**Ekonomická univerzita v Bratislave**

**Fakulta hospodárskej informatiky**

**Manažérske riadenie a informačné technológie**

**Dolnozemská cesta 1**

**852 35 Bratislava**

**Pokročilé využívanie databáz**

**(Cestovná kancelária)**

Roman Teličák Akademický rok 2013/2014

Obsah

[Obsah 2](#_Toc377065337)

[Úvod 3](#_Toc377065338)

[Konceptuálny model: 4](#_Toc377065339)

[Jednoduchý program v jazyku PLS/SQL 5](#_Toc377065340)

[Tvorba tabuľky faktov 6](#_Toc377065341)

[Tvorba základných programových modulov 14](#_Toc377065342)

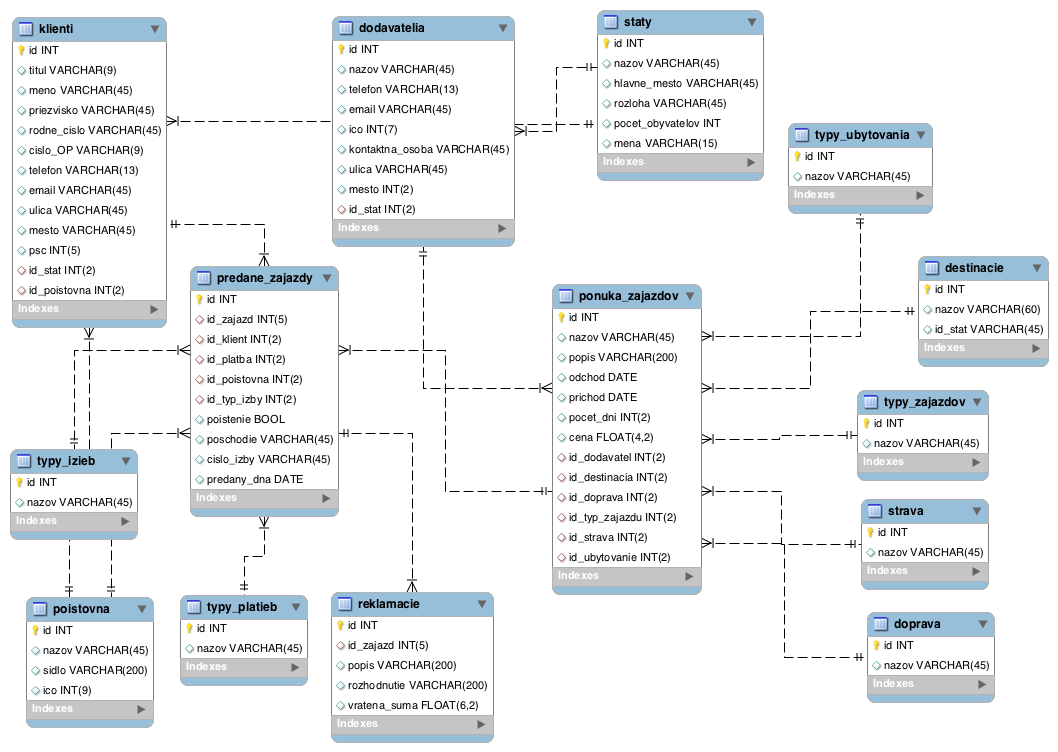
[Prepojenie databázy s webovou aplikáciou 20](#_Toc377065343)

[Prílohy 21](#_Toc377065344)

Úvod

Cestovná kancelária Travaller je sprostredkovateľom rôznych tipov zájazdov a pobytov. Ako taká neponúka vlastné produkty konečnému klientovi, ale sprostredkúva zájazdy od najrôznejších cestovných kancelárií, či už slovenkých alebo zahraničných. Za každý takýcto predaný produkt dostane Traveller províziu vo výške piatich percent z ceny zájazdu. Klienti si u nás zariadia všetky náležitosti pobytu, ako je výber destinácie, typy dopravy, stravy, ubytovania, poistenia ... a my tento pobyt objednáme u dodávateľa, ktorému týmto ušetríme administratívnu prácu.

Konceptuálny model:



Na dáta existuje viac úrovni pohľadu. Najvyššia úroveň je úroveň analytických konceptov, kam patri aj konceptuálna schéma a teda konceptuálny dátovy model. Nižšia úroveň pohľadu je úroveň implementačných konceptov kam patrí databázové schema. Konceptuálny pohľad sa zaoberá modelovaním reality a nie je ovplyvnená budúcimi prostriedkami riešenia (E-R model, Class diagram). Je najvyššou úrovňou abstrakcie.

Jednoduchý program v jazyku PLS/SQL

V tomto zadaní som vytvoril jednoduchý program v jazyku PL/SQL, ktorýdo entity T1(a,b,c,D,odD, x1,x2) vloží čísla a (amin, amax), b (bmin, bmax), c (cmin, cmax) vypočíta Diskritminant a odmocninu z Diskriminantu a hodnoty riešenia rovnice x1,x2.

SET SERVEROUTPUT ON;

DECLARE

aRand number;

bRand number;

cRand number;

discr float;

x1 float;

x2 float;

BEGIN

DELETE FROM T1;

FOR id IN

(

SELECT \* FROM T2

)

LOOP

aRand := ROUND(DBMS\_RANDOM.VALUE(id.amin,id.amax));

bRand := ROUND(DBMS\_RANDOM.VALUE(id.bmin,id.bmax));

cRand := ROUND(DBMS\_RANDOM.VALUE(id.cmin,id.cmax));

discr := (bRand\*brand)-4\*arand\*crand;

IF discr < 0 THEN

x1 := 0;

x2 := 0;

ELSE

x1 := ROUND((-brand+SQRT( discr ))/2\*arand,2);

x2 := ROUND((-brand-SQRT( discr ))/2\*arand,2);

END IF;

INSERT INTO T1 VALUES(arand,brand,crand,discr,SQRT( discr ),x1,x2);

DBMS\_OUTPUT.PUT\_LINE(arand|| ' ' ||brand|| ' ' ||crand|| ' ' ||discr|| ' ' ||x1|| ' ' ||x2);

END LOOP;

COMMIT;

END;

Tvorba tabuľky faktov

V tomto zadaní som vytvoril niekoľko sql dotazov do databázy cestovnej kancelárie (štruktúra je popísaná vyššie v tomto dokument), ktorými som simuloval pozíciu zamestnanca, ktorý chce na základe týchto výsledkov robiť strategické rozhodnutia v riadení podniku. Databáza pozostáva z tabuliek faktov a tabiliek dimenzií. Záznamy do tabuľky dimenzií boli vložené ručne, sú autentické. Záznamy v tabuľke faktov boli vygenerované pomocou programu Microsoft Excel z dát uložených v tabuľkách dimenzií.

**Vytvorte dotaz, ktorý vygeneruje tabuľku faktov obsahujúcu údaje o predaných pobytoch (názov, destinácia, príchod, odchod, cena ...), klientoch, ktoré si tieto pobyty zakúpili, od akého dodávateľa bol zájazd kúpený, v akej poisťovni bol pobyt poistený ...**

CREATE VIEW `pobyty`

AS SELECT

klienti.meno AS klient\_meno, klienti.priezvisko AS klient\_priezvisko,ponuka\_zajazdov.nazov, ponuka\_zajazdov.odchod, ponuka\_zajazdov.prichod, ponuka\_zajazdov.pocet\_dni, ponuka\_zajazdov.cena,

typy\_ubytovania.nazov AS typ\_ubytovania,

dodavatelia.nazov AS dodavatel,

poistovna.nazov AS poistovna,

typy\_zajazdov.nazov AS typ\_zajazdu,

destinacie.nazov AS destinacia,

staty.nazov AS stat

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

JOIN dodavatelia ON ponuka\_zajazdov.id\_dodavatel = dodavatelia.id

JOIN destinacie ON ponuka\_zajazdov.id\_destinacia = destinacie.id

JOIN staty ON destinacie.id\_stat = staty.id

JOIN klienti ON predane\_zajazdy.id\_klient = klienti.id

JOIN typy\_ubytovania ON ponuka\_zajazdov.id\_ubytovanie = typy\_ubytovania.id

JOIN poistovna ON predane\_zajazdy.id\_poistovna = poistovna.id

JOIN typy\_zajazdov ON ponuka\_zajazdov.id\_typ\_zajazdu = typy\_zajazdov.id

ORDER BY predane\_zajazdy.id



**Vytvorte dotaz na prehľad predaja / počet, príjem/ za jednotlivé roky**

SELECT

EXTRACT(year FROM predany\_dna) AS rok,

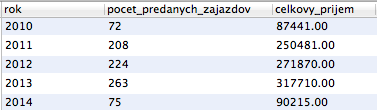
COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

GROUP BY EXTRACT(year FROM predany\_dna)



**Vytvorte dotaz na počet a celkovú sumu predaných zájazdov za rok 2012**

SELECT

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

WHERE predany\_dna <= '2012-12-31' AND predany\_dna >= '2012-01-01'

Macintosh HD:Users:rmn:Desktop:4.png

**Vytvorte dotaz na počet a celková suma predaných zájazdov za februar 2013**

SELECT

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

WHERE predany\_dna >= '2013-01-30' AND predany\_dna <= '2013-02-28'

Macintosh HD:Users:rmn:Desktop:pvd shots:5.png

**Vytvorte dotaz na počet a celková suma predaných zájazdov za 26. februar 2013**

SELECT

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

WHERE predany\_dna = '2013-02-26'

Macintosh HD:Users:rmn:Desktop:pvd shots:6.png

**Vytvorte dotazy na prehľad predaja za jednotlivé destinacie**

SELECT

destinacie.nazov,

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

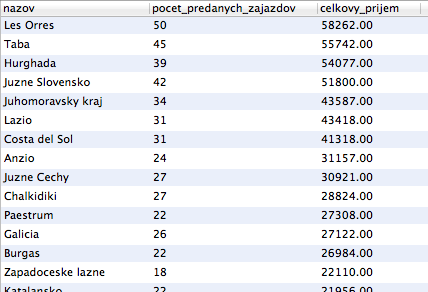
JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

JOIN destinacie ON ponuka\_zajazdov.id\_destinacia = destinacie.id

JOIN staty ON destinacie.id\_stat = staty.id

GROUP BY destinacie.nazov

ORDER BY SUM(ponuka\_zajazdov.cena) DESC



**Vytvorte dotazy na prehľad predaja za jednotlivé krajiny**

SELECT

staty.nazov,

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

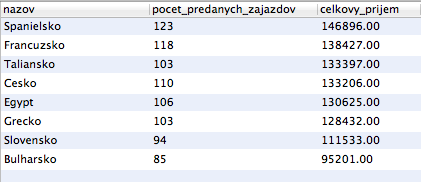
JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

JOIN destinacie ON ponuka\_zajazdov.id\_destinacia = destinacie.id

JOIN staty ON destinacie.id\_stat = staty.id

GROUP BY staty.nazov

ORDER BY SUM(ponuka\_zajazdov.cena) DESC



**Vytvorte dotazy na prehľad predaja jednotlivých typov zajazdov**

SELECT

typy\_zajazdov.nazov,

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

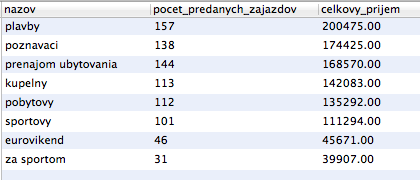
JOIN destinacie ON ponuka\_zajazdov.id\_destinacia = destinacie.id

JOIN staty ON destinacie.id\_stat = staty.id

JOIN typy\_zajazdov ON ponuka\_zajazdov.id\_typ\_zajazdu = typy\_zajazdov.id

GROUP BY typy\_zajazdov.nazov

ORDER BY SUM(ponuka\_zajazdov.cena) DESC



**Vytvorte dotazy na prehľad predaja pobytov podľa jednotlivých typov stravy**

SELECT

strava.nazov,

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

JOIN destinacie ON ponuka\_zajazdov.id\_destinacia = destinacie.id

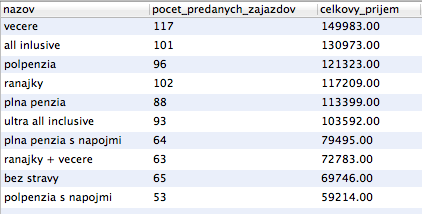
JOIN staty ON destinacie.id\_stat = staty.id

JOIN typy\_zajazdov ON ponuka\_zajazdov.id\_typ\_zajazdu = typy\_zajazdov.id

JOIN strava ON ponuka\_zajazdov.id\_strava = strava.id

GROUP BY strava.nazov

ORDER BY SUM(ponuka\_zajazdov.cena) DESC



**Vytvorte dotazy na prehľad predaja pobytov podľa využitých poisťovní**

SELECT

poistovna.nazov,

COUNT(predane\_zajazdy.id) AS pocet\_predanych\_zajazdov,

SUM(ponuka\_zajazdov.cena) AS celkovy\_prijem

FROM predane\_zajazdy

JOIN ponuka\_zajazdov ON predane\_zajazdy.id\_zajazd = ponuka\_zajazdov.id

JOIN destinacie ON ponuka\_zajazdov.id\_destinacia = destinacie.id

JOIN staty ON destinacie.id\_stat = staty.id

JOIN typy\_zajazdov ON ponuka\_zajazdov.id\_typ\_zajazdu = typy\_zajazdov.id

JOIN poistovna ON predane\_zajazdy.id\_poistovna = poistovna.id

GROUP BY poistovna.nazov

ORDER BY SUM(ponuka\_zajazdov.cena) DESC



Tvorba základných programových modulov

**Vytvorte 1-3 trigre, ktoré budú kontrolovať správnosť zadávaných údajov**

/\* tento trigger zabezpeci, ze aj ked do tabulky FAKTURA budeme \*/

/\* chciet vlozit zaznam s prazdnym datumom objednania, nastavi \*/

/\* sa nam na aktualny den \*/

DELIMITER $$

CREATE TRIGGER faktura\_skontroluj\_datum

BEFORE INSERT ON faktura

FOR EACH ROW BEGIN

IF NEW.datum\_objednania = null THEN

SET NEW.datum\_objednania = CURDATE();

END IF;

END$$

DELIMITER ;

/\* dany trigger skontroluje vkladane data, ak ich hodnoty \*/

/\* su zaporne, vymeni ich za hodnoty nulove, podla ktorych sa \*/

/\* nasledne daju dohladat nekorektne zaznamy \*/

DELIMITER $$

CREATE TRIGGER objednane\_skontroluj\_data

BEFORE INSERT ON objednane

FOR EACH ROW BEGIN

IF NEW.id\_faktura < 0 THEN

SET NEW.datum\_objednania = 0;

END IF;

IF NEW.id\_vyr < 0 THEN

SET NEW.id\_vyr = 0;

END IF;

IF NEW.id\_faktura < 0 THEN

SET NEW.mnozstvo = 0;

END IF;

END$$

DELIMITER ;

/\* predpokladajme, ze si chceme udrzat aj povodne data o odberateloch \*/

/\* tento trigger zabezpeci, ze ked sa updatne zaznam o odberatelovi, \*/

/\* povodne hodnoty sa prenesu do tabulky ODBERATEL\_ZALOHA \*/

/\* predpokladame, ze tabulka ODBERATEL\_ZALOHA ma rovnaku strukturu ako tabulka ODBERATEL \*/

/\* plus obsahuje stlpec ZMENENE, ktory je typu date \*/

DELIMITER $$

CREATE TRIGGER update\_odberatel

BEFORE UPDATE ON odberatel

FOR EACH ROW BEGIN

INSERT INTO odberatel\_zaloha

SET action = 'update',

id\_odberatel = OLD.c\_odb,

ico\_rc = OLD.ico\_rc,

nazov = OLD.nazov,

mesto = OLD.mesto,

ulica = OLD.ulica,

cislo = OLD.cd,

zmenene = NOW();

END$$

DELIMITER ;

**Vytvorte 1-3 jednoduché procedúry**

/\* Procedura pre zistenie mnozstva vyrobkov na sklade podla id vyrobku \*/

DROP PROCEDURE IF EXISTS getQuantity;

DELIMITER //

CREATE PROCEDURE getQuantity( IN itemID INTEGER )

BEGIN

SELECT vyroba.mnozstvo FROM vyroba

WHERE vyroba.id\_mat=itemID;

END//

DELIMITER ;

/\* Procedura pre vypocet ceny vyrobku s DPH na zaklade zadanej sadzby DPH \*/

DROP PROCEDURE IF EXISTS getProductPriceWithTax;

DELIMITER //

CREATE PROCEDURE getProductPriceWithTax( IN itemID INTEGER, IN taxRate INTEGER )

BEGIN

SELECT vyrobky.nazov\_vyr AS Nazov, SUM(vyrobky.cena\_v\*(1+(taxRate/100))) FROM vyrobky

WHERE vyrobky.id\_vyr=itemID;

END//

DELIMITER ;

/\* Procedura pre vypisanie faktur po splatnosti \*/

DROP PROCEDURE IF EXISTS getOverDueInvoices;

DELIMITER //

CREATE PROCEDURE getOverDueInvoices()

BEGIN

SELECT faktura.id\_faktura AS 'ID objednavky', odberatel.nazov AS Odberatel,

DATE\_ADD( faktura.datum\_vystavenia, INTERVAL 14 DAY) AS Splatnost

FROM faktura

JOIN odberatel ON odberatel.c\_odb = faktura.id\_odberatel

WHERE faktura.datum\_vystavenia <= DATE\_ADD( faktura.datum\_vystavenia, INTERVAL 14 DAY );

END//

DELIMITER ;

**Vytvorte 1-3 jednoduché funkcie**

/\* funckcia zaokruhli zadanu sumu ak dostane ako parameter kladne cislo \*/

/\* inak vrati 0 \*/

DROP FUNCTION IF EXISTS zaokruhli\_sumu;

DELIMITER //

CREATE FUNCTION zaokruhli\_sumu(suma FLOAT)

RETURNS INT

BEGIN

IF suma > 0 THEN

RETURN ROUND(suma, 0);

END IF;

RETURN 0;

END //

DELIMITER ;

DELIMITER //

CREATE FUNCTION addTaxToPrice(price FLOAT(5,2))

RETURNS FLOAT(6,2)

BEGIN

declare outPrice FLOAT(6,2);

SET outPrice = price\*(1.2);

RETURN outPrice;

END //

DELIMITER ;

**Vytvorte aspoň 1-3 transakcie, ktoré budú obsahovať priebežné body, body návratu**

/\* demonstracia pouzitia priebeznych bodov navratu \*/

CREATE TABLE t\_test (id INT NOT NULL PRIMARY KEY) ENGINE=InnoDB;

START TRANSACTION;

INSERT

INTO t\_test

VALUES (1);

SELECT \*

FROM t\_test;

id

---

1

SAVEPOINT tran2;

INSERT

INTO t\_test

VALUES (2);

SELECT \*

FROM t\_test;

id

---

1

2

ROLLBACK TO tran2;

SELECT \*

FROM t\_test;

id

---

1

ROLLBACK;

SELECT \*

FROM t\_test;

id

---

**Trigger pre tabulku t1**

CREATE TABLE IF NOT EXISTS t1(a INT(3),b FLOAT(5,2));

/\* ak vkladane cislo do stlpca je delitelne piatimi, \*/

/\* do stlpca b sa zapise hodnota 100 \*/

DELIMITER //

CREATE TRIGGER devideFive

BEFORE INSERT ON t1

FOR EACH ROW BEGIN

IF (NEW.a % 5) = 0 THEN

SET NEW.b = 100;

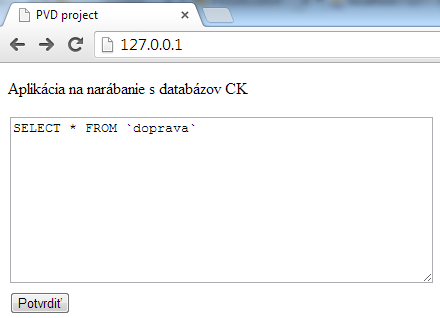
END IF;

END//

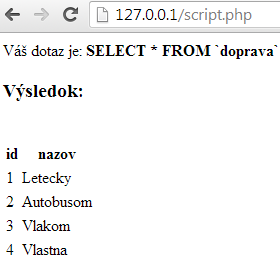
Prepojenie databázy s webovou aplikáciou

Prepojenie databázy som implementoval na lokálnom apache serveri, ktorý som si vytvoril pomocou softwaru XAMPP. Najprv som naimportoval dump databázy a následne som vytvoril html layout, ktorý pozostáva z jednoduchej formy.V nej sa nachádza textarea, do ktorej používateľ napíše dotaz a následne potvrdí tlačidlom. Následne sa zavolá php skript, ktorý nad databázov zavolá vložený dotaz na do stránky prehliadača napíše výsledok vo forme prehľadnej tabuľky.

Používateľské rozhranie webovej aplikácie s vloženým dotazom



Výsledok dotazu:



Prílohy

* konceptualny model.pdf – detail konceptuálneho modelu vo vysokom rozlíšení
* export.sql – štruktúra databázy použitá v tomto projekte
* index.hml – užívateľské rozhranie webovej aplikácie
* script.php – script zabezpečujúci vykonávanie používateľom zadaných dotazov