FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Dokumentácia projektu do predmetu *Databázové systémy*

Hotel

Dávid Bolvanský (xbolva00)
Adrián Tóth (xtotha01)

1. mája 2017

Zadanie

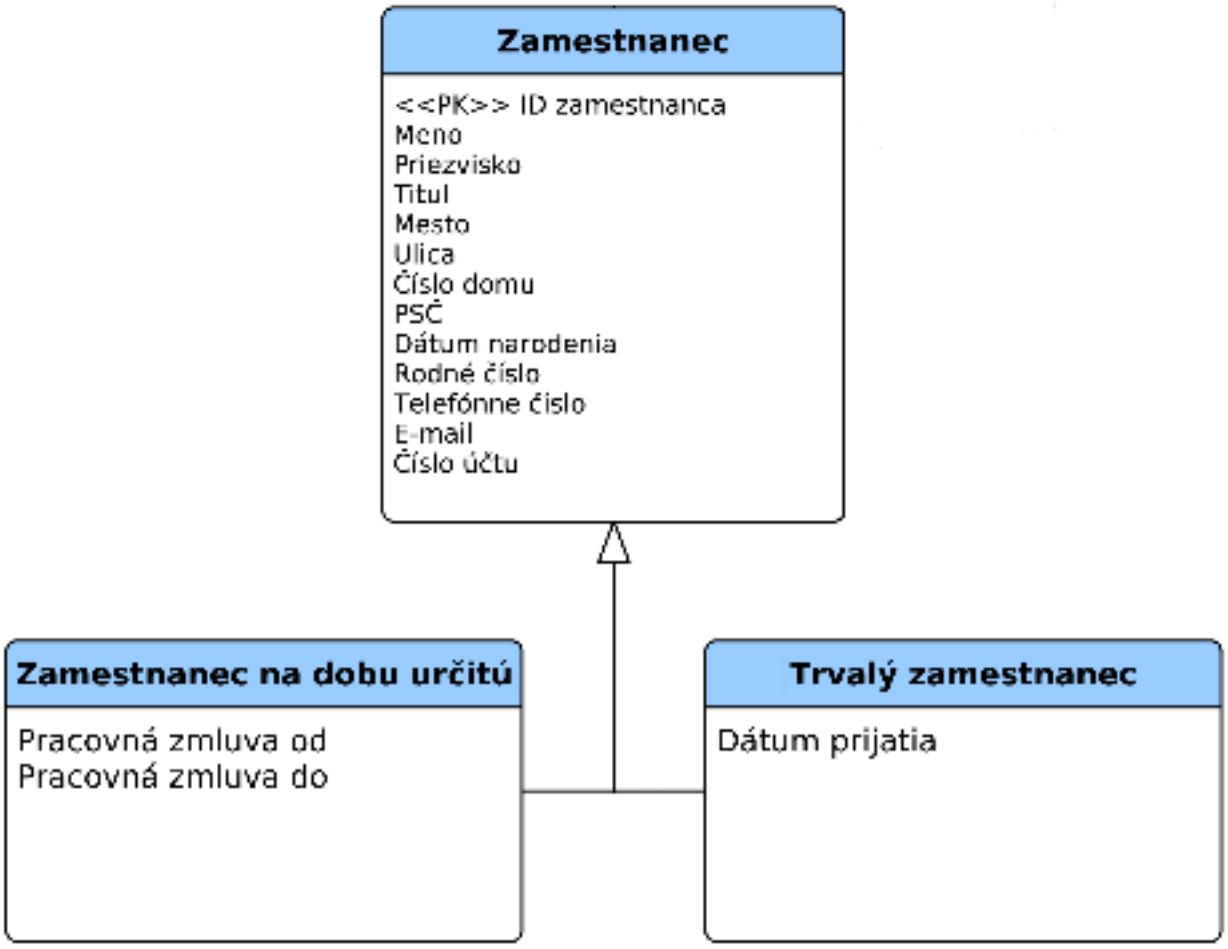
Navrhnite IS hotela, ktorý by poskytoval prehľad o dostupnosti izieb, údaje o hosťoch, ich pobytoch v hoteli, požiadavkách na služby, platby za izby, atď. Zákazníci môžu robiť rezervácie izieb (tým pádom musí zadať svoje osobné údaje). Klient si môže v rámci jednej rezervácie objednať viac izieb, napríklad aj na iný dátum. Hoteli stačí mať informácie iba o jednom klientovi, ktorý zastrešuje celý pobyt v hoteli (o ostatných účastníkoch pobyte nemusia byť dostupné žiadne informácie). Pobyt môže byť vytvorený na základe rezervácie, alebo k rezervácii izby vôbec nemusí dôjsť, ak klient príde priamo na recepcii hotela. Jednotlivé typy izieb majú rôzne ceny, cena izby sa navyše môže líšiť podľa obdobia (turistická sezóna), na ktoré si klient izbu objednáva. Cena izby sa zároveň odvíja od toho, či si hosť izbu rezervuje dopredu, alebo až na mieste. Klient si môže priobjednať k danému pobytu služby v hoteli naviac, ako napr. prenájom bazéna, a to aj viackrát. IS ďalej bude schopný evidovať skutočne absolvované pobyty – tj. ktoré izby klient od kedy do kedy obýval a aké služby skutočne využil (od x do x na ktorej izbe). Pri každom pobyte je evidované, ktorý zamestnanec prevzal platbu za pobyt.

Doplnenie zadania

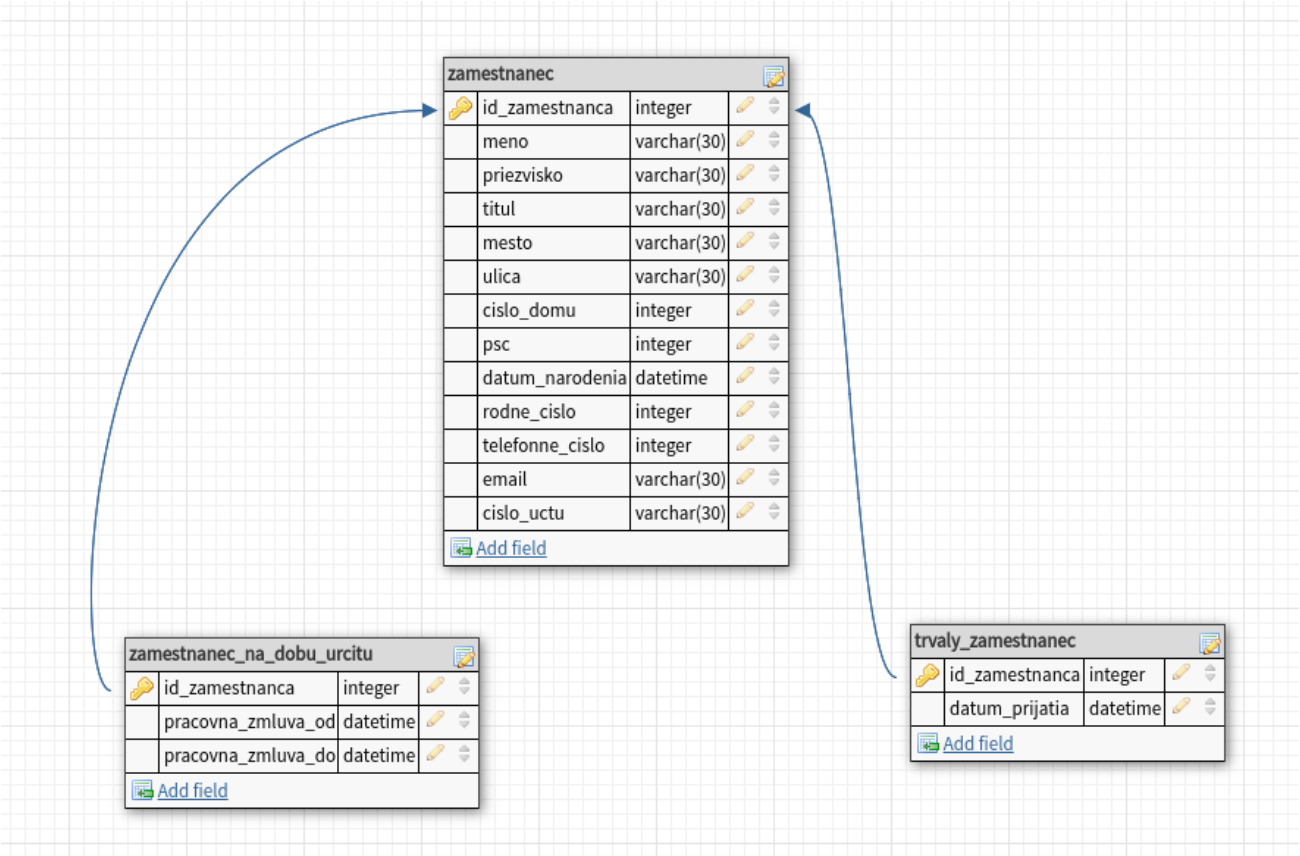
V hoteli pracujú trvalí zamestnanci, ktorí sú zamestnaní na dobu neurčitú a o ktorých je potrebné uchovávať údaj o dátume prijatia do zamestnania v hoteli, a zároveň tu pracujú aj zamestnanci, ktorí sú zamestnaní na dobu určitú a v IS si uchovávame údaj o ich pracovnej zmluve (od – do).

1 Generalizácia / špecializácia

Generalizáciu / špecializáciu sme v našom zadaní využili u zamestnanca hotela. Zamestnanec môže byť buď trvalo zamestnaný, alebo môže byť zamestnaný na dobu určitú.

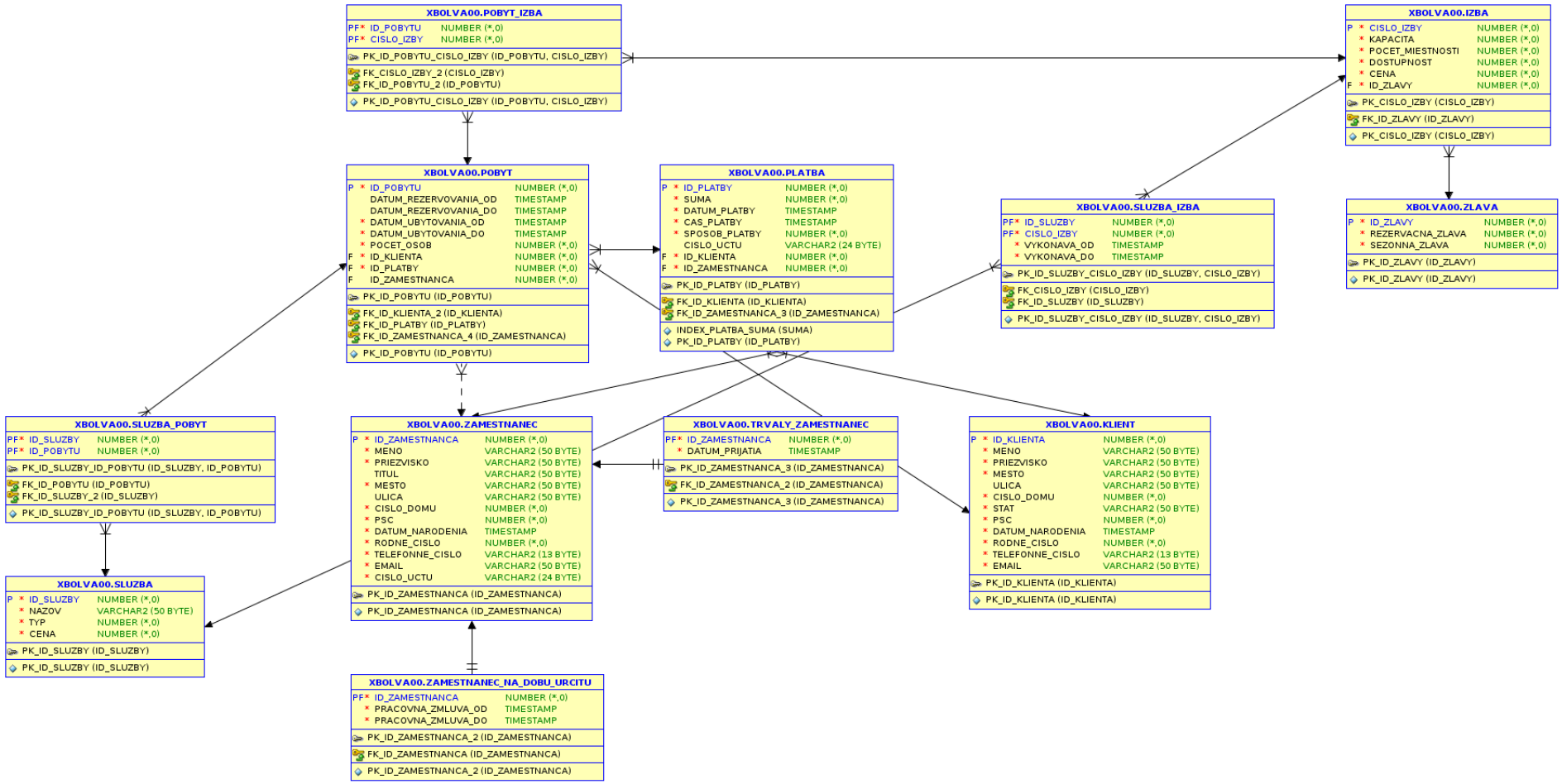


Pri transformácii tejto generalizácie/špecializácie sme vytvorili ďalšie dve tabuľky, a to ako pre trvale zamestnaného, tak aj pre dočasne zamestnaného zamestnanca hotela.



2 Schéma relačnej databázy

Nasledovná schéma popisuje finálnu schému relačnej databázy:


XBOLVA00.POBYT_IZBA
PF* ID_POBYTU NUMBER (*,0)
PF* CISLO_IZBY NUMBER (*,0)
PK_ID_POBYTU_CISLO_IZBY (ID_POBYTU, CISLO_IZBY)
FK_CISLO_IZBY_2 (CISLO_IZBY)
FK_ID_POBYTU_2 (ID_POBYTU)
PK_ID_POBYTU_CISLO_IZBY (ID_POBYTU, CISLO_IZBY)
XBOLVA00.IZBA
P * CISLO_IZBY NUMBER (*,0)
* KAPACITA NUMBER (*,0)
* POCET_MIESTNOSTI NUMBER (*,0)
* DOSTUPNOST NUMBER (*,0)
* CENA NUMBER (*,0)
F * ID_ZLAVY NUMBER (*,0)
PK_CISLO_IZBY (CISLO_IZBY)
FK_ID_ZLAVY (ID_ZLAVY)
PK_CISLO_IZBY (CISLO_IZBY)
XBOLVA00.POBYT
P * ID_POBYTU NUMBER (*,0)
DATUM_REZERVOVANIA_OD TIMESTAMP
DATUM_REZERVOVANIA_DO TIMESTAMP
* DATUM_UBYTOVANIA_OD TIMESTAMP
* DATUM_UBYTOVANIA_DO TIMESTAMP
* POCET_OSOB NUMBER (*,0)
F * ID_KLIENTA NUMBER (*,0)
F * ID_PLATBY NUMBER (*,0)
F ID_ZAMESTNANCA NUMBER (*,0)
PK_ID_POBYTU (ID_POBYTU)
FK_ID_KLIENTA_2 (ID_KLIENTA)
FK_ID_PLATBY (ID_PLATBY)
FK_ID_ZAMESTNANCA_4 (ID_ZAMESTNANCA)
PK_ID_POBYTU (ID_POBYTU)
XBOLVA00.PLATBA
P * ID_PLATBY NUMBER (*,0)
* SUMA NUMBER (*,0)
* DATUM_PLATBY TIMESTAMP
* CAS_PLATBY TIMESTAMP
* SPOSOB_PLATBY NUMBER (*,0)
CISLO_UCTU VARCHAR2 (24 BYTE)
F * ID_KLIENTA NUMBER (*,0)
F * ID_ZAMESTNANCA NUMBER (*,0)
PK_ID_PLATBY (ID_PLATBY)
FK_ID_KLIENTA (ID_KLIENTA)
FK_ID_ZAMESTNANCA_3 (ID_ZAMESTNANCA)
INDEX_PLATBA_SUMA (SUMA)
PK_ID_PLATBY (ID_PLATBY)
XBOLVA00.SLUZBA_IZBA
PF* ID_SLUZBY NUMBER (*,0)
PF* CISLO_IZBY NUMBER (*,0)
* VYKONAVA_OD TIMESTAMP
* VYKONAVA_DO TIMESTAMP
PK_ID_SLUZBY_CISLO_IZBY (ID_SLUZBY, CISLO_IZBY)
FK_CISLO_IZBY (CISLO_IZBY)
FK_ID_SLUZBY (ID_SLUZBY)
PK_ID_SLUZBY_CISLO_IZBY (ID_SLUZBY, CISLO_IZBY)
XBOLVA00.ZLAVA
P * ID_ZLAVY NUMBER (*,0)
* REZERVACNA_ZLAVA NUMBER (*,0)
* SEZONNA_ZLAVA NUMBER (*,0)
PK_ID_ZLAVY (ID_ZLAVY)
PK_ID_ZLAVY (ID_ZLAVY)
XBOLVA00.SLUZBA_POBYT
PF* ID_SLUZBY NUMBER (*,0)
PF* ID_POBYTU NUMBER (*,0)
PK_ID_SLUZBY_ID_POBYTU (ID_SLUZBY, ID_POBYTU)
FK_ID_POBYTU (ID_POBYTU)
FK_ID_SLUZBY_2 (ID_SLUZBY)
PK_ID_SLUZBY_ID_POBYTU (ID_SLUZBY, ID_POBYTU)
XBOLVA00.ZAMESTNANEC
P * ID_ZAMESTNANCA NUMBER (*,0)
* MENO VARCHAR2 (50 BYTE)
* PRIEZVISKO VARCHAR2 (50 BYTE)
TITUL VARCHAR2 (50 BYTE)
* MESTO VARCHAR2 (50 BYTE)
ULICA VARCHAR2 (50 BYTE)
* CISLO_DOMU NUMBER (*,0)
* PSC NUMBER (*,0)
* DATUM_NARODENIA TIMESTAMP
* RODNE_CISLO NUMBER (*,0)
* TELEFONNE_CISLO VARCHAR2 (13 BYTE)
* EMAIL VARCHAR2 (50 BYTE)
* CISLO_UCTU VARCHAR2 (24 BYTE)
PK_ID_ZAMESTNANCA (ID_ZAMESTNANCA)
PK_ID_ZAMESTNANCA (ID_ZAMESTNANCA)
XBOLVA00.TRVALY_ZAMESTNANEC
PF* ID_ZAMESTNANCA NUMBER (*,0)
* DATUM_PRIJATIA TIMESTAMP
PK_ID_ZAMESTNANCA_3 (ID_ZAMESTNANCA)
FK_ID_ZAMESTNANCA_2 (ID_ZAMESTNANCA)
PK_ID_ZAMESTNANCA_3 (ID_ZAMESTNANCA)
XBOLVA00.KLIENT
P * ID_KLIENTA NUMBER (*,0)
* MENO VARCHAR2 (50 BYTE)
* PRIEZVISKO VARCHAR2 (50 BYTE)
* MESTO VARCHAR2 (50 BYTE)
ULICA VARCHAR2 (50 BYTE)
* CISLO_DOMU NUMBER (*,0)
* STAT VARCHAR2 (50 BYTE)
* PSC NUMBER (*,0)
* DATUM_NARODENIA TIMESTAMP
* RODNE_CISLO NUMBER (*,0)
* TELEFONNE_CISLO VARCHAR2 (13 BYTE)
* EMAIL VARCHAR2 (50 BYTE)
PK_ID_KLIENTA (ID_KLIENTA)
PK_ID_KLIENTA (ID_KLIENTA)
XBOLVA00.SLUZBA
P * ID_SLUZBY NUMBER (*,0)
* NAZOV VARCHAR2 (50 BYTE)
* TYP NUMBER (*,0)
* CENA NUMBER (*,0)
PK_ID_SLUZBY (ID_SLUZBY)
PK_ID_SLUZBY (ID_SLUZBY)
XBOLVA00.ZAMESTNANEC_NA_DOBU_URCITU
PF* ID_ZAMESTNANCA NUMBER (*,0)
* PRACOVNA_ZMLUVA_OD TIMESTAMP
* PRACOVNA_ZMLUVA_DO TIMESTAMP
PK_ID_ZAMESTNANCA_2 (ID_ZAMESTNANCA)
FK_ID_ZAMESTNANCA (ID_ZAMESTNANCA)
PK_ID_ZAMESTNANCA_2 (ID_ZAMESTNANCA)

3 SQL skript

Skript na začiatku vyčistí všetky databázové objekty pomocou *DROP* na zabránie potenciálnym konfliktom. Následne sa vytvoria jednotlivé tabuľky, nastavia sa cudzie a primárne kľúče. Tabuľky sa potom naplnia dátami. Skript obsahuje niekoľko *SELECT* príkazov, ktoré pokrývajú požiadavky určené zadaním projektu. Taktiež skript obsahuje triggery, procedúry, materializovaný pohľad, použitie indexu s *EXPLAIN PLAN* a pridelenie prístupových práv druhému členovi tímu. Na potreby ladenia skriptu je v skripte zapnutý výstup servera.

4 Triggery

V rámci projektu sme implementovali tri databázové triggery. Prvým triggerom vyplývajúcim explicitne zo zadania projektu bol trigger na automatické generovanie hodnôt primárneho kľúča – *auto_inkrementacia_cisla_izby*. V našom prípade sme tento trigger použili s tabuľkou *IZBA*. Trigger sa realizoval pomocou sekvencie kvôli uchovaniu posledného čísla použitého ako hodnota primárneho kľúča. Sekvencia začína od čísla 200 a inkrementuje sa po 1.

Ďalší trigger kontroluje správnosť rodného čísla klienta – *kontrola_rodneho_cisla_klienta*. Trigger overuje počet číslic a správnosť dátumu v rodnom čísle. Ďalej rodné číslo musí byť deliteľné číslom 11, na túto kontrolu sme použili funkciu *MOD* na výpočet modula, zvyšku po delení. Taktiež sa kontroluje u rodného čísla s počtom číslic 9, či neobsahuje neprípustnú koncovku 000.

Tretí trigger je automaticky prevádza sumu platby za pobyt z českých korún na eurá (uvažuje sa kurz 27 pre jednoduchosť) – *prevod_na_eura*. Všetky tieto triggery majú príznak *BEFORE INSERT ON*, čo znamená, že sa spúšťajú pred vložením dát do určitej tabuľky.

5 Procedúry

Súčasťou skriptu sú aj tri procedúry. Všetky procedúry obsahujú v sebe použitie kurzora a v každej procedúre bola použitá premenná s dátovým typom odkazujúcim sa na riadok či typ stĺpca tabuľky (*table_name.column_name%TYPE* alebo *table_name%ROWTYPE*). Prvá procedúra výtvara štatistiku o zastúpení klientov z istého mesta v porovnaní s celou klientelou hotela – *zastupenie_mesta_v_klientele*. Procedúra má jeden parameter a ním je práve názov mesta. Súčasťou výpisu je celkový počet klientov, počet klientov z daného mesta a počet klientov z tohto mesta v pomere s celkovým počtom klientov. Tento pomer je vyjadrený v percentách. V procedúre je ošetrený stav delenia nulou, čo v tomto našom prípade znamená, že hotel nemá žiadnych klientov a tento fakt sa pri dosiahnutí výnimky vypíše na výstup. Pri ostatných chybách je vypísané chybové hlásenie s kódom -20005.

Ďalšia procedúra kontroluje správnosť formátu emailovej adresy zamestnancov – *kontrola_emailov_zamestnancov*. Ak tento formát nespĺňa formát *meno@domena*, na výstup je výpisané meno a priezvisko zamestnanca s chybne zadanou emailovou adresou. Ak dôjde k výnimke, je vypísané chybové hlásenie s kódom -20006.

Posledná procedúra informuje o počte voľných izieb vzhľadom na ich kapacity – *volne_izby_podla_kapacity*. Pre zjednodušenie uvažujeme, že hotel má izby s kapacitou od 1 po 4. Na výstupe sú informácie, koľko je voľných izieb na hoteli pre jednotlivé kapacity. Chybové hlásenie s kódom -20007 je vypísané v prípade chyby/výnimky počas analýzy dát v tabuľkách.

6 Materializovaný pohľad

Našou úlohou bolo implementovať materializovaný pohľad patriaci druhému členovi tímu, ktorý používa tabuľky prvého člena tímu pomocou zápisu *login.table*. Bolo potrebné vytvoriť materializované záznamy (logy) obsahujúce zmeny hlavnej tabuľky, ktoré slúžia na to, aby bolo možné používať rýchlu obnovu po potvrdení zmien namiesto kompletnej obnovy, ktorá by vyžadovala spúšťať celý dotaz materializovaného pohľadu, čo by trvalo dlhšie. Následne sme vytvorili samotný materializovaný pohľad, ktorý sa týkal *SELECT* dotazu s prirodzeným spojením tabuliek *POBYT* a *KLIENT* a slúžil na výpis rodného čísla klienta, a dátumov odkedy a dokedy je ubytovaný na hoteli.

Nastavili sme nasledovné možnosti materializovaného pohľadu:
CACHE – optimalizácia čítania z pohľadu
BUILD IMMEDIATE – naplnenie pohľadu po jeho vytvorení
REFRESH FAST ON COMMIT – aktualizácia pohľadov podla záznamov po potvrdení zmien v tabulkách
ENABLE QUERY REWRITE – použitie materializovaného pohľadu optimalizátorom

Pomocou *EXPLAIN PLAN* sme požiadali o vysvetlenie dotazu, čím sme získali prehľad o operáciach vykonávaných počas dotazu obsahujúceho prirodzené spojenie týchto tabuliek.

```
-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  NESTED LOOPS                 |               |
|   2 |   NESTED LOOPS                |               |
|   3 |    TABLE ACCESS FULL          | POBYT         |
|   4 |    INDEX UNIQUE SCAN          | PK_ID_KLIENTA |
|   5 |   TABLE ACCESS BY INDEX ROWID| KLIENT        |
-----------------------------------------------------
```

Povolili sme možnosť *QUERY REWRITE* pre optimalizátor pomocou *ALTER SESSION SET query_rewrite_enabled = TRUE* a znovu sme použili EXPLAIN PLAN na získanie prehľadu o operáciach v dotaze. V tomto získanom prehľade je možné vidieť, že došlo k použitiu materializovaného pohľadu. Optimalizátor bude používať materializovaný pohľad pri vyhodnocovaní dotazu, na ktorom je tento pohľad založený.

```
------------------------------------------------------------
| Id  | Operation                     | Name               |
------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| POHLAD_KLIENT_POBYT |
------------------------------------------------------------
```

7 *EXPLAIN PLAN* a vytvorenie indexu

EXPLAIN PLAN nám slúži na vysvetlenie dotazu databázou, čiže získame plán ako databáza spracováva daný dotaz. *EXPLAIN PLAN* sme použili na *SELECT* dotaz, ktorým získame meno, priezvisko, rodné číslo a počet súm platieb jednotlivých klientov, kde platí, že suma platby za pobyt bola v rozmedzí od 1000 do 1500 českých korún.

EXPLAIN PLAN pre tento dotaz bez využitia indexu vyzerá nasledovne:

```
-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     3 |   318 |     4  (25)| 00:00:01 |
|   1 |  HASH GROUP BY                |               |     3 |   318 |     4  (25)| 00:00:01 |
|   2 |   NESTED LOOPS                |               |     3 |   318 |     3   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |               |     3 |   318 |     3   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL         | PLATBA        |     3 |    78 |     3   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN         | PK_ID_KLIENTA |     1 |       |     0   (0)| 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID| KLIENT        |     1 |    80 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

Rozbor tohto získaného plánu:
SELECT STATEMENT znamená, že sa uskutočnil *SELECT* dotaz. *HASH GROUP BY* značí zoskupovanie položiek podľa hashovacieho kľúča. Ďalej máme dvakrát *NESTED LOOPS*, čo reprezentuje samotné spojenie, kde pre každú položku prvej tabuľky sa prejdú všetky riadky z druhej tabuľky. Nasleduje *TABLE ACCESS FULL*, čo značí prechod celou tabuľkou od začiatku bez použitia indexov. *TABLE UNIQUE SCAN* reprezentuje prístup k tabuľkám cez B–strom, kde získame jeden jedinečný riadok podla primárneho kľúča v tabuľke *KLIENT*.

Následne sme vytvorili index pomocou *CREATE INDEX index_platba_suma ON platba(suma)* a znovu spustili *EXPLAIN PLAN*:

```
-----------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                   |     3 |   318 |     3  (34)| 00:00:01 |
|   1 |  HASH GROUP BY                           |                   |     3 |   318 |     3  (34)| 00:00:01 |
|   2 |   NESTED LOOPS                           |                   |     3 |   318 |     2   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                          |                   |     3 |   318 |     2   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID BATCHED  | PLATBA            |     3 |    78 |     2   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN                    | INDEX_PLATBA_SUMA |     3 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX UNIQUE SCAN                    | PK_ID_KLIENTA     |     1 |       |     0   (0)| 00:00:01 |
|   7 |    TABLE ACCESS BY INDEX ROWID           | KLIENT            |     1 |    80 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------
```

Ako je možné vidieť z výsledného plánu po použití indexu, cena (cost) sa znížila. Plán s použití indexu obsahuje informáciu, že na tabuľke *PLATBA* sa vykonával *TABLE ACCCES BY INDEX ROWID BATCHED*, kedy sa pristupuje do tabuľky cez konkrétny riadok, čo znamená, že databáza použila náš index.

8 Pridelenie prístupových práv

Druhý člen tímu má pridelené práva na úrovni bežného zamestnanca. Znamená to, že nemá práva na úpravu tabuliek s dátami o zamestnancoch, službách a zľavách. Taktiež nemôže použiť procedúru na kontrolu emailov jednotlivých zamestnancov. K ostatným tabuľkám, ako sú napríklad tabuľky s platbami, pobytmi, klientami a izbami, má zamestnanec plný prístup a taktiež môže používať zvyšné dve procedúry. Pridelenie prístupových práv k tabuľkám je realizované

pomocou *GRANT ALL ON [table] TO [user]*, k materializovanému pohľadu pomocou *GRANT ALL ON [materialized view] TO [user]* a k procedúram pomocou *GRANT EXECUTE ON [procedure] TO [user]*. Druhý člen tímu sa pripojí k databáze prvého člena tímu pomocou *ALTER SESSION SET CURRENT_SCHEMA = [schema name]*, kde *schema name* je prihlasovacie meno prvého člena tímu.

9 Zhodnotenie

Skript sme vypracovali v nástroji *SQL Developer*, v prostredí *Oracle* na školskom serveri *Oracle 12c*. Informácie na vypracovanie projektu sme čerpali z materiálov k predmetu *IDS*, kladne hodnotíme aj demonštračné cvičenia, ktoré nám viac ozrejmili jednotlivé časti projektu. Čerpali sme aj z oficiálnej *Oracle* dokumentácie a preštudovali sme si súvisiace články na internete k našej problematike.