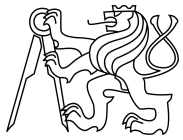
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Nástroj pro verzování APEX aplikací
Student:	Bc. Daniel Zámečník
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

APEX (Application Express) je prostředí pro rychlý vývoj aplikací nad databázovým strojem Oracle. Má řadu větších i menších nasazení. Celá aplikace je umístěna v databázovém stroji a přes www rozhraní se spouští pomocí volání PL/SQL balíků. Prostředí APEX však nemá vyřešenu podporu vývoje typu: vývojové prostředí - produkční prostředí. Tato práce je pokračováním již obhájené bakalářské práce stejného autora. Cílem této práce je zrevidovat hotový prototyp, navázat na něj a pokračovat ve vývoji.

Postupujte dle těchto kroků:
1. Proveďte revizi stávajícího prototypu, který analyzuje a zobrazuje rozdíly mezi dvěma APEX aplikacemi.
2. Navrhněte celou koncepci verzovacího nástroje pro APEX aplikace. Věnujte se též návrhu uživatelského rozhraní (inspirujte se existujícími nástroji pro verzování SW).
3. Návrh implementujte jako APEX aplikaci, nasaďte, zdokumentujte a otestujte.
4. Zhodnoťte přínosy a naznačte další možný rozvoj nástroje.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 14. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práca

Nástroj pro verzování APEX aplikací

Bc. Daniel Zámečník

Vedúci práce: Ing. Michal Valenta, Ph.D.

9. mája 2017

Poďakovanie

Ďakujem Ing. Michalovi Valentovi za podporu, pomoc a prejavený záujem o moju prácu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov. Ďalej prehlasujem, že som s Českým vysokým učením technickým uzavrel licenčnú zmluvu o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona. Táto skutočnosť nemá vplyv na ust. § 47b zákona č. 111/1998 Sb. o vysokých školách.

V Praze 9. mája 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na túto prácu

Zámečník, Daniel. *Nástroj pro verzování APEX aplikací.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Táto diplomová práca sa zaoberá problémom malej podpory Oracle-u pri nasadzovaní APEX aplikácií medzi rôznymi prostrediami a verzovaním aplikácií. Taktiež popisuje aj prostredie APEX, jeho rozdiely medzi verziami 4.2 a 5.1 a spôsob rýchleho vývoja aplikácií pomocou tohto nástroja.

Klíčová slova Oracle, APEX, PL/SQL, databáza, aplikácia

Abstract

This diploma thesis deals with the problem of insufficient support in deploying Oracle APEX applications among different environments and versioning of applications. It also describes the APEX environment, its difference between versions 4.2 and 5.1 and method of rapid development of applications by using this environment.

Keywords Oracle, APEX, PL/SQL, database, application

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Hlavnou motiváciou vzniku tejto práce boli nepostačujúce možnosti nasadzovania aplikácií vo vývojovom prostredí Application Express, čo v niektorých prípadoch značne zvyšuje časovú náročnosť nasadzovania. Preto sa zrodila myšlienka vytvoriť aplikáciu, ktorá by umožňovala vývojárom vidieť rozdiely medzi dvoma aplikáciami, a tým šetrila ich čas. Riešením tejto práce je prototyp aplikácie, ktorý umožňuje robiť snímky (snapshoty) aplikácií a následne dokáže dva z nich porovnať a ukázať rozdiel.

Na úvod v kapitole 2 zhrniem informácie zo softwarového inžinierstva pre lepšie pochopenie kontextu tejto práce a môjho oboru. Technológiu APEX opisujem v kapitole 3. Vysvetľujem jej výhody, možnosti použitia a dôležité pojmy, ktorým je vhodné rozumieť ešte pred zavedením APEXu. Popíšem v nej aj noviky a zmeny, ktorými si toto vývojové prostredie prešlo od verzie 4.2 do 5.1. Na záver kapitoly ešte popíšem jednotlivé časti tohto prostredia. Odporúčané nasadzovanie aplikácií, jeho nedostatky a iné možnosti riešenia vysvetľujem v kapitole 4.

V následujúcich kapitolách popíšem svoju prácu a návrhy ako by sa dala aplikácia zlepšiť a akými funkcionalitami by mohla disponovať. Práca je rozdelená do kapitol podľa toho, čoho sa týka. Aplikáčnej časti a grafickému rozhraniu sa venujem v kapitole 6. Vytvorený dátový model popisujem v kapitole 7. Aplikačná logika je vytvorená v programových balíkoch a tie popíšem v kapitole 8. Prevedeným testom a ich výsledkom venujem kapitolu 9.

Kapitola 2

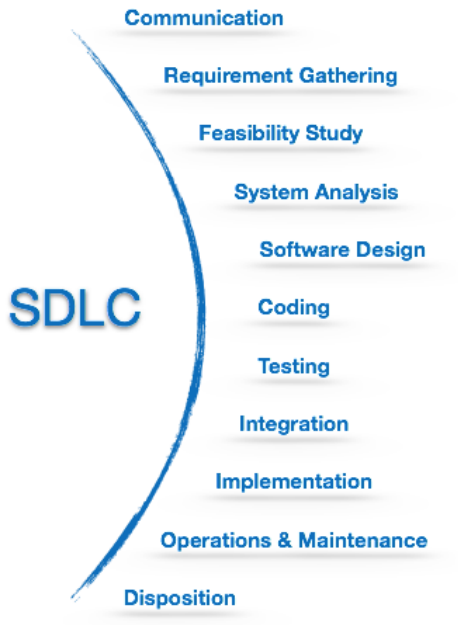
Metodika, teória softwarového inžinierstva, kontext práce

V tejto kapitole by som chcel vysvetliť ako súvisí moja práca s oborom, ktorý študujem. Keďže som na celom projekte pracoval od začiatku sám, tak som si prešiel mnohými časťami životného cyklu vývoja softwaru (Software Development Life Cycle, skrátene SDLC). Z akých častí sa SDLC skladá popíšem v 2.1. Ďalej by som v 2.2 predstavil pár modelov životného cyklu a následne v 2.3 vysvetlil pre aký a prečo som sa rozhodol. Ďalšou súvislosťou je, že moja práca sa venuje podpore pri nasaďovaní aplikácií medzi jednotlivými prostrediami, ktoré softwarové inžinierstvo definuje. Ich potrebu a druhy popíšem v 2.4. Pri písaní tejto kapitoly som najviac vychádzal zo zdrojov [13][12].

2.1 Software Development Life Cycle (SDLC)

Životný cyklus vývoja softwaru (SDLC) predstavuje komplex všetkých činností v softwarovom inžinierstve potrebných pri vývoji softwarového produktu. Zahŕňa teda všetko počnúc počiatočnou komerčnou myšlienkou až po finálnu deinštaláciu produktu po ukončení používania. Jedná sa o kroky, ktorými by sme mali prechádzať, aby sme navrhli a vyvinuli softwarový produkt efektívne. Názvy a počty jednotlivých etáp sa líšia v závislosti od zdroja informácii, ale v podstate pokrývajú rovnaké činnosti.

Ako vidieť na obrázku 2.1, tak zdroj, z ktorého som vychádzal, rozdeľuje životný cyklus na časti: komunikácia, zhromažďovanie požiadaviek, štúdia uskutočniteľnosti, systémová analýza, návrh softwaru, programovanie, testovanie, integrácia, implementácia, prevádzka a údržba, dispozícia.



Obr. 2.1: SDLC Zdroj: [13]

2.1.1 Komunikácia

Komunikácia je prvým krokom, v ktorom odberateľ iniciuje dopyt na softwarový produkt. Kontaktuje poskytovateľa a vyjednávajú sa podmienky vývoja produktu.

2.1.2 Zhromažďovanie požiadaviek

V tomto kroku vstupuje vývojársky tím, ktorý sa snaží na základe diskusií s rôznymi zainteresovanými osobami získať čo najviac informácií potrebných na splnenie vznesených požiadaviek na softwarový produkt. Tieto požiadavky sa po zvážení rozdelia do skupín ako používateľské požiadavky, systémové požiadavky a funkčné požiadavky. Zároveň sa študujú existujúce, či zastaralé súvisiace softwary.

2.1.3 Štúdia uskutočniteľnosti

Po zozbieraní požiadaviek tím ponúkne hrubý plán softwarového procesu. V tomto kroku sa zanalyzuje, či vyvíjaný software môže splniť všetky požiadavky a zistí sa, či projekt je finančne, prakticky a technologicky uskutočniteľný. Pri posudzovaní uskutočniteľnosti môžu byť nápomocné rôzne dostupné algoritmy.

2.1.4 Systémová analýza

Systémová analýza zahŕňa pochopenie limitov softwarového produktu, súvisiacich problémov so zmenou v existujúcom systéme, identifikovanie dopadu projektu na organizáciu a tak ďalej. Projektový tím analyzuje rozsah projektu a plánuje časový harmonogram s prislúchajúcimi zdrojmi.

2.1.5 Návrh softwaru

V tomto kroku sa zužitkujú všetky vstupy od užívateľov a informácie získané pri zhromažďovaní požiadaviek a systémovej analýze na návrh softwarového produktu.

2.1.6 Programovanie

Programovacia fáza predstavuje implementáciu navrhnutého softwaru formou písania programovacieho kódu vo vhodnom programovacom jazyku.

2.1.7 Testovanie

Odhaduje sa, že cca 50% celého vývojového procesu by sa malo testovať. Testovanie sa uskutočňuje počas programovacej fázy na rôznych úrovniach, keďže včasné odhalenie chýb a ich oprava je kľúčom k spoľahlivému softwaru.

2.1.8 Integrácia

Tento krok zahŕňa integrovanie softwaru s vonkajšími entitami, ako napríklad knižnicami, databázami a inými programami.

2.1.9 Implementácia

Implemetovanie znamená inštaláciu softwaru na zariadenia užívateľa. Niekedy si software vyžaduje poinštalačné nastavovanie u koncového užívateľa. Riešia sa prípadné problémy s integráciou v produkčnom prostredí.

2.1.10 Prevádzka a údržba

V tomto kroku sa má zabezpečiť efektívna prevádzka bez chybovosti. Ak je to požadované, vyškolia sa užívatelia a vytvorí sa dokumentácia pre užívateľa, respektíve užívateľská príručka. Software sa udržiava pomocou príležitostných aktualizácií kódu v závislosti od zmien v technológií alebo prostredí používateľa. V tejto fáze sa riešia aj chyby, ktoré neboli zistené testovaním, ale prišlo sa na nich pri reálnom používaní.

2.1.11 Dispozícia

Postupom času software môže výkonostne upadať a môže sa stať zastaralým, či vyžadujúcim rozsiahly aktualizácie a úpravy. Fáza zahŕňa archiváciu dát a potrebných softwarových komponentov, uzatvorenie systému a plánovanie dispozičných aktivít.

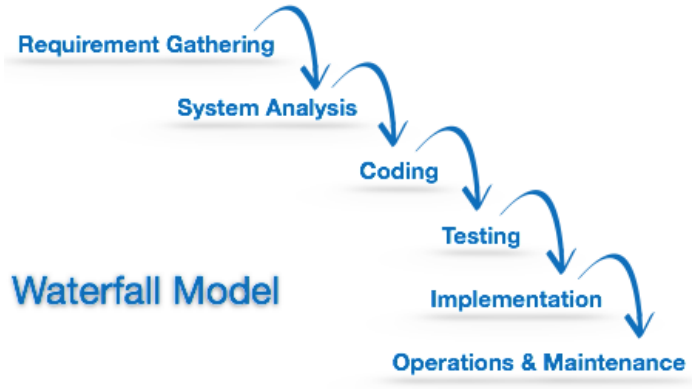
2.2 Modely SDLC

Model SDLC popisuje sekvenciu jednotlivých fáz celého životného cyklu softwarového produktu. Je to jasne definovaná a štruktúrovaná sekvencia jednotlivých etáp v softwarovom inžinierstve potrebných pri vývoji softwarového produktu. Existuje niekoľko modelov životného cyklu vývoja softwaru, pričom každý môžeme charakterizovať určitou postupnosťou krokov, ktorá je typická pre príslušný model s cieľom zaistiť úspešný a efektívny vývojový proces. Každý vyvíjaný software je iný a vyžaduje si špecifický prístup a iný vhodný SDLC model v závislosti od interných aj externých faktorov.

Medzi najbežnejšie a najdôležitejšie modely patria:

- Vodopádový model (Waterfall Model)
- Iteratívny model (Iterative Model)
- Agilný model (Agile Model)

2.2.1 Vodopádový model (Waterfall Model)



Obr. 2.2: Vodopádový model Zdroj: [13]

Vodopádový model je považovaný za najstarší model softwarového procesu, ktorý bol spočiatku široko využívaný. Softwarové programy totiž neboli

také rozsiahle a tým pádom aj požiadaviek bolo menej a bolo tak jednoduchšie ich presne zadefinovať. Táto metóda je veľmi vhodná v prípade, že všetky požiadavky na produkt sú už na začiatku presne definované. Jedná sa o lineárne sekvenčný model životného cyklu, pri ktorom musí byť každá fáza procesu ukončená skôr než sa začne nasledujúca. Vo vodopádovom modeli sa teda žiadne fázy neprelínajú. Ako vidno na obrázku 2.2, tak model sa skladá zo sekvenčných fáz: zhromažďovanie požiadaviek, analýza systému, programovanie, testovanie, nasadenie, údržba.

Použitie tohto modelu je vhodné ak:

- Požiadavky sú jasné, pevne stanovené a dobre zdokumentované
- Definícia produktu je stabilná
- Technológia nie je dynamická
- Nie sú žiadne rozporuplné požiadavky
- Projekt je krátky

Výhody modelu:

- Jasne definované fázy a plán
- Zrozumiteľné míľniky, predvídateľnosť (čas, rozsah, cena)
- Jednoduchá koordinácia práce, definovanie úloh
- Proces a výsledky sú dobre zdokumentované
- Zákazník sa podieľa na projekte v presne stanovených okamžikoch

Nevýhody modelu:

- Nepružnosť modelu, nákladná reakcia na zmeny
- Nutnosť špecifikácie všetkých požiadaviek na začiatku. Nedá sa vyhovieť zmeneným požiadavkám
- Nevhodný model pre dlho prebiehajúce projekty
- Nevhodný pre komplexné a objektovo-orientované projekty

2.2.2 Iteratívny model (Iterative Model)

Iteratívny model sa nepokúša na začiatku o plnú špecifikáciu všetkých požiadaviek. Iteratívny proces sa začne s jednoduchou implementáciou malého súboru požiadaviek a tie sa následne iteratívne rozširujú v ďalších verziách softwaru až kým sa neimplementuje kompletný systém pripravený na nasadenie. Na konci každej iterácie je vytvorená nová verzia, ktorá sa vyhodnotí, identifikujú sa ďalšie požiadavky a modifikuje sa návrh. Kratšie pracovné intervaly a častejšie predkladanie prototypu zákazníkovi vedie k lepšej kontrole a plánovaniu. Iteratívny model tiež viac odhaľuje rozličné predstavy zákazníka a dodávateľa a tým umožňuje tieto odlišné predstavy zjednocovať už počas vývoja upresňovaním špecifikovaných požiadaviek. Jednotlivé iterácie sa vytvárajú vodopádom (Waterfall).

Použitie tohto modelu je vhodné ak:

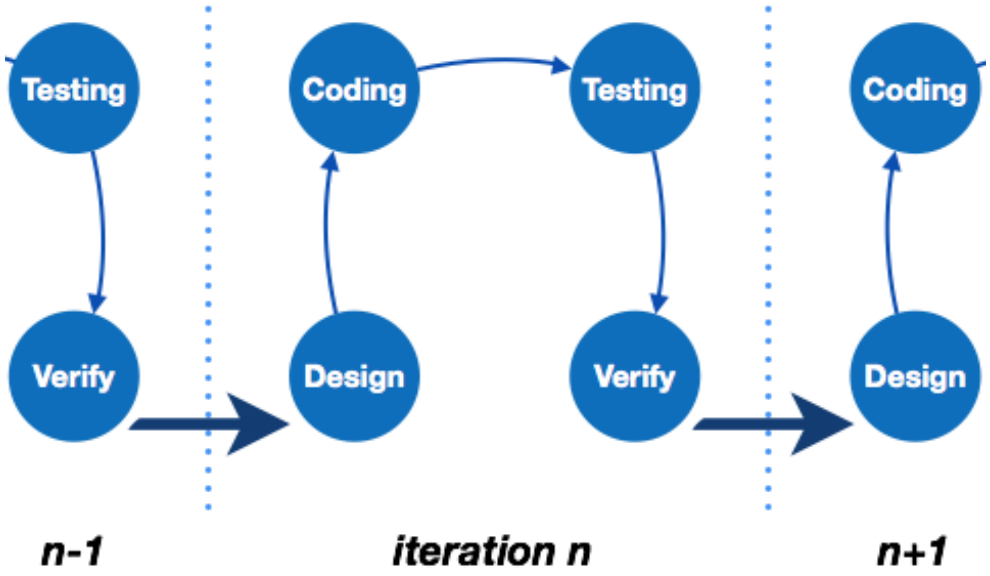
- Projekt je rozsiahly
- Významné požiadavky sú definované, avšak niektoré funkcionality alebo rozšírenie požiadaviek sa môže vyvinúť časom
- Vývojový tím začne využívať novú technológiu počas práce na projekte
- Existujú nejaké rizikové prvky a ciele, ktoré sa môžu v budúcnosti meniť

Výhody modelu:

- S každou iteráciou je doručený funkčný produkt
- Možnosť zakomponovania zmien do ďalšej iterácie
- Zákazník je zapojený do vývoja, má prístup k jednotlivým prototypom a má možnosť upresňovať svoje požiadavky
- Menej nákladná zmena rozsahu a požiadaviek
- Jednoduchšie testovanie a odstraňovanie chýb v malých iteráciach

Nevýhody modelu:

- Komplikovanejšie manažovanie
- Neistý koniec projektu, čo je rizikové
- Nevhodný pre menšie projekty
- Nutnosť udržať konzistentnú dokumentáciu naprieč verziami



Obr. 2.3: Iteratívny model Zdroj: [13]

2.2.3 Agilný model (Agile Model)

Základná myšlienka agilného modelu je, že každý projekt potrebuje iné riešenie a existujúce metódy sa musia prispôsobiť projektu tak, aby sa čo najlepšie hodili k jeho požiadavkám. Pozornosť je upriamená na adaptabilitu procesu a uspokojenie zákazníka rýchlym dodaním funkčného softwarového produktu. Agilná metóda rozloží produkt na malé inkrementačné buildy, ktoré sa poskytujú v iteráciach. Každá iterácia trvá 1-3 týždne. V každej iterácii tímy pracujú simultánne v rôznych oblastiach ako plánovanie, analýza požiadaviek, návrh, programovanie a testovanie. Na konci iterácie je zákazníkovi predstavený funkčný softwarový build s postupne rozšírenými prvkami, až kým finálny build nespĺňa všetky požiadavky zákazníka. Chrbtovou kosťou agilnej metodológie je interakcia so zákazníkom a otvorená komunikácia s minimálnou dokumentáciou. Pracovné tímy musia úzko spolupracovať.

Výhody modelu:

- Vhodný pre fixné aj meniace sa požiadavky
- Dobrý model pre meniace sa prostredia
- Ľahké manažovanie, málo plánovania
- Flexibilný pre vývojárov
- Podpora tímovej práce

Nevýhody modelu:

- Vysoká závislosť na interakcii so zákazníkom, ak je zákazník nezrozumiteľný, môže zviesť pracovný tím nesprávnym smerom
- Vysoká individuálna závislosť kvôli minimálnej dokumentácii
- Vyššia rizikovosť udržateľnosti
- Nevhodný pre zvládnutie zložitých závislostí

2.3 Použitý model

Na realizáciu tejto práce sa najviac hodí iteratívny model primárne z troch dôvodov:

1. Výsledok práce bude prototyp aplikácie (prototyp je výsledkom každej iterácie)
2. Urobiť úplnu analýzu pred začiatkom práce bolo skoro nereálne pre nedostatok dokumentácie o vnútornom fungovaní prostredia APEX
3. Jednotlivé časti riešenia problému pomáhajú k rýchlejšiemu a lepšiemu pochopeniu prostredia APEX, a tým sa ušetrí čas potrebný na analýzu

Okrem faktu, že výstupy jednotlivých iterácií mi pomáhali pri analýze v nasledujúcej iterácii primárne kvôli odhaľovaniu komplikovanosti vnútorného fungovania prostredia Application Express, dáva iteratívny model aj možnosť upravovania plánovaného času na isté fázy iterácie podľa identifikovaných problematických častí. Napríklad iterácia, ktorá vytvára nejakú zložitejšiu logiku, potrebuje omnoho viac času na analýzu a testovanie než implementáciu, a naopak iterácia, zaoberajúca sa napríklad sťahovaním potrebných dát z iných serverov, nepotrebuje veľa času na analýzu, ale môže byť výrazne náročnejšia na implementáciu, prípadne testovanie, ale to už záleží od konkrétneho prípadu.

Navyše tento projekt spĺňa všetky štyri body vhodného použitia iteratívneho modelu, ktoré spomínam v 2.2.2.

2.4 Prostredia (Environments)

Skúsenosťou mnohých softwarových projektov je, že aj keď sa spočiatku zdá, že postačujúce bude 1-2 prostredia, časom sa zistí, že ich je potrebných viac. Neexistuje jednoznačná odpoveď na to, aký počet prostredí je najvýhodnejší, keďže pre každý projekt môže byť vhodné niečo iné. Je úlohou architekta a projektového manažéra stanoviť, ktoré prostredia sú potrebné a zaistiť ich v takom čase a na takom mieste, aby uľahčili prácu vývojárov, testerov a klienta.

Hlavným účelom týchto prostredí je zlepšiť vývojové, testovacie a spúšťacie procesy.
Typickými prostrediami sú:

- **Vývojové prostredie (Development Environment).** Je to prostredie, v ktorom sa software vyvíja. V niektorých prípadoch to môže byť osobný počítač vývojára, inokedy server zdieľaný niekoľkými vývojármi pracujúcimi na rovnakom projekte. Toto prostredie by sa malo čo najviac podobať produkčnému prostrediu, aby sme zabránili odlišnému správaniu sa softwaru v produkcii.

- **Testovacie prostredie (Testing Environment).** Keď sa aplikácia vyvinie do určitého dohodnutého štádia, prepustí sa do testovacieho prostredia. Tu sa testeri snažia zaistiť kvalitu odhaľovaním programových chýb a overovaním napravených chýb. Toto prostredie sa musí veľmi presne podobať produkčnému, pretože je to posledné bezpečné miesto na nájdenie a nápravu chýb súvisiacich s prostredím.

- **Akceptačné testovacie prostredie užívateľa (User Acceptance Test Environment).** Po skončení interného testovania testermi sa software presunie na testovanie ku klientovi. Tu klient verifikuje kvalitu aplikácie a problémové otázky zasiela vendorovi na nápravu. Klient zároveň hodnotí aplikáciu a môže žiadať zmeny, ktoré lepšie zodpovedajú jeho požiadavkám.

- **Pred-produkčné prostredie (Staging / pre production Environment).** Toto prostredie sa zvykne využívať na zostavenie, testovanie a revíziu nových verzií predtým než sa prepustia do produkcie. Software sa obvykle testuje na hardwari odzrkadľujúcom hardware použitý v produkčnom prostredí.

- **Produkčné prostredie (Production Environment).** Je to prostredie, v ktorom sa aplikácia uverejní a uvedie do života.

Kapitola 3

Technológia APEX

3.1 O technológii

Oracle Application Express (Oracle APEX) je framework na vývoj webových aplikácií v Oracle databáze. S počiatkom vo februáry roku 2004 ako vybavenie Oracle Database 10g. S obrovským nárastom záujmu a počtom sťahovaní sa to stalo štandardnou súčasťou Oracle Database 11g a vyššie. Od vtedy Oracle pokračuje v investovaní do vývoja a podpore tohto nástroja. Nová verzia APEX-u vychádza približne raz ročne. Keďže je APEX databázovo orientovaný framework je navrhnutý pre vývojárov znalých SQL a PL/SQL. Keďže je to súčasť databáze ide o bezplatnú funkcionalitu databáze, ktorá je plne podporovaná a funguje všade, kde funguje Oracle databáza.

Prostredníctvom webového prehliadača môžete vyvíjať a nasadzovať širokú škálu aplikácií od jednoduchých po profesionálne. Vyvíjať sa dajú ako pre počítačové zariadenia, tak aj pre mobilné. V priebehu niekoľkých minút je možné premeniť nejaký spreadsheet (napríklad Excelovský dokument) na viacužívateľskú webovú aplikáciu. Na druhom konci spektra môžete vytvoriť obrovskú komplexnú mnohojazyčnú aplikáciu, ako napríklad Oracle Store.

Najhlavnejšími výhodami tohto nástroja je jednoduchosť a rýchlosť. Veľmi pokročilé IDE ponúka mnoho možností ako pre začiatočníkov, tak aj pre pokročilých. Začiatočníci určite ocenia sprievodcov (wizard), pomocou ktorých je možné na pár klikov nastaviť rôzne aj komplikované časti aplikácie bez písania nejakého kódu. Pokročilí tam zas nájdu mnoho spôsobov ako si aplikáciu vedia prispôsobiť dopisovaním kódu, aby dosiahli požadovaného výsledku aj pre náročné požiadavky. Mnoho času vývojárov sa ušetrí aj pri riešení bezpečnosti. Nastavovanie prihlasovania a dostupnosti častí aplikácií pre rôznych užívateľov už asi nemôže byť jednoduchšie. APEX ponúka bezpečnostné funkcie, ako napríklad session-state protection (ochrana stavu sessiony), autentifikačné a autorizačné schémy.

Jednoduchý je aj prechod na vývoj v APEX-e. Keď už máte Oracle databázu, stačí nakonfigurovať APEX a už netreba inštalovať žiaden ďalší klientský

software ani pre vývojárov, ani pre užívateľov. Keď sa všetko nastaví, tak sú definované URL aj pre vývojové prostredie aj pre koncových užívateľov. Stačí spravovať databázový server, keďže vývoj aj runtime beží v databáze.

Rýchlosť vývoja je dosiahnutá aj jednoduchou architektúrou APEX-u. Keďže sú stránky dynamicky generované pomocou metadát uložených v Oracle databáze, tak nedochádza ku žiadnemu generovaniu kódu a tým pádom ani ku kompilovaniu súborov. Aj IDE aj vyvíjaná aplikácia sú dostupné prostredníctvom URL, čiže ak vývojár spraví zmenu v aplikácii a chce vidieť dopad v aplikácii, tak sa len jednoducho preklikne do okna s aplikáciou a obnoví stránku a zmenu je okamžite vidno, keďže zmena nastala pri uložení metadát do databáze. Navyše aj neskúsený vývojár ak si nie je istý zmenami, ktoré ide spraviť, tak si môže vyexportovať stránku, na ktorej ide pracovať a v prípade, že sa mu to nepodarilo, stačí, že importuje pôvodnú stránku a všetko je zas v pôvodnom stave. Dokonca aj keby si zabudol vyexportovať stránku a podarí sa mu pokaziť nejakú funkcionalitu, tak Oracle mu dáva možnosť vyexportovať stránku s odstupom nejakého nastaveného času. Čiže ak hodinu a pol skúšal a nepodarilo sa mu to a nepamätá si ako to bolo predtým, keď to fungovalo, tak si vyexportuje stránku so stavom dve hodiny dozadu a naimportuje ju naspäť a všetky zmeny sa stratia.

APEX sa dá stiahnuť z [6] a podrobné informácie ohľadom inštalácie, aktualizácie, nasadzovania aplikácií, užívateľskom rozhraní, zabezpečení a výkone sa dajú nájsť na [7] a sú zhrnuté a preložené v 3.2. APEX 5.1 je bezplatný plne podporovaný cez Oracle Support Services a je možné ho nainštalovať do všetkých edícií (SE, SE1, SE2, EE) databáz od Oracle verzie 11.2.0.4 a vyššie. Dá sa taktiež používať na Oracle databázach verzie 11g Express Edition (XE), ale je podporovaný len prostredníctvom diskusného fóra OTN. Dôrazne sa odporúča aktualizovať APEX na najnovšiu verziu dostupnú na OTN. APEX nie je možné inštalovať do inej relačnej databázy.

Oracle podporuje inštalácie na nasledujúce platformy:

- Linux (Linux x86, Linux x86-64)
- Microsoft Windows (Microsoft Windows (32-bit), Microsoft Windows x64 (64-bit))
- Oracle Solaris (Oracle Solaris x86-64 (64 bit), Oracle Solaris on SPARC (64 bit))
- HP-UX (HP-UX Itanium, HP-UX PA-RISC (64-bit))
- IBM (IBM AIX on POWER Systems (64-bit), IBM: Linux on System z)

3.2 Inštalácia

Oracle ponúka APEX aj bez inštalácie a to rovno dvoma spôsobmi. Prvý je určený len na demonštračné účely a nie je možné ho použiť na produkčné využitie. Na stránke `apex.oracle.com` si každý vie požiadať o vytvorenie workspace-u, len zadaním mena a e-mailovej adresy. Táto služba dostupná od júna 2002 zažíva obrovský úspech v súčastnosti s viac ako 20 000 aktívnymi workspacemi.

Druhou možnosťou je využitie služby Oracle Database Cloud Service, čo je podnikovo overená profesionálna cloudová služba, ktorá znesie aj veľkú záťaž rozsiahlych produkčných nasadení. Cloudové služby sú v dnešnej dobe využívané vďaka im výhodám ohľadom jednoduchosti a rýchlosti nastavovania výkonu a šetrenia času a investícií pre firmy, ktoré ešte nedisponujú potrebnými zariadeniami a kvalifikovanými zamestnancami. Služba je samozrejme bezpečná a šifruje dáta aj pri prenose aj pri uložení. Taktiež monitoruje a zaznamenáva prístupy do databázy pre audit a kontrolu. Oracle podporuje aj hostovanie iných firiem zverejnením informácií na ich stránke Oracle Application Express Community.

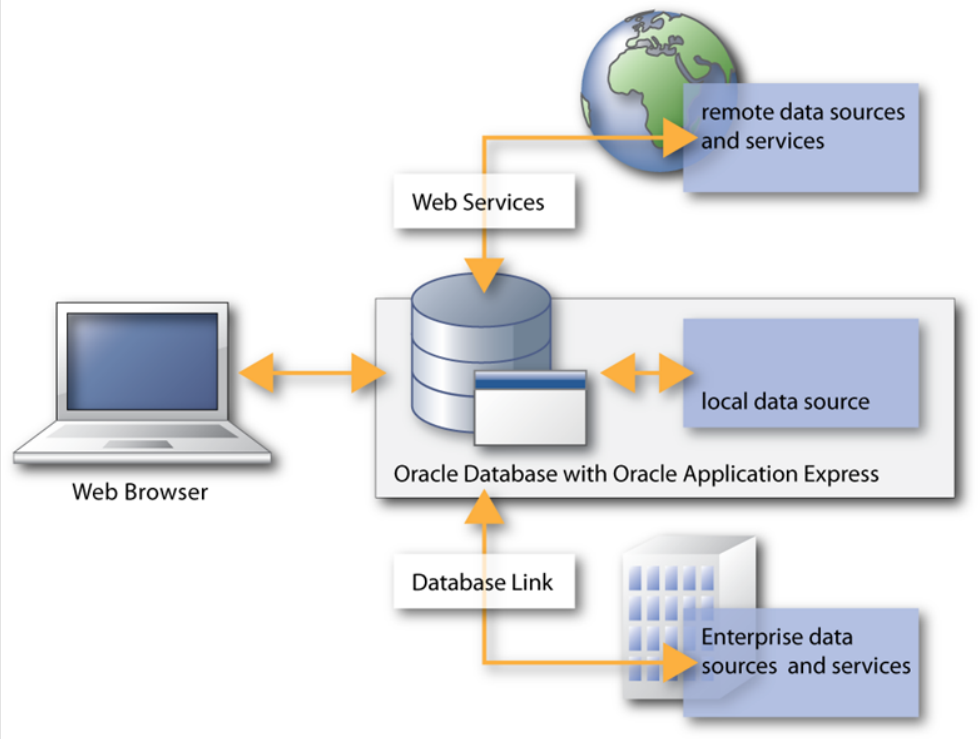
Keďže inštalácia a konfigurácia APEX-u nie je úplne triviálna, tak Oracle ponúka aj možnosť stiahnutia si Oracle VM Virtual Box a následného importovania virtuálneho stroja (dostupné na stiahnutie tu [11]), ktorý obsahuje nainštalované a nastavené Oracle Linux 7, Oracle Database 12c Release 2 EE (12.2.0.1 Linux x86-64), Oracle SQL Developer 4.1.5.21, Oracle Application Express 5.1, Oracle REST Data Services 3.0.9. Tento nástroj je však určený len na účely testovania, takže je nepodporovaný a nemal by byť použitý ako produkčné prostredie.

V prípade, že si chcete nainštalovať a nakonfigurovať APEX existuje niekoľko kľúčových oblastí štruktúry a riadenia, ktorým je dobre porozumieť:

- Database
- Web Listener
- Multi-Tenancy
- Roles and Responsibilities
- Environments
- Database Jobs
- Database Backup

3.2.1 Database

Inštalácia Application Express vytvorí samostatnú schému (APEX_050100 pre APEX 5.1), kde vytvorí všetky tabuľky metadát, pohľady a programové balíčky, ktoré tvoria Application Express engine. Oracle odporúča vytvoriť tablespace (tabuľkový priestor) zvlášť pre engine Application Express a súbory Application Express pre lepšie riadenie a monitorovanie. Napríklad tablespace APEX_TS_050100 by obsahoval engine APEX-u a APEX_TS_FILES súbory APEX-u, ale mená môžu byť ľubovoľné. Overený postup je nainštalovať Oracle Application Express do databázy, ktorá má databázové objekty, ktoré chcete použiť vo vašich aplikáciách. Môžete tiež využiť databázové odkazy (database links) na prepojenie s inými databázami a spojiť svoje aplikácie s webovými službami.

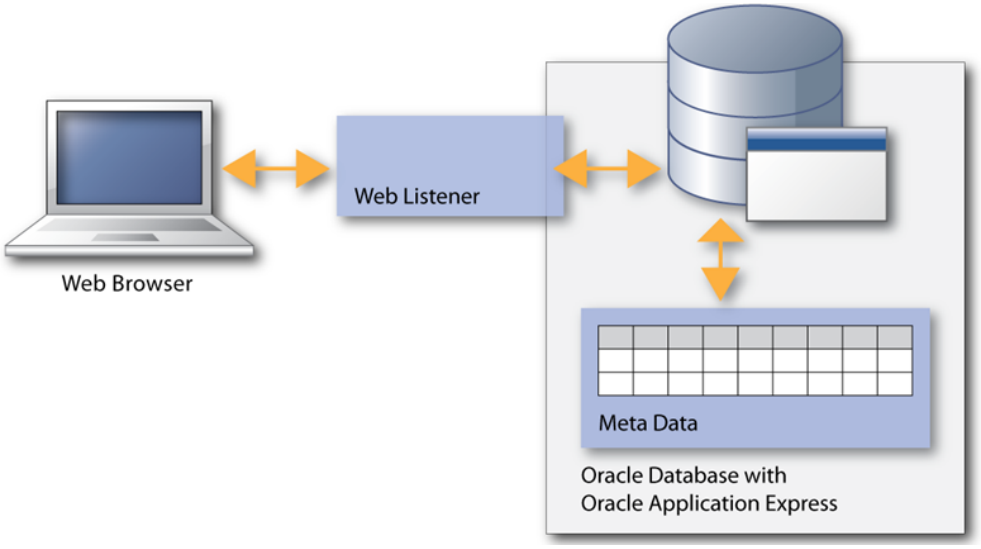


Obr. 3.1: Architektúra APEX-u. Zdroj: [4]

3.2.2 Web Listener

Web Listener funguje ako komunikačný prostriedok medzi webovým prehliadačom a objektami Application Express, ktoré sa nachádzajú v databáze. Ako som spomínal v úvode tejto kapitoly 3.1, tak Application Express využíva jednoduchú architektúru, kde sú stránky dynamicky generované pomocou meta-

dát uložených v databáze. Keďže tu neexistuje žiadne generovanie kódu ani kompilovanie súborov a všetko je k dispozícii na definovaných URL, tak Web Listener mapuje požiadavky prehliadača (browser requests) na volanie databázových procedúr.



Obr. 3.2: Web Listener. Zdroj:[8]

Máte na výber z troch možností Web Listenera:

- **Oracle REST Data Services** (pôvodne APEX Listener) - často označovaný skratkou ORDS je naprogramovaný v jazyku Java a môže byť nainštalovaný na ľubovoľný J2EE kompatibilný webový server a je to preferovaná voľba pre použitie s Oracle Application Express. Jedná sa o bezplatný nástroj, ktorý je plne podporovaný s Oracle WebLogic Server, Oracle Glassfish Server a Apache Tomcat. ORDS je časť referenčnej architektúry, ktorá sa používa na prevádzku Oracle Database Cloud Service.
- **Oracle HTTP Server** (Apache) s mod_plsql pluginom môže byť umiestnený na rovnakom fyzickom počítači ako databáza alebo samostatne. Ak je to na rovnakom stroji, tak je to súčasť licencie s obmedzeným použitím. Ak na samostatnom, tak je potrebné získať licenciu. Mod_plsql je však zastaralé a Oracle odporúča použiť ORDS namiesto tejto možnosti.
- **Embedded PL/SQL Gateway** (EPG) beží na Oracle XML DB Protocal Server v databáze Oracle a obsahuje základné funkcie mod_plsql. Licencia je obsiahnutá v Oracle Database licencii.

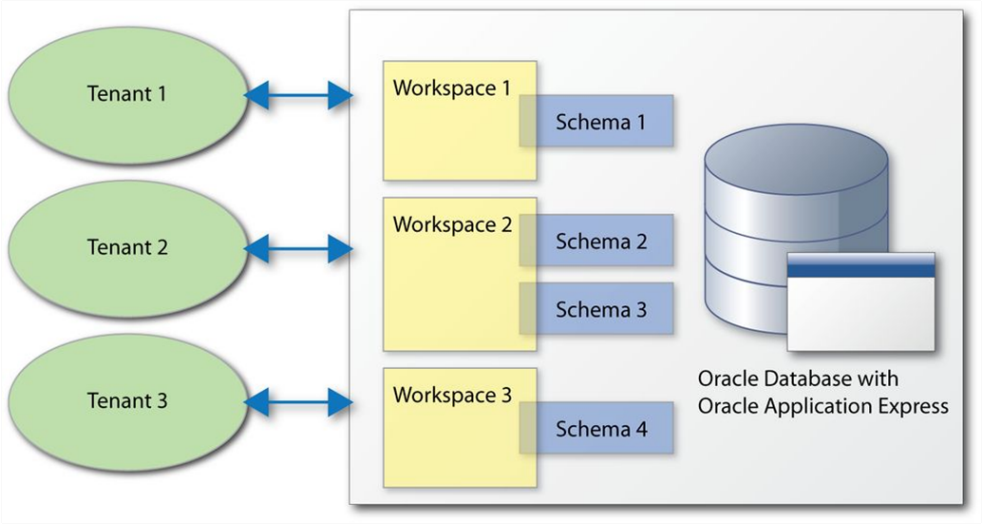
V tabuľke nižšie vidno konkrétne rozdiely medzi jednotlivými Web Listenermi.

Tabuľka 3.1: Porovnanie možností Web Listenerov

	Oracle REST Data Services	**Oracle HTTP Server**	**Embedded PL/SQL Gateway**
Umiestnenie obrázkov	súborový systém	súborový systém	v databáze
Možnosti konfigurácie	GUI release >=2.0 administračné stránky	Database Access Descriptor (DAD)	databázové inicializačné parametre
Nastavenia Connection Pool	JDBC parametre pripojenia	MinSpareServers; MaxSpareServers; MaxClients	SHARED_SERVERS; MAX_SHARED_ SERVERS
RESTful webové služby	áno (release >= 2.0)	nie	nie
Podpora multi-databáz	áno vrátane RAC	áno vrátane RAC	nie
Skenovanie súborov proti vírusom	áno s integráciou ICAP servera	nie	nie
Tlač PDF	áno vrátane podpory FOP (release >= 2.0)	nie	nie
Odporúčané prostredia	všetky	všetky	iba pre vývojové

3.2.3 Multi-Tenancy

Application Express umožňuje jednej databáze hostiť veľké množstvo aplikácií a užívateľov. Vývojári pracujú na vyhradenom pracovnom priestore zvanom workspace na vývoji aplikácií na jednej alebo viacerých schémach. Táto flexibilná architektúra umožňuje jednej inštancii databázy, aby sa chovala ako „Platform as a Service“ (PaaS), ktorá poskytuje možnosti Intranetu v rámci organizácie alebo hosťované internetové služby. Je bežné, že workspace-y sú definované pre jednotlivé oddelenia v rámci organizácie tak, aby každé oddelenie mohlo rozvíjať svoje vlastné databázové objekty a aplikácie nezávisle na sebe.



Obr. 3.3: Multi-Tenancy. Zdroj: [1]

3.2.4 Roles and Responsibilities

Jednotlivci, ktorí pristupujú k prostrediu Oracle Application Express môžu mať rôzne roly, zodpovednosti a práva.

Rozlišujú sa štyri hlavné roly spojené s Application Express:

- **Instance Administrator** zodpovedá za konfiguráciu a monitorovanie inštancie, vrátane zabezpečenia workspace-u, nastavenia funkcií, zabezpečenia a nastavenia inštancie
- **Workspace Administrator** zodpovedá za požiadavky, monitorovanie a udržiavanie vývojárov vo workspace. Všeobecne majú tiež vývojársku zodpovednosť
- **Developer** vyvíja aplikácie a požadované databázové objekty, ak objekty už nie sú k dispozícii.
- **End User** je definovaný užívateľ pre aplikácie, ktoré využívajú Application Express User Authentication Scheme (schéma overovania užívateľa).

Vo všeobecnosti stačí definovať niekoľko administrátorov inštancií pre prácu s databázovými administrátormi a systémovými architektmi, aby správne nakonfigurovali a spravovali prostredie Oracle Application Express. Pre každý workspace musí byť aspoň jeden workspace administrator a ľubovoľný počet vývojárov. Workspace administrátori sú zodpovední za udržiavanie poverení pre vývojárov. Definovanie koncových užívateľov v rámci Application Builder

je nutné len vtedy, ak aplikácia používa Application Express User authentication scheme. Ak sú aplikácie verejné, nevyžadujú overenie alebo iné systémy overenia ako SSO, LDAP atď využívané aplikáciami, tak nebudete musieť definovať koncových užívateľov.

3.2.5 Environments

Ako u každého životného cyklu vývoja softvéru, Oracle dôrazne odporúča mať rôzne prostredia pre vývoj, testovanie / QA a produkciu. Osvedčený postup je nainštalovať „Runtime Only“ Application Express v rámci svojich testovacích a produkčných prostredí. Tým sa odstránia komponenty Application Builder a SQL Workshop z testovacích a produkčných prostredí a vývojári sú nútení robiť všetky zmeny priamo vo vašom vývojovom prostredí. Vývojári by mali zaradiť všetky aplikácie a súvisiace súbory do source control systému a databázoví administrátori (DBA) majú overiť a spúšťať skripty priamo zo source control do testovania a produkcie.

3.2.6 Database Jobs

Po nainštalovaní Oracle Application Express, sú vytvorené štyri database jobs (databázové úlohy). Aby Application Express fungoval správne, musia tieto database jobs bežať pravidelne. Nainštalované database jobs zahŕňajú:

- **ORACLE_APEX_DAILY_MAINTENANCE** beží denne o 1:00 systémového času. Archivuje logy (záznamy) aktivít, očisťuje workspace a maže expirované súbory
- **ORACLE_APEX_PURGE_SESSIONS** beží každú hodinu. Maže informácie o sessione z APEXových tabuliek, ktoré sú staršie ako 12 hodín.
- **ORACLE_APEX_MAIL_QUEUE** beží každých 5 minút. Odosiela správy z e-mailovej fronty.
- **ORACLE_APEX_WS_NOTIFICATIONS** beží každých 30 minút. Odosiela Websheet notifikácie.

3.2.7 Database Backup

Štandardné Oracle zálohovanie (backup) a techniky obnovy (recovery techniques) by mali byť použité na zálohovanie workspace schém a engine-u APEX (APEX_050100 pre APEX 5.1). Avšak, obnovenie engine-u Application Express obnoví metadáta pre všetky aplikácie vo všetkých workspace-och, prepíše všetky aktualizácie aplikácií vykonané od zálohovania. Ak je to nevyhnutné pre obnovenie špecifických aplikácií zo zálohy, je odporúčané, aby

schéma engine-u APEX bola obnovená do samostatnej Oracle databázy a potom exportovaná z tohto zariadenia. Oracle Database má možnosť retrospektívy použitím exportu aplikácie „As Of“, ktorá potenciálne umožní vývojárom vytvárať export z nejakého skoršieho obdobia, ale spomienka sa vzťahuje len na určitú dobu, v závislosti od nastavenia parametra databázy. Vývojári by mali vykonávať vlastné pravidelné zálohovanie svojich aplikácií aj ďalších súborov, pokým nie je kód kompletný a zaradený do source control systému.

3.3 Novinky a zmeny

Spoločnosť Oracle neustále investuje do rozvoja vývojového prostredia Application Express a ako som už v tejto kapitole spomenul, tak nová verzia vychádza približne raz ročne. Keďže táto diplomová práca nadväzuje na moju bakalársku prácu, ktorú som obhájil pred troma rokmi, tak medzičasom si toto prostredie prešlo obrovskými zmenami. Bakalárska práca bola vyvinutá vo verzii APEX 4.2, ktorá vyšla v októbri 2012 a nasledujúce roky vznikali len menšie aktualizácie, ktoré zvyšovali čísla len na tretej úrovni číslovania verzie s poslednou 4.2.6, ktorá vyšla v septembri 2014. Oracle už pravdepodobne v tom čase venoval veľa času vyvíjaniu ohromných rozdielov verzie 5. Verzia 4.2 tak bola po dobu necelých troch rokov najdlhšie aktuálna verzia zo všetkých.

V apríli 2015 sa APEX komunita dočkala verzie 5.0, čo je najväčší krok vo vývoji prostredia Oracle Application Express a preto jej venujem celú sekciu 3.3.1. Nasledujúci rok a pol vyšli štyri menšie aktualizácie bez výraznejších zmien. V decembri 2016 sa Application Express posunul o ďalší väčší krok dopredu, ktorému venujem sekciu 3.3.2. Vo februári tohto roku bol vydaný aj Oracle Application Express Statement of Direction, čo je oficiálne vyhlásenie o smerovaní tohto vývojového prostredia a predpokladaných zmien v novej verzii 5.2. Podarilo sa mi nájsť aj informácie z iného zdroja ohľadom budúcich verzií, ktoré sa dosť týkajú tejto diplomovej práce, tak som sa rozhodol venovať jednu sekciu 3.3.4 aj týmto správam.

3.3.1 APEX 5.0

Táto verzia je výsledok dva a pol ročného úsilia inžinierov v spoločnosti Oracle vylepšiť IDE Application Express. Sám Michael J. Hichwa viceprezident Oracle Developer Tools citovaný na [3] povedal: „Oracle APEX 5 is the largest and most important release in the history of Oracle Application Express“. Čiže verzia 5 je najväčšie a najpodstatnejšie vydanie verzie v histórii Oracle Application Express. Mnoho vylepšení bolo zameraných na zvýšenie produktivity vývojárov a zdokonalenie užívateľského rozhrania aplikácií. Táto verzia priniesla kompletne zmenenú grafiku, čoho výsledkom je jednoduchšie menej farebné modernejšie rozhranie. Veľké zmeny nastali taktiež v časti Page

Designer, čo je najpoužívanejšia stránka pri vyvíjaní aplikácií, ktorú bližšie popíšem v 3.3.3.

Oracle si tiež uvedomuje, že v dnešnej dobe musia webové aplikácie fungovať dokonale aj na mobilných zariadeniach. S touto verziou prišlo hneď niekoľko vymožeností, ktoré podporujú vývoj takých aplikácií. Významnou novinkou je aj **Universal Theme**, ktorý umožňuje jednoducho vybudovať aplikáciu, ktorá funguje dobre na rozličných veľkostiach obrazovky. Táto novinka má však omnoho viac výhod a využití než len vývoj pre mobilné zariadenia, tak ju bližšie popíšem v 3.3.1.1. APEX zahrňuje jQuery Mobile, ktorý používa ľahký framework umožňujúci vytvoriť kompaktné a minimalistické webové stránky obsahujúce len pár obrázkov a CSS súborov. Pomocou jQuery Mobile sú schopné aplikácie bežať na akomkoľvek mobilnom operačnom systéme ako iOS, Android, Blackberry a Windows Mobile. jQuery Mobile podporuje udalosti špecifické pre mobilné zariadenia ako dotykové udalosti alebo zmena orientácie, čo umožňuje kompletne zmeniť zobraznie stránky ak sa preklopí mobilné zariadenie. Je taktiež možné modifikovať CSS pomocou nástroja jQuery ThemeRoller, aby sa zmenilo zobrazenie aplikácie. Umožňuje taktiež definovať podstatnosť stĺpcov, pre prípad zobrazovania na veľmi malé obrazovky. Podstatné budú zobrazované stále, zatiaľ čo nepodstatné budú skrývané ako prvé.

Keďže je Application Express vhodné používať na vývoj databázových aplikácií, tak častou úlohou je vytváranie reportov. Vývojár dokáže rýchlo vytvoriť report len pomocou napísania SQL dotazu. APEX taktiež obsahuje mnoho vstavaných sprievodcov umožňujúcich vytvoriť reporty. Pomocou sprievodcu je možné vytvoriť reporty aj bez písania SQL dotazov. Sprievodca podporuje *drag & drop*, čiže si stačí povyberať tabuľky a prepojiť čiarou stĺpce, podľa ktorých sa majú spojiť a report sa vytvorí bez písania kódu. Existujú tu dva typy reportov a to Classic Report a Interactive Report.

Interaktívny report dáva koncovému užívateľovi mnoho možností upravovať si výstup a obsah reportu. Obsahuje vyhľadávací panel, ponuku záhlaví stĺpcov, meniť ikony a takzvané Action menu. Pomocou tejto akčnej ponuky môže meniť poradie stĺpcov, skrývať a odkrývať stĺpce, aplikovať filtre, zvýrazňovať a radiť obsah reportu. Užívateľ si môže tiež definovať počet záznamov na stránku, agregácie, zoskupovanie (group by), grafy alebo pridať vlastné výpočty. Navyše si vie dané zmeny uložiť a pri ďalšom prihlásení bude report vyzerať tak, ako si ho nastavil, taktiež vie prepínať medzi všetkými takto uloženými nastaveniami. Má možnosť to uložiť aj ako verejný report a vtedy je viditeľný všetkým užívateľom. Vďaka tomu si je každý užívateľ schopný prispôsobí aplikáciu tak, aby vyhovovala jeho požiadavkám a nie je potrebné vytvárať rôzne stránky s reportmi ani zaťažovať vývojárov. Tieto interaktívne reporty boli vo verzii 5.0 tiež prerobené od základu, aby sa zvýši možnosti ako vývojárom, tak aj koncovým užívateľom. Nové interaktívne reporty môžu byť prešstylizované pomocou CSS podobným spôsobom ako ostatné regióny. Veľkou novinkou je, že na jednej stránke už môže byť viac takýchto reportov, čo bolo

obmedzenie predošlých verzií. Ďalšou zmenou je možnosť nastavenia pevných záhlaví reportu, čiže aj pri scrollovaní zostáva záhlavie viditeľné bez ohľadu na to, ktoré záznamy sú zobrazené. V novom interaktívnom reporte si koncový užívateľ môže taktiež definovať pivot report priamo v reporte.

APEX teraz poskytuje aj vstavaného sprievodcu vybudovať kalendáre s mesačným, týždenným, denným alebo zoznamovým pohľadom. Nový kalendár je založený na populárnej FullCalendar knižnici, čiže je ľahko prispôsobiteľný a podporuje *drag & drop*, časové udalosti a je taktiež responzívny.

Príjemnou správou pre vývojárov je aj nová možnosť vytvárať dialógy ako nové stránky APEX-u. Dáva to možnosť dialógom využívať všetky prvky a štandardné procesy stránok. Vývojári tak už nemusia pracne upravovať stránky pomocou JavaScriptu. Stačí, že nastavia typ zobrazenia a vhodnú šablónu a o zvyšok sa postará Application Express.

APEX 5.0 obsahuje aj širokú kolekciu ukážkových aplikácií nazvaných Packaged Applications. Ako názov napovedá sú to zabalené aplikácie a na pár klikov sa dajú rozbaliť a okamžite s nimi pracovať. Sú to pekne vytvorené responzívne aplikácie podporované spoločnosťou Oracle. Je tam 19 použiteľných aplikácií ako napríklad Project Tracking, Survey Builder, Meeting Minutes alebo Group Calendar. ďalej tam je 16 vzorových aplikácií, ktoré sa používajú na prezentáciu funkcií APEX-u od grafov po načítavanie dát. K dispozícii je aj vzorová aplikácia demonštrujúca výkonné priestorové možnosti prítomné v každej Oracle databáze.

3.3.1.1 Universal Theme

Universal Theme je úplne nové užívateľské rozhranie pre aplikácie navrhnuté od základu pre Application Express 5. V predošlých verziách síce APEX obsahoval viacero motívov (themes), avšak ak si ich chcel vývojár prispôsobovať musel to vyriešiť pomocou dopisovania kódu v HTML, CSS alebo JavaScripte. Universal Theme umožňuje vývojárom vybudovať moderné responzivné sofistikované aplikácie bez potreby odborných znalostí týchto jazykov. Aj vyššie citovaný Michael J. Hichwa si pochvaľoval, že vývojárom zameraným na SQL to dramaticky zrýchľuje a uľahčuje vytváranie responzívnych prístupných webových aplikácií bez pomoci CSS/HTML špecialistu. Je to jednoduchšie a aj tak ešte schopnejšie než motívy z predošlých verzií.

Responzívny dizajn umožňuje navrhovať webové stránky tak, aby rozloženie využívalo dostupné miesto podľa rozlíšenia obrazovky bez ohľadu na zariadení, na ktorom sa stránka zobrazuje, či už je to stolný počítač, prenosný počítač, tablet alebo inteligentný telefón. Implementáciou responzívneho dizajnu sa pri zobrazovaní stránky na tabletoch a inteligentných telefónoch rozloženie stránky prispôsobí veľkosti obrazovky konkrétneho zaridenia. Pri zobrazovaní na počítači získa užívateľ možnosť vyskúšania si rozloženia stránky na všetkých možných rozlíšeniach len pomocou zmeny veľkosti okna. Pri menení veľkosti okna sa prvky stránky okamžite presúvajú, menia veľkosť, spôsob

zobrazenia alebo skrývajú. Cieľom responzívneho dizajnu je zobraziť všetok podstatný obsah užívateľsky prívetivým spôsobom pre všetky možné veľkosti obrazoviek.

Universal Theme podporuje jednoduchšie prispôsobenia v reálnom čase pomocou vstavaných funkcionalít v Application Express engine. Je možné využiť viacúrovňového navigačného menu (ponuky) pomocou APEX List, ktorý sa dá umiestniť vľavo alebo hore, čo sú v dnešnej dobe najpoužívanejšie možnosti, prípadne sa dajú použiť obe možnosti v závislosti na zobrazovanom rozlíšení. Navigačné menu môže podporovať aj klávesové skratky. Vlajkovou loďou Universal Theme je Theme Roller, pomocou ktorého sa dá vyladiť mnoho štýlových prvkov a zmeny je vidno okamžite. Až bude vývojár spokojný s prevedenými zmenami môže uložiť zmeny ako Theme Style priamo do metadát aplikácie. Cez Template Options (nastavenie šablóny) je možné zmeniť spôsob akým sú zobrazované tlačítka, regióny, zoznamy a ostatné šablónovo založené komponenty. Jedna šablóna vie byť zobrazená niekoľkými rôznymi spôsobmi vďaka týmto nastaveniam. Dáva to možnosť vývojárom používať menej šablón na vytvorenie oveľa všestrannejších aplikácií. APEX ponúka aj ďalšie výhody Theme Styles a Theme Subscription, čo znamená, že jeden motív môže mať svoj nadradený, a tým pádom zmeny v nadradenom sú propagované do podradených.

3.3.1.2 APEX Tree

APEX Tree je typ regiónu, kde je pomocou hierarchického selectu možné vytvoriť rozklikávaciu stromovú štruktúru. Hierarchický select je taký, kde každý riadok obsahuje relačné stĺpce ID a rodičovské ID a pomocou klauzule START WITH ... CONNECT BY sa vygeneruje stromová štruktúra. Takéto stromové zobrazenie sa síce už nachádzalo aj v starších verziách APEX-u. Bol tak vytvorený aj Page Designer vo verzii 4.2 ako je vidno na obrázku 3.6. Vývojári však mali možnosť vytvárať v tej verzii stromy len pomocou jsTree, čo si vyžadovalo písanie kódu v JavaScripte. Nový APEX Tree je síce taktiež založený na JavaScripte, no vytváranie a úpravy sú omnoho jednoduchšie. Verzia APEX 5.0 už nemá možnosť vytvárať regióny jsTree, dáva len možnosť meniť už existujúce.

Pri konzultáciach ohľadom grafického rozhrania tejto práce sme sa s bývalými kolegami zhodli, že najkrajšie a najintuitívnejšie by bolo zobrazenie rozdielov v stromovej štruktúre, čo sa však počas vypracovávania bakalárskej práce nestihlo a zvolilo sa jednoduchšie tabuľkové zobrazenie. APEX Tree región teda zodpovedá predstavám vývojárov, pre ktorých bola táto práca určená. Túto novinku teda použijem pri prerábaní grafického rozhrania mojej bakalárskej práce.

Tento nový typ regiónu má možnosti klávesovej navigácie a ukladanie stavu. Má tie isté funkcionality ako jsTree, ale má ešte vylepšený výzor a použiteľosť novou implementáciou.

3.3.2 APEX 5.1

Verzia 5.1 je taktiež veľký skok vpred ako pre vývojárov, tak aj pre koncových užívateľov. Rovnako ako verzia 5.0 je táto zameraná na jednoduchší vývoj vyžadujúci menej klikov a príjemných vylepšení. Asi všetky novinky z 5.0 boli ešte viac vylepšené a zjednodušené. Okrem vylepšení sa táto verzia disponuje aj radou nových funkcionalít ako je JET Chart a Interactive Grid (interaktívna mriežka).

Interactive Grid je novinka kombinujúca funkcionality z Interactive Reports a Tabular Form. Ide o región na klientskej strane umožňujúci rýchlu editáciu viacerých riadkov dát v dynamickej mriežke. Zahŕňa väčšinu možností prispôsobovania, ktoré ma k dispozícii interaktívny report, a navyše schopnosť meniť šírku a usporiadanie stĺpcov priamo v reporte pomocou *drag & drop*. Môže sa nastaviť, že je iba na čítanie alebo môže byť editovateľná. V editovateľnom nastavení môže užívateľ priamo v reporte dáta meniť, pridávať alebo mazať. Podporuje taktiež vynikajúcu podporu a použiteľnsť klávesnice. Existujúce tabuľkové formuláre je možné migrovať na interaktívne mriežky pomocou sprievodcu Upgrade Application wizard. Obsahuje všetky očakávané funkcionality ako pevné záhlavia, zamrazné stĺpce, stránkovanie, možnosť aplikovania viacerých filtrov naraz, radenie, agregáciu a ďalšie. Editácia dát môže byť riadená prostredníctvom zoznamu hodnôt alebo takzvaným date pickerom.

Táto nová interaktívna mriežka umožňuje viacúrovňový Master > Detail > Detail vzťah, čím odstránili obmedzenie minulých verzií, kde jedna stránka mohla obsahovať len jeden Master Detail vzťah. V aktuálnej verzii je teda možné stromovo definovať nadväznosť detailných zobrazení či už do hĺbky alebo do šírky ako znázorňuje obrázok 3.4.

Obr. 3.4: Master > Detail > Detail. Zdroj: [2]

Väčšina zákazníkov v dnešnej dobe požaduje vizualizáciu dát prostredníctvom grafov. Application Express síce vedel vytvárať grafy pomocou AnyChart, no aj v tejto oblasti spravil veľký krok integráciu vizualizačného enginu,

ktorý beží na Oracle JET (JavaScript Extension Toolkit). Je to modulárny open source nástroj založený na báze moderných princípov návrhu a vývoja v jazykoch JavaScript, CSS3 a HTML5. Toto vytváranie grafov je vysoko prispôsobiteľné prístupné interaktívne a automaticky zahŕňa podporu responzívneho dizajnu. Je teda možné vytvárať krásne rýchle vysoko prispôsobiteľné a mimoriadne všestranné grafy. Je možné vizualizovať čisté, ale aj hierarchické dáta. Poskytuje desiatky možných spôsobov ako sa dajú dáta vizualizovať. Pre krajšiu prezentáciu podporuje aj animácie.

Rady vylepšení sa dočkal aj Universal Theme na základe úspechu v komunite vývojárov. Boli pridané nové štýly a šablóny. Universal Theme bol zjednodušený a má vylepšený dizajn aj užívateľské rozhranie vo všetkých častiach. Bola pridaná podpora pre jazyky, ktoré sa píšu sprava doľava, aby boli správne vykresľované. Je možné nastaviť preferencie tém na základe preferencií užívateľa. Zaviedli Live Template Options, aby sa ešte zvýšili možnosti upravovania šablón v reálnom čase.

Pre zvýšenie prehľadnosti bola pridaná možnosť nastaviť obrázok aplikácií, ktorý bude zobrazený v Application Builder, kde je zoznam všetkých aplikácií v danom workspacy. Taktiež bola rozšírená aj knižnica ikón, ktorá momentálne obsahuje cez 1100 kusov. Nové ikony boli špeciálne navrhnuté tak, aby dopĺňali vývoj podnikových aplikácií. Graficky sa vylepšilo aj automatické nastavovanie veľkosti dialógov podľa ich obsahu.

Vylepšenia sa dočkali aj dynamické akcie (Dynamic Actions), ktoré momentálne môžu byť asynchronné a umožňujú zobraziť progress bar pre dlhotrvajúce transakcie.

Zjednodušených bolo viacero sprievodcov, ktoré si teraz vyžadujú menej krokov a majú lepšie nastavené štandardné hodnoty, čiže vytváranie komponentov prostredníctvom sprievodcov je ešte rýchlejšie než bolo. Najviac sa to týka sprievodcov vytvárajúcich aplikácie a stránky.

Aplikácie po novom aj upozorňujú koncových užívateľov o neuložených zmenách pri odchádzaní zo stránky.

Do Packaged Applications boli pridané tri nové použiteľné aplikácie a tri ukážkové aplikácie. Tu ma zaujala aplikácia s názvom Quick SQL, kde sa dá prostredníctvom jednoduchého stručného zápisu generovať časti databázového modelu ako tabuľky, vzťahy, triggery, kontrolné obmedzenia a naplniť tabuľky ukážkovými dátami. Keďže mi to príde zaujímavé a inovatívne riešenie rozhodol som sa mu venovať podsekciu 3.3.2.1.

Ďalších príjemných zmien sa dočkal nový Page Designer, ktorého zmeny spolu so zmenami v 5.0 popíšem v časti 3.3.3.

Nový kalendár prešiel tiež početnými vylepšenimi. Pri kliknutí do kalendára sa teraz dajú použiť klávesnícové šípky na navigáciu v kalendári. Vývojárom bola pridaná možnosť definovať dynamické akcie súvisiace s udalosťami zadanými v kalendári, čo výrazne zvyšuje jeho použiteľnosť na rôzne účely. Kalendáru bol pridaný atribút Initialization JavaScript Code, čo umožňuje vloženie kódu pre úpravu inicializácie kalendára.

Pri submite stránky sa už položky stránky neposielajú ako argumenty procedúry wwv_flow.accept, ale prostredníctvom JSON dokumentu, čím sa odstránilo obmedzenie maximálneho počtu 200 položiek na stránku, ktoré mali predošle verzie APEX-u.

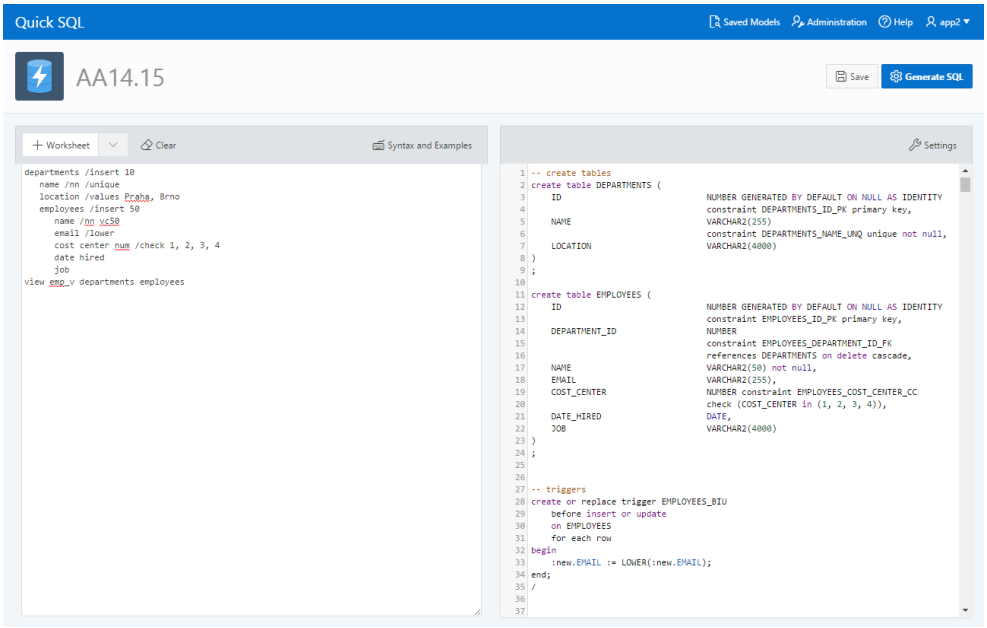
3.3.2.1 Quick SQL

Quick SQL sa dá vyskúšať nie len nainštalovaním z Packeged Applications, ale aj online. Na stránke [10] je odkaz na online verziu, stačí mať Oracle účet. Nachádza sa tam aj dvojminútové video ukazujúce jeden príklad vytvorenia dvoch prepojených tabuliek, pohľadu, triggerov, checkov a naplnenie ukážkovými dátami.

Táto aplikácia nie je určená ako náhrada nástrojov na vytváranie databázového modelu, ani neobsahuje všetky možnosti vytvárani databázových objektov. Avšak vďaka jednoduchosti je možné rýchlo vytvoriť základnú štruktúru a následne sa dá doupravovať prostredníctvom iných nástrojov na modifikáciu databázových objektov ako SQL Developer alebo SQL Workshop v APEX-e.

Dá sa využiť na rýchle vytvorenie prototypu aplikácie, aby sa zákazníkovi dalo v priebehu niekoľkých minút ukázať ako by mohla vyzerať aplikácia aj s nejakými ukážkovými dátami bez nutnosti písania rozsiahlych DDL príkazov.

Demonštrujem jednoduchosť na jednom príklade, ktorý je vidno na obrázku 3.5. Ide v podstate o vzorovú ukážku doplnenú o niekoľko ďalších príkazov.



Obr. 3.5: Ukážka Quick SQL

Jediná forma dokumentácie k tomuto nástroju sa nachádza priamo v aplikácií prístupná po kliknutí na tlačítko „Syntax and Examples“. Nachádzajú sa tam sekcie s tabuľkami príkazov a nastavení s vysvetlením a syntaxou. Dozviete sa tam ako definovať dátové typy, vytvoriť ombedzenia a možnosti definovania akými ukážkovými dátami sa majú tabuľky naplniť. Nachádza sa tam aj 9 príkladov, ktoré ukazujú použitie rôznych kombinácií týchto možností.

Na obrázku vidno, že odsadzovaním textu od kraja sa definujú stĺpce a v prípade, že dojde k ďalšiemu odsadeniu, tak sa jedná o ďalšiu tabuľku, ktorá bude podradená a automaticky sa vygeneruje cudzí kľúč a index. Každej tabuľke sa automaticky generuje aj stĺpec ID ako primárny kľúč. Na obrázku vidno aj 2 definície dátových typov stĺpcov a to name a cost_center v tabuľke employees. Príznak *vc50* značí *varchar2(50)* a *num* značí *number*. Aby si stĺpec vyžadoval hodnotu, tak zaňho jednoducho naíšete */nn* a vygeneruje sa ako *not null*.

V ukážke taktiež znázorňujem ako jednoducho sa vytvárajú constrainty, čo sú databázové konštrukcie obmedzujúce dáta, ktoré môžu byť vložené do daného stĺpca. Ak nie je primárny kľúč definovaný explicitne, automaticky sa vygeneruje stĺpec ID s týmto constraintom. Aj cudzie kľúče sa dajú definovať v prípade, že zápis nie je vytvorený pomocou odsadzovania ako je v ukážke. Taktiež sa dajú jednoducho definovať kontrolné a unikátne obmedzenia, ktoré som zahrnul do príkladu. Unikátny sa vytvorí pomocou príkazu */unique* a kontrolný pomocou príkazu */check*, za ktorým následuje čiarkou oddelený zoznam povolených hodnôt. Podobne sa dá kontrolné obmedzenie vytvoriť aj pomocou príkazu */between*, kde stačí definovať len dolnú a hornú hranicu povolených hodnôt.

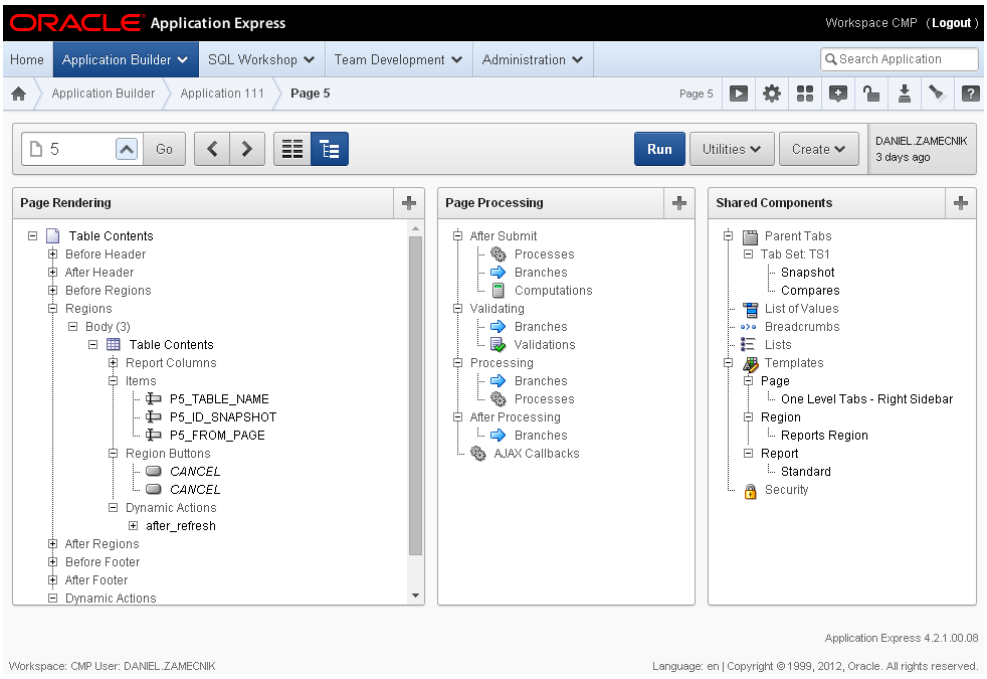
Generovať sa dá aj rada bežne používaných triggerov. Ako vidno v ukážke, tak príkaz */lower*, vytvorí pred uložením dát do tabuľky na hodnotu v danom stĺpci zavolá funkciu lower, ktorá prevedie všetky znaky na malé. Pre opačný efekt sa dá využiť príkaz */upper*. Tieto príkazy sú určené na špecifikáciu chovania pre stĺpce.

Pre vytvorenie najjednoduchšieho spôsobu auditu sa dá použiť príkaz na tabuľku */auditcols*. Ten vytvorí v tabuľke štyri stĺpce updated, updated_by, inserted, inserted_by a odpovedajúce dva triggery, ktoré budú vypĺňať hodnoty týchto stĺpcov.

Ak chcete tabuľky naplniť náhodnými dátami, stačí za tabuľku napísať */insert* a čislo určujúce koľko riadkov sa má vytvoriť. Tento nástroj nejakým spôsobom zvláda rozlišovať čatné stĺpce ako name, email, country, location a podobne. Do týchto stĺpcov generuje rozumné náhodné hodnoty. Nikde som však nenašiel, ktoré všetky možnosti to zvláda. Pri generovaní náhodných dát je možnosť mu povedať aké možné hodnoty má generovať pomocou príkazu */values*, za ktorý čiarkami oddelíte možné hodnoty. Príklad použitia som doplnil do ukážky, kde avšak nevidno vygenerované SQL príkazy, lebo vygenerovaný dokument je rozsiahli.

3.3.3 Page Designer

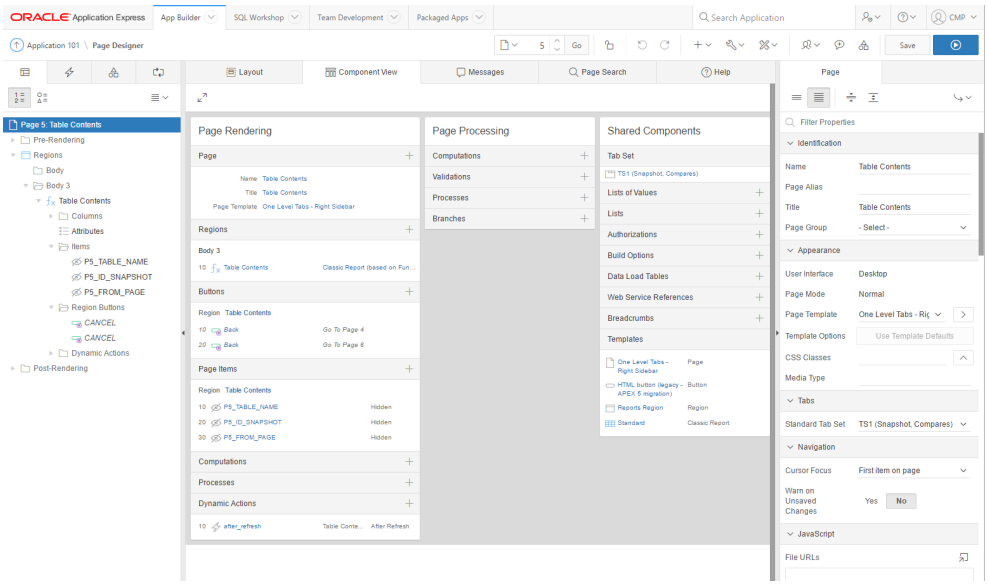
Page Designer je hlavný nástroj na editáciu APEX-ových stránok. Ak chcete upravovať stránky, tak sa najprv musíte prihlásiť do vývojového prostredia Application Express. Po prihlásení sa dostanete na úvodnú stránku, z nej sa preklinete do App Builder, kde nájdete zoznam všetkých aplikácií, ktoré sú vo workspacy, do ktorého ste sa prihlásili. Po kliknutí na nejakú aplikáciu sa otvorí zoznam stránok. Ak chcete upravovať nejakú stránku, stačí na ňu kliknúť a otvorí sa práve tento Page Designer. Pre názornú ukážku rozdielov, ktorými si prešiel priložím obrazovky z oboch verzií tej istej stránky a popíšem ako v novej verzii Oracle dosiahol šetrenie času vývojárov. Na obrázku 3.6 vidno ako vyzerala editácia stránok vo verzii 4.2 a na druhom 3.7 ako to vyzerá v najnovšej verzii 5.1.



Obr. 3.6: Page Designer v APEX 4.2

Nový Page Designer disponuje celou radou noviniek pre rýchlejší vývoj. Jednou je podpora *drag & drop* naprieč viacerých častí APEX-u, čo umožňuje rýchlejší a intuitívnejší vývoj, než hľadanie konkrétnych nastavení. Nová je aj záložka *Layout*, kde je grafická reprezentácia toho, ako bude stránka zobrazená a pomocou *drag & drop* sa dá jednoducho preorganizovať alebo sa dajú dopĺňať nové komponenty pretiahnutím z Component Gallery. Podporuje aj klávesový vstup *Ctrl* a *Shift* pri uznačovaní, ako je bežné aj v ných systémoch, čím sa dá označiť viac komponentov naraz a presunúť alebo kopírovať ich.

Novinka, ktorá najviac šetrí čas vývojárom, je Property Editor, ktorý vidno na obrázku 3.7. Je to panel nachádzajúci sa na pravej strane obrazovky. Ak chcel vývojár vykonať zmenu v predošlých verziách musel kliknúť na prvok, ktorý chcel upraviť, a to ho poslalo na ďalšiu stránku, kde sa nachádzali všetky nastavenia danej komponenty. Po prevedení zmien vývojár stlačil na tlačítko *Save*, čo ho poslalo naspäť do Page Designeru. Nový región Property Editor odstránil všetky tieto zbytočné kroky a načítavania Page Designera stále dookola. V aktuálnej verzii sa po kliknutí na niektorú z komponent len aktualizuje región Property Editor s relevantnými možnosťami nastavenia. Pre ľahšie nájdenie konkrétnych nastavení má Property Editor aj vyhľadávací panel. Taktiež pre lepší prehľad sú zmenené vlastnosti označené modrým rohom, dokým sa stránka neuloží.



Obr. 3.7: Page Designer v APEX 5.1

Ďalšou novinkou sú tlačítka *Undo* a *Redo*, ktorými disponuje väčšina vývojových prostredí. V predošlých verziách ak chcel vývojár vrátiť neúspešné zmeny musel to buď všetko naspäť vyklikať, kde ale hrozilo, že by sa mu to nemuselo podariť správne, alebo bezpečnejšia možnosť bola mať vyexportovanú stránku pred vykonaním zmien. V aktuálnej verzii to je možné za pár sekúnd vrátiť pomocou tlačítka *Undo*.

Vylepšený bol aj Code Editor, čo sú všetky oblasti, kde sa vpisuje SQL a PL/SQL kód. Poskytuje validáciu kódu s inline chybovými hláškami, automatické dopĺňanie kódu, zvýrazňovanie syntaxe, vyhľadávanie s nahradzovaním s podporou regulárnych výrazov aj s podporou *undo* a *redo*.

Aby sa vývojári vyhli zbytočným komplikáciám, majú teraz možnosť zamknúť stránku, čo znemožní menenie stránky ostatným užívateľom, dokým ju

neodomkne užívateľ, pre ktorého je zamknutá. Na to je určené tlačítko v hornom panely Page Designeru, ktoré umožňuje nechať aj komentár pri zamykaní.

Čo sa týka grafických rozdielov, tak vidno, že v novej verzii vsadili na menej farebný dizajn, čo je v dnešnej dobe modernejšie a vo všeobecnosti považované za nadčasovejšie riešenie. Spojením horných dvoch panelov, ktoré aj tak neboli až tak často využívané, ušetrili miesto na obrazovke. Vývojári majú taktiež možnosť si upravovať veľkosti jednotlivých regiónov a preorganizovať záložky podľa toho, ako im to vyhovuje.

3.3.4 Budúcnosť APEX-u

V oficiálnom dokumente od spoločnosti Oracle „Oracle Application Express Statement of Direction“ [9] je v bodoch rozpísané, ktorým oblastiam sa plánujú venovať vo verzii 5.2.

Chcú rozšíriť ich bežné komponenty ako interaktívne mriežky, reporty a grafy o rozhranie používajúce ORDS a REST na prístup k dátam uloženým v zdialených databázach. Plánujú poskytnúť deklaratívne metódy na definovanie odkazov na externé aplikačné REST rozhrania a JSON informačné kanály, aby mohli byť použité ako zdroje dát pre interaktívne mriežky, reporty a formuláre.

Interaktívnej mriežke bude aj vylepšené užívateľské rozhranie a pridané funkcionality, ktorými už disponuje interaktívny report. Medzi ne patrí napríklad zoskupovanie a vytváranie Pivot reportov.

Plánujú vytvoriť niekoľko nových sprievodcov. Prvý je Create App Wizard na vytváranie aplikácií, ktorý by taktiež vedel upravovať už existujúce. Pomocou neho by sa jednoduchšie dali pridávať komponenty a funkcionality. Budú vytvorení aj sprievodcovia na vytváranie komplexnejších komponent ako sú dynamické akcie, formuláre a takzvané Shared Components, čo sú časti aplikácie ktoré sú prístupné pre všetky stránky.

Aktualizované budú Oracle JET a jQuery, ktoré prevezmú najnovšie verzie integrovaných knižníc JavaScriptu, ktoré využívajú novinky ohľadom vizualizácie a nových formulárových nástrojov a ovládania. A ako v každej z posledných verzií, tak aj v následujúcej bude vylepšená oblasť Packaged Applications.

Omnoho podstatnejšie ohľadom témy mojej diplomovej práce sú informácie, ktoré odzneli od vyššie citovného pána Michael J. Hichwa počas prednášok pre APEX World ohľadom budúcnosti APEX-u. Podarilo sa mi nájsť dve fotky [14] [15] z dvoch rôznych prednášok. Ide o posty na sociálnej sieti Twitter od pána Simona Greenwooda a spoločnosti Explorer. Ak by nestačilo, že ide o prednášku viceprezidenta Oracle Developer Tools, tak pre zvýšenie dôveryhodnosti zdroja predstavím aj pána Simona Greenwooda. Ide o riaditeľa vývoja v spomínanej spoločnosti Explorer, ktorý nielenže má viac než dvadsaťročné skúsenosti s vývojom v Oracle technológiách, ale od roku 2005

aj prevzal vedúcu úlohu pri propagácii Application Express zákazníkom spoločnosti Oracle.

Na fotkách sa nachádzajú zoznamy noviniek, ktoré sa dajú očakávať v budúcich verziach, nie len vo verzii 5.2. V zozname sa nachádza vlastný APEX app diff, čo má byť zachytávanie histórie všetkých zmien prevedených v aplikáciách. Na základe tejto informácie sa dá predpokladať, že v priebehu najbližších niekoľkých rokov, bude APEX disponovať nástroju podobnému mojej diplomovej práci. Keďže bude integrovaný vo vývojovom prostredí a vyvíjaný profesionálmi, ktorý poznajú vnútorné fungovanie omnoho lepšie, dá sa očakávať, že nebude vyžadovať žiadne ďalšie nastavovania a kroky pri správe a bude fungovať automaticky a mať lepšie funkcionality, čo sa spôsobom ako to robím ja nedá dosiahnuť. Síce sa tam nespomína nič ohľadom nasadzovania zmien do ďalších prostredí, ale dá sa predpokladať, že až bude hotový tento nátroj, tak prepájanie prostredí môže byť ďalší krok. Táto novinka ukazuje, že si Oracle uvedomuje nedostatku v oblasti verzovania a podpory pri nasadzovaní rozdielov do iných prostredí a začína mu venovať pozornosť.

Zo zoznamu ma ešte zaujali aj ďalšie dva body. Prvým je „Friendly URLs“, čo je celkom príjemné, keďže teraz sa prostredníctvom URL posielali, čísla aplikácií, stránok, sessiony, názvy a hodnoty prvkov stránok a ďalšie informácie. Druhý bod je automatické testovanie APEX aplikácií, čo spolu s APEX app diff naznačuje ako vývojové prostredie Application Express dospieva a stáva sa z neho prvotriedny nástroj na vývoj webových aplikácií.

3.4 Prostredie APEX

Prístup do vývojového prostredia APEX je možný prostredníctvom webového prehliadača na definovanej URL adrese. Na nej sa zobrazí prihlasovacia stránka, kde je potrebné zadať názov workspacu a svoje prihlasovacie údaje. Tým sa dostanete na úvodnú stránku, kde sa nachádza rozcestník do 4 hlavných častí tohto prostredia, vyhľadávanie a nejaké informácie ohľadom obsahu a verzii.

Spomínané 4 hlavné časti sú:

- **App Builder**
- **SQL Workshop**
- **Team Development**
- **Packaged Apps**

Prvé dve bližšie popíšem v podsekciách 3.4.1 a 3.4.2, keďže sú dôležité pre vývoj aplikácií, a druhé dve len stručne predstavím.

Team Development slúži na správu, korigovanie a prehľad ohľadom vývoja aplikácii v danom workspacy. Obsahuje mnoho využiteľných nástrojov v prípade, že v danom workspacy pracuje väčší tím na viacerých aplikáciách. Vedúcemu tímu to dáva dobrý prehľad prostredníctvom grafov a percentuálnych hodnôt. Je tam možné nastavovať míľniky (milestones), zadávať nové vlastnosti, chyby, spätné väzby a takzvaný „To Do“ zoznam.

Packaged Apps obsahuje zoznam aplikácií, ktoré je moźné v priebehu niekoľkých sekúnd nainštalovať a používať. V aktuálnej verzii ich je 40. Rozdeľujú sa na Productivity a Sample. Sample aplikácie slúžia ako ukážky funkcionalít APEX-u. Productivity aplikácie sú hotové riešenia, ktoré majú konkrétne použitie. Nachádza sa tam napríklad Quick SQL, ktorý som popísal v 3.3.2.1. Ďalej sú tam aplikácie ako Bug Tracking, Group Calendar, Customer Tracker, Incident Tracking, Opportunity Tracker, Decision Manager a mnoho ďalších. Navyše sa každou verziou tieto aplikácie aktualizujú a dopĺňajú o nové.

3.4.1 Application Builder

V tejto časti sa nachádza všetko ohľadom vývoja aplikácií. Na úvodnej stránke tejto sekcie sa nachádza zoznam aplikácií, ktoré obsahuje workspace, do ktorého sa užívateľ prihlásil. Má možnosť exportovať existujúce alebo naimportovať nové aplikácie. Po kliknutí na nejakú konkrétnu aplikáciu sa dostaneme na stránku so zoznamom stránok v danej aplikácií, možnosťou dostať sa k nastaveniam aplikácie, zdieľaným komponentom (Shared Components) alebo podporujúcim objektom. Je tam aj tlačítko „Run Application“, ktoré v novom okne otvorí danú aplikáciu. Po rozkliknutí nejakej stránky sa otvorí Page Designer, ktorého novinky a zmeny som už popísal v 3.3.3.

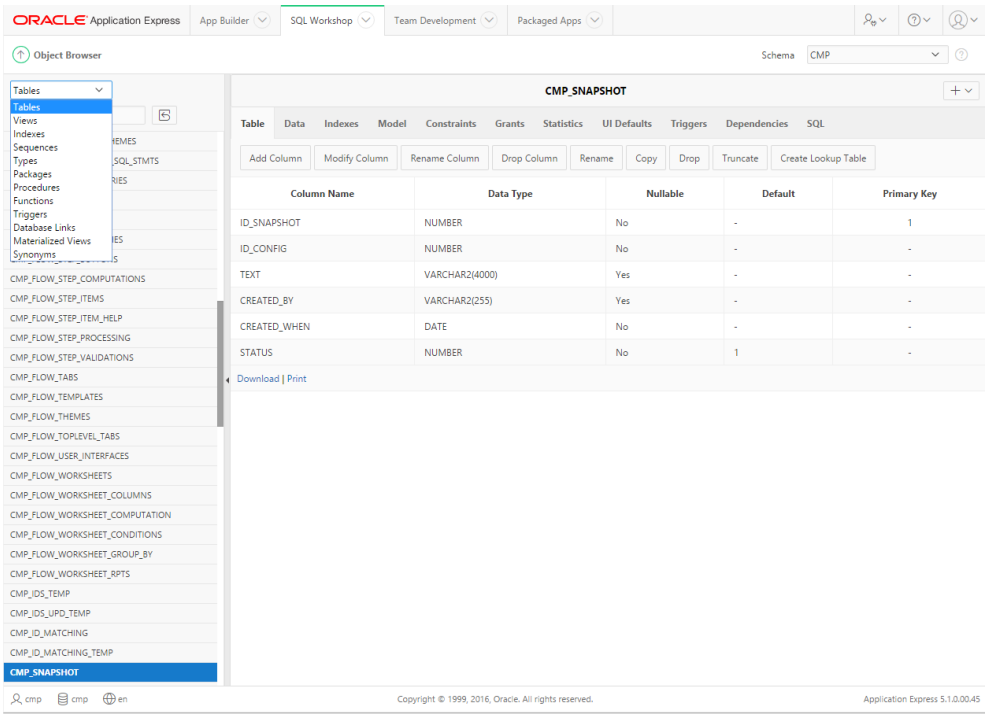
Page Designer je rozdelený do troch hlavných regiónov ako vidno na obrázku 3.7. Naľavo je stromový zoznam obsahu stránky, ktorý sa dá zobraziť dvoma spôsobmi, a to zoskupený podľa poradia vykresľovania alebo podľa typu komponentov. V strede je región s piatimi záložkami, kde najpodstatnejšie sú „Layout“ a „Component View“. V záložke „Layout“ je grafická reprezentácia obsahu stránky a v „Component View“ sú zoznamy všetkých komponentov stránky zoskupených podľa typov. Napravo je región „Property Editor“, v ktorom sú všetky nastavenia komponentu, ktorý bol posledne označený v jednom zo spomínaných dvoch regiónov.

V Shared Components sa dajú ponastavovať rôzne možnosti, ktoré sú potom dostupné zo všetkých stránok aplikácie. Z aplikačnej logiky sa tu dajú nastaviť globálne položky, procesy alebo výpočty. Ohľadom bezpečnosti autorizačné a autentifikačné schémy, ochrana stavu sessiony alebo Build Options. Mnoho ďalších možností ohľadom užívateľského rozhrania, navigácie v aplikácii, súborov, globalizácie, reportov, vzdialených zdrojov dát alebo zásuvných modulov (plug-ins).

3.4.2 SQL Workshop

V SQL Workshope sa nachádzajú všetky nástroje na prácu s databázou. Na úvodnej stránke je rozcestník na päť častí Object Browser, SQL Commands, SQL Scripts, Utilities a RESTful Services. Prvé dve pokrývajú hlavné funkcionality SQL Developera a úplne postačujú na vývoj aplikácií. Na úvodnej stránke sa ešte nachádzajú zoznamy naposledy vytvorených tabuliek a posledne spustených SQL príkazov.

Object Browser umožňuje prehliadať a vytvárať všetky databázové objekty. V prípade, že workspace obsahuje viac databázových schém, tak je tam select list na prepínanie medzi nimi. Ďalší select list slúži na výber typu databázových objektov (tabuľky, pohľady, triggery, procedúry, databázové linky a ďalšie), ktoré sa majú zobraziť v zozname naľavo. Nachádza sa tam aj vyhľadávací panel pre rýchlejšiu prácu. Ukážku vidno na obrázku 3.8



Obr. 3.8: Object Browser

V časti SQL Commands sa dajú vylaďovať SQL požiadavky alebo spúšťať PL/SQL kód. Podobne ako v SQL Developeri tu je možnosť spúšťať SQL výrazy a v tabuľke vidieť výsledok, taktiež je tu možnosť zobraziť Explain Plan. Dajú sa ukladať vyladené príkazy a má to aj históriu spustených. V časti SQL Scripts sa dajú vytvárať, importovať a exportovať skripty.

V časti Utilities sú rôzne nástroje ako načítavanie dát z textových, XML alebo spreadsheet (napríklad Excel) dokumentov; generovanie DDL skriptov z existujúcich databázových objektov, porovnávanie databázových objektov z rôznych schém, sprievodca Query Builder, pomocou ktorého sa dajú vytvárať SQL požiadavky grafickou cestou bez písania kódu a ďalšie.

Kapitola 4

Špecifikácia zadania

S návrhom témy tejto práce prišiel Ing. Kamil Schvarcz, jednateľ spoločnosti APEX Solutions, pre ktorú som pracoval počas môjho bakalárskeho štúdia. Táto firma sa venuje vývoju webových aplikácií a čím ďalej, tým viac sa zameriavala na vývoj v Application Express.

Pri vyvíjaní vtedy najväčšieho APEX-ového projektu sme narazili na zopár komplikácií, ktorých príklad popíšem v 4.2. Okrem nich sme boli so zákazníkom dohodnutý na systéme, že zadávali požiadavky s rôznou úrovňou priorít. Ak prišla urgent požiadavka, tak sa odložilo, na čom sa momentálne pracovalo a bola riešená okamžite. Zvyšné prechádzali štandardnejším spôsobom schvalovania. Čiže vo vývojovom prostredí sa nachádzalo viacero rozpracovaných úloh, ktoré boli dorábane priebežne. Z toho dôvodu sa niekedy stalo, že pri nasadení sa na niečo zabudlo. Pre tieto dôvody sa mnoho času strácalo pri nasadzovaní aplikácií, a tak sme chceli prísť s nejakým lepším riešením.

Predstava bola mať nástroj, ktorý by vedel zobraziť zmeny medzi dvoma aplikáciami a v ideálnom prípade to na kliknutie nasadiť. Keďže mi to prišlo ako zaujímavá téma na diplomovú prácu a lákalo ma riešiť nejaký rozsiahly dlhodobý projekt úplne sám, tak som to celé zobral do svojích rúk od analýzy cez návrh po implementáciu.

Požadované funkcionality sa po krátkom čase rozšírili o to, aby nástroj v sebe udržiaval verzie, čím by sa nahradili predošlé spôsoby verzovania, a vedel zobraziť rozdiely medzi nimi.

Keďže išlo o podporu medzi rôznymi prostrediami, kde produkčné prostredia boli u zákazníkov, tak samozrejmá bola požiadavka vzdialeného prístupu.

4.1 Odporúčané nasadzovanie aplikácií od Oracle

Oracle odporúča [5] pri vývoji aplikácii postupovať podľa štandardných postupov životného cyklu vývoja systému, čiže mať rôzne prostredia pre vývoj, testovanie a produkciu. Vývojári by mali mať možnosť vykonávať zmeny v ap-

likáciách a súvisiacich databázových objektoch iba vo vývojovom prostredí. Na presadenie týchto pravidiel odporúča Oracle nainštalovať Application Express do testovacieho a produkčného prostredia v režíme „Runtime Only“, čo zabráni v prístupe do Application Builder a SQL Workshop. Aktualizovať tieto prostredia by mali vedieť len databázoví administrátori.

Vývojári potrebujú pri práci často vytvárať alebo meniť databázové objekty alebo povolenia, prípadne obsahy tabuliek. Keďže export aplikácie neobsahuje nič z databáze, tak by mali vývojári vytvárať DDL a DML skripty podľa potreby, aby boli všetky zmeny reprodukovateľné. Okrem databázových objektov sa v exporte nenachádzajú ani obrázky, CSS súbory ani JavaScript súbory, ktoré sa musia spravovať samostatne.

V prípade, že sa vo vývojovom prostredí nachádzajú ešte nedokončené stránky alebo komponenty, tak sa dá využiť „Build Options“, kde sa dajú vylúčiť z exportu.

Po tom, čo vývojári dokončia vhodný kus práce, mali by exportovať aplikáciu z Application Buildera a zaradiť ju do source control systému. Za source control systém Oracle považuje akúkoľvek metodiku nasadzovania, akú si firma vyvíjajúca aplikácie zvolila alebo vyrobila. Do source control systému by mali byť zaradené aj DDL a DML skripty, obrázky, CSS súbory, JavaScript súbory a teda všetko čo nie je v exporte aplikácie. Vývojári by potom mali vyplniť build sheet, čo je zoznam zmien od poslednej verzie, aby databázový administrátor mohol vytvoriť nasadenie do testovania a po otestovaní do produkcie.

4.2 Nedostatky odporúčaného nasadzovania

Nedostatky, pre ktoré je odporúčaný postup od Oraclu nepoužiteľný respektíve nepraktický, vysvetlím na príklade z praxe. Majme aplikáciu, ktorá má viac produkčných prostredí a pravidelne prichádzajú požiadavky na zmeny a nové funkcionality. Väčšina zmien sa nasadzuje na všetky produkčné prostredia, no niektoré sú požadované len na jedno prípadne viac produkčných prostredí. Odporúčaný model je nepoužiteľný z dôvodu, že ak by sme mali len jedno vývojové prostredie, tak do produkčného prostredia by sa dostávali aj zmeny, ktoré tam nepatria. To sa síce dá riešiť zavedením programovacích praktík, kde by sa zmeny, určené len pre niektoré prostredia, ošetrovali podmienkami, ale to vedie ku vytváraniu zbytočne komplikovaných aplikácií, a navyše to zaberá viac času. Mať ku každému produkčnému prostrediu jedno vývojové je ešte viac časovo náročnejšie než mať len jedno. Vysoké percento často vykonávaných zmien totiž patrí do všetkých produkčných prostredí a museli by sa vykonávať zvlášť v každom vývojovom prostredí, pretože export a import z jedného vývojového prostredia na iné, by zmazalo všetky zmeny určené len pre jedno prostredie a nahradilo ich funkcionalitami a zmenami, ktoré tam nepatria. Čiže v praxi to vyzeralo tak, že po vyvinutí vo vývojovom prostredí sa zmeny zas pracne ručne vykonávali aj na produkčnom prostredí.

Nepostačujúce možnosti APEX-u pri nasadzovaní aplikácií viedli k hľadaniu podporných aplikácií od iných zdrojov. Nepodarilo sa nájsť žiadne riešenia, ktoré by vedeli porovnávať a ukázať rozdiely medzi dvoma aplikáciami, nie to ešte nasadzovať. Oracle odporúča využívať nejaký version control system (VCS), napríklad SVN (Subversion), na správu verzií vyvíjaných aplikácií. Pre prehľadné verzovanie sa dajú využiť možnosti APEXu na export aplikácií a následne pomocou Application Splittingu rozdeliť export aplikácie na jej časti. SVN by tak ukazoval len zmeny v jednotlivých častiach aplikácie, no zmeny by boli viditeľné len ako odlišné volanie procedúry určenej pre import.

Kapitola 5

Počiatočná analýza

Na pracovných poradách sme spolu so spolupracovníkmi a šéfom zo spoločnosti APEX Solutions, pre ktorú som pred niekoľkými rokmi pracoval, preberali, čo by nám pomohlo pri vývoji a uľahčilo proces nasadzovania. Všetci sme sa zhodli, že najlepšie by bolo mať nástroj, ktorý by vedel porovnať dve aplikácie a ukázal zmeny a následne by vybranú skupinu zmien nasadil na jedno kliknutie.

Keďže sa nikomu z nás nepodarilo nájsť už hotové riešenie, ktoré by riešilo náš problém, tak som sa do toho pustil sám. Nepríjemným zistením bolo, že nebola verejne dostupná žiadna dokumentácia ohľadom vnútorného fungovania prostredia Application Express. Jediná informácia ohľadom metadát a metatabuliek bola, že Oracle neodporúča priamo manipulovať s metadátami v databáze, ale používať na to volania procedúr z programového balíčka určeného na import aplikácií. K týmto procedúram som tiež dokumenáciu nenašiel, ale nehľadal som dôkladne, keďže ako som už spomenul nasadzovanie mal byť posledný krok. Tak jediný spôsob ako sa o fungovaní APEX-u niečo dozvedieť bolo prezerenie systémových tabuliek a vytvorených pohľadov, ktoré boli prospešné pre rýchlejšie pochopenie účelu a obsahu jednotlivých tabuliek. Už pri prvom nahliadnutí do exportu aplikácie bolo jasné, že funkcionalita nasadenia na jeden klik si vyžaduje obrovské množstvo práce, tak prvým cieľom bolo získať aspoň zoznam rozdielov.

Zber požiadaviek na aplikáciu prebiehal primárne na pracovných poradách. Jednou z požiadaviek bolo, že aplikácia bude vedieť zobraziť aj rozdiely medzi staršími verziami. Tým bolo jasné, že si aplikácia musí sťahovať dáta k sebe, aby si pamätala históriu verzií. Pre nedostatok dokumentácie a vedomostí ohľadom toho, čo ma čaká, sme sa rozhodli v tomto projekte postupovať iteratívne. Medzi iteráciami, prípadne po zistení dôležitých informácií, prebiehali stretnutia ohľadom ďalšieho postupu práce.

V prvej iterácii som mal za úlohu definovať množinu tabuliek, ktoré obsahovali metadáta komponentov a funkcionalít, ktoré sme v APEX-e používali. Následne vymyslieť logiku a vytvoriť databázový model, programový balík a

časť aplikácie, kde by sa tieto snapshoty dali vytvárať.

Druhá iterácia mala za cieľ nájsť rozdiely medzi dvoma snapshotmi. Obsahom tejto iterácie malo byť aj vymyslenie logiky ako bude aplikácia párovať jednotlivé komponenty z dvoch aplikácii a ako bude detekovať rozdiely v stĺpcoch obsahujúce ID. Následne doplniť databázový model a programový balík o túto funkcionalitu. Tretia iterácia blízko súvisela s druhou, keďže mala za cieľ vymyslieť ako sa tieto nájdené zmeny budú zobrazovať a doplniť aplikáciu o tieto stránky. Kvôli komplikovanému vytváraniu porovnania som sa rozhodol počas tretej fázy vykonať aj výkonnostný test, ktorý popíšem v kapitole 9.

Svoju prácu spolu s analýzou a návrhom riešenia bližšie popíšem v nasledujúcich kapitolách. Prácu v prostredí Application Express popíšem v kapitole 6, vytvorený databázový model v kapitole 7 a programové balíky v kapitole 8.

Kapitola 6

Aplikácia

V tejto kapitole popíšem ako prototyp aplikácie vyzeral po bakalárskej práci, ako som ho upravil a ako navrhujem, že by mohol vyzerať, keby sa v projekte pokračovalo.

Aplikácia sa skladá z dvoch hlavných častí, ktoré bližšie popíšem v sekciách tejto kapitoly. Prvá je Snapshots, ktorej venujem sekciu 6.2, a druhá je Comparisons, ktorej venujem sekciu 6.3.

V konečnej verzii by som navrhoval mať vytvorené aj administrátorské prostredie, kde by sa spravovali prístupy užívateľov do konkrétnych databáz a workspacov. Ale vzhľadom na to, že pre správne fungovanie tejto aplikácie je potrebná práca databázového administrátora, aby nastavil užívateľov, databázové odkazy (database links) a pridelil práva na selectovanie dát zo schémy, v ktorej sa nachádza engine nástroja Application Express, tak správa prístupov v tejto aplikácii je už len minimum, keďže sa to všetko nachádza len v dvoch tabuľkách.

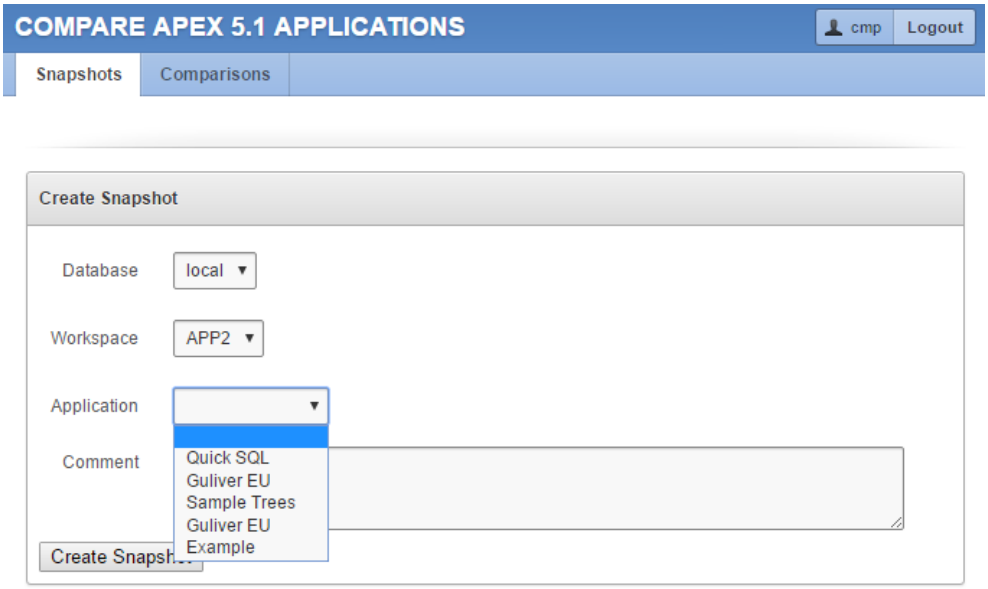
6.1 Aktualizácia na verziu 5.1

Vzhľadom na to, že APEX je spätne kompatibilný, čiže na novších verziách fungujú aplikácie vytvorené v starších, tak import prebehol bez chýb. Keďže ale táto aplikácia pristupuje k metadátam, ktoré sú uložené v schéme APEX-u, tak bolo potrebné opraviť všetky časti, ktoré obsahovali také selecty. Išlo napríklad o zoznam hodnôt (List of Values), ktorý sa nachádzaj v časti Snapshots. Oprava nebola zložitá, lebo stačilo len zmeniť názov schémy, z ktorej sa doťahovali dáta, keďže Oracle mení názov schémy podľa verzie APEX-u. V mojom prípade som teda menil schému z APEX_040200 na APEX_050100.

6.2 Snapshots

Požiadavkou bolo, aby bola aplikácia čo najjednoduchšia a vyžadovala čo najmenej klikov, tak prototyp vytvorený v bakalárskej práci bola len jedna stránka, na ktorej sa dal vytvoriť snapshot.

Ako je vidno na obrázku 6.1, tak aj zloženie stránky bolo celkom jednoduché. Užívateľ si najprv v prvom select liste vyberie databázu, kde sa nachádzala aplikácia, z ktorej chce spraviť snímku. Výberom databázy sa aktualizuje zoznam workspacov. Obsahy oboch týchto zoznamov sú obmedzené podľa nastavených práv prihlásenému užívateľovi. Zobrazia sa mu len možnosti, ku ktorým dostal prístup. Po výbere workspacu sa aktualizuje zoznam aplikácií, ktorý však už siaha priamo do tabuliek APEX-u vybranej databáze a zobrazuje aktuálny zoznam aplikácií, ktoré sa nachádzajú vo workspace.



Obr. 6.1: Ukážka aplikácie: Snapshots

6.2.1 Nedostatky

Keďže táto časť aplikácie obsahuje len jednu stránku, na ktorej sa dajú snapshoty vytvárať, tak za najväčšie nedostatky tejto časti aplikácie považujem absenciu zobrazenia zoznamu snapshotov a možnosti zobrazovať ich obsah, prípadne mazať. Ďalej mi príde nepostačujúce snapshoty idenfikovať len na základe komentáru. Keďže pri vytváraní porovnaní sa pri výbere snapshotov v select liste zbrazuje len čas vytvorenia a užívateľ, ktorý ho vytvoril, by sa dali snapshoty identifikovať lepšie na základe nejakého názvu, respektíve stručnejšieho a výstižnejšieho popisu.

6.2.2 Návrh riešenia nedostatkov

Prvý nedostatok chýbajúceho zoznamu snapshotov je možné vyriešiť jednoduchým pridaním stránky podobnej ako je úvodná stránka časti Comaprisons. V zozname snapshotov by sa samozrejme zobrazovali len snapshoty aplikácií, ku ktorým má prihlásený užívateľ nastavený prístup. Z tejto stránky by sa stala domovská stránka časti Snapshots a bolo by pridané tlačítko „Create Snapshot“, ktoré by viedlo na pôvodnú domovskú stránku, kde sa vytvárajú snapshoty.

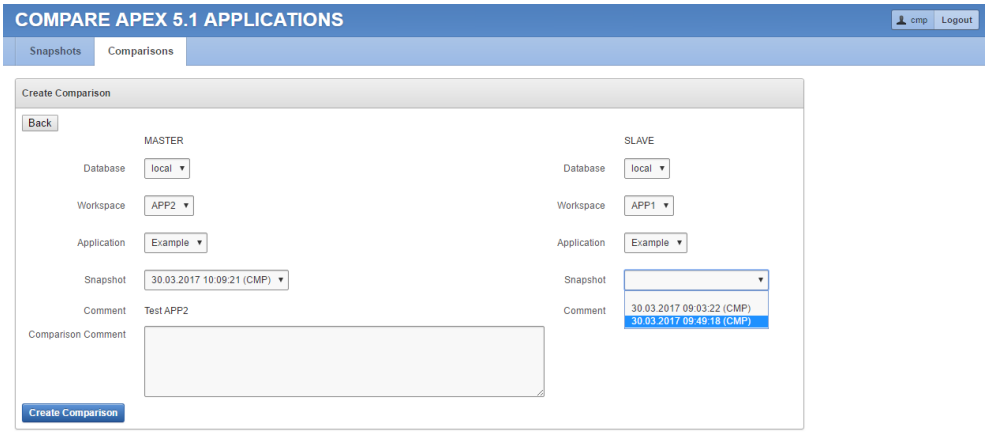
Stránku vytvárajúcu snapshoty by som rozšíril o ďalší prvok na vkladanie názvu, respektíve stručnejšieho popisu, s limitom na počet znakov, aby sa dosiahlo krajšie a čítateľnejšie zobrazovanie v select liste pri vytváraní porovnania.

Ďalšou veľmi užitočnou funkcionalitou by bolo mať možnosť si zobraziť obsah snapshotu, lebo názov a komentár nemusia dávať dobrú predstavu o obsahu a stave aplikácie. Do tabuľky by sa pridal stĺpec s odkazom na stránku, kde by bol celý obsah aplikácie zobrazený v stromovej štruktúre. V novej verzii APEX-u na to existuje typ stránky APEX Tree, ktorý som popísal v 3.3.1.2. Tento typ stránky sa vytvára hierarchickým slectom. Bolo by teda potrebné pre každú tabuľku vytvoriť jeden select, kde by sa doťahovalo ID prvku, ID rodiča a názov s odkazom. Tieto selecty by sa spojili klauzulami „UNION“ a obalili by sa hierarchickým selectom, ktorý by vytvoril požadovanú stromovú štruktúru. Navrhoval by som použiť podobnú štruktúru, ktorá je na ľavej strane v Page Designeri, ktorý popisujem v 3.3.3. Keďže tam ide len o zobrazenie obsahu stránky, tak koreňom stromu je stránka. Pre prípad mojej aplikácie by musela byť pridaná ešte jedna úroveň, čiže koreňom stromu by bola aplikácia a v prvej úrovni by boli stránky a zdieľané komponenty (Shared Components). Stránky by už ďalej mali podobný podstrom ako v Page Designeri. Každý prvok stromu, by bol zároveň odkaz, ktorý by po kliknutí otvoril dialóg obsahujúci všetky jeho nastavenia, ktoré sa v Page Designeri zobrazujú v Property Editore. Vytvorenie takéhoto dôkladného zobrazenia aplikácie by však bolo veľmi časovo náročné, keďže pre každý typ komponentu by musela byť vytvorená nová stránka a všetkým stĺpcom z databáze by museli byť nájdené odpovedajúce názvy v aplikácii, aby sa dosiahlo podobného zobrazenia. Ešte náročnejšie by bolo vytvoriť stránky tak, aby odpovedali logike Property Editora, keďže pri menení nastavení ako napríklad typu komponentu sa zobrazujú iné možnosti nastavenia. Tým by stránka obsahovala toľko prvkov, koľko má tabuľka stĺpcov a museli by im byť ponastavované podmienky kedy sa majú zobraziť alebo skryť. Navyše v prípade stĺpcov obsahujúce ID-čka by sa pre správne zobrazenie ešte muselo siahať do iných časti databáze. Pre niektoré časti by sa asi dali spraviť nejaké zjednodušenia ako zoznamy hodnôt (List of Values) alebo podobné vychytávky APEX-u, no odhadol by som vytvorenie takto detailného zobrazenia aj na sto človeko-dní.

Užitočné by mohlo byť mať možnosť mazať snapshoty. Tu však vzniká pár otázok. Prvá je, že kto by mal mať toto právo, či len administrátor alebo každý užívateľ, ale to je podľa mňa otázka určená pre konkrétnych užívateľov ako by to chceli nastaviť. Druhá otázka je, čo v prípade, že daný snapshot už je súčasťou aspoň jedného porovnania. Jednou možnosťou je, že by sa tým zmazali aj dotknuté porovnania, čo však môže mať väčší dopad než si môže užívateľ predstaviť, tak takú možnosť by som dal len administrátorovi. Ďalšou možnosťou by mohlo byť upraviť databázový model tak, aby porovnanie vedelo existovať aj bez snapshotov. Túto možnosť by som však pre pár dôvodov zamietol. Hlavným je, že aby porovnania nezaberali veľa miesta, tak sa ukladajú len nájdené zmeny, a ak by sa umožnilo mať porovnanie bez zdrojových snapshotov a mal by sa zachovať celý konext, tak by sa do porovnania museli ukladať všetky dáta.

6.3 Comparisons

Domovskou stránkou tejto časti je zoznam porovnaní s tlačítkom „Create“. Okrem informácii o tom, kto a kedy porovnanie vytvoril, obsahuje aj stav porovnania, aby užívateľ videl, či už je porovnanie vytvorené alebo sa na ňom ešte pracuje.



Obr. 6.2: Ukážka aplikácie: Create Comparison

Tlačítkom „Create“ sa užívateľ dostane na stránku, kde si vie vytvoriť porovnanie. Ako je vidieť na obrázku 6.2, tak región má dva stĺpce. Jeden pre master aplikáciu a druhý pre slave. Zvolil som také pomenovania, lebo ma nenapadli vhodnejšie. Slave aplikácia je tá, ktorá by v prípade nasadzovania bola cieľová. V každom stĺpci si užívateľ vyberie databázu, workspace a aplikáciu podobne ako pri vytváraní snapshotu. Rozdiel je ale v tom, že v aplikáciach už sú len na výber tie aplikácie, z ktorých už existujú snapshoty. Je

tam navyše ešte select list na výber konkrétneho snapshotu. Po výbere konkrétneho snapshotu sa dotiahne aj jeho komentár, aby užívateľ videl, či vybral správne. Pred vytvorením porovnania sa k nemu dá dopísať nejaká poznámka.

Po vytvorení porovnania sa dá zobraziť tabuľka, ktorej príklad vidno na obrázku 6.3, kde v každom riadku je názov tabuľky, jej alias, ktorým opisujem, čo obsahuje, počet záznamov v master aplikácii, počet záznamov v slave aplikácii, počet spárovaných záznamov a počet záznamov, v ktorých sa našiel aspoň jeden rozdiel.

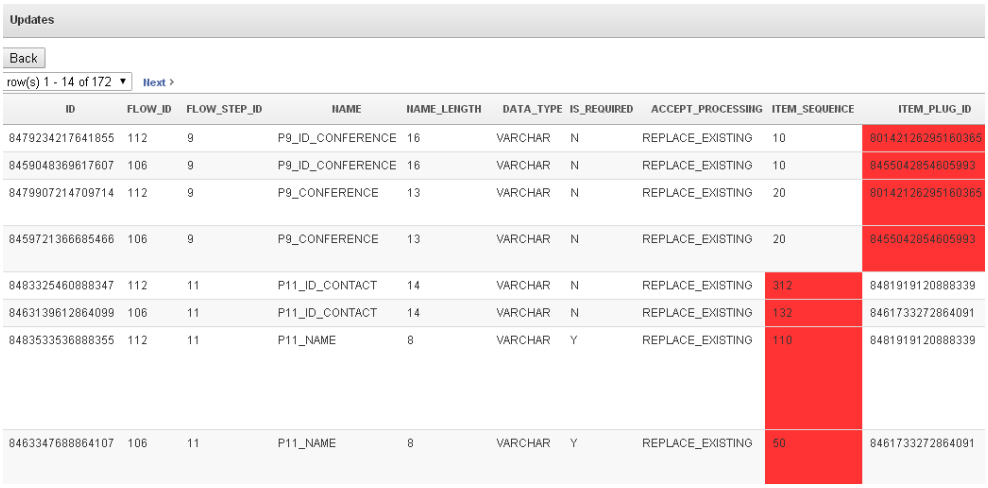
Table Name	Alias	Master	Slave	Matched	Difference
FLOW_QUERY_COLUMN	Structured Query Columns (Wizard Report)	26	26	26	0
FLOW_REPORT_LAYOUTS	Report Layouts	3	3	3	0
FLOW_SHARED_QUERIES	Report Queries	1	1	1	0
FLOW_SHARED_QRY_SQL_STMTS	Report Query Statements	1	1	1	0
FLOW_SHORTCUTS	Shortcuts	1	1	1	0
FLOW_STEPS	Pages	86	85	85	32
FLOW_PAGE_PLUGS	Regions	209	205	205	30
FLOW_REGION_UPD_RPT_COLS	Updateable Report Columns	66	60	59	14
FLOW_PAGE_DA_EVENTS	Dynamic Actions	108	102	102	7
FLOW_PAGE_DA_ACTIONS	Dynamic Action Entries	138	128	128	16
FLOW_STEP_BRANCHES	Page Branches	65	62	62	3
FLOW_STEP_BUTTONS	Page Buttons	169	167	166	9
FLOW_STEP_COMPUTATIONS	Page Computations	21	18	18	4
FLOW_STEP_ITEMS	Page Items	501	493	482	116
FLOW_STEP_ITEM_HELP	Page Item Helps	22	21	21	1

31 - 45

Obr. 6.3: Ukážka aplikácie: Zobrazenie porovnania

Každé číslo v tabuľke je zároveň odkaz na stránku, kde sa dajú zobraziť dané záznamy. Stránky sú si navzájom dosť podobné. Najviac sa líši stránka zobrazujúca rozdiely. Na nej sú tri regióny. Jeden je pre záznamy, ktoré sú len v master aplikácii, a teda pri nasadzovaní by boli pridávané ako nové. Druhý obsahuje záznamy, ktoré sú len v slave aplikácii, čiže pri nasadzovaní by boli riešené ako mazané. Tu je zas otázne či existujú procedúry na mazanie komponentov aplikácie, ale dá sa predpokladať, že áno keďže samotný APEX to musí nejak riešiť. Prípadne by sa dali konkrétne záznamy mazať priamo z tabuliek, čo však nie je odporúčaný postup z logických dôvodov. Tretí región na stránke obsahuje záznamy, v ktorých nastala zmena v aspoň jednom stĺpci. Pre lepšiu viditeľnosť zmien som dané stĺpce podfarbil červenou pomocou

JavaScriptu. Záznamy sú zobrazované po dvojiciach vždy s pravidlom, že prvý riadok je z master aplikácie a druhý zo slave. Navyše v stĺpcoch, ktoré obsahujú ID-čka by užívateľ ani nemal šancu spozorovať zmenu, keďže každá aplikácia má vlastné unkátne ID-čka, ktoré sa neprenášajú exportom tiež z logických dôvodov. Zmenu v stĺpcoch obsahujúce ID-čka má užívateľ možnosť vidieť len vďaka podfarbeniu. Príklad zmien v takom stĺpci je vidieť na obrázku 6.4.


Updates

Back

row(s) 1 - 14 of 172 ▾ Next >

ID	FLOW_ID	FLOW_STEP_ID	NAME	NAME_LENGTH	DATA_TYPE	IS_REQUIRED	ACCEPT_PROCESSING	ITEM_SEQUENCE	ITEM_PLUG_ID
8479234217641855	112	9	P9_ID_CONFERENCE	16	VARCHAR	N	REPLACE_EXISTING	10	80142126295160365
8459048369617607	106	9	P9_ID_CONFERENCE	16	VARCHAR	N	REPLACE_EXISTING	10	8455042854605993
8479907214709714	112	9	P9_CONFERENCE	13	VARCHAR	N	REPLACE_EXISTING	20	80142126295160365
8459721366685466	106	9	P9_CONFERENCE	13	VARCHAR	N	REPLACE_EXISTING	20	8455042854605993
8483325460888347	112	11	P11_ID_CONTACT	14	VARCHAR	N	REPLACE_EXISTING	312	8481919120888339
8463139612864099	106	11	P11_ID_CONTACT	14	VARCHAR	N	REPLACE_EXISTING	132	8461733272864091
8483533536888355	112	11	P11_NAME	8	VARCHAR	Y	REPLACE_EXISTING	110	8481919120888339
8463347688864107	106	11	P11_NAME	8	VARCHAR	Y	REPLACE_EXISTING	50	8461733272864091

Obr. 6.4: Ukážka aplikácie: Región Updates na stránke Differences

6.3.1 Nedostatky

Prvým veľkým nedostatkom je komplikované a neprehľadné zobrazenie porovnania. Nielenže sa zmeny zobrazujú ako obsahy systémových tabuliek, ale prezeranie rozdielov navyše potrebuje veľa preklikávaní medzi stránkami, čo je nielen nepraktické, ale dá sa povedať, že v rozpore s požiadavkou nasadzovania na čo najmenej klikov. Ďalším veľkým nedostatkom je, že takéto zobrazenie je v podstate nepoužiteľné pre prípad, že by sa chcela implementovať možnosť nasadzovania len nejakej podmnožiny nájdených rozdielov. Nedostatkom je samozrejme aj absencia možnosti nasadzovať zmeny.

Nedostatkom je aj chýbajúce automatické párovanie komponentov aplikácií, bez ktorého treba komponenty spárovať ručne alebo budú zmeny zobrazené len ako nové a vymazané. Bolo navrhnuté, že by sa definovali pre každú tabuľku stĺpce, ktoré by identifikovali komponenty. Pri analýze však boli odhalené rôzne komplikácie. Najväčšou bolo, že pre niektoré tabuľky sa to v podstate nedalo definovať jednoznačne. Ďalšou bolo, že zmeny môžu byť vykonávané aj v takých stĺpcoch a čiže aj tak by komponenty neboli spárované, ale identifikované ako nové a vymazané. Pri automatickom párovaní by taktiež hrali rolu stĺpce s ID-čkami, čo celý proces komplikuje. Na tento problém existuje pár možných riešení, ktoré popíšem v 6.3.2.

Menším nedostatkom je, že ak si užívateľ nechá vyhotoviť porovnanie, tak v tabuľke síce vidí stav, v ktorom sa vytváranie porovnania nachádza, ale nie je informovaný o dokončení automaticky. Musí sám obnovoviť stránku, aby zbadal, že sa vytváranie dokončilo. Vytváranie naštastie prebieha rádovo v sekundách.

6.3.2 Návrh riešenia nedostatkov

Nedostatok komplikovaného zobrazenia by sa dal vyriešiť podobne ako popisujem v 6.2.2 zobrazením do stromovej štruktúry pomocou APEX Tree. Listy tohto stromu by sa však líšili. V prípade nových komponentov by sa graficky alebo textovo označili ako pridané a v prípade komponentov, ktoré sa nachádzajú len v slave aplikácii, by sa označili ako odstránené. V prípade nájdenej zmeny by boli komponente pridané listy popisujúce dané zmeny hodnôt. Jedna zmena by predstavovala jeden list, aby sa vyriešil aj nedostatok nasadzovania len podmnožiny nájdených rozdielov. Týmto riešením by sa vyhovelo požiadavku nasadzovania na čo najmenej klikov, keďže po zobrazení porovnania by sa dali všetky zmeny nasadiť na jeden klik.

Absencia automatického párovania má najjednoduchšie riešenie také, že by sa to neimplementovalo vôbec a v prípade prvého nasadenia by si aplikácia uložila nové spárovanie, čo je celkom samozrejmá vlastnosť funkcionality nasadenia. Po prvom nasadení by už bolo všetko správne spárované. Ďalším riešením, ktoré som už naznačil v nedostatkoch, by mohlo byť to automatické párovanie podľa definovania stĺpcov pre každú tabuľku zvlášť, čo by nefungovalo dokonale vo všetkých prípadoch, ale aspoň nejak. Jednoduchšou formou tohto riešenia by bolo, že by to párovalo len identické záznamy, čo by dokázalo spárovať dve na vlas rovnaké aplikácie. Posledné riešenie je vytvoriť rozhranie, kde by sa dali komponenty párovať ručne. Toto riešenie by s kombináciou automatického nedokonalého párovania vytváralo komplexné riešenie. Považujem za zbytočné stratiť toľko času vývojom a následným prípadným dopárovavaním aplikácií, keď prvé spomenuté riešenie je najjednoduchšie a spáruje to dokonale po prvom nasadení.

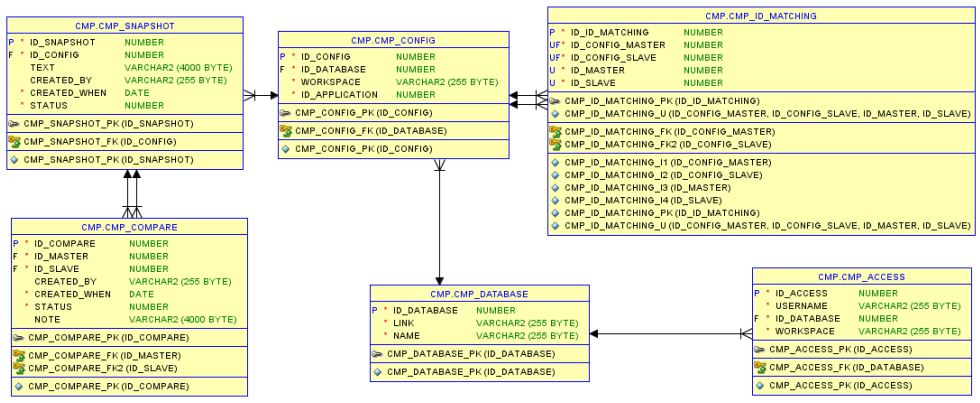
Menší nedostatok informovania užívateľa o dokončení porovnania sa dá vyriešiť novinkou asynchronných dynamických akcií, ktorá prišla s verziou 5.1 a spomínam ju v 3.3.2.

Kapitola 7

Dátový model

Dátový model sa najviac rozvinul počas prvých dvoch iterácií. Ako spomínam v 5, tak kvôli požiadavke uchovávania histórie verzií obsahuje moja aplikácia kópie systémových tabuliek, do ktorých si ukláda kópie všetkých záznamov pri vytvorení snapshotu. Pri analýze systémových tabuliek som zistil, že všetky, ktoré chcem kopírovať začínajú na predponu „WWV_“. Keďže sme aj my v práci používali podobnú praktiku pripájať trojpísmenkovú predponu k názvu tabuľky, aby bolo na prvý pohľad jasné k akej aplikácii tabuľka patrí, tak som predponu v skopírovaných systémových tabuľkách nahradil mojou „CMP_“. Každej takej tabuľke som pridal stĺpec ID_SNAPSHOT, aby sa dali identifikovať záznamy, ktoré patria snapshotu. Ide o cudzí kľúč do tabuľky CMP_SNAPSHOT.

Popíšem ako som navrhol tabuľky, ktoré nevznikli kopírovaním zo systémových tabuliek APEX-u, a pre lepšiu predstavu pripojím aj relačný model vygenerovaný nástrojom SQL Developer Data Modeler, ktorý vidno na obrázku 7.1.



Obr. 7.1: Relačný model časti databáze

Kvôli požadovanej funkcionalite vzdialeného prístupu som vytvoril tabuľku CMP_DATABASE, ktorá je veľmi jednoduchá. Ukladá sa do nej len názov databázového odkazu (database link) a názov, ktorý sa bude zobrazovať v select listoch v aplikácii. Kvôli ďalšej požiadavke, aby sa užívateľom zobrazovali len tie databázy a workspace, ku ktorým dostanú prístup som vytvoril tabuľku CMP_ACCESS. Do nej sa ukladá názov užívateľskeho účtu, ktorému sa prideľuje oprávnenie, ďalej ID_DATABASE, čo je cudzí kľuč do CMP_DATABASE, a názov workspacu, do ktorého má mať prístup. Tieto dve tabuľky sú jediné, s ktorými musí administrátor aplikácie pracovať. V prípade vytvorenia administrátorskeho rozhrania by ani do nich nemusel priamo vkladať, ale vyklikal by oprávnenia v aplikácii. Administrátor aplikácie sa však nevyhne práci v databáze. Pre nakonfigurovanie vzdialeného prístupu musí vytvoriť databázové odkazy a prideliť oprávnenie (príkaz grant select), aby táto aplikácia vedela vyťahovať dáta zo systémových tabuliek APEX-u. V prípade nastavenia prístupu do lokálnej databáze nie je potrebné vytvárať databázový link, ale je potrebné prideliť povolenie „grant select“ užívateľovi tejto aplikácie do schémy APEX_050100. Následne treba do tabuľky CMP_DATABASE do stĺpca LINK vložiť názov služby, na ktorom je databáza dostupná, keďže sa selecty generujú dynamicky so syntaxou @.

Keďže má každý snapshot svoje ID, ale chcem aby sa ukladalo párovanie ID-čiek podľa aplikácie, aby sa pri vytvorení nového snapshotu použilo už existujúce spárovanie a nemuselo sa to párovať znovu, tak som vytvoril tabuľku CMP_CONFIG, ktorá v podstate slúži len na to, aby sa každej aplikácii priradilo jedno ID. Na ňu je napojená tabuľka CMP_ID_MATCHING, ktorá už obsahuje konkrétne párovania. Porovnania sa potom vytvárajú pomocou spárovaných ID-čiek aplikácií. Toto riešenie je nielenže logické a praktické, ale ešte aj šetrí miesto, keďže sa zbytočne nevytvárajú záznamy párovania pri tvorbe nového porovnania už spárovaných aplikácií.

Posledná tabuľka z obrázku je CMP_COMPARE, kde sa ukladajú informácie ohľadom vytvorených porovnaní. Na ňu sú napojené zas kópie systémových tabuliek APEX-u, ktorým som zmenil predponu na „COM_“. Tieto tabuľky majú na rozdiel od systémových tabuliek navyše tri stĺpce. Prvý je ID_COMPARE, ktorým sa identifikujú stĺpce vytvorené porovnaním, druhý je STATEMENT, kde sa ukladá reťazec určujúci, o aký typ rozdielu ide, a stretí je DIFFERENT_COLUMNS, do ktorého sa ukladá zoznam stĺpcov, v ktorých bola nájdená zmena. Táto informácia sa potom využíva v aplikácii na podfarbenie stĺpcov, v ktorých je rozdiel.

Podstatná je aj konfiguračná tabuľka CMP_TABLE_COLUMNS, ktorú na obrázku nevidno. Ide o tabuľku, ktorá nie je nijak prepojená so zvyšnými, ale obsahuje informácie podľa ktorých sa generujú SQL príkazy vytvárajúce porovnanie. Každá tabuľka, ktorá sa má porovnať musí mať záznam v tejto tabuľke. Najpodstatnejšie sú stĺpce TABLE_NAME, ID_COLUMNS a IGNORE_COLUMNS. Do TABLE_NAME patrí názov tabuľky, ktorej sa konfigurácia týka. V ID_COLUMNS sa definujú stĺpce, ktoré obsahujú ID-

čka, a teda sa pri porovnaní použije daná logika porovnávania cez tabuľku CMP_ID_MATCHING. Do IGNORE_COLUMNS patria stĺpce, ktoré sa nemajú porovnávať, napríklad checksum alebo posledná zmena a podobne. Do tejto definície patrí aj stĺpec ID, čo je identifikátor riadku nachádzajúci sa v každej tabuľke.

Kvôli lepšiemu výkonu boli vytvorené aj pomocné tabuľky, do ktorých sa nikdy neukládajú žiadne dáta. Do nich sa vkladajú údaje len počas vytvárania porovnania, ale pred tým než prebehne commit sú zmazané, čiže sú viditeľné len pre transakciu, ktorá si ich tam dočasne vloží, ale nikdy tam neostanú. Bližšie to popisujem v 9.3.

Posledná nepopísaná tabuľka je CMP_TABLE_ID_MATCHES, kde sa pri vytvorení porovnania uložia spárované ID-čka, aby sa správne zobrazovalo porovnanie, keďže pre porovnanie je podstatné aké boli údaje pri vytváraní a nie aktuálne. Napríklad ak sa vytvorí porovnanie aplikácií, ktoré nie sú spárované, tak výsledkom bude, že všetky komponenty slave aplikácie sa majú zmazať a všetky komponenty master aplikácie sa majú vložiť aj v prípade, že sú identické. Ak by sa aplikácie dodatočne spárovali a zobrazenie by vychádzalo z aktuálnych párovaní, tak výsledok by sa nezobrazil správne.

Boli vytvorené aj dva pohľady (view). Prvý je CMP_V_COMPARES, ktorý slúži aplikácii na zobrazenie výsledku porovnania. Druhý som nazval CMP_V_ID_MATCHES a využíva sa pri vytváraní porovnania pri práci so stĺpcami obsahujúcimi ID-čka.

7.1 Aktualizácia na verziu 5.1

Pri tak rozsiahlych zmenách, ktoré nastavi v prostredí APEX, sa dalo očakávať, že sa zmení aj dátový model systémových tabuliek. Po porovnaní náhodných dvoch tabuliek som zistil, že nastali zmeny v oboch, tak som to ďalej neskúmal a nanovo som povytváral kópie tabuliek podľa logiky ako popisujem v tejto kapitole. Ide teda o kópie systémových tabuliek APEX-u, ktorým som zmenil predponu na „COM_“ a „CMP_“.

Počas prerábania týchto tabuliek som zistil, že nová verzia APEX-u neobsahuje štvoricu tabuliek, do ktorých sa ukladali dáta ohľadom Structured Query. To je typ reportu, ktorý sa vytváral pomocou sprievodcu. V podstate išlo o komplikované ukladanie selectu. Išlo o tabuľky, ktoré som v bakalárskej práci popisoval, že vytvárali komplikáciu pri vytváraní snapshotu aj pri párovaní aplikácií. Komplikácia spočívala v tom, že tri z týchto tabuliek neobsahovali napojenie na metadáta aplikácie, ale len na svoju definíciu, čiže sa s nimi nedalo pracovať tak genericky ako s ostatnými tabuľkami, ale muselo sa to spracovávať individuálne. Odstránením týchto štyroch tabuliek sa zmenšil počet kopírovaných tabuliek na 55.

Kapitola 8

Programové balíky

Všetká logika vytvárania snapshotov a porovnaní sa ukrýva v programových balíkoch (packages) databáze. Počas prvej iterácie som vytvoril balík s názvom CMP_SNAPSHOT_PKG, ktorý sa stará o vytváranie snapshotov a popíšem ho v sekcii 8.1. Počas druhej iterácie vznikol o dosť komplikovanejší balík, ktorý porovnáva 2 snapshoty a výsledok zaznamenáva do tabuliek, keďže porovnávanie aplikácii nie je možné v reálnom čase. Balík som nazval CMP_COMPARE_PKG a venujem mu sekciu 8.2.

8.1 CMP_SNAPSHOT_PKG

Tento balík má navonok viditeľnú len jednu procedúru, ktorej sa ako parameter predá ID snapshotu, ktorý sa ma vytvoriť. O vytvorenie záznamu snapshotu sa stará aplikácia a zavolá túto procedúru s vytvoreným ID.

Pomocou daného ID si procedúra dotiahne všetky potrebné informácie informácie ako databázový link a číslo aplikácie. Ako prvý skopíruje záznam o aplikácii. Následne sa volá procedúra, ktorej sa ako parameter podhodí názov tabuľky bez predpony. Táto procedúra generuje reťazec obsahujúci SQL výraz, ktorý kopíruje dáta zo systémovej tabuľky k seme. Všetky tieto volania sú generované a spúšťané pomocou „execute immediate“, kvôli dvom požiadavkám. Prvou je vzdialený prístup, čiže sa za názov tabuľky doplňuje @ a databázový link. Druhou je, že balíky mali byť vytvárané tak, aby boli jednoducho rozšíriteľné. Čiže procedúra vytvárajúca snapshot vyzerá tak, že sa v nej volá procedúra vytvárajúca porovnanie jenej tabuľky podľa podhodeného parametru toľkokrát, koľko tabuliek sa sťahuje. Jediným rozdielom bola spomínaná štvorica tabuliek, ktoré museli byť riešené samostatne. Kvôli ním bola vytvorená bezparametrová procedúra, ktorá najprv vytiahla všetky záznamy definícií a podľa nich kopírovala aj záznamy zo zvyšných troch tabuliek.

Na konci sa aktualizuje príznak stavu vytvárania porovnania, aby dal na vedomie užívateľovi, že snapshot je pripravený.

8.1.1 Aktualizácia na 5.1

Keďže provedúry siahajú to schémy APEX-u, tak bolo potrebné opraviť všetky časti kódu, ktoré generovali SQL reťazce na aktuálny názov schémy APEX_050100.

Ako popisujem v 7.1, tak v novej verzii boli odstránené práve tie štyri tabuľky, kvôli ktorým bola vytvorená najkomplikovanejšia časť tohto balíka, takže volanie procedúry, ktorá sa o to starala stačilo zakomentovať. Teraz už vyzerá tento balík pekne genericky, celkom jednoducho a je ľahko rozšíriteľný už na prvý pohľad o hocikoľko tabuliek.

8.2 CMP_COMPARE_PKG

Podobne ako prvý balík aj tento ma navonok viditeľnú len jednu procedúru, ktorej sa parametrom predá ID porovnania, ktoré sa má vytvoriť. Vo vnútri však už obsahuje omnoho komplikovanejšiu logiku. Podobne ako v prvom balíku sa kvôli rozšíriteľnosti volá opakovane jedna procedúra, ktorej sa parametrom hodí názov tabuľky.

Pre vytvorenie porovnania jednej tabuľky sa generujú rozsiahle SQL výrazy. Jednotlivé časti SQL výrazov sa vytvárajú pomocou troch funkcií. Jedna vytvára zoznam stĺpcov oddelených čiarkou pre potrebu selectovania. Tento zoznam sa vytvára pomocou systémového pohľadu ALL_TAB_COLUMNS. Druhá vytvára podmienky porovnania, ktoré sa doplnia do where klauzule. Tieto podmienky sa vytvárajú tiež pomocou spomínaného pohľadu, keďže pri porovnávaní záleží na dátovom type stĺpca. LOB stĺpce sa musia porovnávať pomocou volania systémovej funkcie, podobne aj RAW typ sa porovnáva pomocou volania funkcie, ale inej. Inak sa vytvára porovnanie dátumu a už kompletne inak sa vytvára porovnávanie stĺpcov obsahujúce ID.

Porovnávanie ID som už v tejto práci spomínal na viacerých miestach, ale popíšem ako s nimi pracuje tento balík. Najprv si z tabuľky obsahujúcej párovania ID-čiek vytiahne relevantné záznamy a uloží si ich do dočasných tabuliek. Tieto tabuľky sa potom využívajú pri spúšťaní vygenerovaných SQL výrazov.

Porovnanie jednej tabuľky prebieha v troch krokoch. V jednom sa vezmú záznamy z master aplikácie, ktoré nemajú pár v slave aplikácii a prihodí sa im príznak INSERT. V druhom zas opačne sa vezmú záznamy zo slave aplikácie, ktoré nemajú pár v master aplikácii a prihodí sa im príznak DELETE. Tretí krok je najkomplikovanejší a tam sa využívajú vygenerované reťazce určené na porovnanie, či v spárovaných záznamoch je zmena. Tieto reťazce sá generujú dva. Jeden je určený do where klauzule a druhým sa vytvára reťazec, ktorý sa plní do stĺpcov DIFFERENT_COLUMNS, ktorý sa využíva pri zobrazovaní v aplikácii.

Kapitola 9

Testovanie

K testovaniu dochádzalo priebežne počas vývoja, vždy po vytvorení nejakého celku. Neboli vytvorené žiadne automatické testy a všetko som testoval sám. Testoval som ako grafické rozhranie, tak aj funkcionality. Kvôli tomu ako sa pre prácu s ID-čkami komplikovane vytvára porovnanie aplikácií a zobrazuje jeho výsledok, keďže je potrebné všetky záznamy spájať pomocou párovacej tabuľky, vznikla otázka ohľadom výkonu. Vykonané výkonnostné testy a riešenia popíšem v sekcii 9.3.

9.1 Grafické rozhranie

Testovanie grafického rozhrania prebieha vlastne počas vývoja. Po pridaní stránky a jej prvkov sa stačí len prekliknúť do ďalšieho okna, kde máte stránku otvorenú, obnoviť ju a všetky zmeny sú okamžite vidieť a je možné skúšať všetky jej komponenty a funkcionality. V prípade mojej aplikácie grafického rozhrania neobsahuje žiadne zložité operácie, a preto nebolo potrebné žiadne väčšie testovanie. Na prvý pohľad bolo vidieť, či stránky zobrazujú to, čo majú.

9.2 Funkcionality

Všetky netriviálne funkcionality boli vytvorené ako procedúry programových balíkov, ktoré vytvárali záznamy v tabuľkách. Čiže testovanie funkcionalít spočívalo v púšťaní procedúr a kontrolou vytvorených dát. V iterácii Snapshot bolo testovanie najjednoduchšie, lebo stačilo len skontrolovať, či si aplikácia dotiahla všetky dáta zo systémových tabuliek.

V mojom riešení dynamického generovania SQL príkazov bola čitateľnosť kódu a jednoduchosť pridávania ďalších systémových tabuliek na úkor jednoduchosti ladenia programu. Keďže pri zle vykonanej zmene kódu vyhlásilo len chybu vo volaní „execute immediate“, bolo potrebné vygenerovaný reťa-

zec vypísať do logu a dané selecty následne vyladiť a nájdenú chybu opraviť v balíku.

9.3 Výkonnosť

V tretej iterácii bolo dôležité otestovať, či je vytvorené riešenie dostatočne rýchle. V tejto iterácii bolo vykonaných najviac testov zo všetkých. Testovanie sa zameriavalo na výkon aplikácie pri zobrazovaní obsahu tabuliek a vytvorení porovnania. V prípade zistenia nepostačujúceho výkonu bolo potrebné tento problém vyriešiť. Kvôli lepšiemu výkonu bola vytvorená pomocná tabuľka CMP_TABLE_ID_MATCHES, aby pomáhala k rýchlejšiemu výpočtu a pamätaniu spárovania pohľadu CMP_V_COMPARES.

Dôležitá bola aj otázka, ako sa bude spomaľovať aplikácia pri plnení tabuľky CMP_ID_MATCHING, keďže všetky porovnávané tabuľky sú spájané práve prostredníctvom tejto tabuľky. Ako záťažový test som sa rozhodol skopírovať osemstokrát záznamy o spárovaní spomínanej najzložitejšej aplikácie určenej na testovanie. To naplnilo tabuľku necelými 2,2 miliónmi záznamov. Z pôvodných okolo 20 sekúnd sa týmto naplnením tabuľky predĺžil proces vytvorenia porovnania na 17 minút. Tento výkonnostný problém som vyriešil pridaním pomocnej tabuľky CMP_ID_MATCHING_TEMP. Tam si procedúra create_compare na začiatky skopíruje všetky relevantné spárované IDčka, ktoré sa týkajú danej dvojice aplikácií a na konci záznamy po sebe zmaže. Z podobných príčin som vytvoril aj ďalšie dve pomocné tabuľky s dočasným obsahom CMP_IDS_TEMP a CMP_IDS_UPD_TEMP. Keďže „commit“ sa v celej procedúre vyskytuje len raz na konci, aby sa zaistila konzistencia dát, a dáta z pomocných dočasných tabuliek sú zmazané ešte pred jeho vykonaním, tak tieto tabuľky nikdy nebudú mať v sebe žiadne dáta (neberiem do úvahy dočasné záznamy v transakcii).

Týmto pridaním dočasných tabuliek som zaručil, že počet záznamov v párujúcej tabuľke CMP_ID_MATCHING má minimálny vplyv na dobu vytvárania porovnania. Vytvorenie porovnania spomínanej testovacej aplikácie sa podarilo dokonca stiahnuť na ešte lepších 15 sekúnd, než bolo pôvodne bez preplnenia tabuľky párujúcej záznamy.

Kapitola 10

Zhodnotenie a záver

10.1 Aktuálny stav projektu

Výsledok tejto práce je hotový prototyp aplikácie s funkcionalitami vytvorenia snímky aplikácie, porovnania dvoch aplikácií a následného zobrazenia rozdielov. Nevýhodou ale je, že pre správne zobrazenie porovnania je potrebné ručne spárovať komponenty aplikácií, keďže nebolo vytvorené automatické párovanie. V práci som popísal aké má aplikácia nedostatky a navrhol som riešenia, ktorými by aplikácia splnila všetky požiadavky. Čo sa týka možných vylepšení v časti aplikácii Snapshots, tak v prípade detailného zobrazenia aplikácie ide o kus práce, ktorý by som odhadoval aj na sto človeko-dní. Podobného rozsahu by bolo vyriešenie nedostatkov v časti Comparisons, kde však ide o podobnú náplň práce, takže po vyriešení jednej časti, by sa dalo použiť vytváranie stromovej štruktúry aj v druhej časti. Ešte väčšieho rozsahu je funkcionalita nasadzovania aplikácií, keďže aj export jednoduchej aplikácie generuje komplikované volania procedúr, ktoré nie sú zdokumentované.

10.2 Budúcnosť projektu

Tento projekt sa nedočká používaniu v praxi pre niekoľko dôvodov. Za najpodstatnejší by som považoval to, že sa Oracle už začal venovať vývoju podobného nástroja ako spomínam v 3.3.4 a dá sa očakávať, že o niekoľko rokov bude k dispozícii. Takže venovanie času vývojom aplikácie, ktorá bude v neďalekej budúcnosti prevalcovaná je zbytočné. Ďalším je, že sa vývoju aplikácií v prostredí Application Express nevenujem už pár rokov, takže v tom nemám osobný záujem. Posledným je, že záujem v pokračovaní projektu neprejavil ani človek, ktorý s návrhom tohto projektu prišiel.

Literatúra

[1] Advanced reporting and charting with oracle application express 4.0. Dostupné z: https://www.slideshare.net/RRomme/advanced-reporting-and-charting-with-oracle-application-express-40 [cit. 24. 4. 2017].

[2] Application express 5.1 new features. Dostupné z: http://www.oracle.com/technetwork/developer-tools/apex/overview/apex51-new-features-3425414.pptx [cit. 24. 4. 2017].

[3] Oracle announces oracle application express 5. Dostupné z: https://blogs.oracle.com/apex/entry/oracle_announces_oracle_application_express [cit. 17. 4. 2017].

[4] Oracle application express architecture. Dostupné z: http://www.oracle.com/technetwork/developer-tools/apex/apex-architecture-160977.ppt [cit. 24. 4. 2017].

[5] Oracle application express deployment. Dostupné z: http://www.oracle.com/technetwork/developer-tools/apex/application-express/apex-deployment-applications-1878446.html [cit. 27. 4. 2017].

[6] Oracle application express download. Dostupné z: http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html [cit. 16. 4. 2017].

[7] Oracle application express install. Dostupné z: http://www.oracle.com/technetwork/developer-tools/apex/application-express/apex-deploy-installation-1878444.html [cit. 15. 4. 2017].

[8] Oracle application express installation overview.
Dostupné z: https://docs.oracle.com/cd/E59726_01/install.50/e39144/overview.htm#HTMIG36
[cit. 24. 4. 2017].

[9] Oracle application express statement of direction.
Dostupné z: http://www.oracle.com/technetwork/developer-tools/apex/application-express/oracle-application-express-sod-1596338.pdf [cit. 23. 4. 2017].

[10] Oracle quick sql.
Dostupné z: https://apex.oracle.com/en/quicksql/ [cit. 21. 4. 2017].

[11] Oracle technology network developer day.
Dostupné z: http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html [cit. 16. 4. 2017].

[12] Sdlc - quick guide.
Dostupné z: https://www.tutorialspoint.com/sdlc/sdlc_quick_guide.htm
[cit. 7. 5. 2017].

[13] Software development life cycle.
Dostupné z: https://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm [cit. 7. 5. 2017].

[14] Twitter post na stránke pána simona greenwooda.
Dostupné z: https://twitter.com/APEXORADEV/status/806493976605900801
[cit. 25. 4. 2017].

[15] Twitter post na stránke spoločnosti explorer uk ltd.
Dostupné z: https://twitter.com/Explorerukltd/status/847362783364530176
[cit. 25. 4. 2017].

Dodatok **A**

Seznam použitých zkratek

APEX Application Express

CSS Cascading Style Sheets

DAD Database Access Descriptor

DBA Database Administrator

DDL Data Definition Language

DML Data Manipulation Language

EE Enterprise Edition

FOP Formatting Objects Processor

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

ICAP Internet Content Adaptation Protocol

ID Identifier

IDE Integrated Development Environment

IOC Initial Operational Capability

J2EE Java 2 Enterprise Edition

JDBC Java Database Connectivity

LCA Life Cycle Architecture

LCO Life Cycle Objectives

LDAP Lightweight Directory Access Protocol

ORDS Oracle REST Data Services

OTN Oracle Technology Network

PaaS Platform as a Service

PDF Portable Document Format

PL/SQL Procedural Language/Structured Query Language

QA Quality assurance

RAC Real Application Clusters

REST Representational state transfer

RUP Rational Unified Process

SDLC Software Development Life Cycle

SE Standard Edition

SE1 Standard Edition One

SoCO Source Control for Oracle

SSO Single sign-on

SVN Subversion

URL Uniform Resource Locator

VCS Version Control System

XE Express Edition

Dodatok B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text`
  - `DP_Zamecnik_Daniel_2017.pdf` ..... text práce vo formáte PDF