UNIVERZITA PARDUBICE  
Fakulta elektrotechniky a informatiky

*systém zubní kliniky /* Databázové systémy I Seminární práce z předmětu Databázové systémy I

Jméno a příjmení: Ahmad Akel

V Pardubice dne 10/12/2021

Obsah

[Analýza 10](#_Toc70355807)

[ERD 13](#_Toc70355808)

[Relační model dat 16](#_Toc70355809)

[Fyzický model dat 17](#_Toc70355810)

[Výčet SQL dotazů 18](#_Toc70355811)

[Závěr 22](#_Toc70355812)

[Přílohy 23](#_Toc70355813)

Analýza

**1)** Databáze zubní kliniky bude obsahovat 13 objektů spolupracujících na poskytování mnoha služeb pro kliniku, lékaře a sestry, kteří na klinice pracují, a pacienty, kteří na kliniku přicházejí.

* První tabulka je tabulka adres, která poskytne údaje o adrese klinik, sester, lékařů, pacientů.
* Klinický tabulka bude obsahovat klinické informace
* Oddělení bude obsahovat informace o odděleních a bude pod stolem kliniky, pak do tohoto stolu budou patřit lékaři a sestry
* Tabulka lékařů obsahuje informace o lékaři a kontakty
* Sestra tabulka obsahuje informace o sestře a kontakty
* Tabulka pacientů obsahuje informace o pacientovi a kontakty
* Schůzky budou obsahovat informace o schůzkách mezi pacientem a lékařem
* Tabulka pojišťoven bude obsahovat informace o pojišťovně a kontakty a poskytne mnoho typů pojištění pro pacienty
* Tabulka pojištění bude obsahovat informace o pojištění pro pacienta
* Tabulka léčby bude obsahovat informace o léčbě a lécích, které budou mezi lékařem a pacientem
* Tabulka léků bude mít název a ceny léků

**2) Strukturální a procedurální pravidla**

Ve své práci jsem nepoužil proceduru

* **3) Address**

1. *address\_id*
2. *country*
3. *city*
4. *zip\_code*
5. *street\_number*

* **Clinic**

1. *Clinic\_id*
2. *Name*
3. *Mail*
4. *Phone\_number*

* **Department**

1. *Department\_id*
2. *Name*
3. *Mail*
4. *Phone\_number*

* **Doctor**

1. *Doctor\_id*
2. *First\_name*
3. *Last\_name*
4. *Mail*
5. *Phone\_number*

* **Nurse**

1. *Nurse\_id*
2. *First\_name*
3. *Last\_name*
4. *Mail*
5. *Phone\_number*

* **Patient**

1. *Patient\_id*
2. *First\_name*
3. *Last\_name*
4. *Mail*
5. *Phone\_number*

* **Child**

1. *Child\_id*
2. *Deciduous\_teeth*

* **Young**

1. *Young\_id*

* **Appointment**

1. *appointment\_id*
2. *date*

* **Insurance\_Company**

1. *Insurance\_company\_id*
2. *Name*
3. *Mail*
4. *Phone\_number*

* **Insurance**

1. *insurance\_id*
2. *insurance\_type*
3. *insurance\_number*

* **Treatemnt**

1. *Treatment\_id*
2. *Fees*
3. *Medicine\_name*
4. *Date*

* **Medicine**

1. *Medicine\_id*
2. *Name*

*price*

**4) CRUD analýza**

systém bude schopen registrovat lékaře, sestry a pacienty a aktualizovat je a mazat, také registrovat informace v ostatních tabulkách poskytujících možnost aktualizace, mazání

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

smyčky nejsou ve schématu

**6) Integritní omezení (IO)**

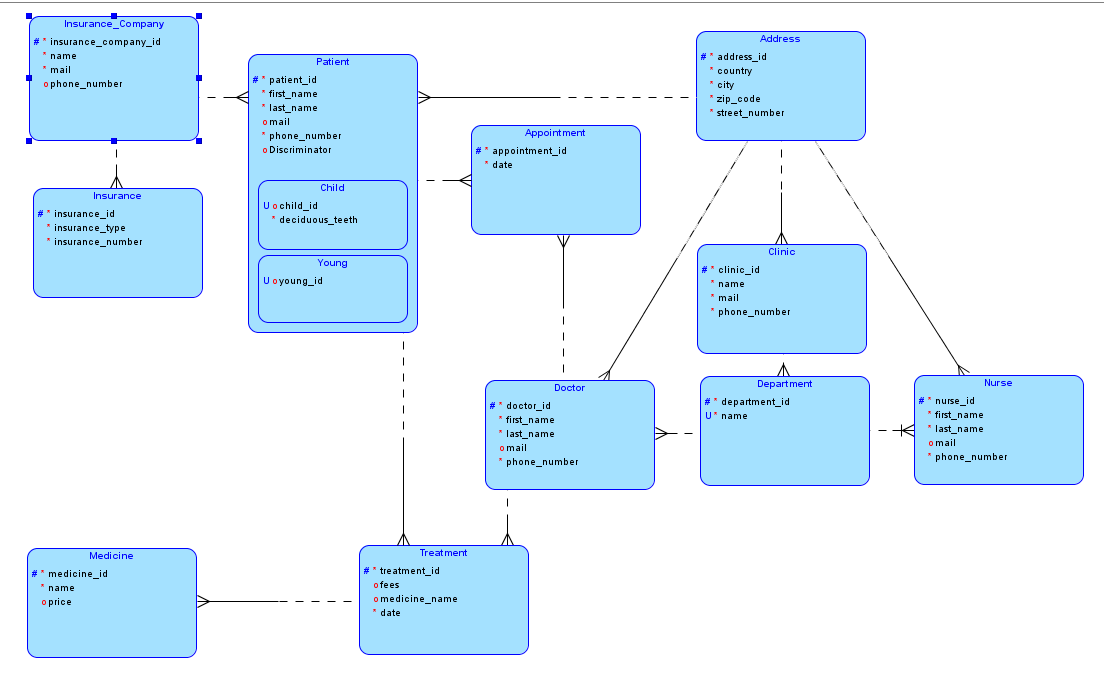
IO1: Lékař a pacient nemohou mít dvě schůzky současně.

IO2: Není možné mít dvě oddělení se stejným názvem.

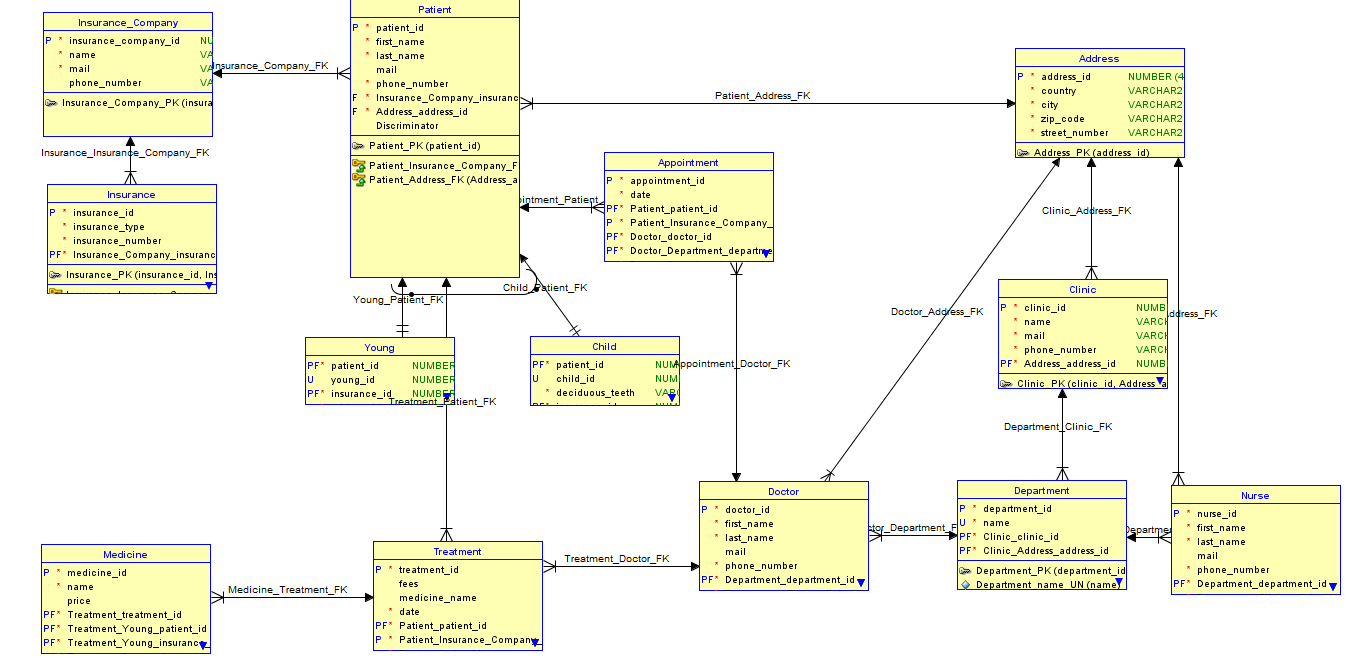
IO3: Děti by neměly dostávat léky pro dospělé.

IO4: Sestra nebo lékař nemohou pracovat na různých odděleních současněIO5: Řidič autobusu nemůže současně řídit pronajatý.

## ERD



## Relační model dat



* Tabulka Address bude mít vztah 1:M s tabulkami:

Clinic, Doctor, Nurse, Patient.

Protože každý řádek v tabulce v každé z těchto tabulek bude mít jednu adresu a každý řádek v tabulce Address může mít mnoho řádků těchto tabulek.

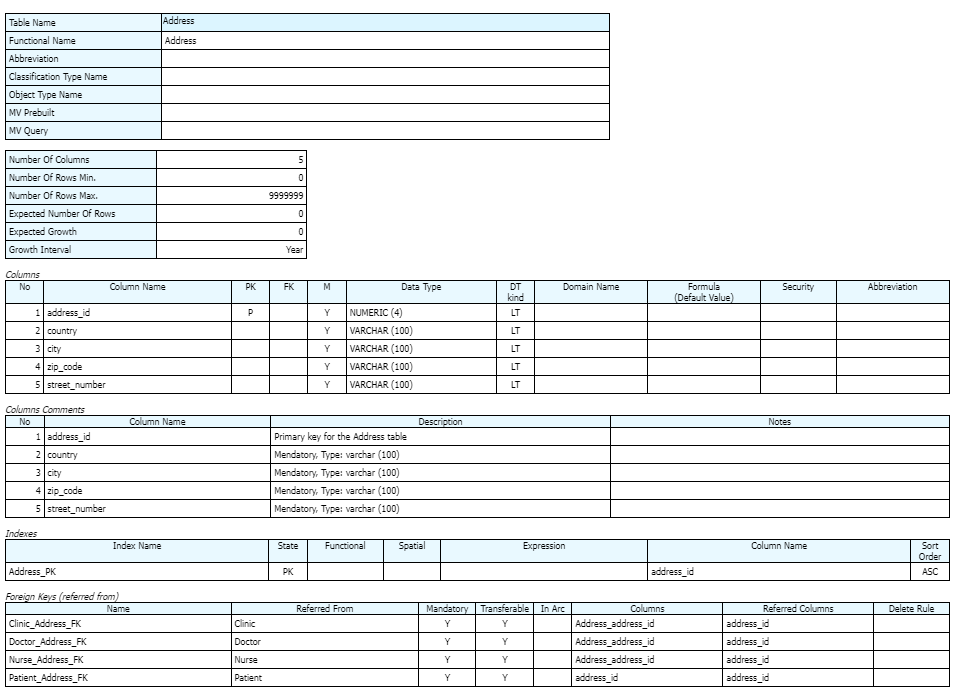
* Každá Clinic může mít mnoho Department a každé Department patří ke Clinic, proto je vztah 1:M.
* Department může mít mnoho lékařů a sester a každý lékař a sestra patří na jedno Department, takže vztah mezi nimi

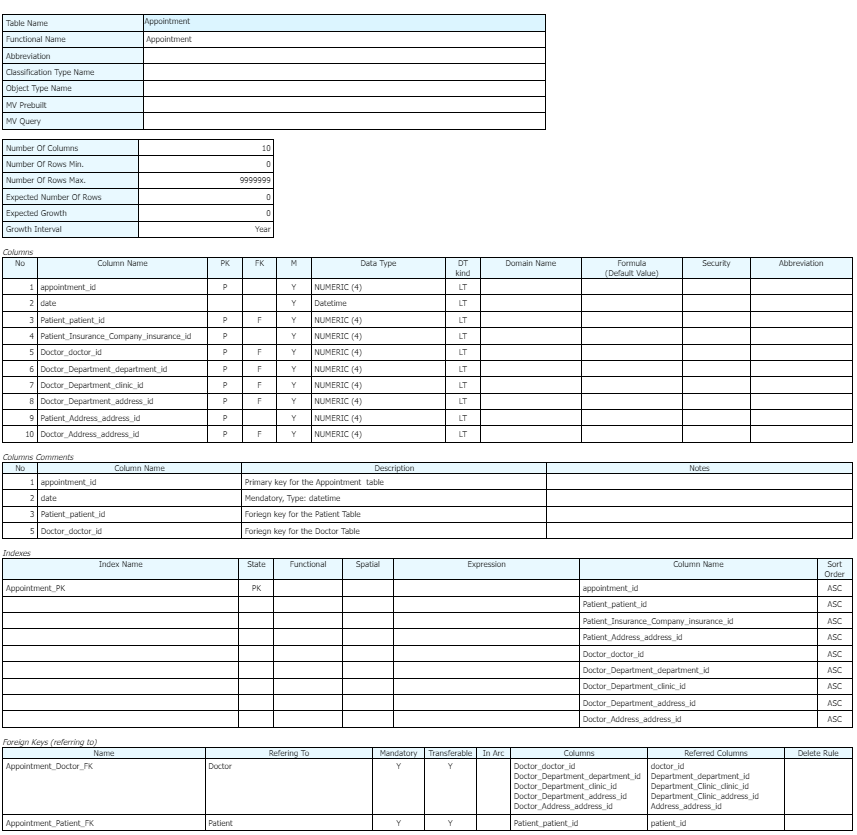
Tabulka Department a tabulce (Doctor, Nurse) je 1:M.

* Tabulky Doctor a Patient mohou mít mnoho schůzek, pak vztah mezi tabulkou Appointment a tabulkami: vztah Doctor, Patient M:1
* Totéž pro léčebný stůl, pacienti a lékaři mohou mít mnoho ošetření a léčba bude mít jednoho lékaře a jednoho pacienta, pak vztah mezi tabulkou Treatment a tabulkami: vztah Doctor, Patient M:1
* vztah mezi Insurance\_company a Patient je 1:M, protože každá pojišťovna může mít hodně pacientů, ale pacient může být registrován u jedné pojišťovny
* vztah mezi Insurance\_company a Insurance je 1:M, protože každá pojišťovna může mít mnoho druhů pojištění, ale pojištění může patřit jedné pojišťovně
* vztah mezi Treatment a Medicineje 1:M, protože každá Léčbamůže mít mnoho druhů medicína, ale medicína může patřit jedné Léčbamůže
* pro podtabulky Child a Young zdědí atributy supertabulkové tabulky pacientů, aby určili věk pacienta, a rozdělí je do dvou tabulek, aby se s každou z nich zacházelo samostatně.

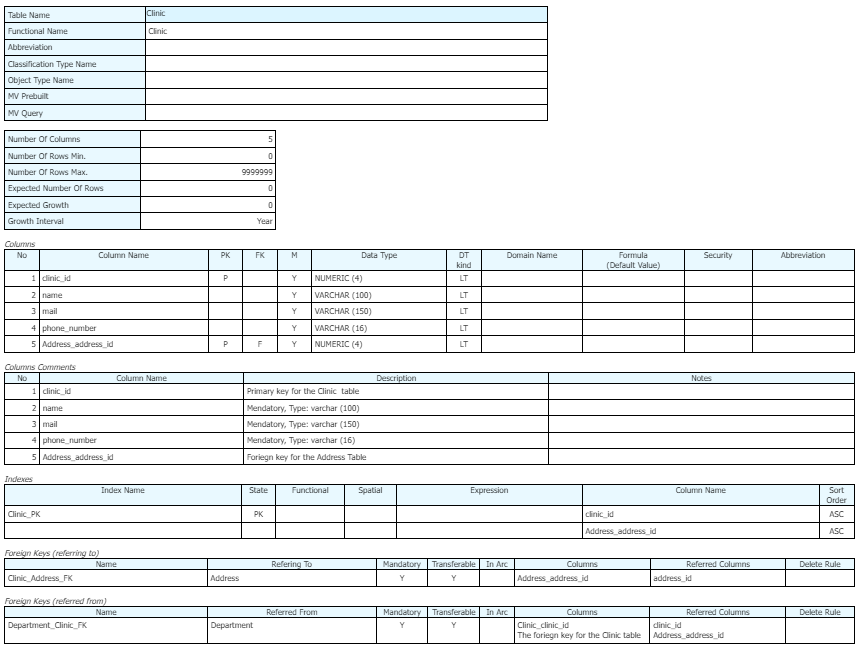
Fyzický model dat

1-Addresses

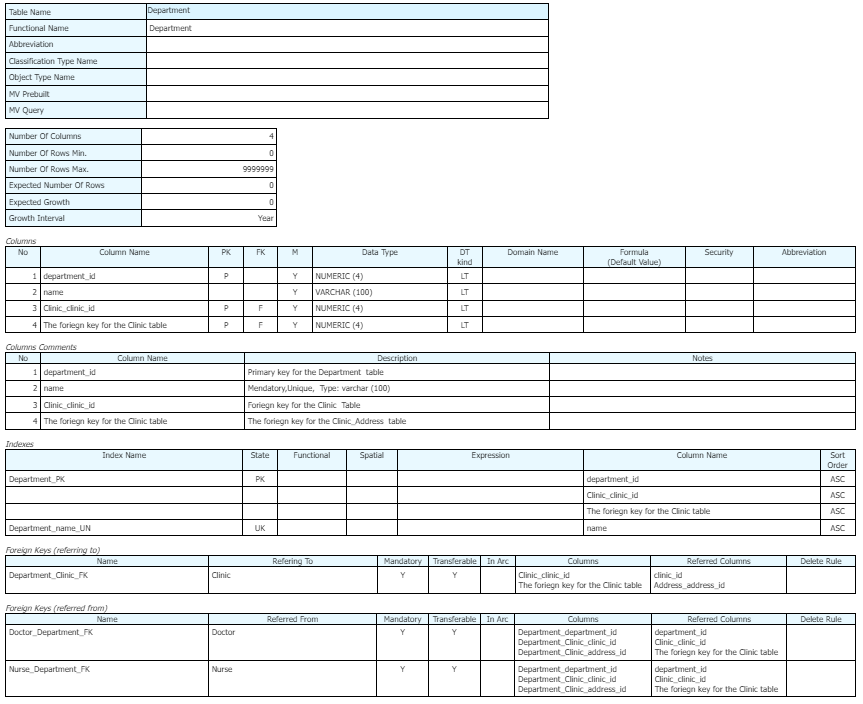


2-Appointment

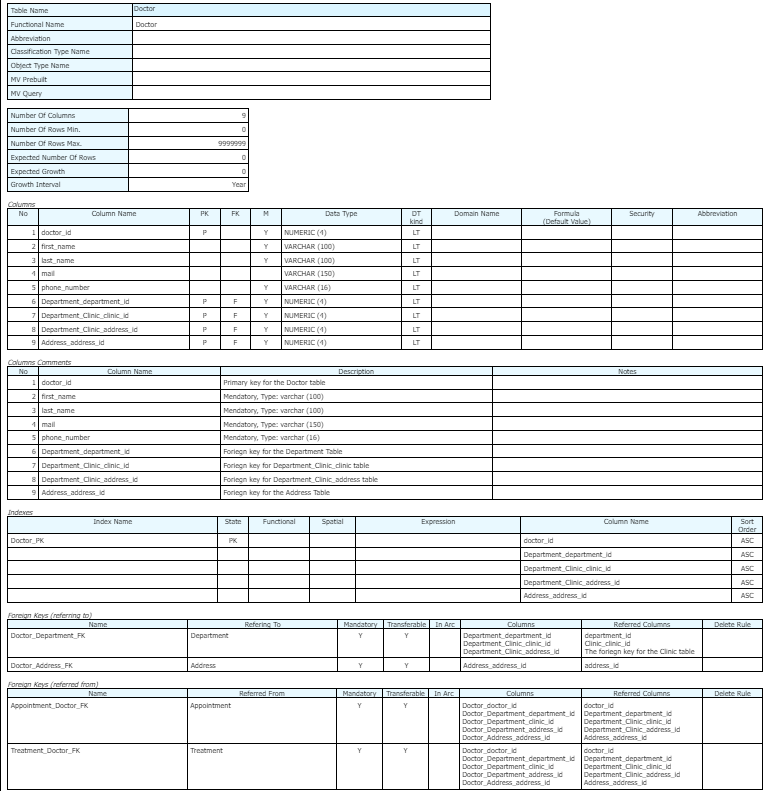
3-Clinic



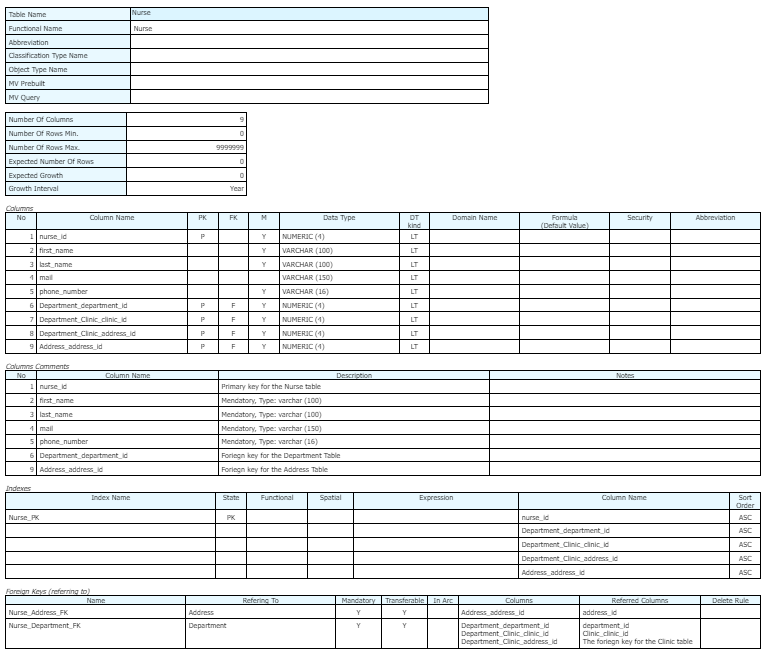
4-Department



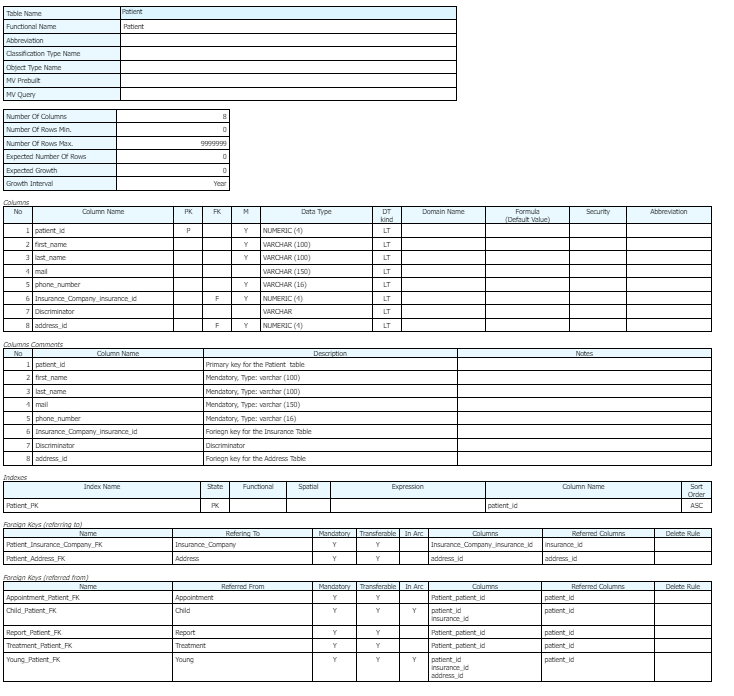
5-Doctor



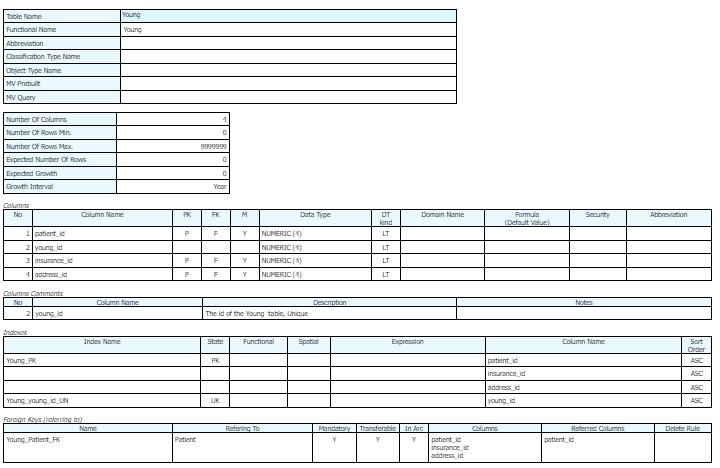
6-Nurse



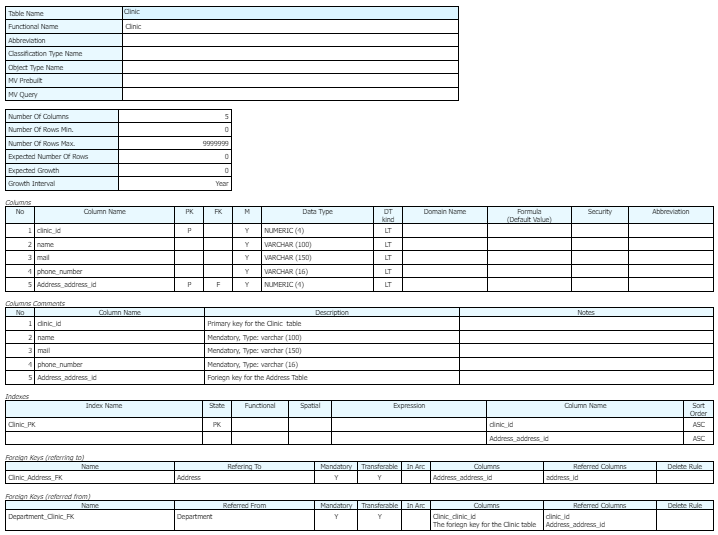
7-Patient



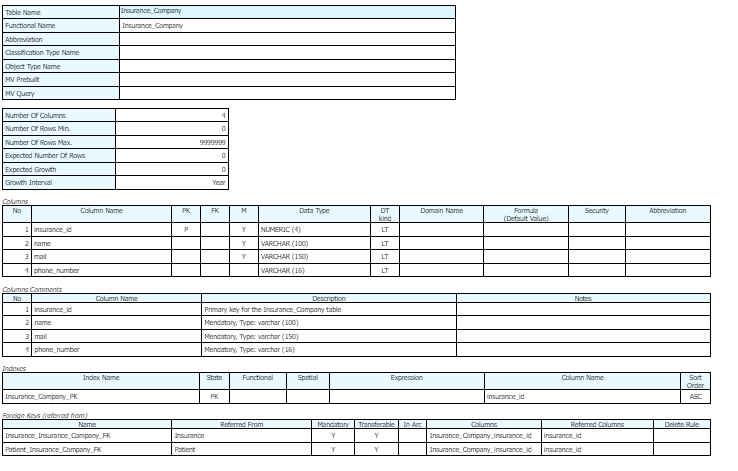
8-Young



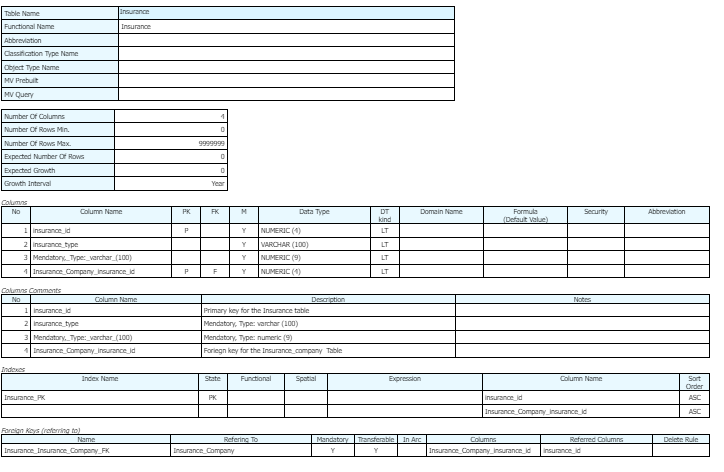
9-Child



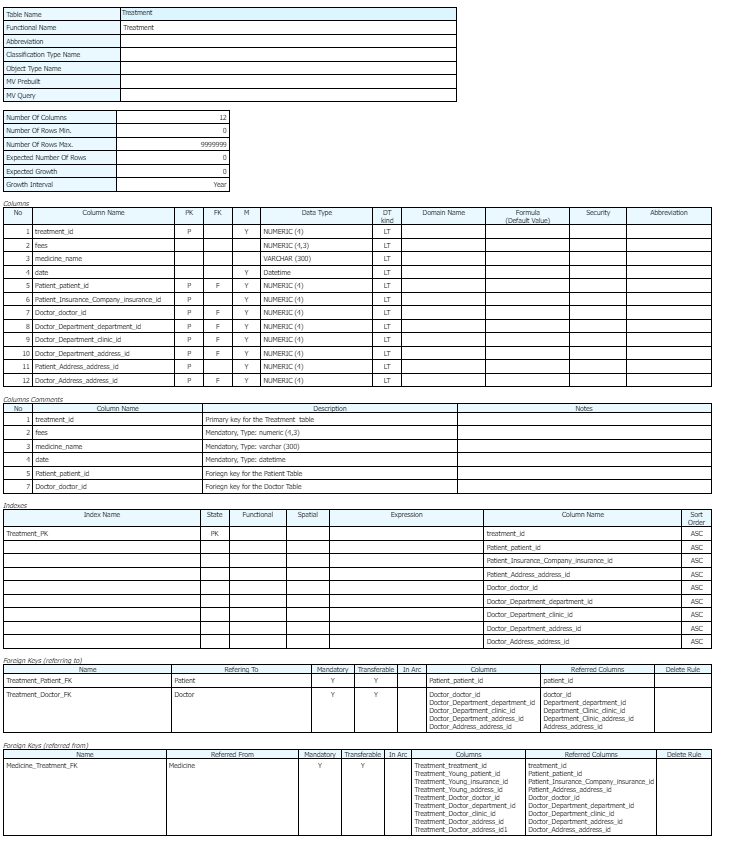
10-Insurance\_Company



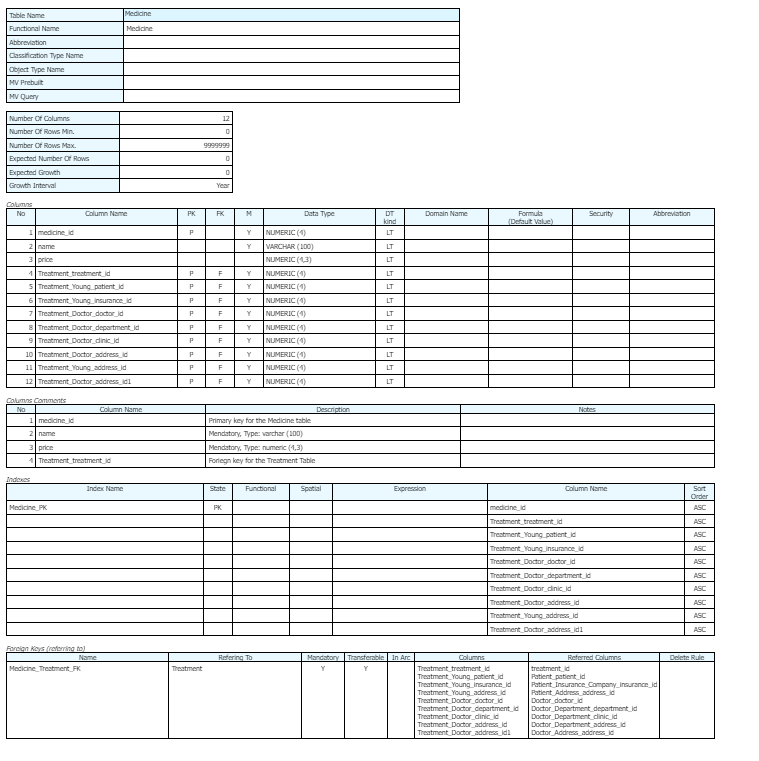
11-Insurance



12-Treatment



13-Medicine

*SQL*

D1: retrieve data from patient table   
SELECT \*  
FROM patient  
 WHERE patient\_id>1  
 AND last\_name ='Sam'  
ORDER BY last\_name ASC;

D2: retrieve data from patient table

SELECT \*  
FROM patient  
 WHERE last\_name='Sam'  
 AND patient\_id NOT IN (3,4);

D3: retrieve the patients first\_name and last\_name who visited Doctors : Marek Filip and Filip Marek and not visited Doctor: Marek Jan

SELECT first\_name,last\_name  
FROM patient  
 WHERE doctor\_id IN (2,1) AND patient.doctor\_id NOT IN (3);

D4,D5: list all patients with their doctors (INNER USING)  
SELECT patient.first\_name,patient.last\_name,doctor.first\_name,doctor.last\_name,doctor.patient\_id  
FROM patient  
JOIN doctor  
 USING (doctor\_id);

D6: select the doctors and nurses names who have the same addresses (JOIN ON)  
SELECT doctor.first\_name,doctor.last\_name,nurse.first\_name,nurse.last\_name  
FROM doctor  
 INNER JOIN NURSE  
 ON doctor.address\_id=nurse.ADDRESS\_ID;

D7: (NATURAL JOIN)  
SELECT doctor.first\_name,doctor.last\_name,department.name  
FROM doctor  
 NATURAL JOIN department;

D8: select all the doctors first and last names and their departments where they belong  
SELECT doctor.first\_name,doctor.last\_name,department.name Department\_Name  
FROM doctor  
 CROSS JOIN department;

D9: (LEFT JOIN)  
SELECT nurse.nurse\_id Nurse\_ID,nurse.first\_name,nurse.last\_name, doctor.doctor\_id Doctor\_ID,doctor.first\_name,doctor.last\_name  
FROM nurse  
 LEFT JOIN doctor ON doctor.doctor\_id=nurse.nurse\_id;

D10: (RIGHT JOIN)  
SELECT nurse.nurse\_id Nurse\_ID,nurse.first\_name,nurse.last\_name, doctor.doctor\_id Doctor\_ID,doctor.first\_name,doctor.last\_name  
FROM nurse  
 RIGHT JOIN doctor ON doctor.doctor\_id=nurse.nurse\_id;

D11: list all treatments and their medicines where price is null (FULL OUTER JOIN)  
SELECT \*  
FROM medicine  
FULL OUTER JOIN treatment ON treatment.treatment\_id=medicine.treatment\_id  
WHERE price IS NULL  
 ORDER BY medicine.name;  
select \* from medicine;

D12,D13,D14: retrieve data from patient table   
SELECT \*  
FROM patient  
 WHERE patient\_id>1  
 AND last\_name ='Sam'  
ORDER BY last\_name ASC;

D15:

SELECT  
first\_name,last\_name,doctor\_id  
FROM patient p  
WHERE  
EXISTS(  
 SELECT 1  
 FROM appointment  
 WHERE p.patient\_id=appointment.patient\_id  
 )

ORDER BY first\_name;

D16: (UNION)  
SELECT nurse\_id,first\_name,last\_name  
FROM nurse  
 WHERE nurse\_id >=2  
UNION  
SELECT doctor\_id,first\_name,last\_name  
FROM doctor  
 WHERE first\_name='Marek'  
ORDER BY 2;

D17: (MINUS)  
SELECT first\_name,last\_name  
FROM doctor  
 WHERE first\_name='Marek'  
MINUS  
SELECT first\_name,last\_name  
FROM nurse  
 ORDER BY last\_name;

D18: (INTERSECT)  
SELECT address\_id  
FROM doctor  
 INTERSECT  
SELECT address\_id  
FROM NURSE  
 ORDER BY address\_id;

D19: Using string function (LTRIM) to make a shortcut of the doctor's last name  
SELECT last\_name,  
 LTRIM(last\_name,'Kope') last\_name\_shortcut  
FROM patient  
 WHERE last\_name LIKE '%Kope%'  
ORDER BY last\_name;

D20: select the price average using AVG numeric function  
SELECT AVG(price) price\_average  
FROM medicine;

D21: calculating how many months between two appointments dates using date function (MONTHS\_BETWEEN)  
SELECT MONTHS\_BETWEEN('01-JAN-21','01-AUG-20') waiting\_time  
FROM DUAL;

D22: counting the maximum price using (MAX) function  
SELECT MAX(price) max\_price  
FROM medicine;

counting the minimum price using (MIN) function  
SELECT MIN(price) min\_price  
FROM medicine;

counting the total price using (SUM) function  
SELECT SUM(price) total\_price  
FROM medicine;

select the price average using AVG numeric function  
SELECT AVG(price) price\_average  
FROM medicine;

D23,D24,D25: return number of medicines which having total price bigger than 5 usd on each treatment using (GROUP BY,HAVING) clause and (COUNT)  
SELECT treatment\_id,COUNT(medicine\_id) number\_of\_medicines  
FROM medicine  
GROUP BY treatment\_id  
HAVING SUM(price\*10) >5  
ORDER BY treatment\_id;

D26,D27: Patient VIEW  
CREATE VIEW patient\_view AS  
 SELECT  
patient\_id,  
first\_name || ' ' || last\_name full\_name  
 FROM patient;

D28:

INSERT INTO patient  
 ( patient\_id, first\_name, last\_name, mail, phone\_number, address\_id, insurance\_id)  
VALUES  
 (1,'Ahmad','Akel','ahmad@gmail.com',722939931,1,1),  
 (2,'Jan','Sam','jan@gmail.com',722939941,2,1),  
 (3,'Filip','Sam','Filip@gmail.com',722546321,3,1);

D29: update zip\_code in address table  
UPDATE  
 address  
SET  
 zip\_code=53008  
WHERE  
address\_id=3;

D30: delete repeated column from patient table  
DELETE FROM patient  
WHERE patient\_id=5;

Závěr

Myslím, že databáze, kterou jsem vytvořil, bude dobře fungovat s jednoduchým systémem Dental Clinic a samozřejmě je zde prostor pro její rozvoj, aby byla pokročilejší, přeskočil jsem část 4 v sekci analýzy, protože jsem nevěděl co přesně by tam mělo být, Vynechal jsem také spoušť a sekvenci, protože jsem neměl moc času odevzdat práci a nevěděl jsem, jak přesně bude použita v mé databázi.

Naučil jsem se hodně o tom, jak se vyhnout překážkám při vytváření databáze, zejména pokud jde o vztahy.

a jak organizovat databázi pomocí analýzy.

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.