SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

**AUTOBUSOVÝ DOPRAVCA**  
Semestrálna práca z DBS

Cvičenie streda 15:00   
Zimný semester 2014/2015 Jozef Dutko 64138  
2. ročník, Aplikovaná informatika Ján Dzvoník 72803

# Úvod

## Účel dokumentu

Tento dokument obsahuje špecifikáciu softvérového systému pre autobusového dopravcu. Dokument je výsledkom študentského projektu v predmete Databázové systémy. Je určený pre vedúceho projektu.

## Prehľad dokumentu

Dokument analyzuje databázovú časť softvérového systému pre autobusového dopravcu. V kapitole 1 sa nachádza znenie zadania. V kapitola 2 sú analyzované požiadavky a spracovaný konceptuálny návrh.

# Zadanie

Úloha: D03

Názov: Autobusový dopravca

Popis požiadaviek:  
Založil Ste si dopravnú spoločnosť, ktorá sa bude zaoberať osobnou dopravou a pre začiatok zakúpila jeden vlastný autobus, ktorý preváža cestujúcich na jednej trase. Vypracujte pre vlastné potreby informačno-databázový systém pre rezerváciu, predaj lístkov. Sledujte Vaše príjmy, vyťaženosť autobusu, atď.

# Konceptuálny návrh

## Analýza požiadaviek

Nasledujúce požiadavky vznikli komunikáciou so zákazníkom. Zhotoviteľ je povinný zabezpečiť všetku funkcionalitu a zákazník sa zaväzuje nemeniť požiadavky v čase vývoja systému.

Dopravná spoločnosť zhrnula požiadavky na systém do niekoľkých bodov, nefunkčnosť jednej požiadavky znamená nesplnenie úlohy.

Systém by mal mať možnosť aktualizácie autobusov, ale na začiatok stačí, keď bude v databáze iba jeden autobus, ktorý bude jazdiť na jednej diaľkovej linke.

Chceme mať možnosť pridať novú trasu, aktualizovať existujúcu trasu.

Vyžaduje sa taktiež možnosť vytvorenia rezervácie. Rezervácia sa vzťahuje na jeden spoj a na jedno miesto v autobuse.

Potrebujeme mať prehľad o predaných rezerváciách týkajúcich sa zadaného spoja. Chceme vedieť zistiť podiel cestujúcich s rezerváciou na celkovom počte cestujúcich.

V súvislosti s cestovnými lístkami budeme v systéme evidovať iba údaje, ktoré sú pre firmu významné z ekonomického hľadiska. Nevyžaduje sa napríklad uchovávať nástupnú a výstupnú zastávku cestujúceho, ale precestovanú vzdialenosť.

Zamestnanci by mali mať možnosť zistiť vyťaženosť konkrétneho spoja, vyťaženosť k určitému dňu v týždni, celkovú priemernú vyťaženosť spojov.

Systém by mal poskytnúť údaje, či sa preferuje zakúpenie rezervácie alebo kúpa cestovného lístku bez rezervácie.

Systém by mal slúžiť aj manažérom na zistenie príjmov dosiahnutých za určité časové obdobie.

Systém má obsahovať cenník. Cenník požadujeme vytvoriť tak, že intervalu vzdialeností bude priradená zodpovedajúca cena. V cenníku musia byť rozlíšené 2 cenové úrovne: obyčajné a zľavnené cestovné. Cenník musí byť možné pravidelne meniť, kedy sa starý cenník vymení za nový.

Vyžaduje sa schopnosť zobraziť cestovný poriadok zadanej linky.

## Entity

Entita reprezentuje objekt reálneho sveta, o ktorom chceme mať v databáze informácie.

Potrebujeme, aby existovala entita **LÍSTOK**. LÍSTOK uchováva informácie o cenovej úrovni, vzdialenosti a hodnote Lístku. Z nej bude možné získať informácie o tržbách, vyťaženosti autobusu atď.  
Systém má ponúkať možnosť rezervácie miest, tento fakt vyžaduje entitu **REZERVÁCIA**. Rezervácia je samostatný produkt. Rezervácia miesta sa vzťahuje na istý spoj. V tejto entite chceme zachovávať aj cenu platenú cestujúcim za rezerváciu miesta.   
Cestujúceho bude zaujímať **SPOJ**. V entite SPOJ budú inštanciami konkrétne spoje realizované v daný deň. Kúpiť si cestovný lístok alebo rezervovať miesto je možné na konkrétny SPOJ.  
Potrebujeme evidovať **ZASTÁVKY**. Spoj obsluhujúci danú trasu musí mať vopred určené zastávky. V entite zastávky máme priestor na zachytenie podstatných informácií týkajúcich sa cestovného poriadku na trase, ako napríklad čas zastavenia a podobne.  
Z požiadaviek vyplýva, že zadávateľ plánuje prevádzkovať jeden autobus na jednej trase. Aj pri takto jednoduchom modeli musíme vyriešiť problém, že po trase je možné ísť tam a späť. Preto zavedieme entitu **LINKA**. S entitou LINKA tiež získavame možnosť v budúcnosti vytvárať nové trasy. V LINKE sme zaviedli aj čas odchodu, ktorý bude potrebný pre identifikáciu správnej linky. Linke budú prislúchať ZASTÁVKY a konkrétny výkon linky bude SPOJ.  
Ďalej potrebujeme, aby existoval **CENNÍK**, podľa ktorého bude cestujúcemu určené cestovné. Musíme byť schopní priradiť vzdialenosti príslušnú cenu. Platnosť cenníka je vhodné ohraničiť na vopred určené obdobie. My sme sa rozhodli cenník meniť raz za rok, čiže táto skutočnosť sa v projekte nevyskytne, keďže naša spoločnosť začala podnikať iba tento rok. Po skončení platnosti by mal byť vytvorený a používaný nový cenník. Predpokladáme, že starý cenník nám ako entita, ktorá bude slúžiť na porovnávanie cien oproti minulým rokom, taktiež ho potrebujeme ďalej evidovať kvôli väzbe s lístkami, keďže predpokladáme, že lístky nám ostanú v entite LÍSTOK archivované.  
Zavedieme si systematické riešenie aj pre evidenciu autobusov, entitu **AUTOBUS**, kde sa budú nachádzať informácie o vozidle, ktoré potrebujeme mať v systéme zaznamenané. Pri návrhu databázy by bola chyba implicitne predpokladať existenciu jediného autobusu počas celého životného cyklu firmy.

## Atribúty

Atribúty sú vlastnosti entít, ktoré potrebujeme zaznamenať.

Lístok: **spoj, vzdialenosť, cenová úroveň**.  
Lístok sa vzťahuje na konkrétny spoj, chceme zaznamenať precestovanú vzdialenosť - vzdialenosť zastávok, ktorú zistíme z rozdielu vzdialeností výstupnej a nástupnej zastávky, a potrebujeme informáciu o cenovej úrovni cestujúceho. Pomocou týchto údajov dokážeme určiť cenu podľa cenníka.

Cenník: **vzdialenosť\_od, vzdialenosť\_do, cenová úroveň, cena.**Cenník potrebujeme na to, aby sme vzdialenosti v intervale vzdialenosť\_od - vzdialenosť\_do priradili cestovné. Cestovné sme rozdelili do dvoch úrovní: obyčajné a zľavnené cestovné. Úrovne sú reprezentované číslom. Na základe vzdialenosti a cenovej úrovne vieme identifikovať cenu.

Rezervácia: **spoj, miesto v autobuse, cena**.  
Pri rezerváciách nás zaujíma, na ktorý spoj máme rezerváciu, sumu ktorú cestujúci zaplatí navyše za rezervovanie a číslo miesta v autobuse.

Spoj: **dátum, linka, autobus**.  
Spoj je určený dátumom a linkou. Z informácie o linke chceme byť schopní jednoznačne určiť informáciu o cestovnom poriadku. Ak sa linka v daný deň vykoná viackrát, potrebujeme sa odkázať na správnu linku. Pri spoji chceme evidovať výpravu vozidiel, takže potrebujeme atribút, ktorý bude hovoriť o autobuse, ktorý vykonal daný spoj.

Linka: **označenie linky, smer, čas odchodu**.  
Budeme evidovať označenie linky, čas odchodu a smer.

Autobus: **EČV vozidla, kapacita**.  
Zaujíma nás kapacita autobusu. EČV vozidla potom môže slúžiť na jednoznačnú identifikáciu vozidla.

Zastávky: **linka, názov zastávky, čas zastavenia, vzdialenosť**.  
Pre danú linku budeme evidovať názov zastávky, čas zastavenia a vzdialenosť v kilometroch z východzej stanice. Z týchto údajov sa zostavuje vzdialenosť zastávok pri lístku a dáta sa dajú tiež použiť napríklad na tlač cestovného poriadku danej linky.

## Matrix diagram

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Lístok | Rezervácia | Spoj | Zastávky | Linka | Cenník | Autobus |
| Lístok | - | - | Zakúpený na | - | - | Má cenu podľa | - |
| Rezervácia | - | - | Uskutočnená na | - | - | - | - |
| Spoj | Ponúka predaj | Ponúka možnosť | - | - | Jazdí po | - | Vykonaný |
| Zastávky | - | - | - | - | Sú priradené | - | - |
| Linka | - | - | Obslúžená | Má priradené | - | - | - |
| Cenník | Určuje cenu | - | - | - | - | - | - |
| Autobus | - | - | Vykonáva | - | - | - | - |

## Vzťahy entít

V maticovom diagrame nemáme zachytené informácie o tom, aké vlastnosti majú vzťahy medzi entitami. Každému vzťahu potrebujeme priradiť kardinalitu. Kardinalita je počet objektov entít, ktoré sa zúčastňujú vzťahu na jednotlivých stranách. Druhou vlastnosťou je parcialita. Určuje povinnosť alebo voliteľnosť účasti entity na vzťahu. Ďalšou vlastnosťou vzťahu je transferabilita. Vyjadruje možnosť zámeny inštancií entít podieľajúcich sa na vzťahu.  
Medzi entitami sme identifikovali tieto binárne vzťahy:

SPOJ - LÍSTOK  
Kardinalita: 1:N - na jeden spoj možno kúpiť viac lístkov, jeden lístok sa spája s práve jedným spojom.  
Parcialita: neúplná účasť na strane spoja - na každý spoj nemusí byť vždy predaný lístok, úplná účasť na strane lístka - k lístku musí existovať konkrétny spoj.

CENNÍK - LÍSTOK  
Kardinalita: 1:N - k jednému cenníku sa môže vzťahovať viac lístkov, jeden lístok sa vzťahuje na práve jeden cenník.  
Parcialita: neúplná účasť na strane cenníka - cenníku nemusí byť vo vzťahu so žiadnym lístkom, úplná účasť na strane lístka - lístok musí byť vo vzťahu s cenníkom.

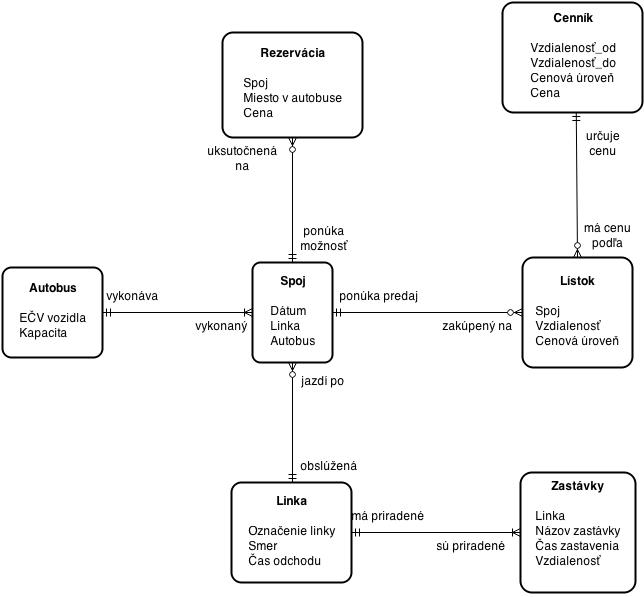
SPOJ - REZERVÁCIA  
Kardinalita: 1:N - k jednému spoju môže existovať viac rezervácií, jedna rezervácia je vo vzťahu s práve jedným spojom.  
Parcialita: neúplná účasť zo strany spoja - môžeme mať spoj, ktorý nie je vo vzťahu so žiadnou rezerváciou, úplná účasť zo strany rezervácie - každá rezervácia je vo vzťahu so spojom.

AUTOBUS - SPOJ  
Kardinalita: 1:N - každý spoj je vykonaný jediným autobusom, jeden autobus vykonáva viac spojov.  
Parcialita: úplná účasť zo strany autobusu - autobus vykonáva spoje (inak ho nemá zmysel vlastniť), úplná účasť zo strany spoja - spoj musí byť vykonaný autobusom.

LINKA - SPOJ  
Kardinalita: 1:N - spoj je označený práve jednou linkou, linka je obslúžená viacerými spojmi.  
Parcialita: neúplná účasť zo strany linky - môžeme evidovať linku, ktorá sa v praxi nevykonáva, úplná účasť zo strany spoja - spoj musí mať vždy pri radenú linku.

LINKA - ZASTÁVKY  
Kardinalita: 1:N - inštancia entity zastávky je priradená práve jednej linke, jedna linka má priradených viac inštancií zastávok.  
Parcialita: úplná účasť na strane linky - každá linka musí mať priradené zastávky (minimálne počiatočnú a konečnú), úplná účasť na strane zastávky - každá zastávka musí byť priradená práve jednej linke, pretože v entite zastávka zachytávame informácie o cestovnom poriadku vzhľadom na konkrétnu linku.

## ERA diagram



untitled diagram 2 (1)

# Logická úroveň

## Určenie primárnych kľúčov

Každá entita potrebuje mať svoj primárny kľúč. Bude to atribút (prípadne kombinácia atribútov) entity, ktorý bude mať podľa definície nasledujúce vlastnosti:

- bude unikátny v celej tabuľke popisujúcej danú entitu. To znamená, že sa v celej tabuľke nenájdu dva riadky, ktoré by mali rovnakú hodnotu tohto atribútu.

- jeho hodnota nesmie byť prázdna (NULL) po celú dobu existencie tabuľky.

- už raz zadaná hodnota tohto identifikátoru sa nesmie v danom riadku tabuľky nikdy zmeniť po dobu existencie tabuľky.

Existuje pravidlo, ktoré hovorí, že primárny kľúč má byť čo najmenší. Z toho nám vyplýva, že za primárny kľúč by bolo dobre si zvoliť ID skladajúce sa z čísiel.

Entita autobus má vhodný kandidát na primárny kľúč EČV vozidla, ktorý je jedinečný pre každý autobus, ale sa skladá z čísiel a písmen. Takže zvolíme si radšej iba číselnú formu, aby sme splnili podmienku, že primárny kľúč má byť čo najmenší. Preto si zvolíme ako primárny kľúč ID. Čiže môžeme skoro pre všetky ENTITY priradiť atribút ID\_ + názov entity. Okrem cenníku, kde je vhodné si za primárny kľúče zvoliť trojicu atribútov od ktorých je cena úplne závislá. Je to cenová úroveň, vzdialenosť\_od a vzdialenosť\_do. Toto určenie nám pomohlo splniť druhú normálnu formu a tretiu normálnu formu. Pri entite ZASTAVKY sme si zvolili za primárny kľúč dvojicu atribútov ID\_LINKA a nazov\_zastavky. Zabezpečili sme, že atribúty vzdialenost a cas\_zastavenia budú úplne závislé od primárneho kľúča a tabuľka zastávky spĺňa druhú a tretiu normálnu formu.

## Normalizácia

### Prvá normálna forma (1NF)

O relácii (tabuľke) hovoríme, že je v prvej normálnej forme, ak sú všetky jej atribúty atomické , t.j. ďalej nedeliteľne . Inými slovami - každý atribút relácie môže mať iba jednu hodnotu, teda nemôže byť jeho hodnotou ďalšia relácia. Túto skutočnosť sme splnili tým, že sme už v konceptuálnej úrovni návrhu oddelili entitu LINKA od entity SPOJ, taktiež sme oddelili potom entitu ZASTAVKY od entity LINKA. Tento krok sme dokončili pridaním príslušných primárnych kľúčov.

### Druhá normálna forma (2NF)

Tabuľka je v druhej normálnej forme, ak je v prvej normálnej forme a najviac každý atribút, ktorý nie je primárnym kľúčom je na primárnom kľúči úplne závislý. Naša entita REZRVACIE nespĺňa túto podmienku, lebo cena rezervácie nezávisí na primárnom kľúči. Preto sme sa rozhodli spraviť novú entitu CENNIK\_REZ, ktorá bude obsahovať cenu a ID ako primárny kľúč. Vzniknutý nový vzťah bude mať kardinalitu 1:N. Teraz môžeme vyhlásiť, že naše entity sú v druhej normálnej forme. Môžeme si to ukázať na entite CENNIK, kde máme 3jicu primárnych kľúčov a atribút cena je na všetkých závislý.

### Tretia normálna forma

O relácii hovoríme, že je v tretej normálnej forme, ak je v druhej normálnej forme a zároveň všetky jej atribúty, ktoré netvoria primárny kľúč, sú na sebe nezávisle. Ak by existoval atribút, ktorý je závislý na inom atribúte, musíme ho presunúť do novej entity. U nás sa po určení primárneho kľúča ako trojica atribútov v entite cenník splnila podmienka nezávislosti atribútov, ktoré nie sú primárnymi kľúčami. Keby sme v tomto prípade zvolili za kľúč iba cenovú úroveň, tak si môžeme všimnúť závislosť medzi vzdialenosťami a cenou, čo by viedlo k porušeniu nezávislosti nekľúčových atribútov.

## Mapovanie entít, atribútov a ich vzťahov do tabuliek

Tento krok ma nasledujúce pravidlá:

1) Každá entita v konceptuálnom modeli sa stáva samostatnou tabuľkou. Menom tabuľky sa obvykle stáva meno entity v pluráli. Vytvoríme teda tabuľky AUTOBUSY, SPOJE, ZASTAVKY, LINKY, CENNIK, LISTKY, CENNIK\_REZ, REZERVACIE.

2) Identifikátor entity sa stáva primárnym kľúčom. Naše tabuľky teda budú mať ako primárne kľúče priradené ich ID čísla. CENNIK bude mať priradenú trojicu atribútov, ZASTAVKY budú mať ako primárny kľúč dvojicu atribútov ID\_LINKY a nazov\_zastavky.

3) Ak je súčasťou vzťahu jeden alebo viac atribútov, je nutné vytvoriť novú (väzbovú) entitu

podobne ako u vzťahu M:N a z nej vytvoriť tabuľku. Naše vzťahy nemajú kardinalitu n:m, takže tento bod môžeme preskočiť.

4) Pri každom vzťahu zvolíme tabuľku, ktorá bude obsahovať cudzí kľúč ako odkaz do inej tabuľky. Pre voľbu tabuľky použijeme tieto pravidlá:

a) Ak je vzťah typu 1:N, bude cudzí kľúč pridaný do tabuľky na strane N. Preto pridávame cudzie kľúče do tabuliek :

SPOJE na vyjadrenie referencie na tabuľky AUTOBUS a LINKY pridáme na strane SPOJE atribúty ID\_AUTOBUS, ID\_LINKA.

Tak isto pridáme aj k entitám REZERVACIE a LISTKY atribút ID\_SPOJ vyjadrujúci vzťah

k entite SPOJE. Entita REZERVACIE bude mať ešte aj cudzí kľúč ID, ktorá je referenciou do tabuľky CENNIK\_REZ.

Entita LISTKY bude obsahovať cudzie kľúče cenová úroveň a vzdialenosť, ktoré odkazujú na CENNÍK. Ako posledné zvolime v entite ZASTÁVKY za cudzý kľúč primárny kľúč ID\_LINKA.

b) Ak sa jedná o parciálny vzťah 1:1, bude cudzí kľúč pridaný do tabuľky, ktorá sa vyskytuje vo vzťahu nepovinne. Takýto vzťah sa u nás nenachádza.

c) Ak sa jedná o neparciálny vzťah 1:1, volíme tabuľku, ktorá je vecne „podriadená“. V našom návrhu máme vzťah 1:1 nezavádzali

d) Vzťahy typu M:N najprv dekomponujeme na dva vzťahy 1:N a potom postupujeme podľa pravidiel pre tento typ vzťahov. Tento vzťah sa u nás taktiež nenachádza.

5) Ak sa vyskytne v konceptuálnom modeli špecializácia, máme viacero možností, optimálnym riešením je vytvorenie tabuľky, ktorá bude obsahovať stĺpce spoločné pre všetky typy entít. Pre každú špecializáciu (entitu) potom vytvoríme tabuľku ďalšiu, ktorá bude obsahovať stĺpce špecifické pre túto entitu. Musíme však zaistiť referenčnú integritu dát medzi špecializovanými tabuľkami a tabuľkou hlavnou. (Toto nie je náš prípad, takéto tabuľky sa v našej schéme nevyskytujú.)

Po mapovaní vyzerajú naše tabuľky takto:

AUTOBUSY:

|  |  |  |
| --- | --- | --- |
| ID\_AUTOBUS | ECV vozidla | kapacita |

SPOJE:

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_SPOJ | ID\_AUTOBUS | ID\_LINKA | datum |

LINKY:

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_LINKA | oznacenie\_linky | smer | cas\_odchodu |

ZASTAVKY:

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_LINKA | vzdialenost | nazov\_zastavky | cas\_zastavenia |

REZERVACIE:

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_REZERVACIE | ID\_SPOJ | miesto\_na\_sedenie | ID |

LISTKY:

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_LISTOK | ID\_SPOJ | vzdialenost | cenova\_uroven |

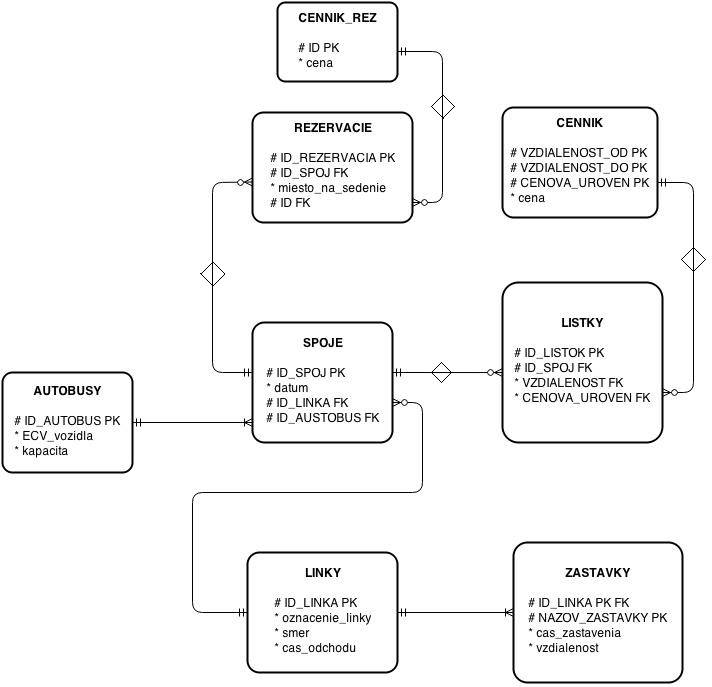
CENNIK:

|  |  |  |  |
| --- | --- | --- | --- |
| vzdialenost\_od | vzdialenost\_do | cenova\_uroven | cena |

CENNIK\_REZ:

|  |  |
| --- | --- |
| ID | cena |

## Relačná schéma

untitled diagram 3 (5)  
(nakoniec ešte doplníme non-transferability značky...)

# Implementačná úroveň

V tejto úrovni sa nám mení návrh na konkrétne technické prostriedky. Na vývoj našej databázy použijeme ORACLE Database Aplication Express a ORACLE SQL Developer.

Entity sa stávajú tabuľkami vo fyzickej databáze. Atribúty sa stávajú stĺpcami v tabuľke, pričom musíme zvoliť príslušný dátový typ pre každý stĺpec. Unikátne identifikátory sa stavajú primárnymi kľúčmi v tabuľke, vzťahy medzi entitami modelujeme ako cudzie kľúče.

## 5.1. Slovný popis návrhu

### Dátové typy

Na prácu s reťazcami sme zvolili dátový typ VARCHAR2, kvôli jeho úspornosti pamäti oproti typu CHAR, kde reťazec, ktorý je kratší ako jeho stanovená maximálna dĺžka je automaticky doplnení znakmi, aby jeho dĺžka zodpovedala maximálnej dĺžke (max 2000, min 1, implicitne 1). Pri VARCHAR2 sa reťazec nedopĺňa do maximálnej dĺžky. Pri prekročení maximálnej dĺžky nahlási systém chybu. Povinným argumentom pri VARCHAR2 je maximálna dĺžka reťazca (max 4000, min 1).

Pri práci s číslami využijeme dátový typ NUMBER, ktorý môže vystupovať ako:

1. Celé číslo

NUMBER(p), kde p je maximální počet číslic; zápis je ekvivalentný tomuto: NUMBER(p,0).

2. Fixed point číslo

NUMBER(p,s), kde s je počet číslic vpravo od desatinnej čiarky v rozsahu <-84, 127> - záporná hodnota vynutí nulovosť prísl. počtu cifier vľavo od desať. čiarky - zaokrouhľovaný na stovky a pod., p stále značí max. počet číslic vľavo od desatinnej čiarky.

3. Floating point číslo

NUMBER, povoľuje 38 platných číslic, rozsah 1.0 x 10-130 až 9.9...9 x 10125. Oracle samozrejme zase kontroluje, či zadávané hodnoty zodpovedajú datovému typu: ak je to potrebné, zaokrouhľuje; ale ak nejde číslo za daných podmienok uložiť (p je malé a pod.), zahlási chybu.

Ako posledné použijeme pri práci s dátumom dátový typ DATE. Pre každú hodnotu typu DATE, Oracle skladuje tieto informacie: storočie, rok, mesiac, deň, hodina, minúta a sekunda. Keď zadáme iba dátum tak sa bude brať implicitne čas 12:00 am (polnoc).

V našom prípade máme entitu AUTOBUSY, kde máme atribúty kapacita, ID a EČV vozidla. Kapacita v tomto prípade bude predstavovať celé číslo, čiže využijeme NUMBER(p,0), kde p nám postačí trojmiestne. EČV vozidla bude reťazec skladajúci sa z čísiel a znakov, preto využijeme VARCHAR2 s dĺžkou 8, kde jedno miesto je vyhradené pre pomlčku. Zostalo nám definovať ID, ktoré bude v našom prípade maximálne 16 miestne číslo a minimálne 1 miestne číslo, čiže NUMBER(16,0).

Entita SPOJE má atribúty ID, ID\_LINKA, ID\_AUTOBUS a dátum. IDečka majú všetky formát NUMBER(16,0). Dátum má formát DATE s časom 12:00 am. Formát je v tvare „16-JAN-99”.

Entita LINKY má ID, označenie linky, smer a čas odchodu. Smer linky je charakterizovaný reťazcom hovoriacim o cieľovej zastávke, preto využijeme VARCHAR2 s dĺžkou 20, čo zodpovedá počtu písmen najdlhšieho názvu slovenského mesta. Označenie linky bude reprezentované ako trojmiestnu kombináciu písmen a čísiel, VARCHAR2(3). Čas odchodu si určíme ako VARCHAR2(5).

Entita ZASTAVKY obsahuje atribúty ID\_LINKA, názov zastávky, vzdialenosť, Čas zastavenia. Názov zastávky bude 20 písmenový reťazec. Vzdialenosť bude vyjadrená 7 znakmi, kde rátame s možnou, že vzdialenosť bude zadaná buď v metroch alebo kilometroch. Čas zastavenia vyjadríme formátom VARCHAR2 s veľkosťou 5 pre úsporu pamäti, keďže čas nám slúži len ako informácia o zastávke.

Entita REZERVACIE obsahuje ID\_REZERVACIE, ID, ID\_SPOJ, miesto na sedenie. Miesto na sedenie je maximalne trojmiestne číslo, čiže NUMBER(3,0).

Entita CENNIK\_REZ obsahuje ID a cenu, ktorá bude charakterizovaná ako NUMBER(3,2).

Entita LISTKY obsahuje vzdialenosť, cenovú úroveň a ID\_SPOJ. Cenová úroveň je v našom prípade maximálne jednomiestne číslo. Vzdialenosť je ako v predchádzajúcom prípade maximálne 7 miestne číslo.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabuľka** | **Stĺpec** | **Dátový typ** | **Poznámka** |
| AUTOBUSY | ID\_AUTOBUS | NUMBER(16,0) | Primárny kľúč |
|  | ECV\_vozidla | VARCHAR2(8) |  |
|  | Kapacita | NUMBER(3,0) |  |
| SPOJE | ID\_SPOJ | NUMBER(16,0) | Primárny kľúč |
|  | ID\_AUTOBUS | NUMBER(16,0) | Cudzí kľúč |
|  | ID\_LINKA | NUMBER(16,0) | Cudzí kľúč |
|  | Datum | DATE |  |
| LINKY | ID\_LINKA | NUMBER(16,0) | Primárny kľúč |
|  | Smer | VARCHAR2(20) |  |
|  | Oznacenie\_linky | VARCHAR2(3) |  |
|  | Cas\_odchodu | VARCHAR2(5) |  |
| ZASTAVKY | Nazov\_zastavky | VARCHAR2(20) | Primárny kľúč |
|  | ID\_LINKA | NUMBER(16,0) | Primárny, cudzí kľúč |
|  | Vzdialenost | NUMBER(7,0) |  |
|  | Cas\_zastavenia | VARCHAR2(5) |  |
| REZERVACIE | ID\_REZERVACIE | NUMBER(16,0) | Primárny kľúč |
|  | ID\_SPOJ | NUMBER(16,0) | Cudzí kľúč |
|  | Miesto\_na\_sedenie | NUMBER(3,0) |  |
|  | ID | NUMBER(16,0) | Cudzí kľúč |
| CENNIK\_REZ | ID | NUMBER(16,0) | Primárny kľúč |
|  | Cena | NUMBER(3,2) |  |
| LISTKY | ID\_LISTOK | NUMBER(16,0) | Primárny kľúč |
|  | ID\_SPOJ | NUMBER(16,0) | Cudzí kľúč |
|  | Vzdialenost | NUMBER(7,0) | Cudzí kľúč |
|  | Cenova\_uroven | NUMBER(1,0) | Cudzí kľúč |
| CENNIK | Vzdialenost\_od | NUMBER(7,0) | Primárny kľúč |
|  | Vzdialenost\_do | NUMBER(7,0) | Primárny kľúč |
|  | Cenova\_uroven | NUMBER(1,0) | Primárny kľúč |
|  | Cena | NUMBER(5,2) |  |

## ID

Väčšina naších entít má ako primárny kľúč priradené ID číslo, ktoré sme si zvolili ako NUMBRE(16,0). Tieto primárne kľúče sú tvorené pomocou sekvencií a triggerov, ktoré zabezpečujú jedinečnosť identifikátora.

Pre vytvorenie sekvencie používame tento kód:

CREATE SEQUENCE seq\_nazov;

Naša sekvencia má tieto parametre:

MIN\_VALUE 1

MAX\_VALUE 999999999999999999999999999

START WITH 1

INCREMENT BY 1

CACHE 20

Ďalej sme vytvorili trigger. Trigger je procesný kľúč, ktorý je automatický vykonaný v reakcií na určitú udalosť. V našom prípade sa vytvorí trigger, ktorý pred každým zavolaním funkcie INSERT do príslušnej tabuľky zavolá sekvenciu, ktorá vygeneruje jedinečné číslo a to následne vloží do tabuľky.

Kód pre vytvorenie triggeru a jeho následné definovanie:

CREATE OR REPLACE TRIGGER nazov\_bir

BEFORE INSERT ON NAZOV\_TABLE

FOR EACH ROW

BEGIN

SELECT nazov\_seq.NEXTVAL

INTO :new.ID

FROM dual;

END;

## Vytvorenie tabuliek

Tabuľky sa vytvárali nasledujúcim spôsobom

CREATE TABLE NAZOV (

COLUMN1 DATATYPE NULL|NOT NULL,

COLUMN2 DATATYPE NULL|NOT NULL,

...);

Uvedieme si príklad pre tvorbu tabuľky CENNIK:

CREATE TABLE CENNIK (

vzdialenost\_od NUMBER(7,0) NOT NULL,

vzdialenost\_do NUMBER(7,0) NOT NULL,

cenova\_uroven NUMBER(1,0) NOT NULL,

cena NUMBER(5,2) NOT NULL

);

Na základe takto vloženého príkazu sa nám vytvorí tabuľka cenník, kde stĺpce budú vzdialenost\_od, vzdialenost\_do, cenova\_uroven, cena. Tieto stĺpce sme ošetrili pomocou zadania do príkazu CREATE TABLE NOT NULL. Toto ošetrenie sa nám v praxi prejaví tak, že po zadaní inej hodnoty ako nami určený dátový typ, nám nedovolí napríklad pridať riadok do tabuľky.

Po vytvorení tabuľky si potrebujeme určiť primárne kľúče a to nasledujúcim príkazom:

ALTER TABLE nazov\_tabulky ADD CONSTRAINT nazov\_pk PRIMARY KEY (column1, column2...);

V našom prípade sme pre určenie primárnych kľúčov v tabuľkv CENNIK použili tento kód:

ALTER TABLE CENNIK ADD CONSTRAINT CENNIK\_pk PRIMARY KEY (vzdialenost\_od, vzdialenost\_do, cenova\_uroven);

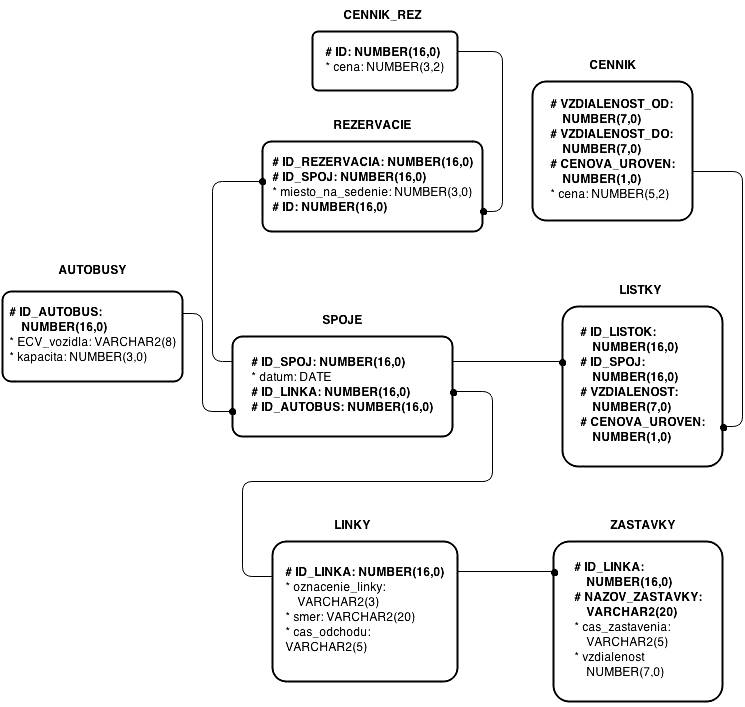
Ako posledný krok pre dokončenie procesu vytvárania tabuliek, sme si potrebovali určiť cudzie kľúče a to týmto príkazom:

alter table tabulka\_name add constraint FK\_tabulka\_name2 FOREIGN KEY (ID) references tabulka\_name2;

Príklad pre vyjadrenie cudzieho kľúča ID v našej tabuľke REZERVACIE:

alter table rezervacie add constraint FK\_cennik\_rez FOREIGN KEY (id) references cennik\_rez;

## Fyzický model



## Prehľad potrebných SQL dopytov

Naša databáza má ponúkať možnosť použitia SQL príkazov SELECT, INSERT, DELETE, UPDATE. Pri tomto bode budeme vychádzať z požiadaviek vytvorených po komunikácií zo zákazníkom.

### Vkladanie údajov do tabuliek, zmena dátových typov stĺpcov

1. Vkladanie hodnôt ID do tabuliek LISTKY, CENNIK\_REZ, AUTOBUSY, REZERVACIE, SPOJE, LINKY pomocou sekvencií.
2. Pridávanie údajov do tabuliek.
3. Manuál pri rezervácií.
4. Archivácia cenníkov.
5. Zmena dátových typov stĺpcov pre účely šetrenia pamäte
6. Pridanie cestovného lístka
7. Vkladanie hodnôt ID do tabuliek LISTKY, CENNIK\_REZ, AUTOBUSY, REZERVACIE, SPOJE, LINKY pomocou sekvencií.

V našom prípade pri použití sekvecnií a triggerov môžeme použiť nasledovné syntaxi príkazov insert, ktoré si ukážeme na našej databáze:

1. Lístok môžeme pridať aj bez zadania primárneho kľúča.

insert into listky (id\_spoj, vzdialenost, cenova\_uroven) values (1,150,0);

1. Dá sa pridať aj za pomoci, že ID označíme ako NULL.

insert into listky (id\_listok, id\_spoj, vzdialenost, cenova\_uroven) values (null,2,200,1);

1. Môžeme pridať lístok aj za pomoci napísania ľubovolné čísla, kde sa toto číslo bude ignorovať a namiesto neho sa pred insertom pridá číslo zo sekvencie.

insert into listky (id\_listok, id\_spoj, vzdialenost, cenova\_uroven) values (9999,2,200,1);

1. Môžeme využiť listok\_seq.nextval.

insert into listky (id\_listok, id\_spoj, vzdialenost, cenova\_uroven) values (listok\_seq.nextval,2,150,1);

1. Pridávanie údajov do tabuliek
2. Pridávanie do tabuľky AUTOBUSY

insert into autobusy (kapacita, ECV\_vozidla) values (60,'ZH-355AB');

1. Pridávanie do tabuľky CENNIK\_REZ

insert into cennik\_rez (cena) values (1);

1. Pridávanie do tabuľky CENNIK

insert into cennik (vzdialenost\_od, vzdialenost\_do, cenova\_uroven,cena) values (150,200,1,2);

1. Pridávanie do tabuľky SPOJE

insert into spoje (ID\_linka,ID\_autobus,datum) values (1,1, '01-dec-14');

1. Pridávanie do tabuľky ZASTAVKY

insert into zastavky (nazov\_zastavky, cas\_zastavenia, ID\_linka, vzdialenost) values ('Bratislava', '8:05',1,233);

1. Pridávanie do tabuľky LINKY

insert into LINKY(smer,oznacenie\_linky,cas\_odchodu) values ('Bratislava', 'R13', '5:00');

1. Manuál pri rezervácií

Rezervácia u nás funguje na princípe, že užívateľ zadá spoj, dátum, čas odchodu, smer a oznčenie linky. Na základe týchto dát vieme jednoznačne spojiť cestovný lístok s identifikátormi spoja a linky. Ak je ešte v spoji voľné miesto, čiže cestujúci nepresiahne svojou prítomnosťou kapacitu, bude mu pridelené nasledujúce voľné miesto automaticky. Zamestnanec cestujúcemu vygeneruje doklad o rezervácii. Cestujúci sa v reálnych podmienkach preukáže dokladom s označením ID\_rezervácie.

Príkladný kód pre pridanie rezervácie:

insert into rezervacie (id\_rezervacia, id\_spoj, miesto\_na\_sedenie, id) values (

null

,(select id\_spoj from spoje where datum = to\_date('03-dec-14')

and id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00'))

,case when((select count(\*) from rezervacie where id\_spoj =

(select id\_spoj from spoje where datum = to\_date('03-dec-14')

and id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00'))

) +1 ) <= (select kapacita from autobusy where id\_autobus =

(select id\_autobus from spoje where id\_spoj = (

(select id\_spoj from spoje where datum = to\_date('03-dec-14')

and id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00')))))

then ((select count(\*) from rezervacie where

id\_spoj = (select id\_spoj from spoje where datum = to\_date('03-dec-14')

and id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00'))

) +1 )

else null

end

,1);

Po vytvorení rezervácie môžeme vygenerovať túto rezerváciu za pomoci ID:

select ID\_REZERVACIA as id, oznacenie\_linky as linka, smer, datum, miesto\_na\_sedenie as miesto, to\_char(cena) || '€' as cena

from (( (REZERVACIE JOIN CENNIK\_REZ USING(ID))

JOIN SPOJE USING (ID\_SPOJ))

JOIN LINKY USING (ID\_LINKA))

where ID\_REZERVACIA = 5;

1. Archivácia cenníkov

V našom prípade chceme, aby sa cenníky archivovali tým spôsobom, že ich nevymažeme z databázy ale pridáme jednoducho nový cenník, toto by nám malo slúžiť na porovnávanie cien. V našom modeli, sme ešte neuvažovali s existenciou viacerých cenníkov, alebo existencií archívu cenníkov. V prípade existencií viacerých cenníkov by sme pridali nový cenník:

CREATE TABLE CENNIK (

vzdialenost\_od NUMBER(7,0) NOT NULL,

vzdialenost\_do NUMBER(7,0) NOT NULL,

cenova\_uroven NUMBER(1,0) NOT NULL,

cena NUMBER(5,2) NOT NULL

);

Časom by sme však mohli predpokladať existenciu tabuľky CENNIK\_arch:

CREATE TABLE CENNIK\_arch AS(SELECT \* FROM CENNIK);

1. Zmena hodnoty stĺpcov pre účely šetrenia pamäte

V naše databáze máme napríklad ID, ktoré je dátového typu NUMBER(16,0), zaberá to určítu pamät pre ušetrenie pamäti môžeme zmeniť dľžku NUMBER a to zadaním príkazu

alter table table\_name modify ID NUMBER(5,0);

1. Pridanie cestovného lístka

insert into listky (id\_listok,

id\_spoj,

vzdialenost,

cenova\_uroven) values

( 9

,(select id\_spoj from spoje where datum = to\_date('04-dec-14')

and id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00'))

,(select vzdialenost from zastavky where zastavky.id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00')

and nazov\_zastavky = 'Bratislava') /\* vystupna zastavka \*/

-

(select vzdialenost from zastavky where id\_linka = (select id\_linka from linky where oznacenie\_linky = 'R13'

and smer = 'Bratislava'

and cas\_odchodu = '5:00')

and nazov\_zastavky = 'Banská Bystrica') /\* nastupna zastavka \*/

,0

);

### Vymazávanie a zmena údajov

V našej databáze chceme riešiť aj situáciu, kde môže nastať chyba pri zadávaní údajov do tabuliek. Pri zlom zadaní údajov existujú príkazy pomocou ktorých vieme tento zlý údaj vymazať – DELETE alebo ho upravit – UPDATE.

Môžeme predpokladať napríklad situáciu, kde zamestnanec našej dopravnej spoločnosti zadá zle údaje o cestovnom lístku. Riešenie pomocou DELETE by vypadalo takto:

DELETE FROM LISTKY

WHERE ID\_Listok = 81;

Príkaz nám vymaže lístok s ID = 81 z tabuľky LISTKY, taktiež by sme mohli zvoliť vymazanie všetkých lístkov s určitým ID\_spojom a podobne.

Riešenie pomocou UPDATE:

UPDATE LISTKY

SET vzdialenost = 100

WHERE ID\_LISTOK = 41;

Príkaz nám zmení údaje v stĺpci vzdialenosť na 100. Pomocou WHERE sme určili komu chceme tento údaj zmeniť, taktiež môžeme určiť pomocou ID\_SPOJA lístky, ktoré chceme zmeniť.

Podobne by sa rieišili zlé údaje v cenníku:

DELETE FROM cennik

WHERE cena = 2;

alebo:

UPDATE cennik

SET cena = 3

WHERE cena = 2;

Problém nastane v prípade vymazania autobusu, ale tento problém sme nemuseli riešiť, pretože si danú funkcionalitu zákazník nevyžiadal. Pri vytváraní štatistik ako vyťaženosť autobusu za jednotlivé roky, potrebujeme evidovať kapacitu autobusu. Preto je vhodné uchovávať naďalej údaje o vyradenom autobuse. Keby nastala situácia, kedy už nepotrebujeme údaje o vyradenom autobuse a ani o cestovných lístkov, ktoré sa viažu na existenciu autobusu, môžeme jednoducho použiť tento súbor príkazov v nasledujúcom poradí:

delete from listky

WHERE ID\_spoj in (select ID\_spoj from SPOJE where ID\_autobus = 21);

delete from rezervacie

WHERE ID\_spoj in (select ID\_spoj from SPOJE where ID\_autobus = 21);

delete from spoje

WHERE ID\_autobus = 21;

Delete from autobusy

WHERE ID\_autobus = 21;

### Získavanie údajov z tabuľky

1. Všetky predanné lístky.
2. Predanné lístky do určitého dátumu.
3. Predanné lístky na spoj.
4. Priemerná dĺžka trasy pripadajúca na jedného cestujúceho.
5. Podiel cestujúcich s rezerváciou na celkovom počte cestujúcich.
6. Vyťaženosť spoja podľa ID\_SPOJa.
7. Celková priemerná vyťaženosť spoja do dátumu.
8. Priemerná vyťaženosť vzhľadom na deň v týždni.
9. Údaje o rezervácií.
10. Zobrazenie rezervácií na daný spoj.
11. Cestovný lístok.
12. Zobrazenie cestovného poriadku podľa linky.
13. Vyhľadať spoje, ktoré išli v daný dátum.
14. Zistenie prijmov z listkov za deň.
15. Zistenie príjmov z rezervácií za deň.
16. Zistenie celkových príjmov.
17. Predanné lístky.

select count(\*) as "celkovy pocet listkov" from LISTKY;

1. Predanné lístky do určitého dátumu.

select datum, count(\*) as "Prepravene osoby" from LISTKY join SPOJE USING (ID\_SPOJ) where datum <= to\_date('5.dec.14') GROUP BY datum;

1. Predanné lístky na spoj

select ID\_SPOJ, datum, count(\*) as "Prepravene osoby" from LISTKY join SPOJE USING (ID\_SPOJ) GROUP BY datum, ID\_SPOJ order by datum;

1. Priemerná dĺžka trasy pripadajúca na jedného cestujúceho.

select to\_char(round(( (select sum(vzdialenost) from listky)

/ (select count(\*) from listky)

), 2)

) || ' km'

as "Priemerna dlzka trasy"

from dual;

1. Podiel cestujúcich s rezerváciou na celkovom počte cestujúcich.

select

(select to\_char(

round(

((select count(\*) from rezervacie natural join spoje

where datum <= to\_date('4-dec-14')) /

(select count(\*) from listky natural join spoje

where datum <= to\_date('4-dec-14'))),2)

\* 100

) || '%'

from dual) as "Podiel rezervacii",

to\_date('03-dec-14') as "Do datumu"

from dual;

1. Vyťaženosť spoja podľa ID\_SPOJa.

select distinct datum, oznacenie\_linky, nazov\_zastavky as odkial, smer as kam, cas\_odchodu,

to\_char(round(( (select count(\*) from listky join spoje using (ID\_SPOJ)

where ID\_SPOJ = 1) /

(select sum (kapacita) from spoje join autobusy using (ID\_AUTOBUS)

where ID\_SPOJ = 1)

),3) \* 100) || '%' as "Vytazenost"

from ((((SPOJE join LISTKY using (ID\_SPOJ))

join AUTOBUSY using (ID\_AUTOBUS))

join LINKY using (ID\_LINKA))

join zastavky using (ID\_LINKA))

where ID\_SPOJ = 1 and ZASTAVKY.vzdialenost = 0;

1. Celková priemerná vyťaženosť spoja do dátumu.

select

to\_date('04-dec-14') as "Do datumu",

to\_char(round(( (select count(\*) from listky join spoje using (ID\_SPOJ)

where datum <= to\_date('04-dec-14')) /

(select sum (kapacita) from spoje join autobusy using (ID\_AUTOBUS)

where datum <= to\_date('04-dec-14'))

),3) \* 100) || '%' as "Vytazenost"

from dual;

1. Priemerná vyťaženosť vzhľadom na deň v týždni.

select

to\_char(to\_date('03-dec-14'), 'Day') as "Den v tyzdni",

to\_char(round(( (select count(\*) from listky join spoje using (ID\_SPOJ)

where to\_char(datum, 'fmDay') = 'Wednesday') /

(select sum (kapacita) from spoje join autobusy using (ID\_AUTOBUS)

where to\_char(datum, 'fmDay') = 'Wednesday'))

,3) \* 100) || '%' as "Vytazenost"

from dual;

1. Údaje o rezervácií.

select ID\_REZERVACIA as id, oznacenie\_linky as linka, smer, datum, miesto\_na\_sedenie as miesto, to\_char(cena) || '€' as cena

from (( (REZERVACIE JOIN CENNIK\_REZ USING(ID))

JOIN SPOJE USING (ID\_SPOJ))

JOIN LINKY USING (ID\_LINKA))

where ID\_REZERVACIA = 5;

1. Zobrazenie rezervácií na daný spoj.

select id\_rezervacia, oznacenie\_linky, smer, datum, miesto\_na\_sedenie

from (rezervacie join spoje using(ID\_SPOJ))

join linky using(ID\_LINKA)

where ID\_SPOJ = 1

order by miesto\_na\_sedenie;

1. Cestovný lístok.

select ID\_listok, to\_char(vzdialenost) || ' km' as vzdialenost, to\_char(cena) || '€' as cena, CASE LISTKY.cenova\_uroven WHEN 1 THEN 'Celý'

When 0 THEN 'Oby?ajný'

ELSE ' ' END as typ\_listka,

datum, oznacenie\_linky as linka, smer FROM

LISTKY JOIN CENNIK ON ((LISTKY.vzdialenost between CENNIK.vzdialenost\_od AND vzdialenost\_do)

and (listky.cenova\_uroven = cennik.cenova\_uroven))

JOIN SPOJE USING(ID\_SPOJ) JOIN LINKY USING(ID\_LINKA)

WHERE ID\_LISTOK = 1;

1. Zobrazenie cestovného poriadku podľa linky

select vzdialenost as km, nazov\_zastavky, cas\_zastavenia

from zastavky where id\_linka = 1 order by km;

1. Vyhľadať spoje, ktoré išli v daný dátum.

select spoje.datum, spoje.ID\_SPOJ, linky.oznacenie\_linky, linky.smer, linky.CAS\_ODCHODU from

spoje left outer join linky using (ID\_LINKA)

where datum = to\_date('04-12-2014');

1. Zistenie prijmov z listkov za deň.

select sum(cena) as "Prijem z listkov" from

( listky join cennik on

( ( round(listky.VZDIALENOST) between cennik.vzdialenost\_od and cennik.VZDIALENOST\_DO)

and (listky.CENOVA\_UROVEN = cennik.CENOVA\_UROVEN) )

)

join spoje using(ID\_SPOJ)

where datum = to\_date('03-12-14');

1. Zistenie príjmov z rezervácií za deň.

select sum(cena) as "Prijem z rezervacii" from

(rezervacie natural join cennik\_rez)

join spoje using (ID\_SPOJ)

where datum = to\_date('04-12-14');

1. Zistenie celkových príjmov

select

(select sum(cena) from

listky join cennik on

( ( round(listky.VZDIALENOST) between cennik.vzdialenost\_od and cennik.VZDIALENOST\_DO)

and (listky.CENOVA\_UROVEN = cennik.CENOVA\_UROVEN) )) as "Prijem z listkov",

(select sum(cena) from

rezervacie natural join cennik\_rez) as "Prijem z rezervacii",

( (select sum(cena) from

listky join cennik on

( ( round(listky.VZDIALENOST) between cennik.vzdialenost\_od and cennik.VZDIALENOST\_DO)

and (listky.CENOVA\_UROVEN = cennik.CENOVA\_UROVEN) ))

+

(select sum(cena) from

rezervacie natural join cennik\_rez)

) as "Prijem celkom"

from dual;