**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт компьютерных наук и технологий**

**Кафедра Компьютерных Систем и Программных Технологий**

**Отчет по дисциплине**

**«Базы данных»**

[**Язык SQL-DML**](http://tiger.ftk.spbstu.ru/trac/edu-db-2016/ticket/33)

**Работу выполнил студент группы №:** 43501/3 Крутихин М.В.

**Работу принял преподаватель:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мяснов А.В.

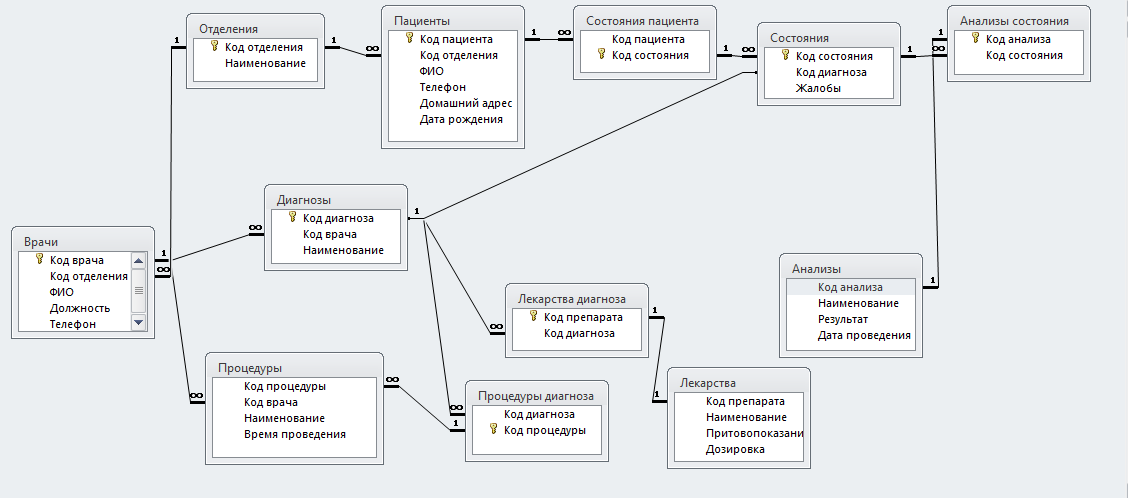
**Санкт-Петербург**

**2016 г.**

**Программа работы**

1. Самостоятельное изучение SQL-DDL
2. Создание скрипта БД в соответствии с согласованной схемой (должны присутствовать первичные и внешние ключи, ограничения на диапазоны значений). Продемонстрировать скрипт преподавателю.
3. Создайте скрипт, заполняющий все таблицы БД данными
4. Выполнение SQL-запросов, изменяющих схему созданной БД по заданию преподавателя. Продемонстрировать их работу преподавателю.
5. Изучите основные возможности IBExpert. Получите ER-диаграмму созданной БД с помощью Database Designer.
6. Автоматически сгенерируйте данные при помощи IBExpert (для трех или большего числа таблиц, не менее 100000 записей в каждой из выбранных таблиц)

**Ход работы:**



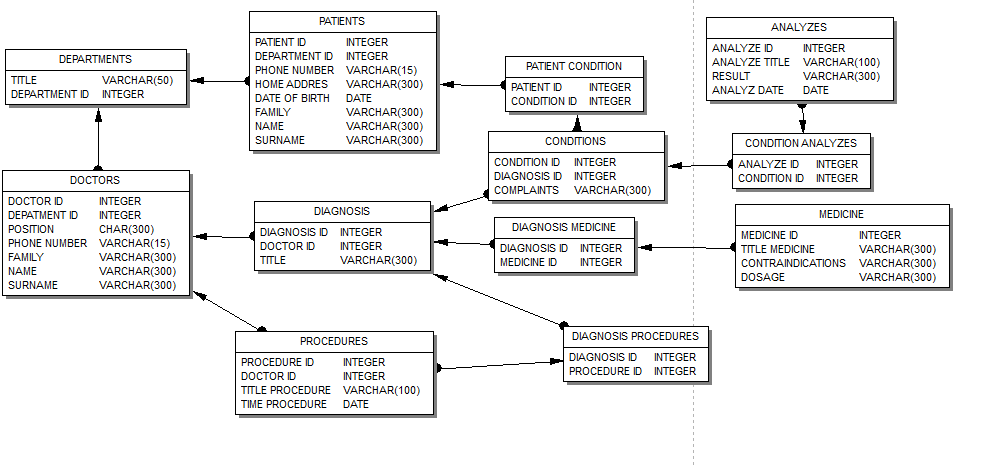
*Рис.1 SQL-схема БД*

На основе данной схемы был написан скрипт, создающий соответствующую БД:

|  |
| --- |
| SET SQL DIALECT 3;  SET NAMES WIN1251;  SET CLIENTLIB ‘C:\Program Files (x86)\Firebird\Firebird\_3\_0\fbclient.dll’;  CREATE DATABASE ‘C:\DataBases\HOSPITAL.FDB’  USER ‘SYSDBA’ PASSWORD ‘masterkey’  PAGE\_SIZE 16384  DEFAULT CHARACTER SET WIN1251 COLLATION WIN1251;  CREATE GENERATOR GEN\_DEPARTMENTS\_ID START WITH 0 INCREMENT BY 1;  SET GENERATOR GEN\_DEPARTMENTS\_ID TO 1;  CREATE TABLE ANALYZES (  “ANALYZE ID” INTEGER NOT NULL,  “ANALYZE TITLE” VARCHAR(100) NOT NULL,  RESULT VARCHAR(300) NOT NULL,  “ANALYZ DATE” DATE NOT NULL  );  CREATE TABLE “CONDITION ANALYZES” (  “ANALYZE ID” INTEGER NOT NULL,  “CONDITION ID” INTEGER NOT NULL  );  CREATE TABLE CONDITIONS (  “CONDITION ID” INTEGER NOT NULL,  “DIAGNOSIS ID” INTEGER NOT NULL,  COMPLAINTS VARCHAR(300) NOT NULL  );  CREATE TABLE DEPARTMENTS (  TITLE VARCHAR(50) NOT NULL,  “DEPARTMENT ID” INTEGER NOT NULL  );  CREATE TABLE DIAGNOSIS (  “DIAGNOSIS ID” INTEGER NOT NULL,  “DOCTOR ID” INTEGER,  TITLE VARCHAR(300) CHARACTER SET NONE NOT NULL  );  CREATE TABLE “DIAGNOSIS MEDICINE” (  “DIAGNOSIS ID” INTEGER NOT NULL,  “MEDICINE ID” INTEGER NOT NULL  );  CREATE TABLE “DIAGNOSIS PROCEDURES” (  “DIAGNOSIS ID” INTEGER NOT NULL,  “PROCEDURE ID” INTEGER NOT NULL  );  CREATE TABLE DOCTORS (  “DOCTOR ID” INTEGER NOT NULL,  “DEPATMENT ID” INTEGER NOT NULL,  “POSITION” CHAR(300) NOT NULL,  “PHONE NUMBER” VARCHAR(15) CHARACTER SET NONE NOT NULL,  FAMILY VARCHAR(300) NOT NULL,  NAME VARCHAR(300) NOT NULL,  SURNAME VARCHAR(300) NOT NULL  );  CREATE TABLE MEDICINE (  “MEDICINE ID” INTEGER NOT NULL,  “TITLE MEDICINE” VARCHAR(300) NOT NULL,  CONTRAINDICATIONS VARCHAR(300) NOT NULL,  DOSAGE VARCHAR(300) NOT NULL  );  CREATE TABLE “PATIENT CONDITION” (  “PATIENT ID” INTEGER NOT NULL,  “CONDITION ID” INTEGER NOT NULL  );  CREATE TABLE PATIENTS (  “PATIENT ID” INTEGER NOT NULL,  “DEPARTMENT ID” INTEGER NOT NULL,  “PHONE NUMBER” VARCHAR(15) CHARACTER SET NONE NOT NULL,  “HOME ADDRES” VARCHAR(300) CHARACTER SET NONE,  “DATE OF BIRTH” DATE,  FAMILY VARCHAR(300) NOT NULL,  NAME VARCHAR(300) NOT NULL,  SURNAME VARCHAR(300) NOT NULL  );  CREATE TABLE PROCEDURES (  “PROCEDURE ID” INTEGER NOT NULL,  “DOCTOR ID” INTEGER,  “TITLE PROCEDURE” VARCHAR(100) NOT NULL,  “TIME PROCEDURE” DATE NOT NULL  );  ALTER TABLE ANALYZES ADD CONSTRAINT UNQ1\_ANALYZES UNIQUE (“ANALYZE ID”);  ALTER TABLE CONDITIONS ADD CONSTRAINT UNQ1\_CONDITIONS UNIQUE (“CONDITION ID”);  ALTER TABLE MEDICINE ADD CONSTRAINT UNQ1\_MEDICINE UNIQUE (“MEDICINE ID”);  ALTER TABLE PROCEDURES ADD CONSTRAINT UNQ1\_PROCEDURES UNIQUE (“PROCEDURE ID”);  ALTER TABLE “CONDITION ANALYZES” ADD CONSTRAINT “PK\_CONDITION ANALYZES” PRIMARY KEY (“ANALYZE ID”);  ALTER TABLE DEPARTMENTS ADD CONSTRAINT PK\_DEPARTMENTS PRIMARY KEY (“DEPARTMENT ID”);  ALTER TABLE DIAGNOSIS ADD CONSTRAINT PK\_DIAGNOSIS PRIMARY KEY (“DIAGNOSIS ID”);  ALTER TABLE “DIAGNOSIS MEDICINE” ADD CONSTRAINT “PK\_DIAGNOSIS MEDICINE” PRIMARY KEY (“MEDICINE ID”);  ALTER TABLE “DIAGNOSIS PROCEDURES” ADD CONSTRAINT “PK\_DIAGNOSIS PROCEDURES” PRIMARY KEY (“PROCEDURE ID”);  ALTER TABLE DOCTORS ADD CONSTRAINT PK\_DOCTORS PRIMARY KEY (“DOCTOR ID”);  ALTER TABLE “PATIENT CONDITION” ADD CONSTRAINT “PK\_PATIENT CONDITION” PRIMARY KEY (“CONDITION ID”);  ALTER TABLE PATIENTS ADD CONSTRAINT PK\_PATIENTS PRIMARY KEY (“PATIENT ID”);  ALTER TABLE ANALYZES ADD CONSTRAINT FK\_ANALYZES\_1 FOREIGN KEY (“ANALYZE ID”) REFERENCES “CONDITION ANALYZES” (“ANALYZE ID”);  ALTER TABLE “CONDITION ANALYZES” ADD CONSTRAINT “FK\_CONDITION ANALYZES\_1” FOREIGN KEY (“CONDITION ID”) REFERENCES CONDITIONS (“CONDITION ID”);  ALTER TABLE CONDITIONS ADD CONSTRAINT FK\_CONDITIONS\_1 FOREIGN KEY (“DIAGNOSIS ID”) REFERENCES DIAGNOSIS (“DIAGNOSIS ID”);  ALTER TABLE CONDITIONS ADD CONSTRAINT FK\_CONDITIONS\_2 FOREIGN KEY (“CONDITION ID”) REFERENCES “PATIENT CONDITION” (“CONDITION ID”);  ALTER TABLE DIAGNOSIS ADD CONSTRAINT FK\_DIAGNOSIS\_1 FOREIGN KEY (“DOCTOR ID”) REFERENCES DOCTORS (“DOCTOR ID”);  ALTER TABLE “DIAGNOSIS MEDICINE” ADD CONSTRAINT “FK\_DIAGNOSIS MEDICINE\_1” FOREIGN KEY (“DIAGNOSIS ID”) REFERENCES DIAGNOSIS (“DIAGNOSIS ID”);  ALTER TABLE “DIAGNOSIS PROCEDURES” ADD CONSTRAINT “FK\_DIAGNOSIS PROCEDURES\_1” FOREIGN KEY (“DIAGNOSIS ID”) REFERENCES DIAGNOSIS (“DIAGNOSIS ID”);  ALTER TABLE DOCTORS ADD CONSTRAINT FK\_DOCTORS\_1 FOREIGN KEY (“DEPATMENT ID”) REFERENCES DEPARTMENTS (“DEPARTMENT ID”);  ALTER TABLE MEDICINE ADD CONSTRAINT FK\_MEDICINE\_1 FOREIGN KEY (“MEDICINE ID”) REFERENCES “DIAGNOSIS MEDICINE” (“MEDICINE ID”);  ALTER TABLE “PATIENT CONDITION” ADD CONSTRAINT “FK\_PATIENT CONDITION\_1” FOREIGN KEY (“PATIENT ID”) REFERENCES PATIENTS (“PATIENT ID”);  ALTER TABLE PATIENTS ADD CONSTRAINT FK\_PATIENTS\_1 FOREIGN KEY (“DEPARTMENT ID”) REFERENCES DEPARTMENTS (“DEPARTMENT ID”);  ALTER TABLE PROCEDURES ADD CONSTRAINT FK\_PROCEDURES\_1 FOREIGN KEY (“PROCEDURE ID”) REFERENCES “DIAGNOSIS PROCEDURES” (“PROCEDURE ID”);  ALTER TABLE PROCEDURES ADD CONSTRAINT FK\_PROCEDURES\_2 FOREIGN KEY (“DOCTOR ID”) REFERENCES DOCTORS (“DOCTOR ID”); |

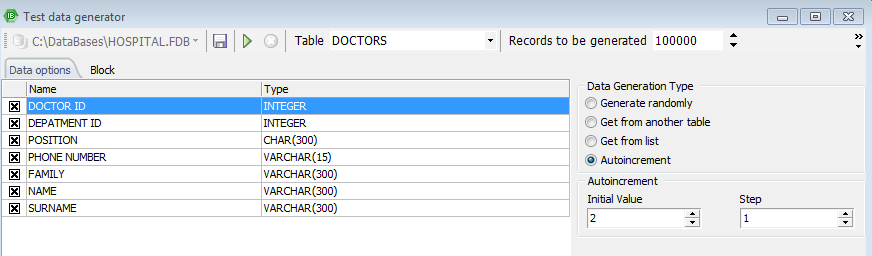
*Листинг 1. Скрипт для создания БД*

Ниже приведена ER-диаграмма, полученная в databse designer:

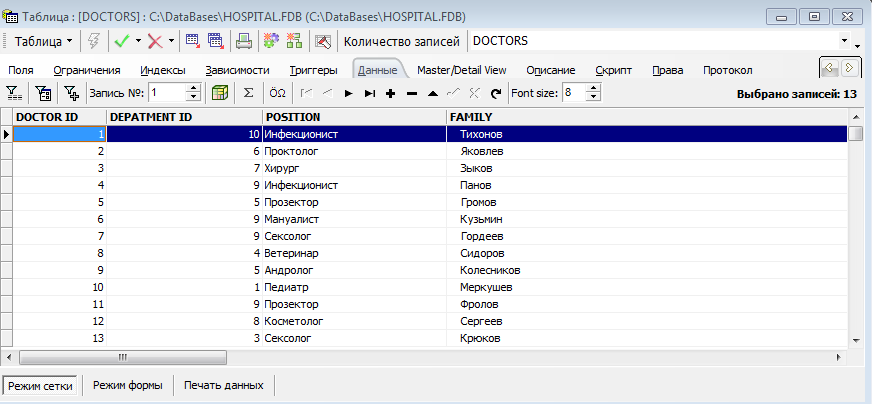


*Рис. 2 ER- диаграмма базы данных.*

Далее при помощи инструмента «Генератор тестовых данных» были автоматически сгенерированы данные для таблиц Patients, Doctors, departments, patiend condition:



*Рис.3 Инструмент «Генератор тестовых данных»*



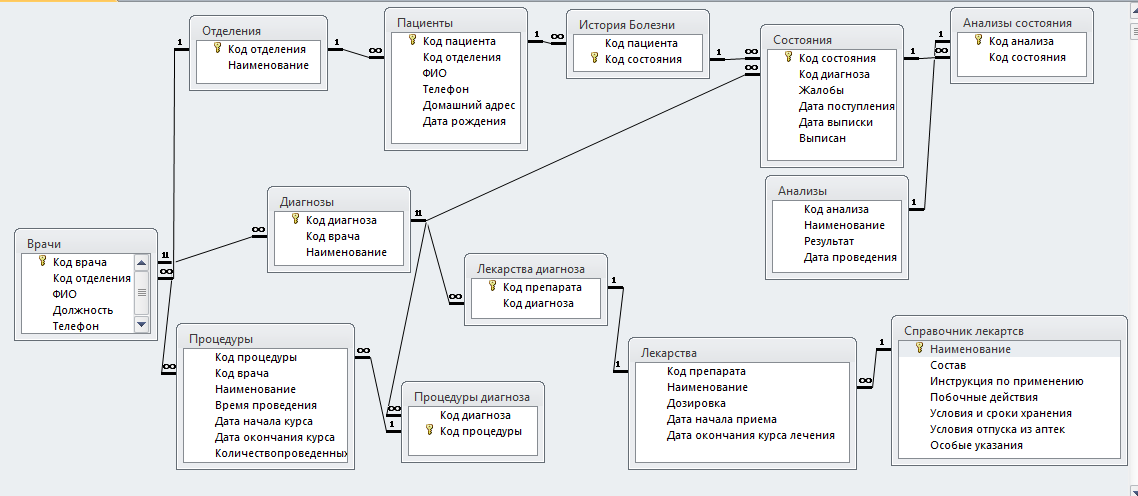
*Рис.4 Пример данных сгенерированных для таблицы Doctors*

**Выполнение индивидуального задания.**

Формулировка задания:

Ввести учет истории болезни: поступление, выдача лекарств и проведение процедур, изменение состояния, назначения при выписке.

На рисунке 4 представлена измененная SQL-схема базы данных:



*Рис.5 Измененная SQL-схема БД*

Список изменений:

1. Для возможности учета истории болезни в таблицу «Состояния» были добавлены поля даты поступления и даты выписки (дата выписки может быть пустой). Кроме того, в эту же таблицу было добавлено логическое поле «Выписан» для упрощения идентификации состояний, которые уже были «вылечены».

2. Была добавлена таблица «Справочник лекарств», в которой содержится подробная информацию о каждом препарате.

3. Таблица лекарства была изменена, теперь в ней содержится информация о дате начала приема и окончания приема средства, а так же дозировка, написанная врачом. Это было сделано для того, чтобы обеспечить более точный учет выданных лекарств и при этом убрать избыток информации (противопоказания, состав и прочие поля, которые могу быть не нужны врачу в определенной ситуации – теперь находятся в таблице «Справочник лекарств».)

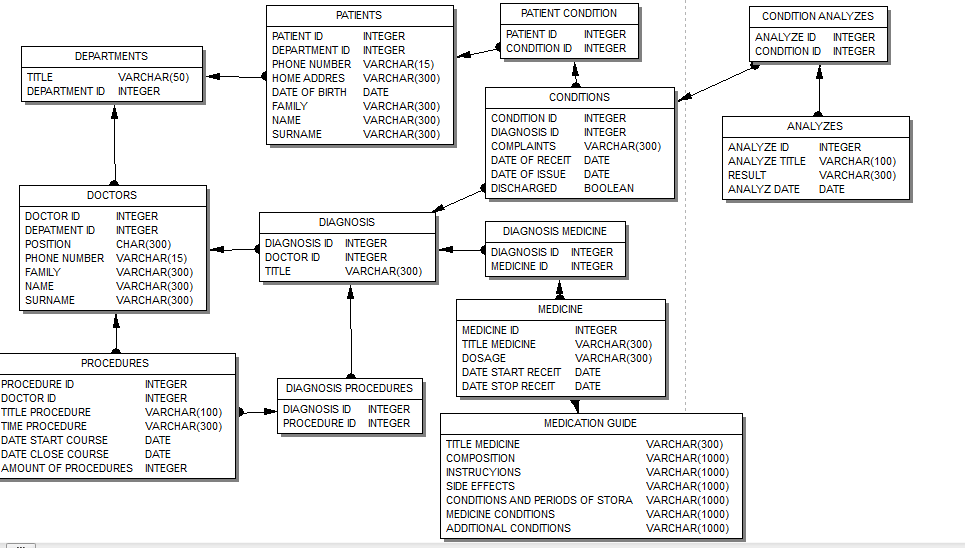
4. Была изменена таблица «Процедуры». В нее были добавлены поля для даты начала курса лечения и даты его окончания, так же целочисленное поле для учета количества визитов пациента в процедурный кабинет.

Скрипт, изменяющий БД в соответствии с новой sql - схемой:

|  |
| --- |
| CREATE TABLE "MEDICATION GUIDE" (  "TITLE MEDICINE" VARCHAR(300) NOT NULL,  COMPOSITION VARCHAR(1000) NOT NULL,  INSTRUCYIONS VARCHAR(1000) NOT NULL,  "SIDE EFFECTS" VARCHAR(1000),  "CONDITIONS AND PERIODS OF STORA" VARCHAR(1000) NOT NULL,  "MEDICINE CONDITIONS" VARCHAR(1000) NOT NULL,  "ADDITIONAL CONDITIONS" VARCHAR(1000)  );  ALTER TABLE "MEDICATION GUIDE" ADD CONSTRAINT "PK\_MEDICATION GUIDE" PRIMARY KEY ("TITLE MEDICINE");  ALTER TABLE CONDITIONS ADD DATE\_OF\_RECEIT DATE NOT NULL;  ALTER TABLE CONDITIONS ADD DATE\_OF\_ISSUE DATE;  ALTER TABLE CONDITIONS ADD DISCHARGED boolean;  ALTER TABLE MEDICINE ADD DATE\_START\_RECEIT DATE NOT NULL;  ALTER TABLE MEDICINE ADD DATE\_STOP\_RECEIT DATE;  ALTER TABLE PROCEDURES ADD DATE\_START\_COURSE DATE NOT NULL;  ALTER TABLE PROCEDURES ADD DATE\_STOP\_COURSE DATE;  ALTER TABLE PROCEDURES ADD AMOUNT\_OF\_PROCEDURES INTEGER; |

*Листинг 2. Скрипт для изменения БД.*

ER –диаграмма, соответствующая измененной базе данных:



*Рис.6 ER-диаграмма измененной БД*

**Вывод:**

В ходе работы был изучен язык определения данных DDL. С помощью данного языка можно определять, изменять и удалять какие-либо структуры данных.

Так же был получен опыт работы с СУБД IBExpert, графический интерфейс, который позволяет работать сразу с несколькими БД, а так же в целом значительно упрощает работу. На мой взгляд, наиболее важными инструментами данного интерфейса являются: дизайнер БД, позволяющий создавать ER-диаграммы для проверки результатов работы и генератор тестовых данных, позволяющий заполнить базу достаточно большим количеством значений.