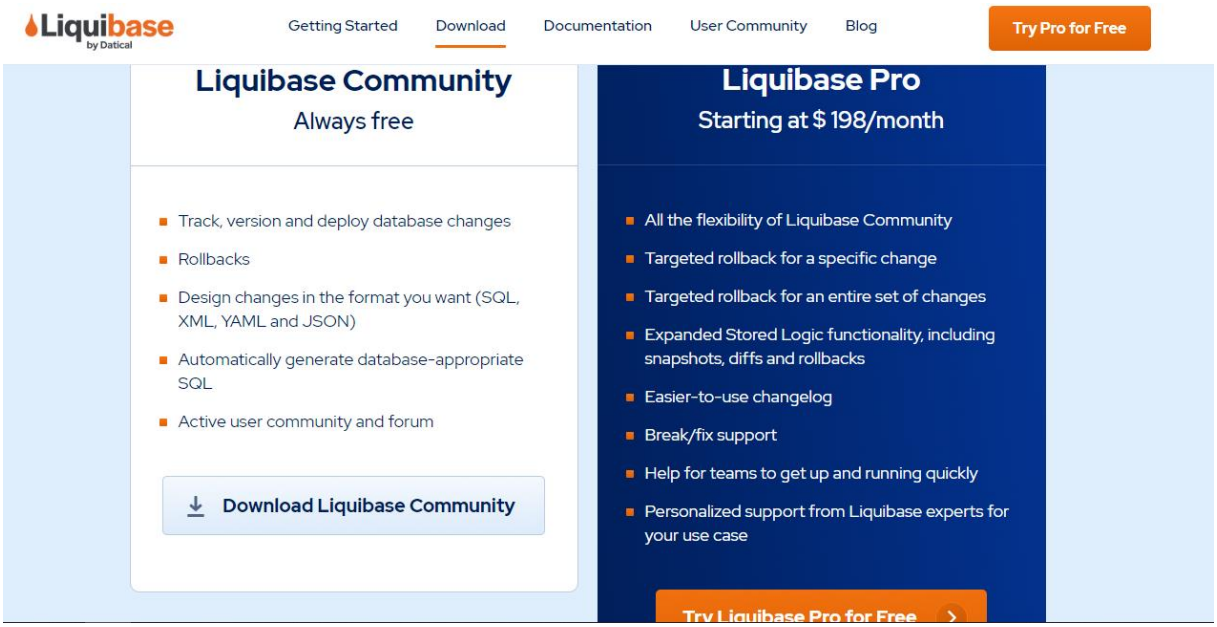
3.1 Liquibase

První možností, která již je také prakticky využitelná, je nástroj Liquibase.

Liquibase je nástroj od společnosti Datical, který slouží ke kontrole verzí databáze a umožňuje jejich snadné spojování a rozdělování. Dokáže evidovat změny a vytvářet o nich výsledné zprávy, které obsahují informace o tom, kdy, kde a jak byly změny provedeny. Schopností této aplikace je také oprava chyb v databázi. Kromě klasického SQL formátu dokáže rovněž pracovat s formáty JSON, XML a YAML. Mezi podporované databázové systémy patří kromě MS SQL Server, který je v této práci, také všechny další široce využívané databáze, jako je MySQL, Oracle, PostgreSQL či MariaDB. (Liquibase, 2020b)

Liquibase je možné stáhnout na webových stránkách https://www.liquibase.org. Je to open source software a nabízí verzi Liquibase Community, která je zcela zdarma, nebo Liquibase Pro, což je verze placená. I placenou verzi však lze vyzkoušet na limitovaný čas bez nutnosti placení. (Liquibase, 2020b)

Obrázek 17: Liquibase download



Zdroj: Liquibase (2020a)

Nástroj disponuje dvěma diskuzními fóry – uživatelským a vývojářským –, která však nevypadají, že by byla příliš aktivní. Na webových stránkách se kromě těchto diskuzních fór nabízí také možnost kontaktovat přímo vývojáře Liquibase. Dále je zde k dispozici YouTube kanál, který ale neobsahuje velké množství výukových videí. Dokumentace však nabízí poměrně obsáhlý, leč trochu nepřehledný individuální tutoriál pro každý databázový systém, který je nástrojem podporován.

Aby bylo možné s touto aplikací vůbec pracovat, je potřeba ji v první řadě nainstalovat. To je možné provést skrze instalační soubor, který lze stáhnout na webových stránkách po kliknutí na *Download Liquibase Community* a vyplnění e-mailové adresy. Dalším krokem je vytvoření souboru s parametry databáze, který je základním pilířem práce s Liquibase. Právě na modifikování tohoto souboru a provádění příkazů nad ním je nástroj stavěn. Pro účely této práce jsou k dispozici v rámci aplikace Liquibase dva možné příkazy. Prvním ze dvou příkazů je diff a druhým je diffChangeLog.

3.1.1 Příkaz diff

Příkaz diff v Liquibase umožňuje porovnat dvě databáze stejného nebo rozdílného typu. Příkaz diff se typicky využívá na konci projektu, protože umožňuje uživateli ověřit, že všechny změny, které uživatel očekával, se nacházejí v changelogu. Příkaz diff je rovněž užitečný pro následující problémy – nalezení chybějícího objektu v databázi,

zjištění, zda byla změna provedena v databázi, a nalezení neočekávaných položek v databázi. (Liquibase, 2020b)

Tento příkaz je tedy vhodný pro porovnání dvou databázových modelů. Výsledná zpráva sestává ze seznamu skládajícího se ze tří kategorií – Missing, Unexpected a Changed. Kategorie Missing slouží k evidování objektů, které nebyly nalezeny ve zdrojové databázi a chybí v ní tedy. Unexpected má za úkol naopak nalézat objekty, které se nachází ve zdrojové databázi, ale nejsou součástí databáze cílové. Changed zaznamenává objekty, které se nachází ve zdrojové i cílové databázi, avšak jsou nějakým způsobem odlišné. (Liquibase, 2020b)

Na obrázcích 18 a 19, které následují, je znázorněn příklad použití příkazu diff v rámci souboru s definovanými parametry databáze a příklad výstupní zprávy, která pojednává o rozdílech mezi databázemi.

Obrázek 18: Příkaz diff

```
liquibase
--outputFile=mydiff.txt
--driver=oracle.jdbc.OracleDriver
--classpath=ojdbc14.jar
--url="jdbc:oracle:thin:@<IP OR HOSTNAME>:<PORT>:<SERVICE NAME OR SID>"
--username=<USERNAME>
--password=<PASSWORD>
diff
--referenceUrl="jdbc:oracle:thin:@<IP OR HOSTNAME>:<PORT>:<SERVICE NAME OR SID>"
--referenceUsername=<USERNAME>
--referencePassword=<PASSWORD>
```

Zdroj: Liquibase (2020b)

Obrázek 19: Příkaz diff – výsledná zpráva

```
Diff Results:
Reference Database: MYSCHEMA2 @ jdbc:oracle:thin:@localhost:1521:ORCL (Default Schema:
Comparison Database: MYSCHEMA @ jdbc:oracle:thin:@localhost:1521:ORCL (Default Schema:
Compared Schemas: MYSCHEMA2 -> MYSCHEMA
Product Name: EQUAL
Product Version: EQUAL
Missing Index(s): NONE
Unexpected Index(s):
     PK_DATABASECHANGELOGLOCK UNIQUE  ON MYSCHEMA.DATABASECHANGELOGLOCK(ID)
     PK_DEPARTMENT UNIQUE  ON MYSCHEMA.DEPARTMENT(ID)
     PK_SERVICETECH UNIQUE  ON MYSCHEMA.SERVICETECH(ID)
     PK_SERVICETECH2 UNIQUE  ON MYSCHEMA.SERVICETECH2(ID)
     PK_SERVICETECH3 UNIQUE  ON MYSCHEMA.SERVICETECH3(ID)
Changed Index(s): NONE
Missing Table(s): NONE
Unexpected Table(s):
     DATABASECHANGELOG
     DATABASECHANGELOGLOCK
     DEPARTMENT
     SERVICETECH
     SERVICETECH2
     SERVICETECH3
Changed Table(s): NONE
Missing View(s): NONE
Unexpected View(s):
     VIEW1
Changed View(s): NONE
Liquibase command 'diff' was executed successfully.
```

Zdroj: Liquibase (2020b)

Ve výsledné zprávě lze vidět, že kromě rozdělení podle jednotlivých kategorií, je seznam dělen také na základě konkrétních databázových objektů. Je tedy vytvořen seznam samostatně pro tabulky, pohledy, indexy a všechny ostatní objekty, které jsou dále kategorizovány dle výše popsaných typů rozdílů.

3.1.2 Příkaz diffChangeLog

Příkaz diffChangeLog poskytuje informace o rozdílech mezi dvěma databázemi. Konkrétně je schopen poukázat na rozdíly obecně a vygenerovat změny k vyřešení většiny z nich. Příkaz diffChangeLog se obvykle používá, když chce uživatel vytvořit changelog, aby s jeho pomocí synchronizoval několik databází. Příkaz diffChangeLog rovněž poskytuje více informací o chybějících objektech v databázi, změnách provedených v databázi a neočekávaných položek v databázi. (Liquibase, 2020c)

Stejně jako příkaz diff v předešlé podkapitole lze stejným způsobem – tedy pomocí souboru s definovanými parametry databáze – použít příkaz diffChangeLog. Odlišnost mezi těmito dvěma příkazy spočívá ve schopnosti odstranění rozdílů v databázích. Příkaz diffChangeLog na rozdíl od příkazu diff nejenže rozdíly mezi vybranými databázemi odhalí, ale je schopen také vytvořit skript, který tyto rozdíly odstraní a databáze se tak stanou identickými.

3.1.3 Shrnutí Liquibase

Na základě výše popsaných informací lze tvrdit, že nástroj Liquibase je schopen splnit požadavky na hlavní cíl práce, tedy porovnat objekty ve dvou databázových modelech. Podpora produktu však nevypadá, že by byla příliš silná, neboť obě uživatelská fóra ani YouTube kanál nedisponují vysokou aktivitou. Práce s Liquibase je založená na definovaných příkazech, které jsou užívány v rámci souboru s definovanými parametry databází. Tento soubor je nutné ručně vytvořit. Výstupy příkazů jsou zobrazovány v textové podobě v příkazovém řádku. Tyto skutečnosti činí z nástroje poměrně uživatelsky nepřívětivé prostředí.

3.2 SQL Server Data Tools (SSDT)

SQL Server Data Tools je sada nástrojů, která funguje jako rozšíření pro prostředí Visual Studio od společnosti Microsoft. Tyto nástroje obsahují funkci, která je schopna porovnat dva databázové modely. Kromě právě připojené databáze lze využít také databázový projekt SQL Server, databázový otisk nebo .dacpac soubor. Výstupem porovnávací funkce je sada akcí, které je potřeba provést, aby se cílová databáze dostala do stavu databáze zdrojové. Výsledná zpráva je tedy sjednocena v první úrovni dle těchto akcí a následně podle jednotlivých objektů, které se ve vybraných databázích liší. Nástroj SSDT rovněž umožňuje vygenerovat a aktivovat skript, který tyto akce spustí a

převede tak cílovou databázi do stavu zdrojové databáze. Vygenerovanou sadu akcí je také možné uložit pro pozdější využití do samostatného souboru. (Microsoft, 2020c)

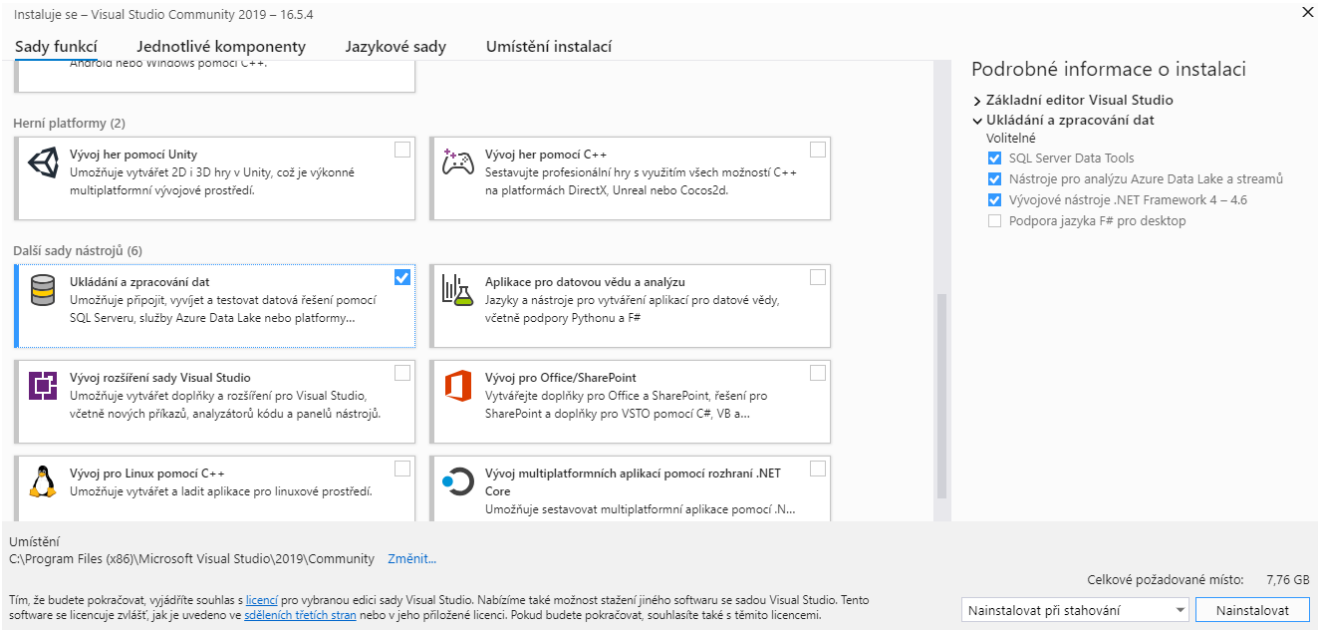
Jak již bylo řečeno v úvodu kapitoly, SQL Server Data Tools je rozšíření pro Visual Studio. V první řadě je tedy nutné stáhnout a nainstalovat samotné Visual Studio. To lze snadno provést na webových stránkách visualstudio.microsoft.com. Stejně jako nástroj Liquibase, tak i Visual Studio disponuje několika verzemi. Na stránkách společnosti Microsoft lze konkrétně nalézt verzi Komunita, která je jediná plně zdarma, a dále verze Professional a Enterprise, které jsou již placené. Důležité je, že sadu SSDT je možné využít již se základní Komunita verzí. Dokumentaci nástroje je vzhledem k obsáhlosti dokumentačních stránek společnosti Microsoft poměrně složité vyhledat, nicméně samotná dokumentace je již zpracovaná velice přehledně a srozumitelně.

Obrázek 20: Visual Studio download

Zdroj: Microsoft (2020b)

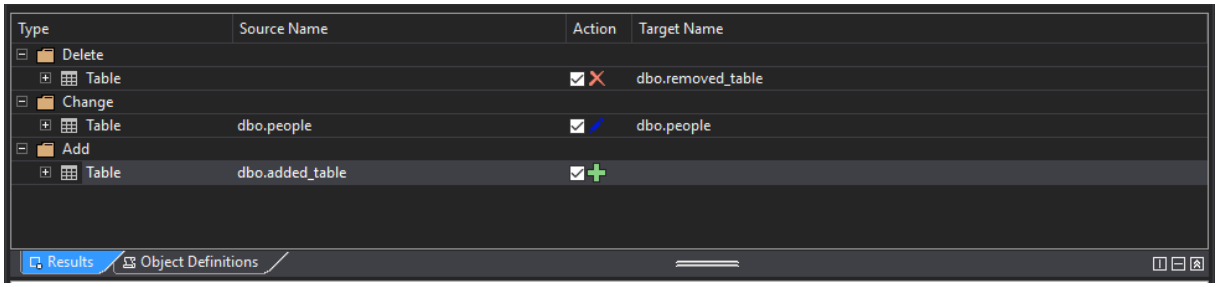
Obrázek číslo 20 ukazuje možnosti verzí Visual Studio 2019, které lze opatřit. Kromě stažení a instalace tohoto softwaru je nutné získat dále sadu SSDT. Toho lze docílit velice snadno, neboť instalační průvodce sám uživatele vyzve k vybrání nástrojů, které je třeba kromě samotného Visual Studio nainstalovat. Stačí tedy vybrat možnost *Ukládání a zpracování dat*, která se nachází v kategorii *Další sady nástrojů* a obsahuje potřebnou sadu nástrojů SQL Server Data Tools, jak je znázorněno na obrázku číslo 21, který následuje.

Obrázek 21: Instalace nástroje SSDT



Zdroj: Visual Studio (2020), zpracováno autorem

Obrázek 22: Výstup nástroje SSDT



Zdroj: SSDT (2020), zpracováno autorem

Obrázek číslo 22 znázorňuje výstup zpracování komparační analýzy nástroje SSDT. Lze si také všimnout výše popsaného rozdělení. Změny se zde řadí do tří kategorií – delete, change a add. Po rozkliknutí možnosti *Object Definitions* lze také nahlédnout na změny v podobě SQL kódu.

3.2.1 Shrnutí SSDT

Jak již bylo řečeno, funkce SSDT pro srovnání dvou databázových modelů umí zobrazit sadu akcí a vytvořit a spustit skript, který tak převede cílovou databázi do stavu zdrojové databáze. Dokáže tedy zkombinovat oba popsané příkazy nástroje Liquibase a je schopen naplnit hlavní cíl. Jeho instalace je velice jednoduchá a fakt, že mnoho uživatelů již disponuje softwarem Visual Studio (a také se tím pádem orientuje v tomto

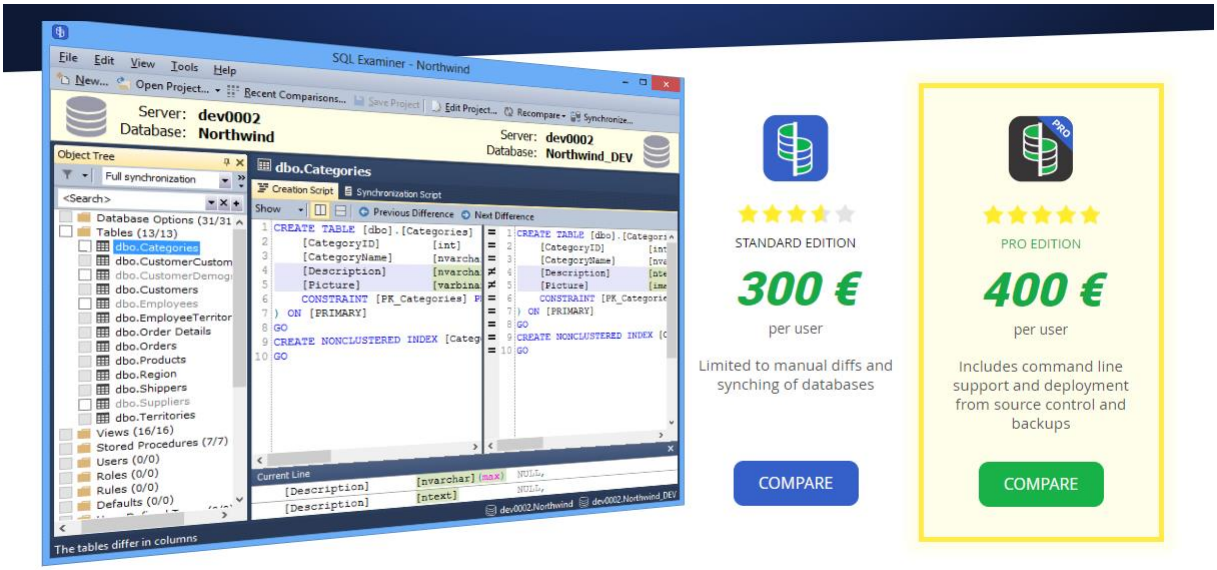
prostředí), tuto instalaci ještě usnadňuje, neboť stačí pouze doplnit balíček nástrojů, který obsahuje i SSDT. Jednotlivé položky výstupu, který je tvořen sadou akcí, je možné dále rozbalit a dozvědět se o nich více informací. SSDT však disponuje také jednou podstatnou nevýhodou – je vytvořen exkluzivně pro MS SQL Server databáze. To však v rámci této práce nevadí, neboť právě tento databázový systém je v ní využíván.

3.3 SQL Examiner

SQL Examiner je nástroj vytvořený společností Intelligent Database Solutions, který dokáže srovnat databáze a zobrazit objekty, které byly vytvořeny, smazány či změněny. Jeho další schopností je vybrat změny a aplikovat je na cílovou databázi. SQL Examiner disponuje podporou všech předních databázových systémů, mezi které patří MS SQL Server, MySQL, MariaDB, Oracle nebo PostreSQL. Porovnávání je možné v rámci živých databází, databázových záloh, otisků či SQL skriptů generujících databáze. (Intelligent Database Solutions, 2020).

Společnost Intelligent Database Solutions nabízí nástroj ve dvou možných variantách. Žádná z těchto variant však není zdarma, a je tedy nutné zaplatit poměrně vysoký licenční poplatek. Standard Edition je nabízena za 300 dolarů a Pro Edition za 400 dolarů. Uživatel také může využít možnost pořízení balíčku SQL Examiner Suite, ve kterém se kromě nástroje SQL Examiner porovnávajícího databázové modely nachází také nástroj SQL Examiner Data, který umí nalézt rozdíly v datech ve dvou databázích. V tomto případě jsou pak licence o sto dolarů dražší a vyjdou tedy na 400 a 500 dolarů. Oba nástroje je nicméně možné vyzkoušet v rámci zkušebního patnáctidenního trial období.

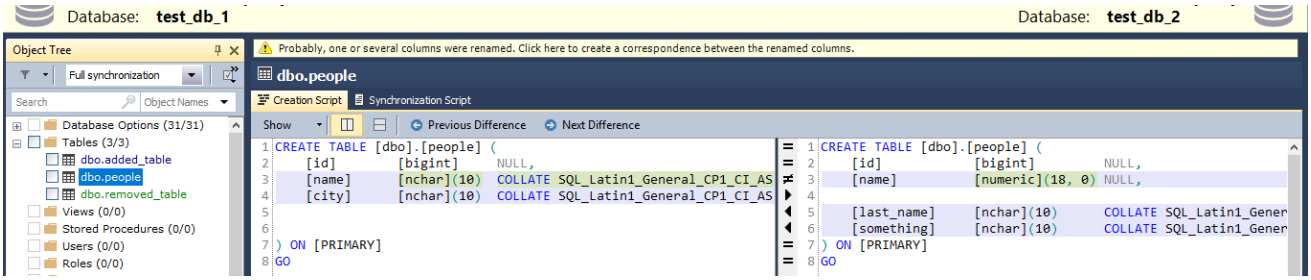
Obrázek 23: Verze nástroje SQL Examiner



Zdroj: Intelligent Database Solutions (2020)

Na stránkách nástroje, jimiž jsou www.sqlaccessories.com/sql-examiner, lze nalézt dokumentaci s návody na jednotlivé aktivity, které SQL Examiner Suite poskytuje. Tato dokumentace je velice přehledně a stručně zpracována a obsahuje návody rozdělené na základě používaného databázového systému. Výsledná zpráva sestává ze tří kategorií odlišností – objekty, které se nacházejí v obou databázích, ale jsou odlišné, objekty, které se nacházejí pouze v cílové databázi, a objekty, které se nachází pouze ve zdrojové databázi. Výstup kromě těchto tří kategorií odlišnosti disponuje také kategorií, která obsahuje objekty, jež se v rámci porovnávaných databází nijak neliší. Objekty je dále možné blíže prozkoumat po kliknutí na ně.

Obrázek 24: Výstup nástroje SQL Examiner



Zdroj: SQL Examiner (2020), zpracováno autorem

Na obrázku 23 je vidět výsledná zpráva, která pojednává o rozdílech mezi databázovými modely. Na levé straně se nachází sloupec se všemi objekty, které jsou

rozděleny podle typu objektu. Dále si lze všimnout také barevného rozdělení podle kategorie rozdílu.

3.3.1 Shrnutí SQL Examiner

Rovněž třetí vybraný nástroj – SQL Examiner – obsahuje funkce, které jsou vhodné pro splnění hlavního cíle práce. Dokumentace je přehledně a srozumitelně zpracována a rovněž výstupy jsou velice uživatelsky přívětivé. Také instalace je velmi rychlá a nenáročná. Tyto výhody jsou však vykoupeny absencí neplacené verze a nutností zaplatit poměrně vysoký licenční poplatek.

3.4 Vyhodnocení nástrojů

Na základě informací, které byly zjištěny o třech výše popsaných nástrojích, lze usoudit, že Liquibase, SQL Server Data Tools i SQL Examiner jsou nástroji, které disponují funkcemi, jež dokáží naplnit hlavní cíl této práce. Byly rovněž nastíněny výhody a nevýhody na základě různých parametrů.

Softwarový nástroj Liquibase se zdá být nejméně vhodný z pohledu přívětivosti uživatelského prostředí. Nejenže je oproti ostatním dvěma možnostem poměrně složité s ním pracovat, ale také výstupy jsou ve velice jednoduché textové formě. Rovněž jeho dokumentace vypadá nejméně přehledně ze všech výše popsaných nástrojů.

Nástroje SSDT a SQL Examiner nabízí přehlednější a využitelnější výsledné zprávy. V oblasti dokumentace se zdá být lepším nástrojem SQL Examiner, neboť nabízí návody, které ukazují krok za krokem postupy i s přidanými obrázky, avšak také SSDT disponuje poměrně srozumitelnou dokumentací. Z informací o obou variantách, které byly popsány v předešlých kapitolách, lze říci, že jsou si nástroje značně podobné v klíčových faktorech, tedy formát výstupu, zpracování dokumentace, složitost zprovoznění a využití. SQL Examiner však oproti SSDT disponuje jednou esenciální nevýhodou – nutností zaplatit po 15 dnech licenční poplatek. Tato skutečnost uvádí SSDT do značné výhody vzhledem k velké podobnosti v ostatních aspektech. Další faktor, který autor považuje za výhodu SSDT, je skutečnost, že SSDT je stejně jako MS SQL Server, který je v práci využíván, produktem společnosti Microsoft a je tedy očekávána bezproblémová podpora. Tato skutečnost s sebou však nese jednu již

zmíněnou nevýhodu, kterou je podpora pouze pro databáze MS SQL Server. To ale v rámci této práce není překážkou.

3.4.1 Finální výběr

Na základě vyhodnocení a výše popsaných důvodů se autor této práce rozhodl využít pro účely splnění hlavního cíle nástroj SQL Server Data Tools (SSDT). Tento software tedy bude použit v následující části práce pro porovnání rozdílů mezi objekty v modelech testovacích databází a jejich následnou interpretaci.

Tento nástroj byl vybrán, neboť autorovým cílem je poskytnout, popsat a otestovat pro firmu Diebold Nixdorf postup, který bude schopen odhalit rozdíly mezi dvěma modely databázového systému MS SQL Server a který bude případně firma následně dále využívat v reálné praxi, aby byla schopna odhalit, zda někdo nemanipuloval s objekty v jejich databázi. Autor tedy v této práci provedl průzkum, v rámci kterého zjistil, jaké existují možnosti k dosažení požadovaného cíle. Následně tyto možnosti porovnal a vybral z nich tu, kterou vyhodnotil jako nejvhodnější pro konkrétní zadání této práce.

4 Postup pro porovnání objektů dvou databázových modelů

Pro dosažení cílů, které byly stanoveny na začátku této práce, bude potřeba zprovoznit a využít několik softwarových produktů. Prvním z nich je VirtualBox od společnosti Oracle. Tento software umožňuje vytvoření virtuálního počítače. Dalším z množiny programových nástrojů bude Microsoft SQL Server Management Studio. Tento nástroj bude využit již přímo ve VirtualBoxu a umožní prohlížení a práci s databází a jejími objekty. Posledním nástrojem, ve kterém bude i samotný proces zpracováván, je SQL Server Data Tools. Tento software nabízí možnost porovnání dvou databázových modelů. Vzhledem k tomu, že se jedná o rozšíření pro prostředí Visual Studio, bude nutné opatřit rovněž tento software.

4.1 Zprovoznění nástroje VirtualBox

VirtualBox je software, jenž umožňuje vytvoření virtuálního stroje pro firemní i domácí využití. Je to open source software vyvíjený společností Oracle. Mezi nabízenými možnostmi se nachází jak Windows, tak i operační systémy, které jsou založené na unixovém jádru. (Oracle, 2020)

Důvodem potřeby využití aplikace VirtualBox je ten, že společností Diebold Nixdorf byl dodán virtuální otisk počítače, na kterém se nacházejí testovací databáze. Pro účely této práce bude vyhovující operační systém Windows 10.

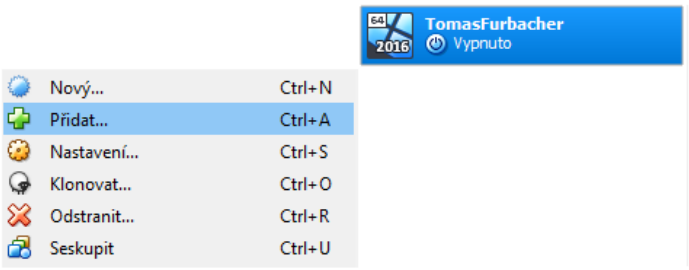
V první řadě je potřeba VirtualBox stáhnout z webových stránek virtualbox.org. Pro tuto práci byla vybrána verze 5.2.26. Po kliknutí na možnost *Downloads*, která se nachází na levé straně domovských stránek, je třeba zvolit odkaz *VirtualBox 5.2 builds*. Po rozkliknutí tohoto odkazu je nutné nalézt správnou verzi a po jejím nalezení kliknout na možnost *Windows hosts*. Touto aktivitou se spustí stahování instalačního souboru. Tento postup je zobrazen na obrázku 25.

Obrázek 25: VirtualBox download

Zdroj: Oracle (2020)

Dalším krokem po dokončení instalace je vytvoření nového virtuálního stroje. Toho dosáhneme spuštěním programu VirtualBox, kliknutím na záložku *Počítač* a následně na možnost *Přidat*. Firma Diebold Nixdorf poskytla potřebný soubor pro nahrání nového virtuálního stroje. Je tedy potřeba vybrat tento soubor. Po nějaké době se v nabídce virtuálních strojů objeví nová možnost reprezentující přidaný počítač. Tento počítač se spustí pomocí dvojitého kliknutí.

Obrázek 26: Vytvoření virtuálního počítače



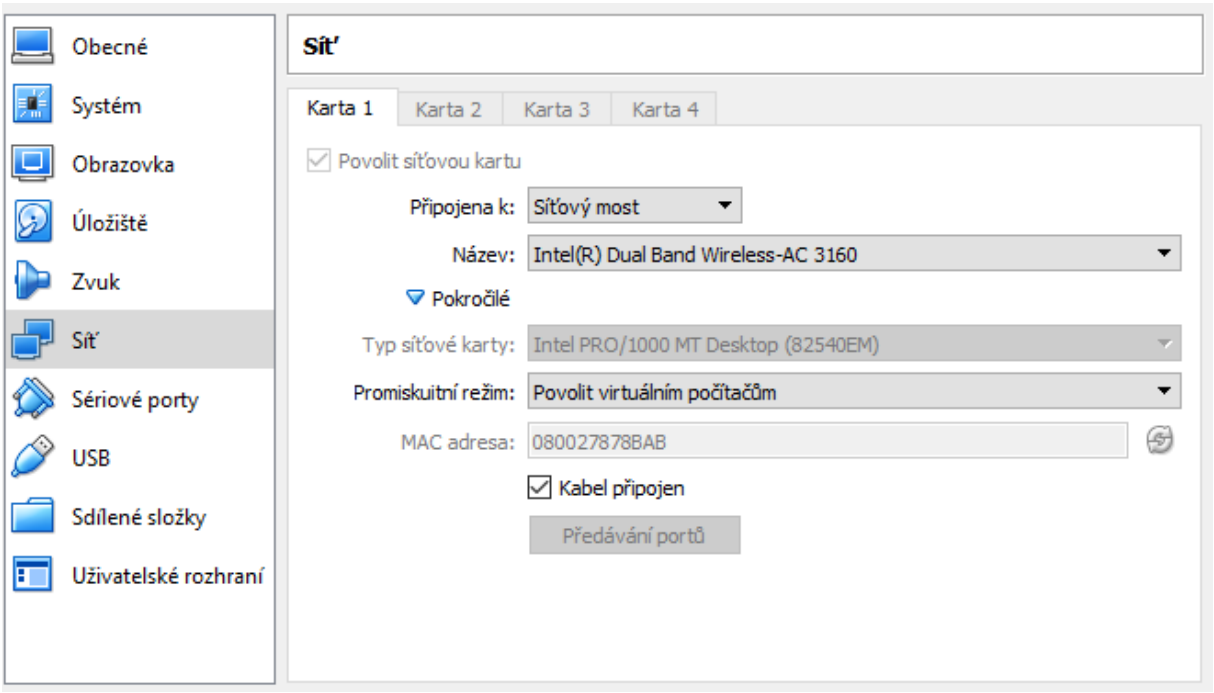
Zdroj: VirtualBox (2020), zpracováno autorem

4.1.1 Zprovoznění internetové sítě

Tato podkapitola se v práci nachází z toho důvodu, že zprovoznění internetu v prostředí VirtualBox nemusí být triviální záležitostí. Vzhledem k tomu, že bude potřeba stáhnout

a nainstalovat několik softwarových nástrojů přímo ve virtuálním počítači, je zprovoznění připojení nezbytné. Pro prvotní připojení k internetu v nástroji VirtualBox je nejprve nutné provést několik kroků. V první řadě jde o síťové nastavení přímo v aplikaci. K tomuto nastavení se uživatel dostane kliknutím pravého tlačítka myši na vytvořený virtuální stroj a následným zvolením možnosti *Nastavení*. Následně je nutno přepnout na záložku *Síť*. Zde je třeba změnit připojení na *Síťový most* a následně vybrat odpovídající název. Následně je také potřeba rozbalit záložku *Pokročilé* a nastavit zde *Promiskuitní režim* na možnost *Povolit virtuálním počítačům*. Následně stačí potvrdit nastavení a tím jsou splněny veškeré kroky, které jsou potřebné vykonat přímo v prostředí VirtualBox.

Obrázek 27: Nastavení sítě v prostředí VirtualBox

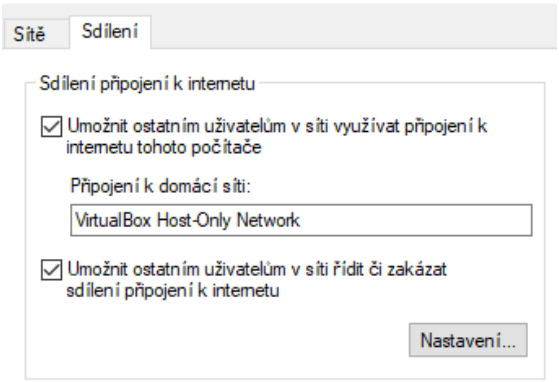


Zdroj: VirtualBox (2020), zpracováno autorem

Takto upravené nastavení však není dostačující k tomu, aby byl zprovozněn internet na virtuálním stroji. Nyní ještě musíme povolit sdílení internetu v hostujícím počítači. Tohoto stavu dosáhneme spuštěním *Ovládacích panelů* následně vybráním možnosti *Centrum síťových připojení a sdílení*. Na nově otevřené kartě je následně třeba zvolit možnost *Změnit nastavení adaptéru*, které se nachází v levé horní části. Tato činnost způsobí otevření nové záložky se seznamem možných připojení. Zde vybereme možnost, ke které chceme, aby byl virtuální stroj připojen, klikneme na ni pravým

tlačítkem myši a zvolíme možnost *Nastavení*. Přepnutím na kartu *Sdílení*, zaškrtnutím prvního políčka a potvrzením celého procesu docílíme zprovoznění internetu ve virtuálním stroji.

Obrázek 28: Nastavení sítě v prostředí Windows



Zdroj: Windows (2020), zpracováno autorem

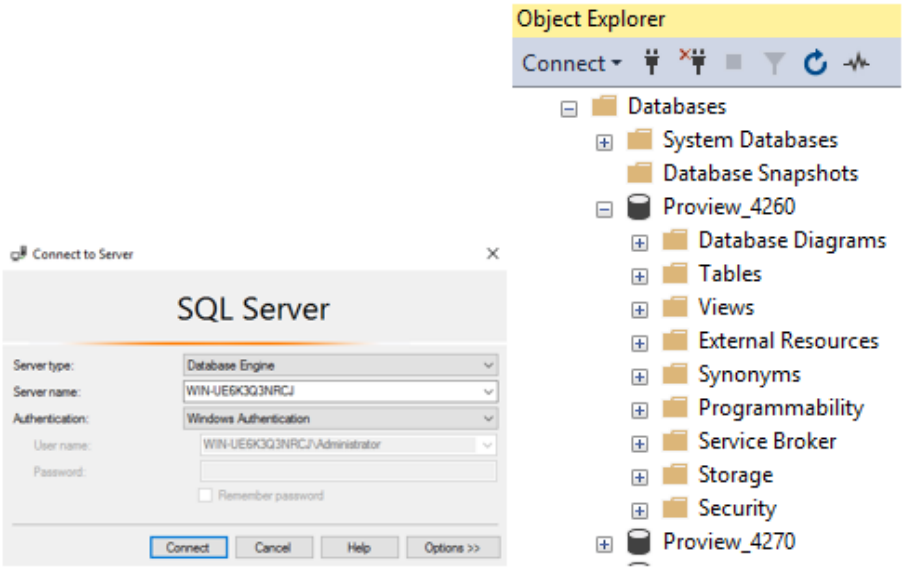
4.2 Zprovoznění nástroje Microsoft SQL Server Management Studio (SSMS)

Tento nástroj vyvíjený společností Microsoft slouží ke správě databází založených na technologii SQL. SSMS umožňuje monitorovat a upravovat datové komponenty a vytvářet SQL dotazy a skripty. (Microsoft, 2020a)

Microsoft SQL Server Management Studio, konkrétně verze 2017, je již nainstalovaný a připravený na virtuálním počítači, který byl poskytnut firmou Diebold Nixdorf, a není ho tedy potřeba znovu stahovat. Zároveň již obsahuje testovací databáze, na kterých bude aplikován postup pro jejich porovnávání.

Microsoft SQL Server Management Studio lze spustit pomocí ikony na ploše. Následně se nástroj otevře a objeví se nové okno, na kterém je třeba kliknout na *Connect*. Po této operaci se již zobrazí lišta s připojenými databázemi. Pro účel testování pro tuto práci jsou určeny hned čtyři databáze a to *proview_4260*, *proview_4270*, *ReportServer* a *ReportServerTempDB*. Po kliknutí na symbol „+“ u těchto databází jsou již vidět jednotlivé databázové objekty – například tabulky či pohledy – které jsou předmětem této práce. Tyto objekty lze mimo jiné vidět na následujícím obrázku.

Obrázek 29: Prostředí nástroje SSMS



Zdroj: SSMS (2020), zpracováno autorem

Jak již bylo řečeno, nacházejí se v SSMS čtyři testovací databáze. Ty neobsahují žádná data, neboť ta nejsou předmětem práce a nejsou tedy zapotřebí. Obsahují však velké množství databázových objektů, které jsou naopak pro práci naprosto zásadní. Kromě tradičního objektu – tabulky – s různými klíči, omezeními a datovými typy lze v databázích nalézt také celou řadu dalších, v první části práce popsaných databázových objektů. Na obrázku 29 lze kromě zmíněných tabulek objevit také pohledy (views). Jiné objekty jsou skryté pod dalšími složkami. Například ve složce *Programmability* lze najít kromě procedur a funkcí také triggery. Ve složce *Security* naopak najdeme různé typy databázových klíčů. Objektů pro potřeby testování se tedy v databázích nachází velké množství. Kromě jejich množství je také jejich rozdílnost značná a také tato vlastnost je pro potřeby práce velice důležitá.

4.3 Spuštění Visual Studio a SSDT

Instalaci a spuštění prostředí Visual Studio a jeho rozšiřujícího nástroje se již věnovala kapitola 3.2 a nebude tedy v této části již znovu popisována, pouze je vhodné zmínit, že autor pracuje s verzí Visual Studio 2019.

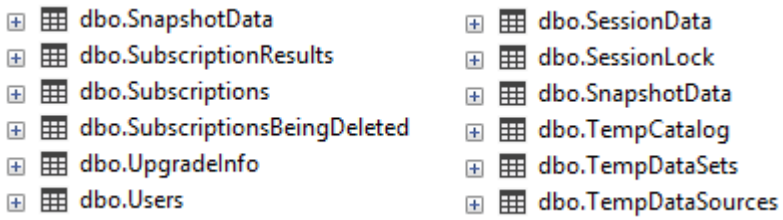
4.4 Popis postupu

Tato kapitola představuje stěžejní část práce, neboť půjde o samotný popis postupu pro porovnání databázových modelů a jeho následnou aplikaci na testovací databáze, jak je definováno v hlavním cíli práce.

4.4.1 Rozdíly mezi databázemi v SQL Server Management Studio

V první řadě je nutné spustit VirtualBox a zapnout virtuální počítač, jenž byl importován v jedné z předchozích kapitol. Počítač se může spouštět poměrně dlouho – i několik minut. Poté, co se zapne, je třeba spustit SQL Server Management Studio a připojit se k serveru s databázemi, což bylo rovněž popsáno již v jedné z předešlých kapitol. V této fázi budou využity pro srovnání databáze *ReportServer* a *ReportServerTempDB*. V těchto databázích je totiž možné si hned na první pohled povšimnout mnoha rozdílů. Příkladem rozdílu mohou být například tabulky *TempCatalog*, *TempDataSets* a *TempDataSources*, které se nacházejí v databázi *ReportServerTempDB*, avšak v databázi *ReportServer* již chybí. Naopak například tabulky *UpgradeInfo* a *Users* se nachází v databázi *ReportServer,* ale nejsou v databázi *ReportServerTempDB*. Vše je znázorněno na obrázku číslo 30. Takových rozdílů mezi těmito dvěma databázemi lze najít mnoho i mezi jinými objekty, než jsou jen tabulky, ale pro demonstraci funkcionalit nástroje SSDT toto prozatím postačí.

Obrázek 30: Rozdíly v databázích



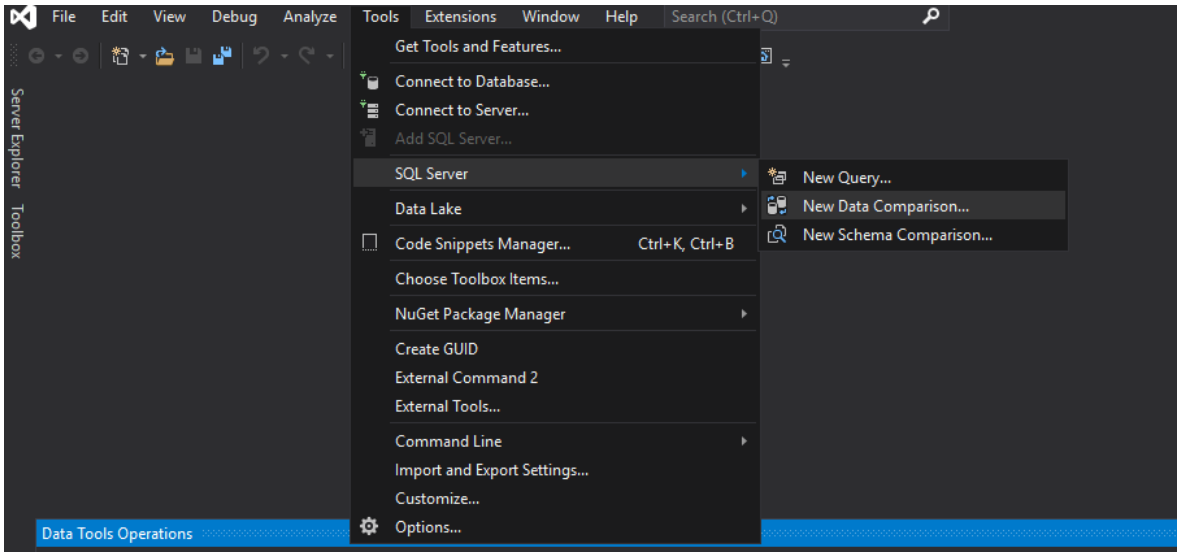
Zdroj: SSMS (2020), zpracováno autorem

4.4.2 Práce v SQL Server Data Tools

Nyní se přesuneme do samotného nástroje SSDT. Do toho je možné dostat se pomocí spuštění prostředí Visual Studio. Pro potřeby srovnání testovacích databází je potřeba zvolit možnost *Continue without code*. Po tomto úkonu se již spustí samotné prostředí Visual Studio a zde je nutné otevřít záložku *Tools*, následně zvolit možnost *SQL Server*

a konečně kliknout na *New Scheme Comparison*, jak je ukázáno na obrázku číslo 31, který nyní následuje.

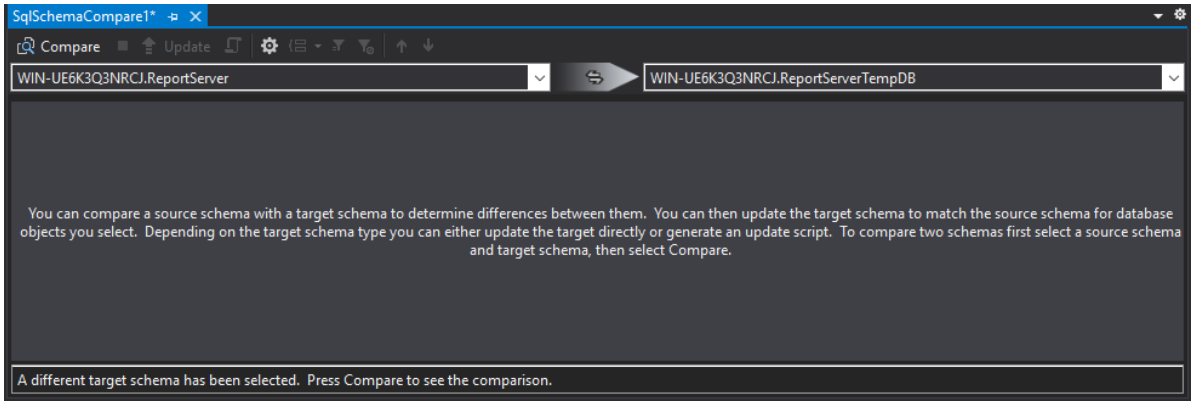
Obrázek 31: Spuštění porovnávacího nástroje



Zdroj: SSDT (2020), zpracováno autorem

Po tomto aktu se již otevře karta pro porovnávání. Je potřeba zvolit zdrojovou databázi (*Select Source*). Tato databáze představuje tu, do jejíhož stavu chceme dostat cílovou databázi. Je třeba zvolit možnost *Database* a poté *Select Connection*. Následně se otevře nová karta, na které zvolíme možnost *Browse*, poté *Local* a zde vybereme příslušný server (v našem případě *WIN-UE6K3Q3NRCJ*), zvolíme databázi (pro nás *ReportServer*) a klikneme na tlačítko *Connect*. Stejným způsobem navolíme také cílovou databázi (*Select Target*), kterou bude *ReportServerTempDB*. Výsledek nyní popsaného postupu je zobrazen na obrázku 32.

Obrázek 32: Výběr testovaných databází



Zdroj: SSDT (2020), zpracováno autorem

Nyní je vše připraveno k porovnání a stačí kliknout na tlačítko *Compare*. Generování výstupů může nějakou chvíli trvat. Samozřejmě záleží na velikosti databází, počtu objektů, které se v nich nachází, a také množství rozdílů. Nicméně ve spojení s VirtualBoxem zabere tato operace přibližně několik desítek vteřin.

Obrázek 33: Výstup srovnání databází

Type	Source Name	Action	Target Name
Delete			
Table			dbo.ExecutionCache
Table			dbo.newTableAddedByFurbacher
Table			dbo.PersistedStream
Table			dbo.SessionData
Table			dbo.SessionLock
Table			dbo.TempCatalog
Table			dbo.TempDataSets
Table			dbo.TempDataSources
Change			
Table	dbo.SegmentedChunk		dbo.SegmentedChunk
Table	dbo.SnapshotData		dbo.SnapshotData
Procedure	dbo.GetDBVersion		dbo.GetDBVersion
Add			
Table	dbo.ActiveSubscriptions		
Table	dbo.Batch		
Table	dbo.CachePolicy		
Table	dbo.Catalog		
Table	dbo.ConfigurationInfo		

Zdroj: SSDT (2020), zpracováno autorem

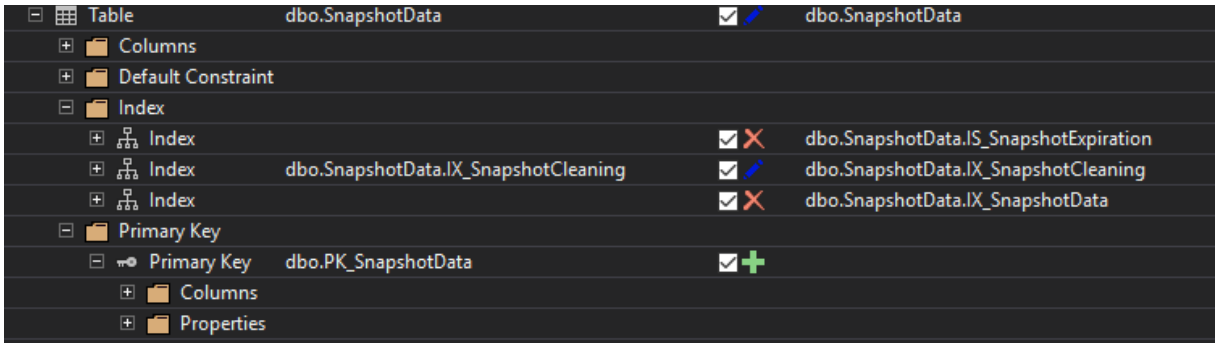
Výstupem je tedy, jak už bylo zmíněno, seznam operací, které je třeba provést, aby se cílová databáze dostala do stavu zdrojové databáze. Vyskytují se zde tři typy operací:

- delete – tato operace určuje, které databázové objekty je potřeba smazat z cílové databáze. Vzhledem k tomu, že cílem je dostat se ze stavu cílové databáze do stavu zdrojové databáze, znamená to, že objekty, které jsou označeny akcí *delete*, se vyskytují pouze v cílové databázi a ve zdrojové už nejsou. Operace je graficky znázorněna jako červený křížek.
- change – operace *change* označuje, že je potřeba provést nějakou změnu v definici objektu. Intepretace je v tomto případě taková, že objekt s označením *change* se sice nachází v obou databázích, nicméně je v každé databázi jinak definovaný. Operace je graficky znázorněna jako modrá tužka.
- add – poslední operací je *add*. Ta označuje, že je objekt naopak nutné do cílové databáze přidat. V praxi to tedy znamená, že takový objekt se nachází pouze ve zdrojové databázi a v cílové databázi chybí. Operace je graficky znázorněna jako zelené znaménko plus.

Na obrázku číslo 33 lze vidět, že objekty jsou kromě shlukování podle akcí, které je třeba provést, rozdělené také podle typu objektu – zde jsou vyobrazeny objekty typu *table* a *procedure*. Na obrázku je také možné si všimnout změny avizované již před spuštěním komparativního nástroje. Mezi objekty, které jsou označeny operací *delete*, se nachází také tři tabulky *TempCatalog*, *TempDataSets* a *TempDataSources*. Toto značí, že tabulky se nacházejí pouze v cílové databázi a je tedy potřeba je smazat, což je přesně to, co jsme si ukázali před aplikací nástroje SSDT.

Každou jednu změnu je také možné rozbalit a podívat se na ni detailněji. Lze tedy vidět, které konkrétní součásti objektu mají být smazány, změněny či přidány. Tato skutečnost je pak velice užitečná především u objektů označených operací *change*.

Obrázek 34: Detail operace change

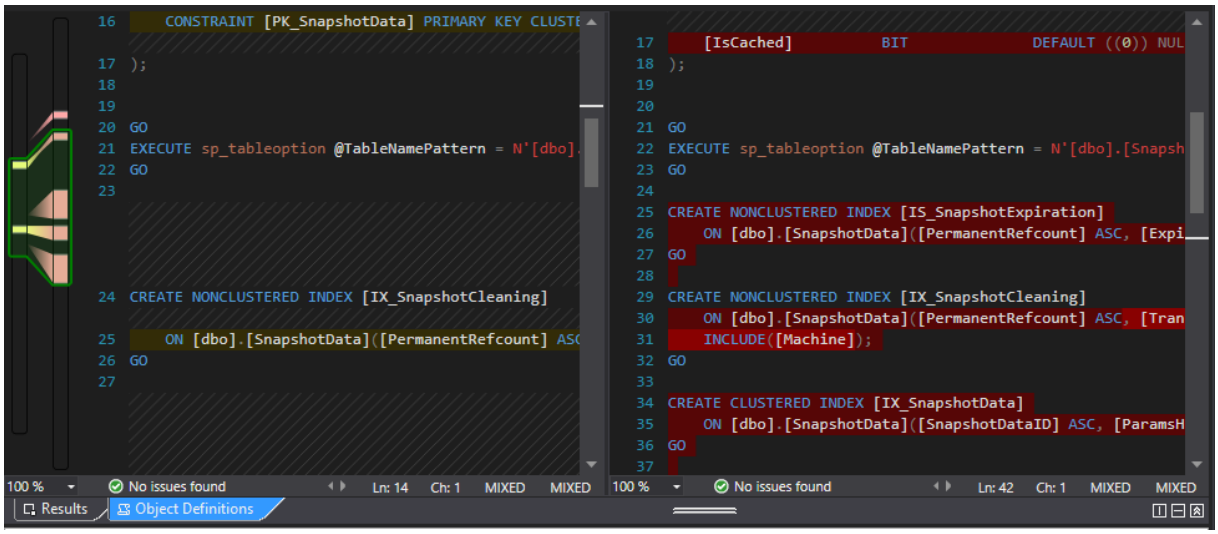


Zdroj: SSDT (2020), zpracováno autorem

Na obrázku 34, jenž lze vidět výše, je znázorněno částečné rozbalení tabulky *SnapshotData*, která nese označení *change*, což je poznat již na obrázku číslo 33. Lze vidět, že tabulka se liší ve čtyřech odlišných částech. Kromě sloupců a omezení je tabulka rozdílná také v indexech a primárním klíči. Poslední dvě jmenované součásti jsou pro příklad dále rozbaleny. Objekty jsou tedy v tomto případě situovány do jakéhosi stromového systému a po postupném rozbalování je možné vidět, kde konkrétně se objekty liší a jakou operaci je zapotřebí na konkrétních místech vykonat. V operaci *change* se tedy skrývá soubor dalších operací, které mohou kromě dalších operací *change* nabývat také již zmíněných hodnot *add* a *delete*. Toto všechno vidíme po rozbalení tabulky *SnapshotData* na obrázku 34.

Nástroj SSDT však umožňuje jít ještě více do hloubky.

Obrázek 35: SSDT – Object Definitions



Zdroj: SSDT (2020), zpracováno autorem

Po přepnutí na možnost *Object Definitions*, která se na obrázku číslo 35 nachází na levé dolní straně, umožňuje SSDT nahlédnout na rozdíly mezi objekty také v podobě definičního kódu objektů. Toto lze opět aplikovat na jednotlivé části objektů a je tedy možné podívat se pouze na to, jak se liší definice primárního klíče nebo indexů.

Obrázek 36: Tabulky v kategorii Add

Zdroj: SSDT (2020), zpracováno autorem

Pouze pro úplnost si na obrázku 36 ukážeme, že dvě dříve zmíněné tabulky *UpgradeInfo* a *Users*, které naopak chybí v cílové databázi, nesou označení *add* a je tedy nutné je kompletně přidat.

4.4.3 Manipulace s databázovými objekty

Nyní si ještě ukážeme, jak se projeví obsah výstupu poté, co se přidají nové databázové objekty.

Do databáze *ReportServer* přidáme novou tabulku, kterou nazveme *AddedTable*. Půjde tedy o případ, kdy se objekt bude nacházet pouze ve zdrojové databázi. Tabulka bude obsahovat sloupec *ID*, jenž bude datového typu bigint, sloupce *Name* a *LastName*, které budou datového typu nvarchar, a sloupec *Phone*, jemuž bude přidělen datový typ int. Definice tabulky je zobrazena na následujícím obrázku.

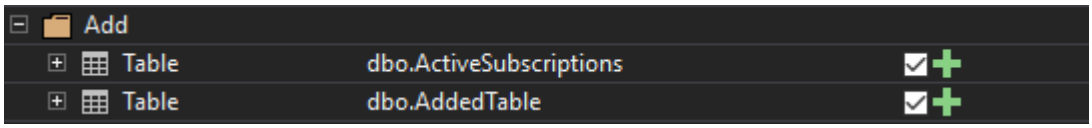
Obrázek 37: Sloupce tabulky AddedTable v databázi ReportServer

Column Name	Data Type	Allow Nulls
ID	bigint	☐
Name	nvarchar(50)	☑
LastName	nvarchar(50)	☑
Phone	int	☑

Zdroj: SSMS (2020), zpracováno autorem

Nyní se opět přesuneme do nástroje SSDT a stejným způsobem, jaký byl popsán v předešlé podkapitole, provedeme komparaci objektů stejných testovacích databází.

Obrázek 38: Komparace po přidání tabulky do databáze ReportServer



Zdroj: SSDT (2020), zpracováno autorem

Na obrázku číslo 38 je znázorněn očekávaný výsledek. V kategorii pro operaci *add* se objevil nový objekt – tabulka *AddedTable*. Tedy přesně ta nová tabulka, která byla vytvořena v předchozím kroku.

Další změnou, která bude v databázi provedena, je vytvoření tabulky stejného názvu, avšak s nepatrně rozdílnou definicí, v druhé testovací databázi *ReportServerTempDB*. Sloupec *ID* bude v tomto případě povolovat hodnotu null. Atributu *Name* bude změněn datový typ z nvarchar na nchar. *LastName* zůstane nezměněno a místo sloupce *Phone* bude v této tabulce vytvořen sloupec *City* s datovým typem nvarchar. Definice tabulky je opět znázorněna na obrázku 39, který nyní následuje.

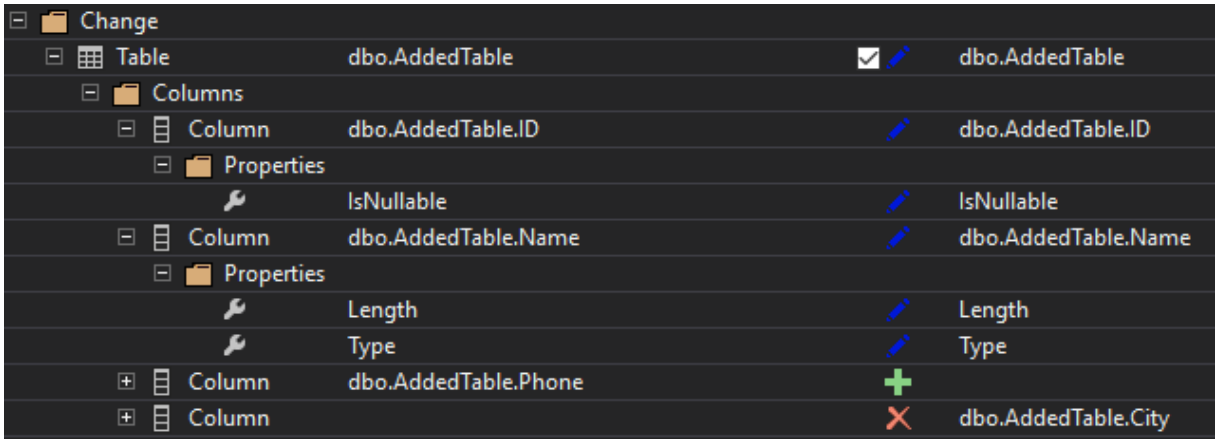
Obrázek 39: Sloupce tabulky AddedTable v databázi ReportServerTempDB

Column Name	Data Type	Allow Nulls
ID	bigint	☑
Name	nchar(10)	☑
LastName	nvarchar(50)	☑
City	nvarchar(50)	☑

Zdroj: SSMS (2020), zpracováno autorem

Opět spustíme funkci pro porovnání databázových modelů popsanou výše.

Obrázek 40: Komparace po přidání tabulky AddedTable do databáze ReportServerTempDB

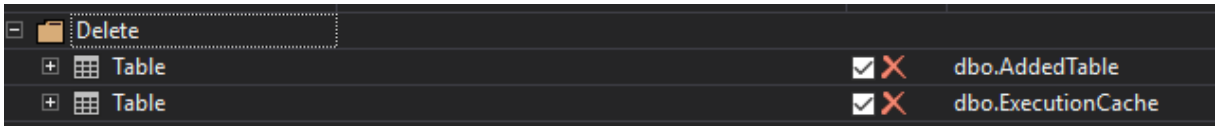


Zdroj: SSDT (2020), zpracováno autorem

Dle očekávání se tabulka *AddedTable* přesunula do kategorie *change*, neboť nyní se již nachází v obou testovaných databázích, avšak definována je rozdílně v každé z nich. Po rozbalení jednotlivých částí je možné vidět každou jednu změnu, která byla popsána výše při definování tabulky. Výsledná zpráva poukazuje na to, že je nutno přidat sloupec *Phone* a naopak odebrat sloupec *City*. Rovněž popisuje změny, které je třeba provést na atributech *ID* a *Name*, tedy změnit možnost hodnoty null v případě sloupce *ID* a upravit datový typ (a s ním spojenou délku) u sloupce *Name*.

Jako poslední změnu, která bude provedena v rámci demonstrace schopností nástroje SSDT, je smazání tabulky *AddedTable* z databáze *ReportServer*. Tuto operaci provedeme kvůli tomu, že jsme již měli možnost vidět přírůstky v kategoriích *add* a *change*, zatím však ne v poslední kategorii *delete*, což vyvolá právě tato změna.

Obrázek 41: Komparace po smazání tabulky AddedTable z databáze ReportServer



Zdroj: SSDT (2020), zpracováno autorem

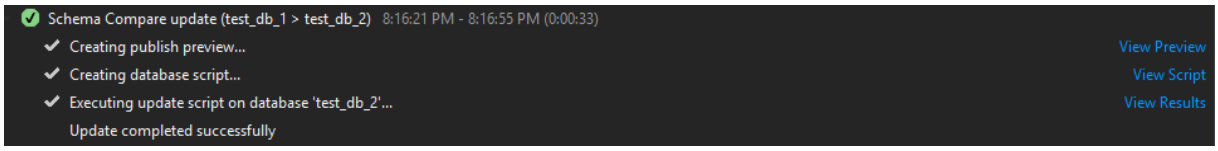
Na obrázku číslo 41 lze vidět, že výsledná zpráva dopadla dle předpokladů a tabulka *AddedTable* se opravdu nyní přesunula do poslední možné kategorie *delete*. Těmito

třemi operacemi jsme si demonstrovali, v jaké formě nástroj SSDT generuje výstupy. Opět je vše možné sledovat také v režimu *Object Definitions*.

4.4.4 Update

Výsledná zpráva se souborem operací však není, jak již bylo naznačeno, jediná důležitá funkce, kterou nástroj SSDT nabízí. Kromě tohoto je schopen také vygenerovat skript, který převede cílovou databázi do stavu zdrojové databáze a provede tak všechny změny popsané ve výstupu komparační analýzy. Na vyžádání uživatele navíc tento skript spustí a databáze tím nabydou identického stavu. Spuštění převáděcího skriptu je možné vykonat kliknutím na tlačítko *Update*, které se nachází v levé horní části, jak je možné vidět na obrázku číslo 32. Čas, po který bude skript databázi převádět, se odvíjí opět od velikosti databází, počtu objektů a především od množství změn, které je třeba provést. Po vykonání skriptu nástroj SSDT poskytne výslednou zprávu podobnou té, která je zobrazena na obrázku 42. Také je zde vidět, že nástroj poskytuje uživateli možnost zkontrolovat si náhled, skript a výsledky operace update.

Obrázek 42: Výsledná zpráva operace update



Zdroj: SSDT (2020), zpracováno autorem

Na závěr je jen pro úplnost dobré dodat, že výsledky porovnání lze také uložit ve formátu .scmp pro pozdější využití.

Závěr

Hlavním cílem práce bylo nalezení způsobu, kterým bude možné porovnat databázové modely systému MS SQL Server, a aplikovat jej na testovací databáze poskytnuté firmou Diebold Nixdorf. Tento požadavek byl splněn za pomoci nástroje SQL Server Data Tools, jenž je rozšířením pro prostředí Visual Studio. Pomocí tohoto nástroje byly porovnány testovací databáze a interpretovány výstupy procesu.

Rovněž dílčí cíle práce byly splněny, neboť v rámci čtyř částí, na které je práce rozdělena, byly objasněny principy relačních databází a byl popsán ERA model včetně doplňujících ukázek z prostředí phpMyAdmin, MySQL Workbench a pomocí fragmentů kódu SQL jazyka. Následně byla představena firma Diebold Nixdorf. Popsána byla kromě její nejpodstatnější části – produktu – také historie a znázorněna organizační struktura. Dále byla zanalyzována řešení, která již existují a jež disponují funkcemi, pomocí kterých je možné dosáhnout splnění hlavního cíle práce. Tato řešení byla poté vyhodnocena a byl z nich vybrán nástroj SSDT, díky kterému byl proveden samotný proces porovnání databázových modelů. Současně byly také představeny softwarové produkty, které byly kromě samotného SSDT použity jako podpůrné prostředky. Práce se věnovala rovněž stažení, instalaci a zprovoznění těchto prostředků.

Nástroj SSDT byl vybrán díky tomu, že mezi třemi představenými možnostmi byl posouzen jako nejvhodnější pro tuto práci, a to především pro své velice přehledně a detailně generované výstupy a skutečnosti, že tento nástroj poskytuje potřebné funkcionality zcela zdarma. Kromě toho také Microsoft – výrobce nástroje – nabízí k nástroji přehlednou a srozumitelnou dokumentaci. Uživatelské prostředí je velice příjemné a celý proces porovnání je velmi jednoduše a rychle realizovatelný. Na základě těchto skutečností se autor domnívá, že je nástroj i popsaný postup skutečně využitelný ve společnosti Diebold Nixdorf. Není totiž nutné za poskytnuté výsledky platit jakékoli poplatky či složitě zavádět nějaké komplexní řešení. Celá instalace i práce s SSDT je velice rychlá, snadná a intuitivní a vygenerované výsledky, které jsou jednoduše interpretovatelné, poskytují veškeré informace o rozdílech v testovaných databázových modelech.

Seznam použité literatury

CAREERRIDE, 2020. SQL Server - Database Objects. *Interview questions and answers, Interview experiences, Aptitude, Reasoning, Current affairs, CV, GD, Test etc.* [online]. [cit. 30.04.2020]. Dostupné z: https://www.careerride.com/SQL-Server-database-objects.aspx

DIEBOLD NIXDROF, 2019. *Company* [online]. [cit. 29.12.2019]. Dostupné z: https://www.dieboldnixdorf.com/en-us/about-us

DIEBOLD NIXDORF. *Global centre Pilsen introduction* [prezentace]. Plzeň, 2019.

DIEBOLD NIXDROF, 2019. *O nás* [online]. [cit. 30.12.2019]. Dostupné z: https://www.dnpilsen.cz/o-nas/

GANGUR, Mikuláš. *Základy implementace webového informačního systému: praktický průvodce*. V Plzni: Západočeská univerzita, 2014. ISBN 978-80-261-0485-8.

INTELLIGENT DATABASE SOLUTIONS, 2020. *SQL Compare and Synchronization Tools* [online]. [cit. 25.04.2020]. Dostupné z: https://www.sqlaccessories.com/sql-examiner/

KROENKE, David, AUER, David. *Databáze*. Brno: Computer Press, 2015. ISBN 978-80-251-4352-0

LIQUIBASE, 2020. *Version Control for Your Database*. [online]. [cit. 24.04.2020]. Dostupné z: https://www.liquibase.org/

LIQUIBASE, 2020. Docs. Diff Command. *Liquibase - Version Control for Your Database* [online]. [cit. 24.04.2020]. Dostupné z: https://www.liquibase.org/documentation/diff.html

LIQUIBASE, 2020. Docs. diffChangeLog Command. *Liquibase - Version Control for Your Database* [online]. [cit. 25.04.2020]. Dostupné z: https://www.liquibase.org/documentation/diffChangeLog.html

MICROSOFT, 2020. Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS). *Microsoft Docs*. [online]. [cit. 26.04.2020]. Dostupné z: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

MICROSOFT, 2020. Stáhnout Visual Studio 2019 pro Windows a Mac. *Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio* [online]. [cit. 25.04.2020]. Dostupné z: https://visualstudio.microsoft.com/cs/downloads/?rr=https%3A%2F%2Fdocs.microsoft.com%2Fcs-cz%2Fvisualstudio%2Finstall%2Finstall-visual-studio%3Fview%3Dvs-2019

MICROSOFT, 2020. Use Schema Compare to Compare Different Database Definitions - SQL Server Data Tools (SSDT). *Microsoft Docs*. [online]. [cit. 25.04.2020]. Dostupné z: https://docs.microsoft.com/en-us/sql/ssdt/how-to-use-schema-compare-to-compare-different-database-definitions?view=sql-server-ver15

ORACLE, 2019. *Co je relační databáze?* [online]. [cit. 27.12.2019]. Dostupné z: https://www.oracle.com/cz/database/what-is-a-relational-database/

ORACLE, 2020. *Welcome to VirtualBox.box!* [online]. [cit. 26.04.2020]. Dostupné z: https://www.virtualbox.org/

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

RYCHLÍK, Jan. *Databázové systémy 1* [přednáška]. Plzeň, 2016.

TEOREY, Toby J. *Database modeling and design: logical design*. 5th ed. Burlington, MA: Morgan Kaufmann, c2011. ISBN 0123820200.

W3SCHOOLS, 2019. *Introduction to SQL* [online]. [cit. 28.12.2019]. Dostupné z: https://www.w3schools.com/sql/sql_intro.asp

ZHANG, Preston. *Practical guide to Oracle SQL, T-SQL and MySQL*. Boca Raton: CRC Press, Taylor & Francis Group, [2018]. ISBN 9781138105188.

Seznam obrázků

Obrázek 1: Databázová tabulka ... 13
Obrázek 2: Databázová tabulka s cizím klíčem ... 14
Obrázek 3: ERA model ... 16
Obrázek 4: ERA model s vazbou M:N ... 17
Obrázek 5: DDL – příkaz CREATE TABLE ... 20
Obrázek 6: DDL – příkaz ALTER TABLE ... 21
Obrázek 7: Tabulka interpret před aplikováním příkazu SELECT ... 22
Obrázek 8: DML – příkaz SELECT ... 22
Obrázek 9: Tabulka interpret po aplikování příkazu SELECT ... 23
Obrázek 10: Tabulka pisen před aplikováním příkazu SELECT ... 24
Obrázek 11: DML – příkaz SELECT s více tabulkami ... 24
Obrázek 12: Tabulka pisen po aplikování příkazu SELECT ... 25
Obrázek 13: DML – příkazy INSERT a DELETE ... 25
Obrázek 14: DML – příkaz UPDATE ... 26
Obrázek 15: Historie společnosti Diebold Nixdorf ... 29
Obrázek 16: Organizační struktura společnosti Diebold Nixdorf ... 32
Obrázek 17: Liquibase download ... 34
Obrázek 18: Příkaz diff ... 36
Obrázek 19: Příkaz diff – výsledná zpráva ... 36
Obrázek 20: Visual Studio download ... 38
Obrázek 21: Instalace nástroje SSDT ... 39
Obrázek 22: Výstup nástroje SSDT ... 39
Obrázek 23: Verze nástroje SQL Examiner ... 41
Obrázek 24: Výstup nástroje SQL Examiner ... 41
Obrázek 25: VirtualBox download ... 45
Obrázek 26: Vytvoření virtuálního počítače ... 45
Obrázek 27: Nastavení sítě v prostředí VirtualBox ... 46
Obrázek 28: Nastavení sítě v prostředí Windows ... 47
Obrázek 29: Prostředí nástroje SSMS ... 48
Obrázek 30: Rozdíly v databázích ... 49
Obrázek 31: Spuštění porovnávacího nástroje ... 50
Obrázek 32: Výběr testovaných databází ... 50
Obrázek 33: Výstup srovnání databází ... 51
Obrázek 34: Detail operace change ... 52
Obrázek 35: SSDT – Object Definitions ... 53
Obrázek 36: Tabulky v kategorii Add ... 53
Obrázek 37: Sloupce tabulky AddedTable v databázi ReportServer ... 54
Obrázek 38: Komparace po přidání tabulky do databáze ReportServer ... 54
Obrázek 39: Sloupce tabulky AddedTable v databázi ReportServerTempDB ... 54
Obrázek 40: Komparace po přidání tabulky AddedTable do databáze ReportServerTempDB ... 55
Obrázek 41: Komparace po smazání tabulky AddedTable z databáze ReportServer ... 55
Obrázek 42: Výsledná zpráva operace update ... 56

Seznam použitých zkratek a značek

1NF – první normální forma

2NF – druhá normální forma

ATM – Automated Teller Machine

DCL – Data Control Language

DDL – Data Definition Language

DML – Data Manipulation Language

DQL – Data Query Language

ERA – Entity Relationship Attribute

JSON – JavaScript Object Notation

SQL – Structured Query Language

SŘBD – systém řízení báze dat

SSDT – SQL Studio Data Tools

SSMS – SQL Server Management Studio

XML – Extensible Markup Language

YAML – YAML Ain’t Markup Language

Abstrakt

Fürbacher, T. (2020) *Porovnání databázových modelů* (Bakalářská práce), Západočeská univerzita v Plzni, Fakulta ekonomická, Česko

Klíčová slova: relační databáze, databázový model, databázový objekt, MS SQL Server, SQL Studio Data Tools, SQL

Tato práce se zabývá porovnáním dvou modelů relační databáze systému MS SQL Server z pohledu odlišností mezi jejich objekty. Společnosti si svoje databáze hlídají a často chtějí vědět, zda s nimi někdo nemanipuloval. Hlavním cílem je poskytnout popis postupu, na jehož výstupu bude přehled o všech objektech, ve kterých se testované databáze liší a jakým způsobem jsou rozdílné. Součástí tohoto cíle je také aplikace popsaného postupu na testovacích databázích. V práci jsou představeny hlavní principy relačních databází včetně popisu jejich modelů a objektů. Posléze je provedeno porovnání několika nástrojů schopných naplnění hlavního cíle práce a na základě jejich výhod a nevýhod jsou vyhodnoceny a je vybrána nejvhodnější varianta. Následně je popsán postup pro porovnání, který je ve zvoleném nástroji aplikován na testovacích databázích. Výsledky porovnání jsou poté dále interpretovány.

Abstract

Fürbacher, T. (2020) *Comparison of Database Models* (Bachelor Thesis), University of West Bohemia, Faculty of Economics, Czech Republic

Key words: relational database, database model, database object, MS SQL Server, SQL Studio Data Tools, SQL

This thesis deals with a comparison of two models of relational databases of system MS SQL Server from differences between its objects point of view. Companies guard their databases and they often want to know if anybody manipulated with them. The main objective is to provide a description of a method whose results will be an overview of all objects, which are different and in which way they differ. The part of the thesis object is also an application of the described method on testing databases. In the thesis, there are introduced main specifics of relational databases including a description of its models and objects. Then a comparison of a few tools, which are able to fulfill the main object, is performed. They are evaluated based on their advantages and disadvantages and the most suitable one is chosen. Then the method for a comparison is described. This method is applied through the chosen tool for testing databases. Next the results are interpreted.