UNIVERZITA PARDUBICE  
Fakulta elektrotechniky a informatiky

*Informační systém pojišťovny*

Seminární práce z předmětu Databázové systémy I

Jméno a příjmení:

Jiří Král

V Pardubicích dne 12.12.2022

Obsah

[Analýza 10](#_Toc121737829)

[ERD 13](#_Toc121737830)

[Relační model dat 15](#_Toc121737831)

[Fyzický model dat 15](#_Toc121737832)

[Výčet SQL dotazů 29](#_Toc121737833)

[Závěr 34](#_Toc121737834)

[Přílohy 35](#_Toc121737835)

Analýza

Středně velká pojišťovna z České republiky si přeje přejít na nový informační systém. Po hromadě testů a ladění jsme jim jakožto KING s.r.o. vyhověli. Tato databáze vypadá jako oříšek, protože si pojišťovna NEXT (dále jen zákazník nebo pojišťovna) si nemůže z několika na sobě nezávislých důvodů přejít na zcela nové in-house řešení. Řešením tohoto problému je tedy systém, který bude následujících 5 až 10 let v provozu. Systém proto bude v některých částech používat kombinaci nových a starých standardů zákazníka. I přes to že bude v budoucnu potřeba kód předělat na nový systém znovu, dává to i možnosti zkoušet hromadu experimentálních funkcí, které mají čas být vyvinuty spolu se samotným konceptem databáze. Další velkou výhodou tohoto postupu je že můžeme v klidu přejít na jiný standart jazyka SQL. Neméně důležité kolem celého projektu je otázka ochrana osobních údajů. Celý projekt podléhá GDPR – Obecnému nařízení o ochraně osobních údajů (General Data Protection Regulation), plným názvem Nařízení Evropského parlamentu a Rady (EU) č. 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES obecné nařízení o ochraně osobních údajů. Kvůli tomuto je v této demonstraci využito ukázkových dat.

**1) Kompletní scénář min. 350 znaků, kterému musí odpovídat modely!!!**

Po několika osobních schůzkách a výměně emailu byla identifikovány tyto údaje a podmínky pro funkce databáze.

**Pojistka** je jedna z nejdůležitějších tabulek. **Pojistka** musí identifikovat *kdy* byla sjednána, do které *hodnoty* je kryto pojistkou a o jaký *druh pojistky* se jedná. A to konkrétně: **Připojištění**, **Úrazové** **pojištění** anebo **Majetkové** **pojištění**. Kromě toho je u každé pojistky evidováno, co smí a musí dělat v rámci smlouvy, díky **Závazkům** a **Pohledávkám**. Co si také klient přál bylo ukládání dat o tom, který zaměstnanec vykonal danou pojistku. Pro další evidenci zaměstnanců slouží i tabulka **Zaměstnanec**. O **Zaměstnanci** je mimo jména nutné ukládat i co vykonávají za **Pozici**, pod kterou **Pobočku** patří. **Klient** a **Banka** ukládají svoje standartní identifikátory jako například *název*, oproti tomu bude Kontakt jako svá vlastní tabulka pouze v pozici odkazu – neboli cizího klíče. Stejnou implementací je připojen **Kontakt** k **Zaměstnancovi**. K budoucímu spojení bude docházet mezi **Klientem** a **Bankou**. Ta momentálně není přidělená, protože jednotlivé **Banky** nedali integrovanému spojeni **KLIENT** x **POJISTOVNA** x **BANKA** zelenou. Adresa bude sloužit na uložení standartních údajů a hlavně města. To je kvůli následnému filtrování pomocí Kraj umožní dávat požadovaná anonymizovaná data analytickému a marketingovému týmu pojišťovny.

**2) Strukturální a procedurální pravidla**

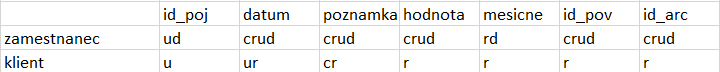
* Adresa musí mít Kraj
* ID\_kraj nemůže být vyšší než 14
* Zaměstnanec musí mít Pozici
* V Adresa, Patra se počítají od 0 = přízemí, 1 = první podlaží
* Datum ukončení smlouvy nesmí být před uzavřením smlouvy
* Hodnota platba\_mesicne nesmí přesahovat 100% pojištěné hodnoty

**3) Jednoduchá analýza vyplývající ze scénáře, která zahrnuje označení entit (např. tučně), označení atributů (kurzívou).**

*Atributy* jsou označeny kurzívou, **Entity** jsou označeny tučným písmem.

**4) CRUD analýza**

* Zaměstnanec vytváří Pojistku. Klient vytváří Pojistku.



**5) Diskuze smyček -** [**diskuse možných datových nekonzistencí v důsledku smyček ve schématu**](https://dbs.fit.cvut.cz/sw/editor/loop-discussion/)**.**

* Smyček má toto řešení několik, u všech by byl opatrný, všude se snažím používat ID srovnání a CK
* Klient, Kontakt, Banka
* Klient, Kontakt, Banka, Adresa
* Pojistka, Klient, Adresa, Zaměstnanec
* Zaměstnanec, Pobočka, Adresa
* Pojistka, Klient, Kontakt, Zaměstnanec

**6) Integritní omezení (IO)**

IO1: Klient musí mít svoji Adresu.

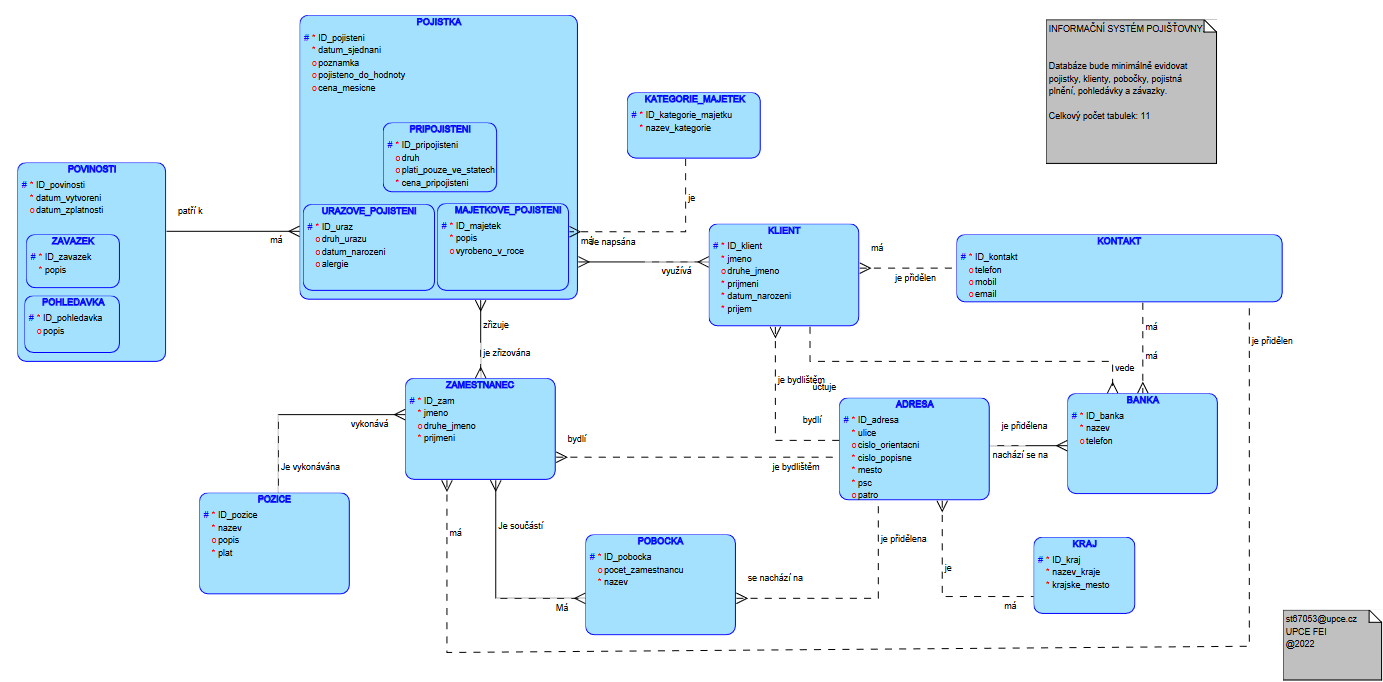
IO2: Plat zaměstnance musí být minimálně ve výši minimální mzdy.

IO3: Pojištění Majetek musí spadat do nějaké kategorie

IO4: Adresa nerozlišuje městské části Prahy

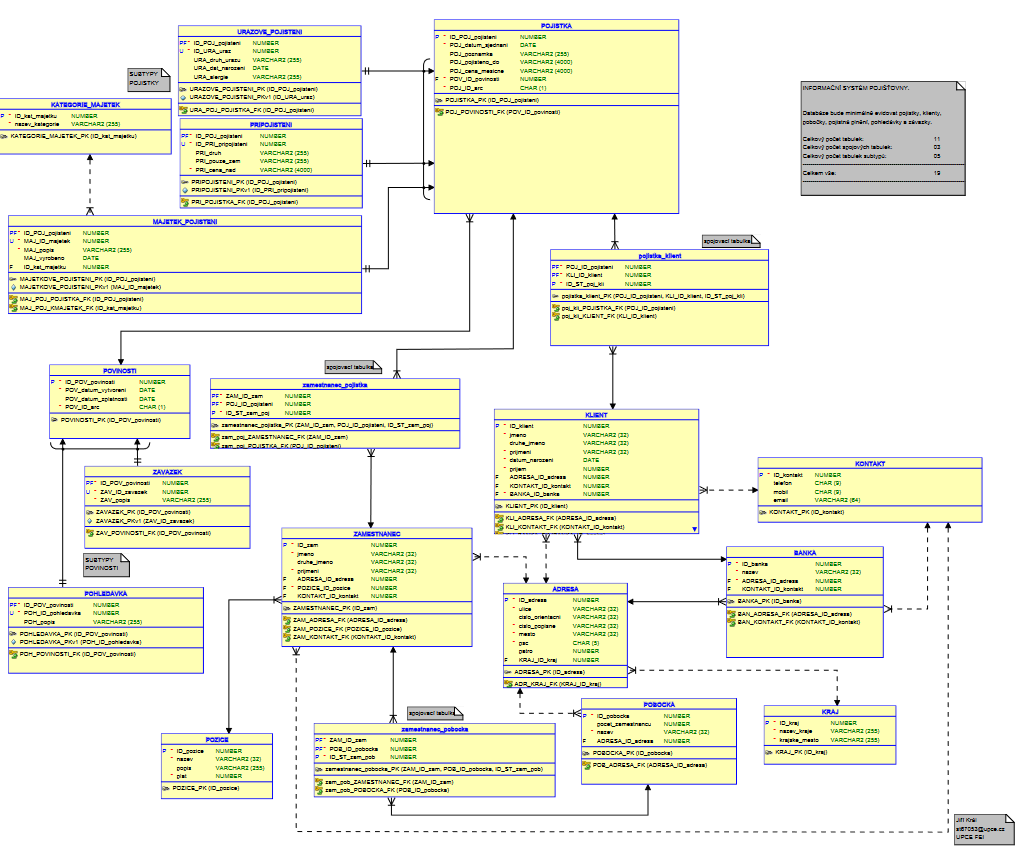
IO5: Pojistka musí mít přiřazené Povinnosti a kategorii pojistky.

## ERD



* ERDish věty:
  + Každé Povinnosti musí a patří k jedné nebo více Pojistkám
  + Každé Pojistky musí mít jednu a pouze jednu Povinnost
  + Každá Kategorie majetku může být u jednoho nebo více Majetkového pojištění
  + Každé Majetkové pojištění může mít jednu a pouze jednu Kategorii majetku
  + Každý Pojistka může být napsaná u jednoho nebo více Klientů
  + Každý Klient musí využívat jednu nebo více Pojistek
  + Každý Klient může mít jeden a pouze jeden Kontakt
  + Každý Kontakt může mít přidělen jednoho nebo více Klientů
  + Každý Klient může účtovat u jedné nebo více Bank
  + Každá Banka může vést jednoho a jenom jednoho Klienta
  + Každý Klient může být bydlištěm na jedné a pouze jedné Adrese
  + Každý Kontakt může mít jednu nebo více Bank
  + Každá Banka může mít jeden a pouze jeden Kontakt
  + Každá Banka musí se nacházet na jedné a pouze jedné Adrese
  + Každá Adresa může být přidělena jedné nebo více Bank
  + Každý Kontakt může být přidělen jednomu nebo více Zaměstnancovy
  + Každá Adresa může být u jednoho a pouze jednoho Kraje
  + Každý Kraj může mít jednu nebo více Adres
  + Každá Adresa může být přidělena jedné nebo více Pobočkám
  + Každá Pobočka může se nacházet na jedné a pouze jedné Adrese
  + Každá Adresa může být u jednoho nebo více zaměstnanců
  + Každý Zaměstnanec může bydlet na jedné a pouze jedné Adrese
  + Každá Pobočka může mít jednoho nebo více Zaměstnanců
  + Každý Zaměstnanec musí být součástí jednoho nebo více Poboček
  + Každý Zaměstnanec musí vykonávat jednu a pouze jednu Pozici
  + Každá Pozice může být vykonávána jedním nebo více Zaměstnancem
  + Každá Pojistka musí být zřizována jedním nebo více Zaměstnanci
  + Každý Zaměstnanec může zřizovat jednu nebo více Pojistek

## Relační model dat



K vytvoření relačního modelu jsem využil funkce Oracle SQL Developer Data Modeler „Engenieer to Relation model“ neboli postav do Relačního modelu. Pak jen zvývalo upravit názvy CK, které přesahovali povolených 30 znaků a uspořádání modelu do čitelné podoby.

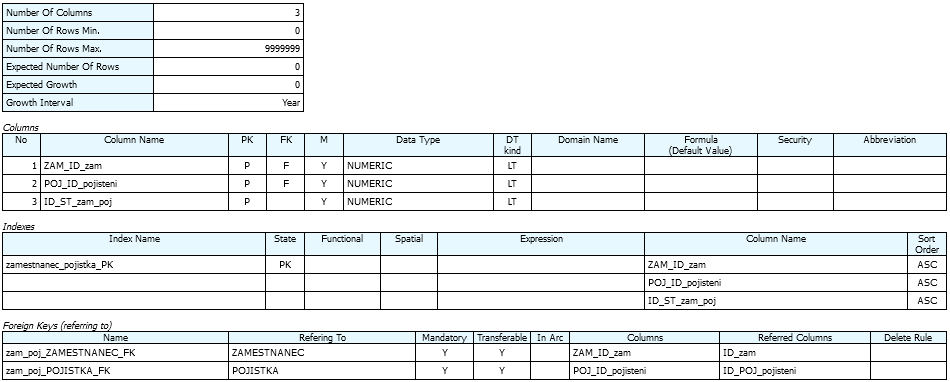
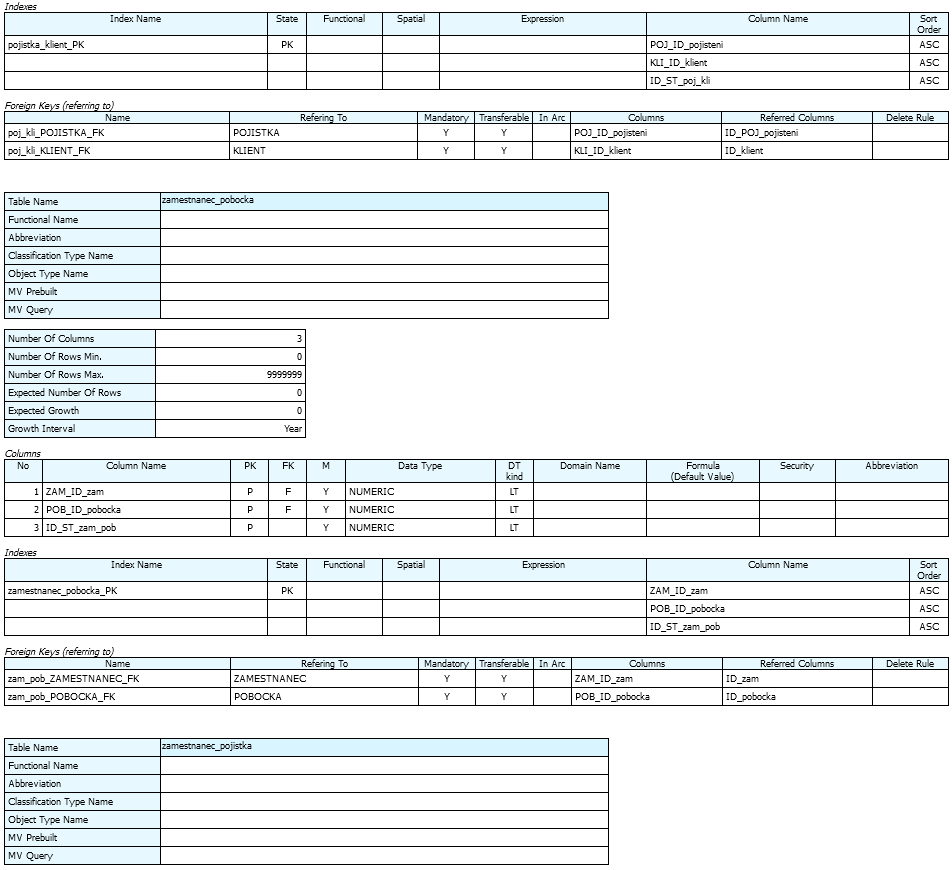
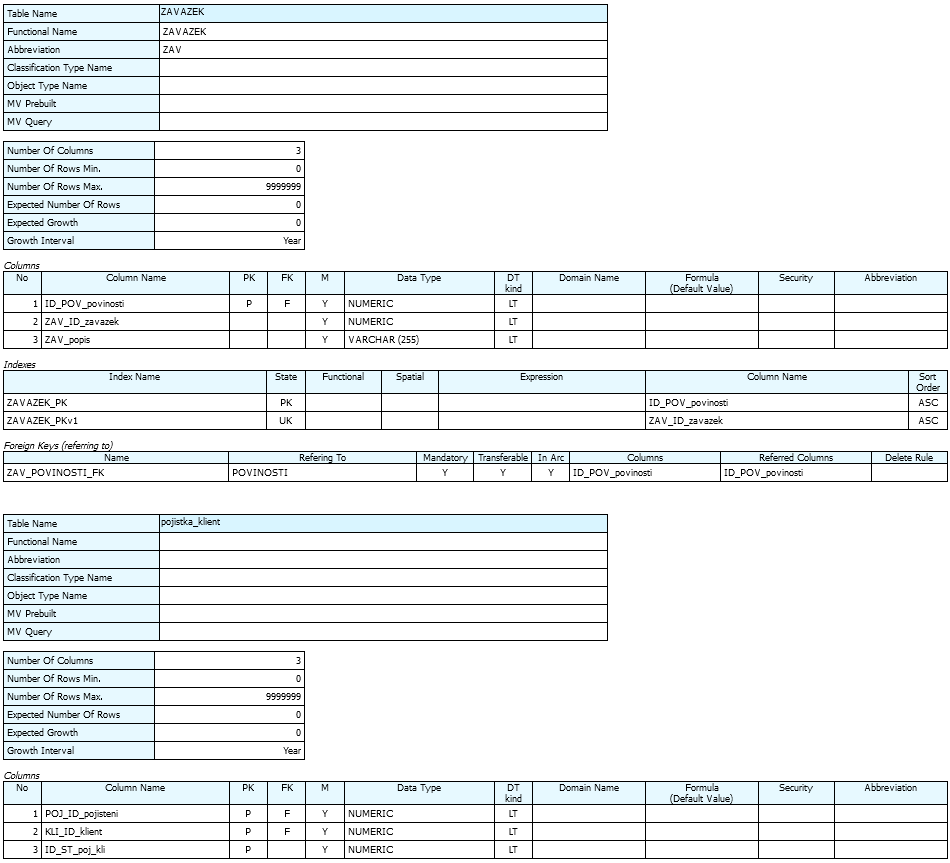
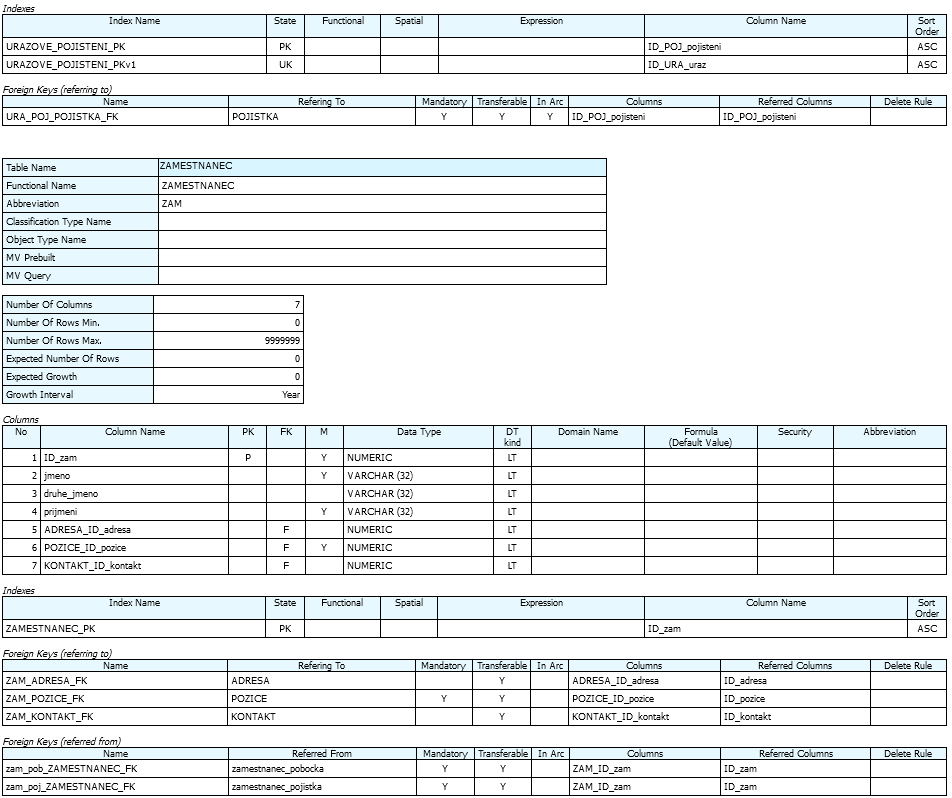
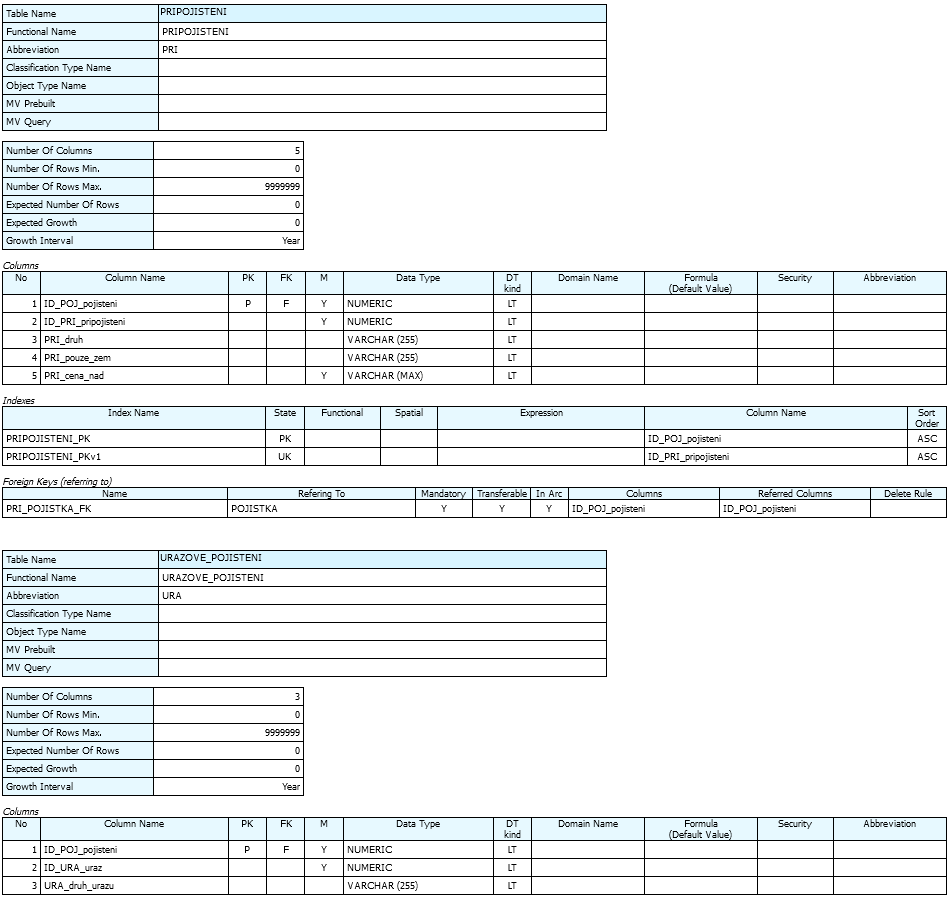
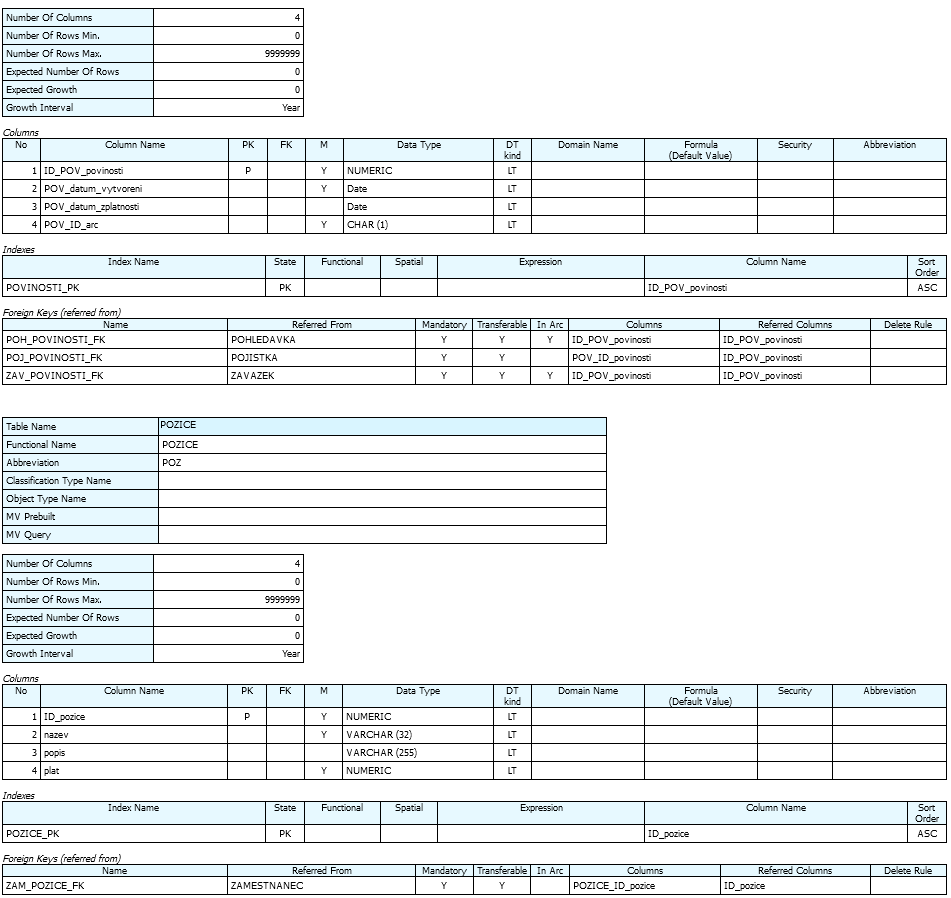
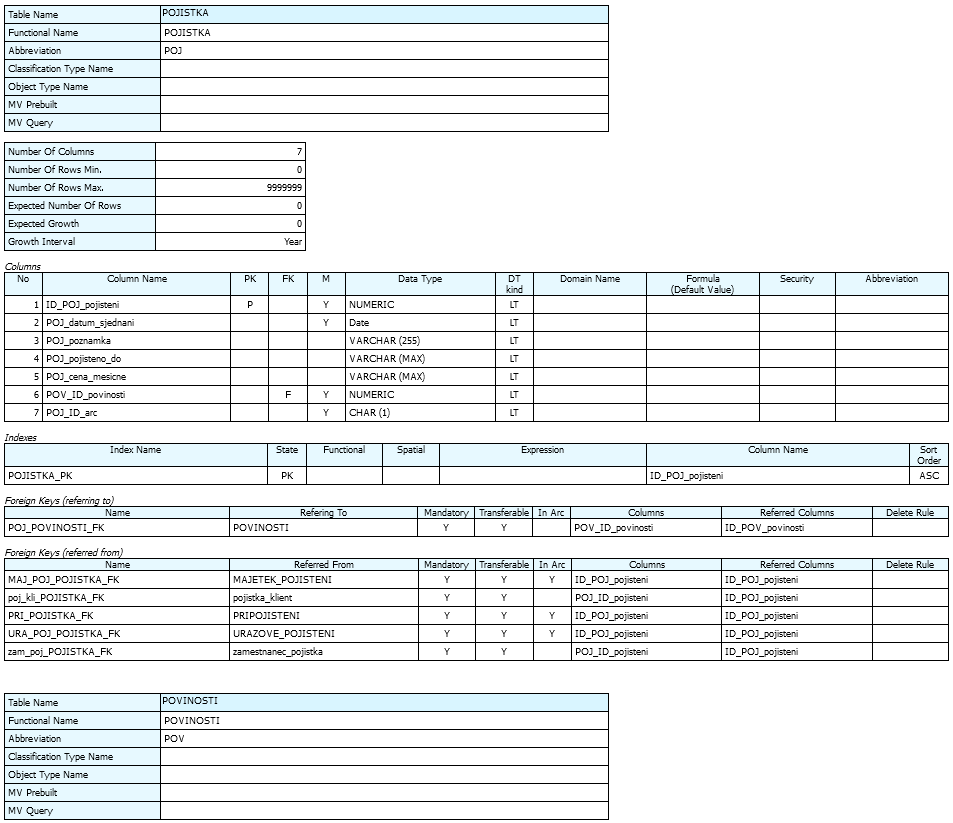
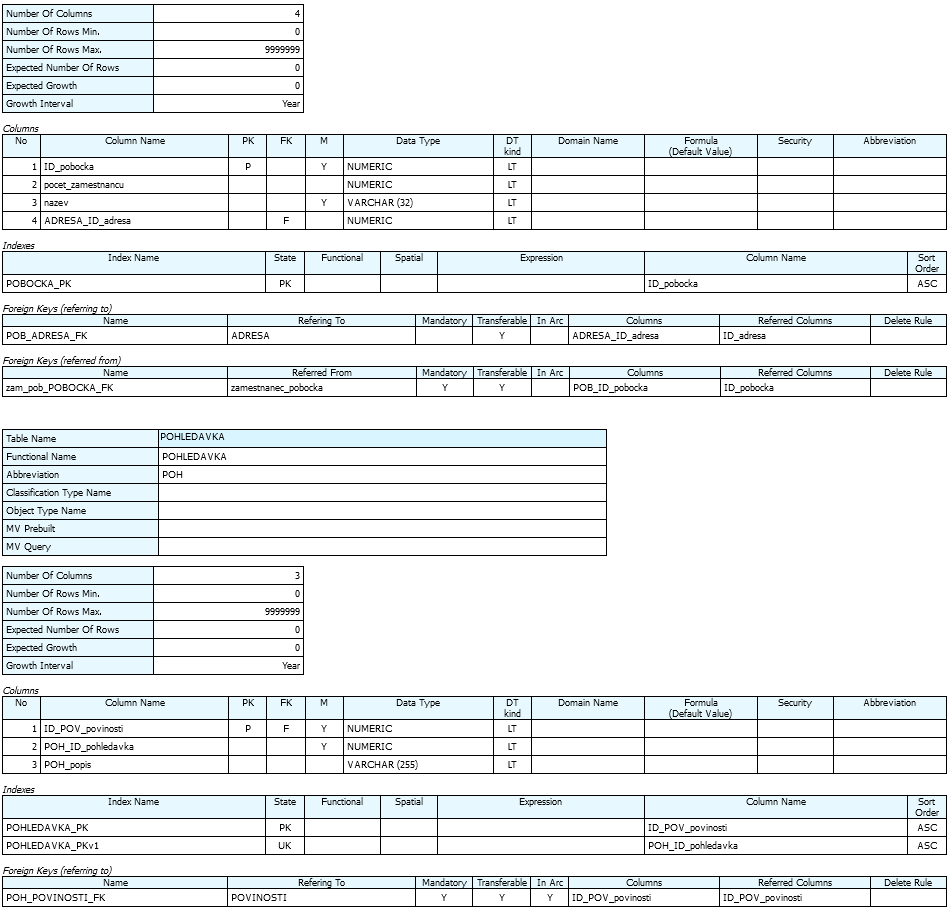
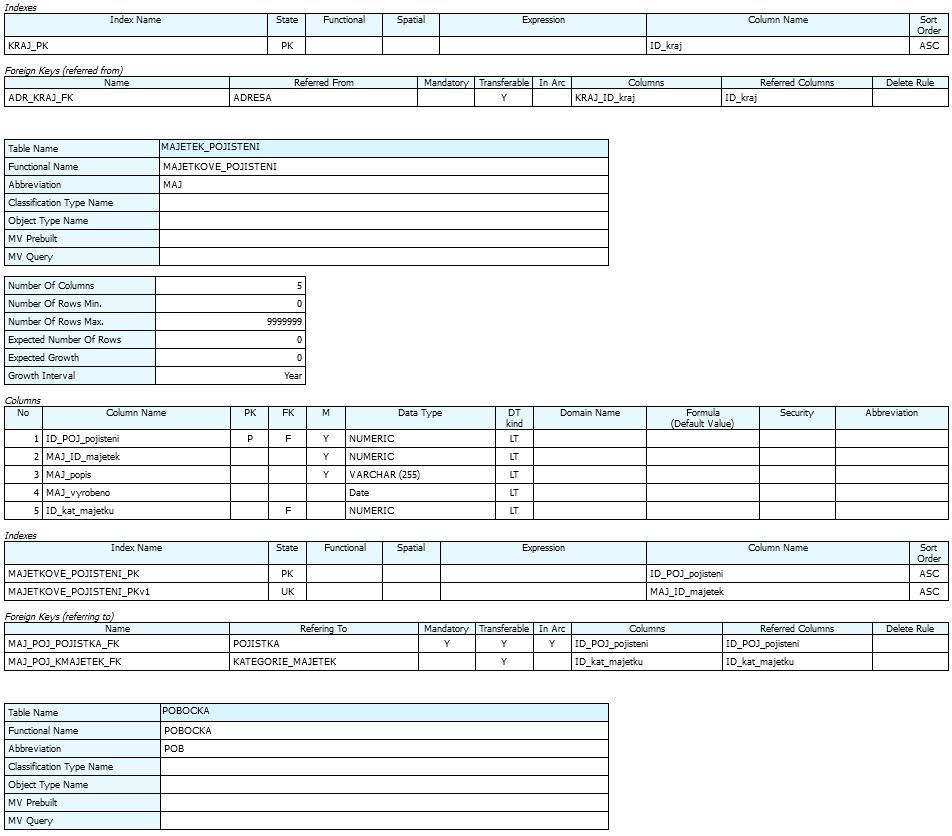
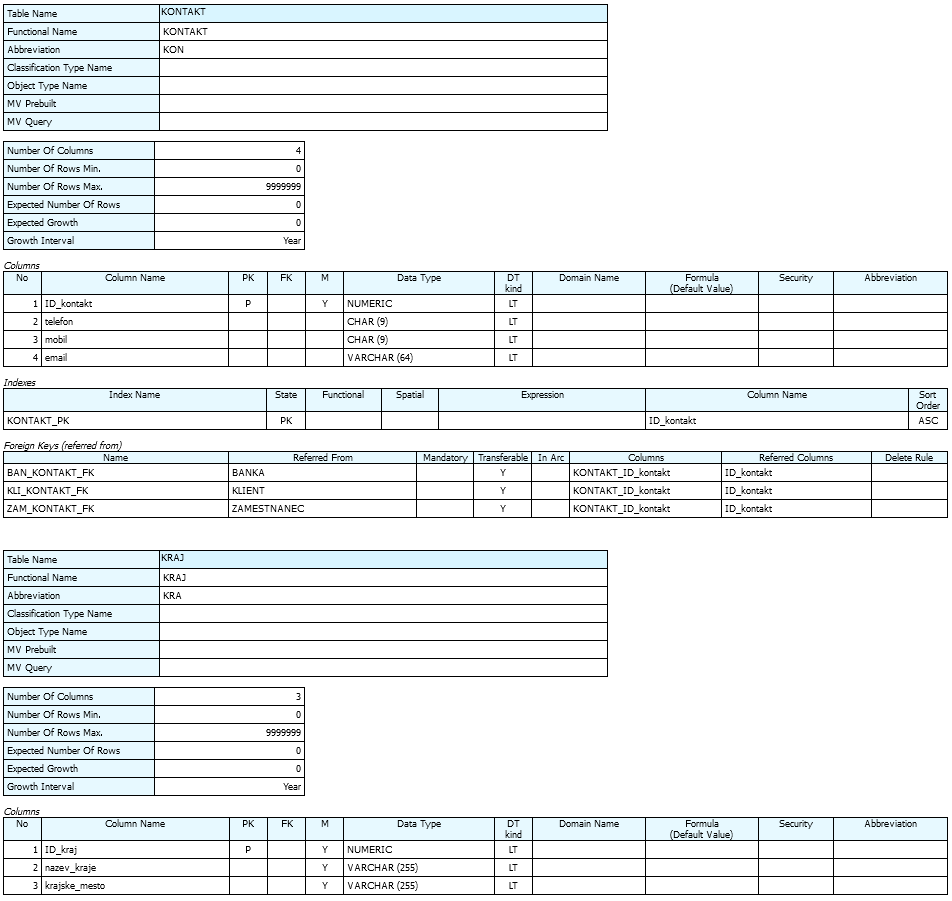
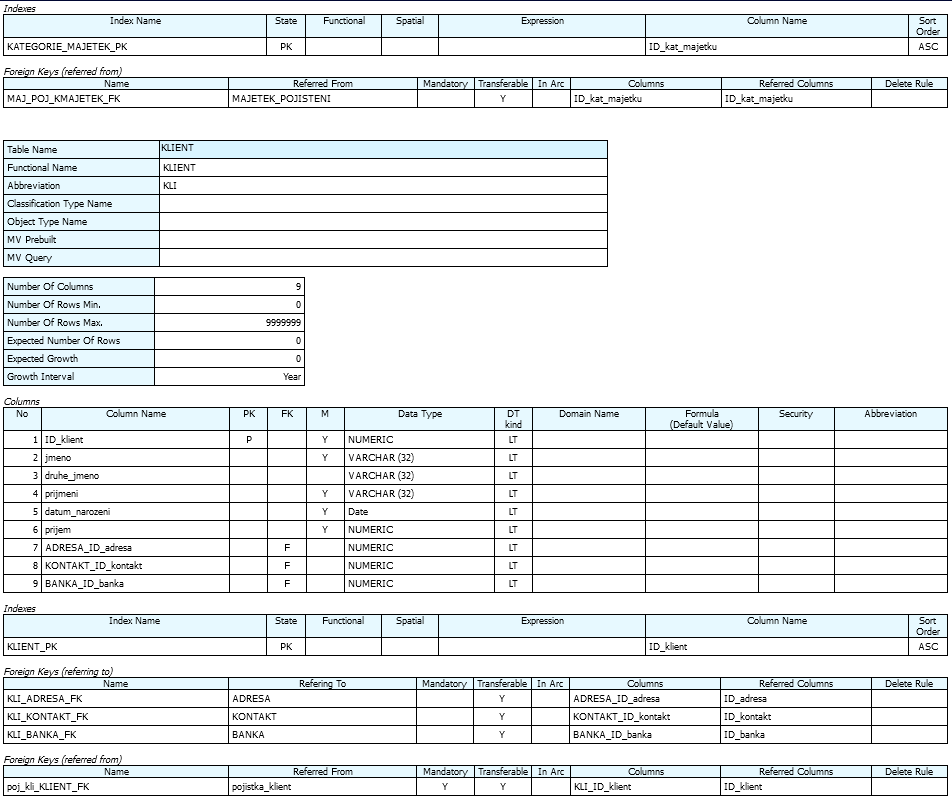
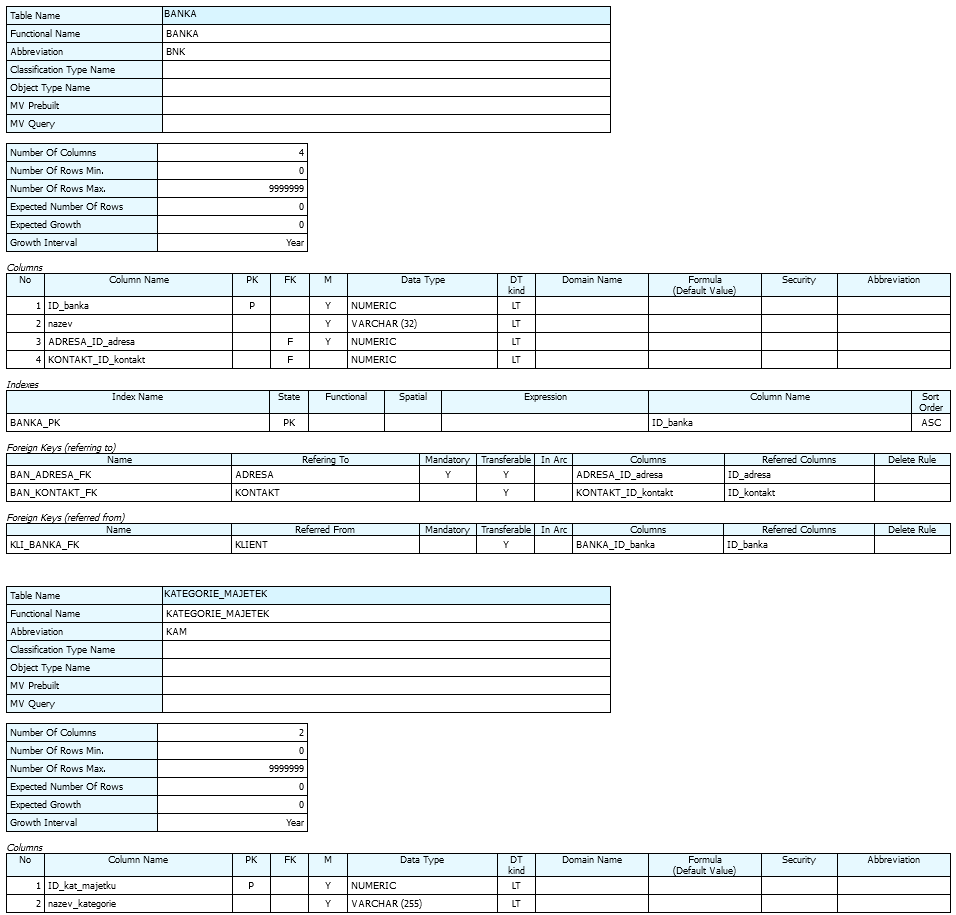
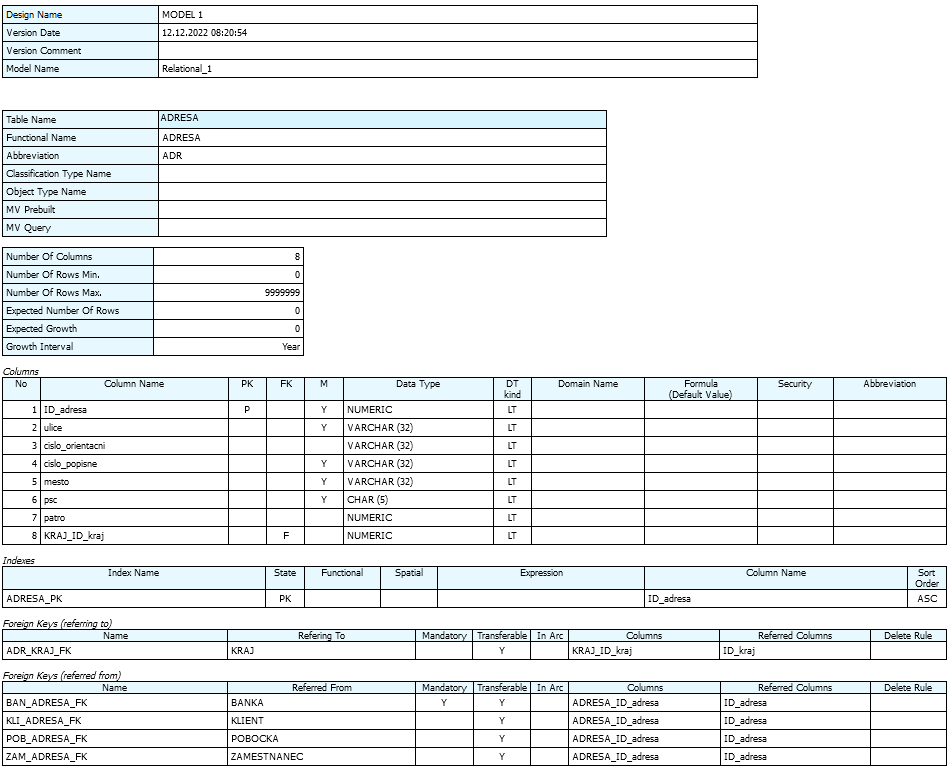
Fyzický model dat

Používám 5 trigerů pro spravování ARC, lišší se převážně podle znaménka, kde v Povinnosti se používá malé **z** nebo **p**, tak v Pojistce se používají velká písmena a to konkrétně **U, M, P**

Obsah obrázku text

Popis byl vytvořen automaticky

Následuje dokumentace vytvořená pomocí data modeleru:



Výčet SQL dotazů

|  |  |
| --- | --- |
| **Kategorie** | **Charekteristika kategorie příkazu** |
| D1 | pozitivní dotaz nad spojením alespoň dvou tabulek například *„vyber seznam se jmény a adresami pacientů, kteří byli vyšetřováni MUDr. Čmelákem“* |
| D2 | negativní dotaz nad spojením alespoň dvou tabulek například: *„seznam se jmény a adresami pacientů, kteří nenavštívili lékaře MUDr. Čmeláka“* |
| D3 | Vyber ty, kteří mají vztah POUZE k …například „*seznam se jmény a adresami pacientů, kteří navštívili pouze MUDr. Čmeláka ~ navštívili doktora Čmeláka a nenavštívili žádného jiného lékaře“* |
| D4 | Vyber ty, kteří/které jsou ve vztahu se všemi - dotaz s univerzální kvantifikací například: *„seznam se jmény a adresami lékařů, které navštívili VŠICHNI pacienti, kteří někdy navštívili MUDr. Čmeláka“* |
| D5 | spojení - JOIN USING |
| D6 | spojení - JOIN ON |
| D7 | spojení - NATURAL JOIN |
| D8 | spojení - CROSS JOIN |
| D9 | polospojení (vnější) - LEFT OUTER JOIN |
| D10 | polospojení (vnější) - RIGHT OUTER JOIN |
| D11 | plné (vnější) spojení - FULL (OUTER) JOIN |
| D12 | vnořený dotaz v klauzuli WHERE |
| D13 | vnořený dotaz v klauzuli FROM |
| D14 | vnořený dotaz v klauzuli SELECT |
| D15 | vztažený vnořený dotaz (EXISTS | NOT EXISTS) |
| D16 | množinové sjednocení - UNION |
| D17 | množinový rozdíl – MINUS |
| D18 | množinový průnik - INTERSECT |
| D19 | Funkce pro práci s řetězci |
| D20 | Funkce pro práci s čísly |
| D21 | Funkce pro práci s datumy |
| D22 | agregační funkce (count | sum | min | max| avg) |
| D23 | agregační funkce nad seskupenými řádky - GROUP BY (HAVING) |
| D24 | stejný dotaz ve třech různých formulacích SQL |
| D25 | všechny klauzule - SELECT FROM WHERE GROUP BY HAVING ORDER BY |
| D26 | pohled VIEW |
| D27 | dotaz nad pohledem alespoň s podmínkou |
| D28 | Příkaz pro vložení množiny řádků - INSERT bez klauzule VALUES, např. „rezervuj pacientovi č. 6 v různých časech všechny lékaře (zubař, neurolog a praktický lékař) |
| D29 | UPDATE s vnořeným SELECT příkazem |
| D30 | DELETE s vnořeným SELECT příkazem |

Dotazy zapisujte postupně takto:

*Označení dotazu:„Zápis dotazu v přirozeném jazyce“, Označení hlavní kategorie např. HK-D1, případně i označení vedlejší kategorie např. VK-D2*

*SQL*

**Příklady:**

A1:"Zobrazí ID\_pobočky pojišťovny a název kraje kde ID\_kraje = 2 '.", HK: D1, VK: D26

CREATE OR REPLACE VIEW A1 AS

select p.ID\_pobocka,k.nazev\_kraje, p.nazev

from pobocka p

JOIN adresa a

ON p.ADRESA\_ID\_ADRESA = a.id\_adresa

JOIN kraj k

ON a.kraj\_ID\_kraj = k.ID\_kraj

where kraj\_id\_kraj = 2;

select \* from A1;

A2:" Zobrazí všechny ID\_poboček pojišťovny a názvy kraje kde ID\_kraje není 2.", HK: D2, VK: D26  
CREATE OR REPLACE VIEW A2 AS

select p.ID\_pobocka,k.nazev\_kraje, p.nazev

from pobocka p

JOIN adresa a

ON p.ADRESA\_ID\_ADRESA = a.id\_adresa

JOIN kraj k

ON a.kraj\_ID\_kraj = k.ID\_kraj

where kraj\_id\_kraj != 2;

select \* from A2;

/

A3:"Zobraz všechny města které nejsou v Praze (ID = 1) nebo v Jihomoravském kraji", HK: D3 , VK: D26

CREATE OR REPLACE VIEW D3 AS

select a.mesto , k.nazev\_kraje

from adresa a

JOIN kraj k

ON a.kraj\_ID\_kraj = k.ID\_kraj

where k.ID\_kraj > 2 and (k.nazev\_kraje != 'Jihomoravsky kraj');

select \* from D3;

/

A4: „Zjišťování jestli město patří pod Moravskoslezský nebo Jihomoravský kraj.“, HK:D4, VK: D26

CREATE OR REPLACE VIEW D4 AS

select a.mesto nazev\_mesta, k.nazev\_kraje "Pod který kraj toto město patří patří:"

from adresa a

join kraj k

on a.kraj\_ID\_kraj = k.ID\_kraj

where k.nazev\_kraje = 'Jihomoravsky kraj' OR k.nazev\_kraje = 'Moravskoslezsky kraj'

order by k.nazev\_kraje DESC;

select \* from D4;

/

A5: „Seznam pojištěných věcí spolu s kategorií, kam patří.“, HK:D5, VK: D26

CREATE OR REPLACE VIEW D5 AS

select KATEGORIE\_MAJETEK.nazev\_kategorie "Kategorie pojištění", MAJETEK\_POJISTENI.MAJ\_popis "pojištěná věc"

from KATEGORIE\_MAJETEK join MAJETEK\_POJISTENI

using (ID\_KAT\_majetku);

select \* from D5;

A8:“Jméno zaměstnance a jeho pozice“, HK: D8, VK: D26

CREATE OR REPLACE VIEW D8 AS

SELECT zam.POZICE\_ID\_pozice, zam.jmeno, poz.nazev "Název postu"

FROM zamestnanec zam

CROSS JOIN pozice poz

where ID\_pozice = POZICE\_ID\_pozice

order by zam.jmeno;

select \* from d8 ;

A9:“Příjmení, id a název pracoviště kde pracuje Zaměstnanec, řazeno podle příjmení“, HK:D9, VK: D26

CREATE OR REPLACE VIEW D9 AS

SELECT z.prijmeni "Přijmení zamšstnance",z.ID\_zam "id Zaměstnance", p.nazev "Název postu"

FROM zamestnanec z

LEFT JOIN pozice p ON z.POZICE\_ID\_pozice = p.ID\_pozice

ORDER BY z.prijmeni;

select \* from d9;

A10:“Seznam všech přirřazených i nepřiřazených pozic. Řazeno podle příjmení, null hodnoty poté“, HK:D10, VK: D26

CREATE OR REPLACE VIEW D10 AS

SELECT z.prijmeni "Přijmení zamšstnance",z.ID\_zam "id Zaměstnance", p.nazev "Název postu"

FROM zamestnanec z

RIGHT JOIN pozice p ON z.POZICE\_ID\_pozice = p.ID\_pozice

ORDER BY z.prijmeni;

select \* from d10;

A11:“Seřad zaměstnance podle platu, nejvyšší plat na začátek “, HK:D11, VK: D26

CREATE OR REPLACE VIEW D11 AS

SELECT z.prijmeni "Přijmení zamšstnance",z.ID\_zam "id Zaměstnance", p.nazev "Název postu", p.plat "výplata"

FROM zamestnanec z

FULL OUTER JOIN pozice p

ON z.POZICE\_ID\_pozice = p.ID\_pozice

WHERE ID\_pozice = POZICE\_ID\_pozice

ORDER BY p.plat DESC;

select \* from d11;

A12:“vybere všechny klienty, kteří mají vyplněny všechny údaje“, HK:D12, VK: D26

CREATE OR REPLACE VIEW D12 AS

SELECT klient.ID\_klient, klient.jmeno, klient.druhe\_jmeno, klient.prijmeni, SUB1.telefon, SUB1.mobil, SUB1.email

FROM klient,

(SELECT kontakt.ID\_kontakt, kontakt.telefon, kontakt.mobil, kontakt.email

FROM kontakt

WHERE kontakt.telefon != 'null') SUB1

WHERE SUB1.ID\_kontakt = klient.KONTAKT\_ID\_kontakt AND

klient.druhe\_jmeno != 'null';

select \* from D12;

A13:“vybere všechny informace o klientovy“, HK:D13, VK: D26

idk;

A14:“vybere všechny informace o klientovy“, HK:D14, VK: D26

CREATE OR REPLACE VIEW D14 AS

select \*

from (select \* from klient);

select \* from D14;

A15:“Zobrazí pozice, které jsou přiděleny zaměstnancům a berou více jak 50000“, HK:D15, VK: D26

CREATE OR REPLACE VIEW D15 AS

SELECT \*

FROM pozice p

WHERE EXISTS (SELECT z.ID\_zam, z.jmeno

FROM zamestnanec z

WHERE z.POZICE\_ID\_pozice = p.ID\_pozice

AND p.plat > 50000);

select \* from D15;

A16:“ “, HK:D13, VK: D26

;

A17:“Vyber zaměstnance co jsou mimo rozsah“, HK:D17, VK: D26

CREATE OR REPLACE VIEW D17 AS

SELECT zam.ID\_zam ,zam.jmeno, zam.prijmeni, poz.plat FROM zamestnanec zam JOIN pozice poz ON poz.ID\_pozice = zam.POZICE\_ID\_pozice where poz.plat > 10000

MINUS

SELECT zam.ID\_zam ,zam.jmeno, zam.prijmeni, poz.plat FROM zamestnanec zam JOIN pozice poz ON poz.ID\_pozice = zam.POZICE\_ID\_pozice where poz.plat < 80000;

select \* from D17;

A18:“ Najdi kraj, který je v rozsahu id\_kraj 10 až 12“, HK:D18, VK: D26

CREATE OR REPLACE VIEW D18 AS

select id\_kraj , nazev\_kraje from kraj where id\_kraj > 10

intersect

select id\_kraj , nazev\_kraje from kraj where id\_kraj < 12;

select \* from D18;

A19:“ Zjišťuje délku druhu úrazu“, HK:D19, VK: D26

CREATE OR REPLACE VIEW D19 AS

select URA\_druh\_urazu, LENGTH(URA\_druh\_urazu) as delka from urazove\_pojisteni;

select \* from D19;

A20:“Hledání jestli je vyšší číslo pevné linky nad mobilním telefonem “, HK:D20, VK: D26

CREATE OR REPLACE VIEW D20 AS

select ID\_kontakt, mobil, telefon from kontakt where telefon > mobil order by ID\_kontakt asc;

select \* from D20;

A21:“ Seřaď Klienty podle toho, kolik let uplynulo od jejich narození“, HK:D21, VK: D26

CREATE OR REPLACE VIEW D21 AS

select ID\_klient, jmeno, prijmeni,datum\_narozeni,TO\_CHAR((MONTHS\_BETWEEN(CURRENT\_DATE, datum\_narozeni) / 12), 999.99) as "uplynulo let"

from klient

order by "uplynulo let" desc;

select \* from D21;

A22:“ Výpočet průměrného platu v CZK a USD“, HK:D22, VK: D26

CREATE OR REPLACE VIEW D22 AS

select TO\_CHAR(AVG(plat), '999,999,999.99') AS "Průměrný plat v CZK" , TO\_CHAR(AVG(plat \* 24), '$999,999,999.99') as "Průměrný plat v USD" from pozice;

select \* from D22;

A23:“Vyhledává nejnižší plat v oddělení “, HK:D23, VK: D26

CREATE OR REPLACE VIEW D23 AS

SELECT nazev, MIN(plat) AS "nejnižší plat v oddělení"

FROM pozice

GROUP BY nazev;

select \* from D23;

A24:“Vypisování veškerých dat z kraje, různé styly zápisu “, HK:D24, VK: D26

CREATE OR REPLACE VIEW D24a AS

select \* from kraj where nazev\_kraje != null OR ID\_kraj < 15;

CREATE OR REPLACE VIEW D24b AS

select ID\_kraj, nazev\_kraje from kraj;

CREATE OR REPLACE VIEW D24c AS

select \* from kraj where ID\_kraj <= 14 AND nazev\_kraje IS NOT NULL;

select \* from D24a;

select \* from D24b;

select \* from D24c;

A25:“Najdi města kde jsou nejvyšší patra “, HK:D25, VK: D26

CREATE OR REPLACE VIEW D25 AS

SELECT mesto, MAX(patro) AS "nejvyšší patro"

FROM adresa

GROUP BY mesto

HAVING MAX(patro) > 2

order by MAX(patro) desc;

select \* from D25;

A26:“vytvoř pohled na select co vybírá adresu “, HK:D26

CREATE OR REPLACE VIEW D26 AS

select ulice, cislo\_popisne ,cislo\_orientacni

from adresa;

select \* from D26;

A27:“spojování view a následný upravený select “, HK:D27, VK: D26

CREATE OR REPLACE VIEW D27 AS

select b.nazev\_kraje, a.krajske\_mesto

FROM D24a a

inner join D24b b

on a.ID\_kraj = b.ID\_kraj;

select \* from D27 where krajske\_mesto = 'Praha';

A28:“ vkládání do databáze bez VALUES“, HK:D28

INSERT INTO pozice

(ID\_pozice, nazev, popis, plat )

SELECT 100, 'Automizator vsehomira','zapina lidem pocitace', 87452

FROM dual

WHERE NOT EXISTS (SELECT \* FROM pozice WHERE pozice.ID\_pozice = 100);

A29:“Úprava dat “, HK:D29

UPDATE pozice

SET nazev = 'Automizator nehonira'

WHERE ID\_pozice = 100;

A30:“Odstranení dat “, HK:D30

DELETE FROM pozice

WHERE ID\_pozice = (

SELECT ID\_pozice FROM pozice

WHERE nazev ='Automizator nehonira');

Závěr

I přes porodní bolesti jsem dokázal připravit práci do odevzdatelné podoby. Z mého pohledu by tato databáze zasloužila ještě hromadu optimalizace než bude z mého pohledu připravena na vyslání pro reálnou pojišťovnu. Jinak práce mi ukázala jak jednoduché je ztratit čas na blbostech, kolik kofeinu zvládne lidské tělo nasát a jak moc je důležité začít včas. Práce mi taky připomněla, že být megalomanem se nevyplácí, model je zbytečně složitý a spolu s hromadou smyček má i hromadu problémů. I přes to celý systém funguje, což jsem upřímně rád.

Přílohy

1. DDL skript pro vytvoření tabulek, omezení, sekvencí, indexů, triggerů, apod.
2. DML skript pro naplnění tabulek daty.
3. Soubor dmd včetně složky (Datový model musí souhlasit s DDL skriptem)
4. Skript se všemi dotazy.