

Санкт-Петербургский Государственный Политехнический  
Университет Петра Великого  
Институт Компьютерных Наук и Технологий  
**Кафедра Компьютерных Систем и Программных Технологий**

**Отчёт по лабораторной работе**  
**Дисциплина:** Базы данных  
**Тема:** Ознакомление с основами SQL-DDL

Выполнил студент группы 43501/3

\_\_\_\_\_  
(подпись) Круминьш Д.В.

Преподаватель

\_\_\_\_\_  
(подпись) Мяснов А.В.

## 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

По итогам предыдущей работы имеется следующая SQL - схема:

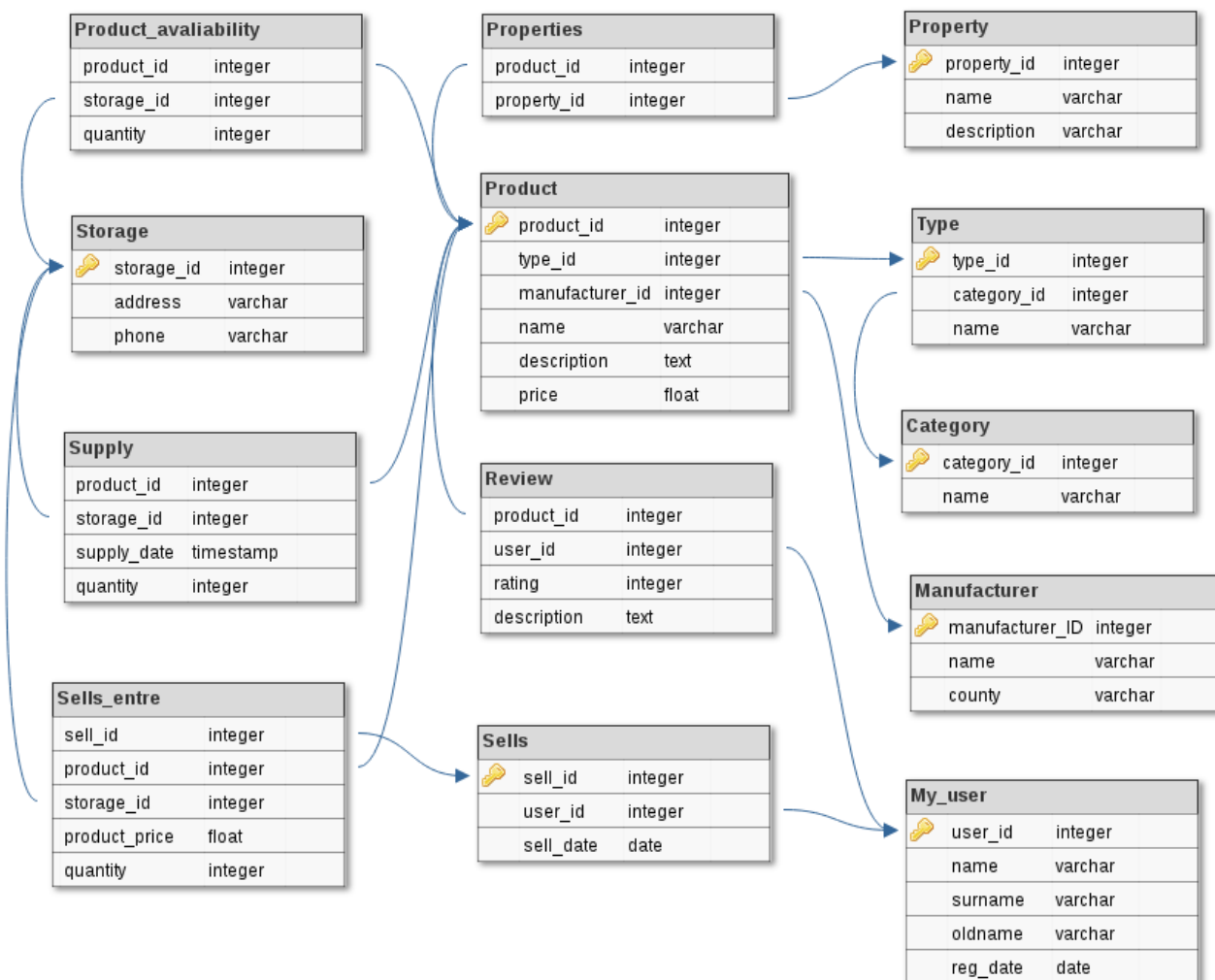


Рис. 1: SQL-схема БД

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

```
1 CREATE DATABASE 'C:\elmart.fdb' user 'SYSDBA' password '1111' DEFAULT
  ↳ CHARACTER SET WIN1251;
2 CREATE TABLE CATEGORY(
3     category_id          INTEGER          NOT NULL ,
4     name                  VARCHAR(256)     NOT NULL ,
5     CONSTRAINT PK_CATEGORY PRIMARY KEY (category_id)
6 );
7 CREATE TABLE TYPE(
8     type_id              INTEGER          NOT NULL ,
9     category_id          INTEGER          NOT NULL REFERENCES
  ↳ CATEGORY,
10    name                  VARCHAR(256)     NOT NULL ,
11    CONSTRAINT PK_TYPE PRIMARY KEY (type_id)
12 );
13 CREATE TABLE PROPERTY(
14     property_id          INTEGER          NOT NULL ,
15     name                  VARCHAR(256)     NOT NULL ,
16     description           VARCHAR(256)     NOT NULL ,
17     CONSTRAINT PK_PROPERTY PRIMARY KEY (property_id)
18 );
19 CREATE TABLE MANUFACTURER(
20     manufacturer_id      INTEGER          NOT NULL ,
21     name                  VARCHAR(256)     NOT NULL ,
22     country               VARCHAR(256)     NOT NULL ,
23     CONSTRAINT PK_MANUFACTURER PRIMARY KEY (manufacturer_id)
24 );
25 CREATE TABLE PRODUCT(
26     product_id           INTEGER          NOT NULL ,
27     type_id              INTEGER          NOT NULL REFERENCES
  ↳ TYPE,
28     manufacturer_id      INTEGER          NOT NULL REFERENCES
  ↳ MANUFACTURER,
29     name                  VARCHAR(256)     NOT NULL ,
30     description           VARCHAR(256)     NOT NULL ,
31     price                 FLOAT           NOT NULL ,
32     CONSTRAINT PK_PRODUCT PRIMARY KEY (product_id)
33 );
34 CREATE TABLE PROPERTIES(
35     product_id           INTEGER          NOT NULL REFERENCES
  ↳ PRODUCT,
36     property_id          INTEGER          NOT NULL REFERENCES
  ↳ PROPERTY
37 );
38 CREATE TABLE MY_USER(
39     user_id              INTEGER          NOT NULL ,
40     name                  VARCHAR(256)     NOT NULL ,
41     surname               VARCHAR(256)     NOT NULL ,
42     oldname               VARCHAR(256)     NOT NULL ,
43     reg_date              DATE            NOT NULL ,
44     CONSTRAINT PK_MY_USER PRIMARY KEY (user_id)
45 );
46 CREATE TABLE REVIEW(
```

```

47     product_id          INTEGER          NOT NULL REFERENCES
    ↪ PRODUCT,
48     user_id             INTEGER          NOT NULL REFERENCES
    ↪ MY_USER,
49     rating              INTEGER          NOT NULL,
50     description          VARCHAR(256)     NOT NULL
51 );
52 CREATE TABLE SELLS(
53     sell_id              INTEGER          NOT NULL,
54     user_id              INTEGER          NOT NULL REFERENCES
    ↪ MY_USER,
55     sell_date            DATE              NOT NULL,
56     CONSTRAINT PK SELLS PRIMARY KEY (sell_id)
57 );
58 CREATE TABLE STORAGE(
59     storage_id           INTEGER          NOT NULL,
60     address              VARCHAR(256)     NOT NULL,
61     phone                VARCHAR(256)     NOT NULL,
62     CONSTRAINT PK STORAGE PRIMARY KEY (storage_id)
63 );
64 CREATE TABLE PRODUCT_AVALIABILITY(
65     product_id           INTEGER          NOT NULL REFERENCES
    ↪ PRODUCT,
66     storage_id           INTEGER          NOT NULL REFERENCES
    ↪ STORAGE,
67     quantity             INTEGER          NOT NULL
68 );
69 CREATE TABLE SUPPLY(
70     product_id           INTEGER          NOT NULL REFERENCES
    ↪ PRODUCT,
71     storage_id           INTEGER          NOT NULL REFERENCES
    ↪ STORAGE,
72     supply_date          DATE              NOT NULL,
73     quantity             INTEGER          NOT NULL
74 );
75 CREATE TABLE SELLS_ENTRE(
76     sell_id              INTEGER          NOT NULL REFERENCES SELLS,
77     product_id           INTEGER          NOT NULL REFERENCES
    ↪ PRODUCT,
78     storage_id           INTEGER          NOT NULL REFERENCES
    ↪ STORAGE,
79     product_price         FLOAT            NOT NULL,
80     quantity             INTEGER          NOT NULL
81 );
82 COMMIT;

```

Листинг 1: Скрипт для создания БД

Далее был написан скрипт немного наполняющий БД.

```

1  INSERT INTO CATEGORY(category_id, name) VALUES (1, 'Компьютер');
2  INSERT INTO CATEGORY(category_id, name) VALUES (2, 'Бытовая техника');
3  INSERT INTO CATEGORY(category_id, name) VALUES (3, 'Комплектующие для
    ↪ ПК');
4

```

```

5 INSERT INTO TYPE(type_id, category_id, name) VALUES (1,3,'Видеокарта')
  ↳ ;
6 INSERT INTO TYPE(type_id, category_id, name) VALUES (2,3,'Материнская
  ↳ плата');
7 INSERT INTO TYPE(type_id, category_id, name) VALUES (3,3,'Процессор');
8
9 INSERT INTO PROPERTY(property_id,name,description) VALUES(1,'Количество
  ↳ о ядер','4');
10 INSERT INTO PROPERTY(property_id,name,description) VALUES(2,'Тип сокет
  ↳ а','AM3+');
11 INSERT INTO PROPERTY(property_id,name,description) VALUES(3,'Тип проце
  ↳ ссора','Мобильный');
12
13 INSERT INTO MANUFACTURER(manufacturer_id,name,country) VALUES(1,'НЕОБИ
  ↳ Т','Россия');
14
15 INSERT INTO PRODUCT(product_id,type_id,manufacturer_id,name,
  ↳ description,price) VALUES(1,3,1,'AMD X9000','Видеокарта нового п
  ↳ околениа', 12222.4);
16
17 INSERT INTO PROPERTIES(product_id,property_id) VALUES(1,1);
18
19 INSERT INTO MY_USER(user_id, name, surname, oldname, reg_date) VALUES
  ↳ (1, 'Денис', 'Круминьш', 'Валерьевич','25.6.2011');
20
21 INSERT INTO REVIEW(product_id, user_id, rating, description) VALUES
  ↳ (1,1,4,'Можно было-бы и дешевле');
22
23 INSERT INTO STORAGE(storage_id, address, phone) VALUES(1,'ул. Никакая,
  ↳ д. 123', '8 (905) 123-45-67');
24
25 INSERT INTO SUPPLY(product_id, storage_id, supply_date, quantity)
  ↳ VALUES(1, 1, '15.9.2012', 14);
26
27 INSERT INTO PRODUCT_AVALIABILITY(product_id, storage_id, quantity)
  ↳ VALUES (1,1,13);
28
29 INSERT INTO SELLS(sell_id, user_id, sell_date) VALUES(1,1,'16.9.2012')
  ↳ ;
30
31 INSERT INTO SELLS_ENTRE(sell_id, product_id, storage_id, product_price
  ↳ , quantity) VALUES(1,1,1, 12222.4, 1);

```

Листинг 2: Наполнения БД

## Индивидуальное задание

1. Реализовать иерархическую структуру категорий товаров с возможностью принадлежности товара к нескольким категориям.
2. Реализовать структуру хранения характеристик категорий товаров и значений этих характеристик для каждой товарной позиции.

Для реализации иерархической структуры таблица Category была удалена, а таблица Type претерпела изменения. В частности появилось поле p\_id, которое ссылается(может и не ссылаться) на поле type\_id из этой же таблицы, что обеспечивает иерархию категорий товаров.

Была добавлена таблица Type\_prop для хранения характеристик той или иной категории.

Так-же претерпели изменению таблицы Properties и Property. Поле Description из таблицы Property перешло в таблицу Properties и изменило название на prop\_value. Это было необходимо для более корректной работы с базой данных.

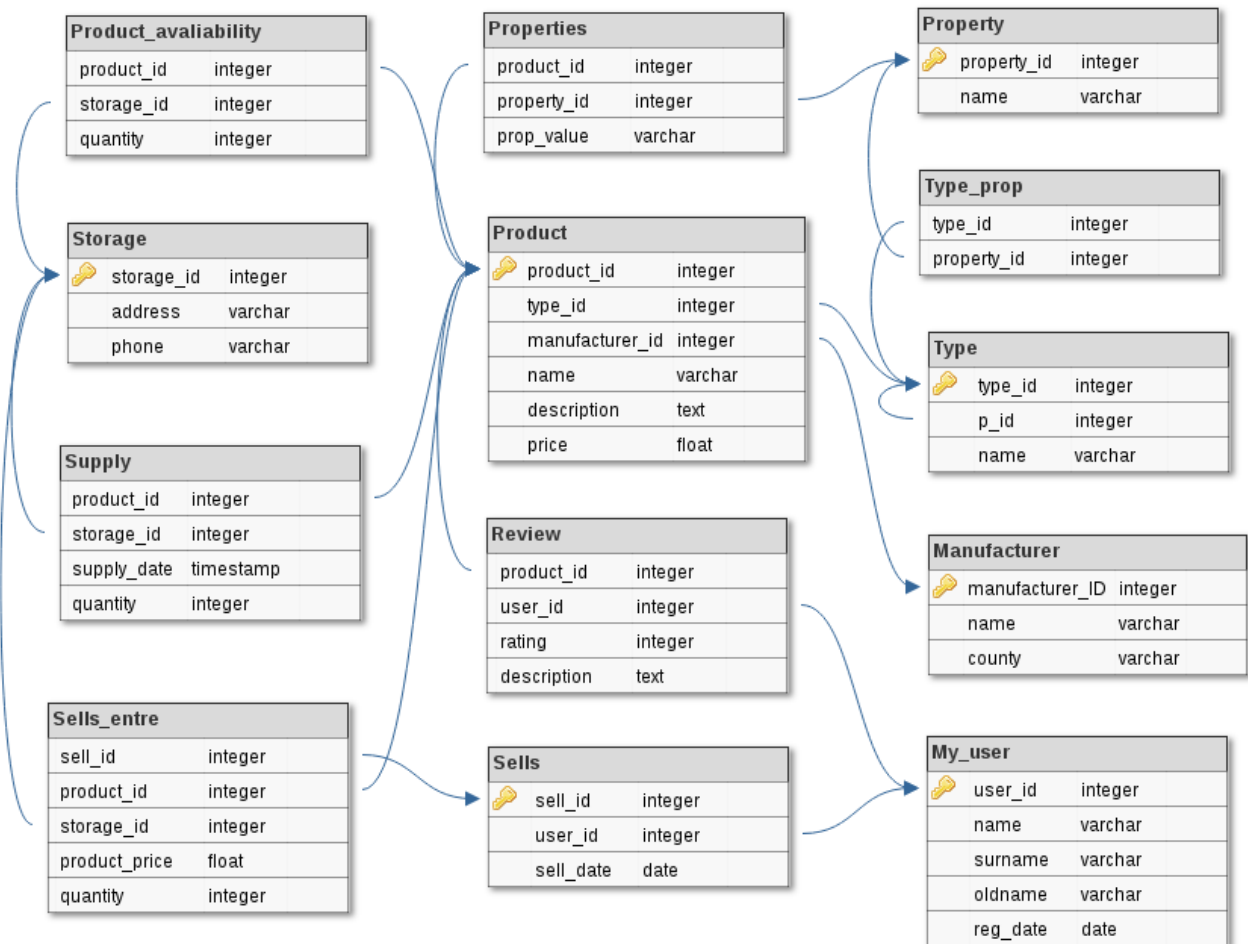


Рис. 2: Измененная SQL-схема БД

Далее представлен скрипт изменяющий БД в соответствии с новой SQL-схемой.

```

1 CONNECT 'C:\ELMART.FDB' user 'SYSDBA' password '1111';
2 ALTER TABLE TYPE DROP CONSTRAINT INTEG_67;
3 DROP TABLE CATEGORY;
4 ALTER TABLE TYPE ALTER COLUMN CATEGORY_ID TO P_ID;
5 ALTER TABLE TYPE ADD FOREIGN KEY (P_ID) REFERENCES TYPE(TYPE_ID);
6 ALTER TABLE PROPERTIES ADD PROP_VALUE VARCHAR(256) NOT NULL;
7 ALTER TABLE TYPE ALTER P_ID DROP NOT NULL;
8 ALTER TABLE PROPERTY DROP DESCRIPTION;
9 UPDATE TYPE SET P_ID=2, NAME='Мобильная видеокарта' WHERE TYPE_ID=3;
10 UPDATE TYPE SET P_ID=1, NAME='Видеокарта' WHERE TYPE_ID=2;
11 UPDATE TYPE SET P_ID=NULL, NAME='Комплектующие для ПК' WHERE TYPE_ID
    ↳ =1;
12 CREATE TABLE TYPE_PROP(

```

```

13      type_id          INTEGER          NOT NULL    REFERENCES
      ↪ TYPE,
14      property_id      INTEGER          NOT NULL    REFERENCES
      ↪ PROPERTY
15 );
16 COMMIT;

```

Листинг 3: Скрипт для изменения БД

Дополнительно приведена ER - диаграмма, созданная с помощью инструмента Database designer.

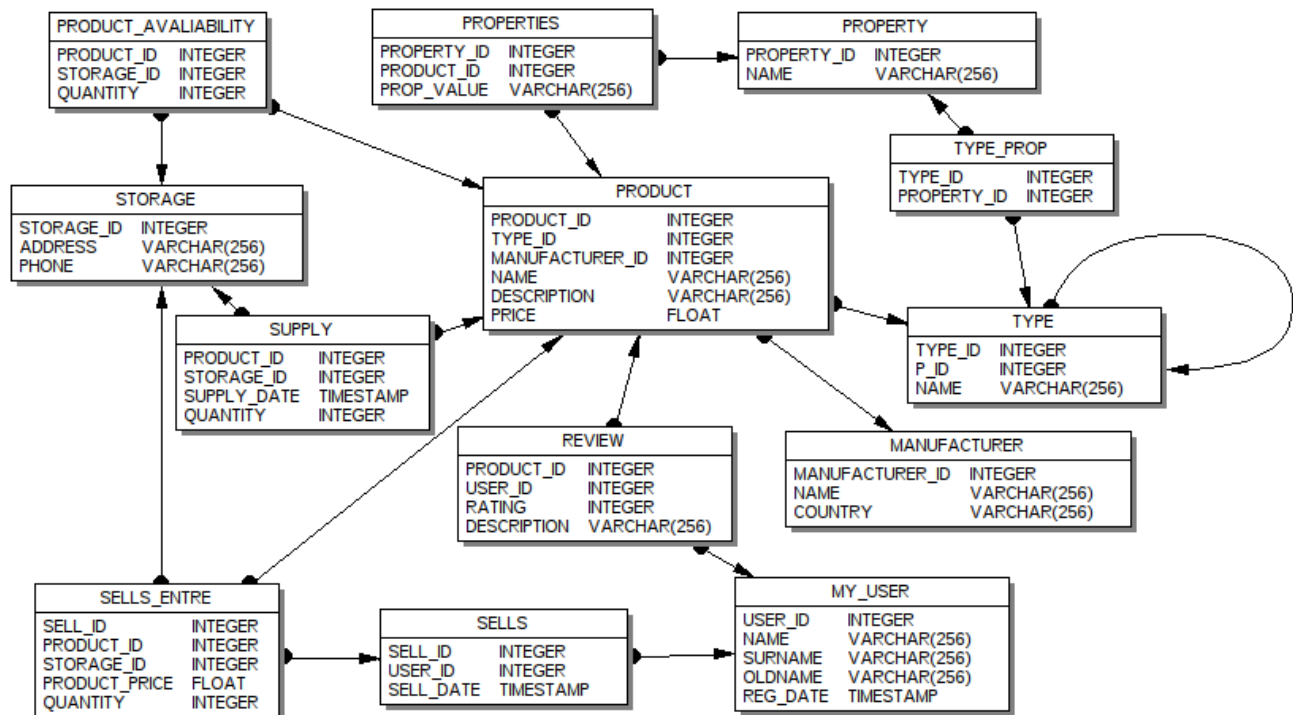


Рис. 3: ER - диаграмма, используя database designer

Для заполнения БД случайными значениями был использован инструмент Test data generator, при помощи которого было сформировано 100 000 записей для таблиц: My\_user, Product, Review.

Данный инструмент выглядит следующим образом:

Test data generator

Table: MY\_USER Records to be generated: 100000 Commit after: 500

Name	Type
<input type="checkbox"/> USER_ID	INTEGER
<input type="checkbox"/> NAME	VARCHAR(256)
<input type="checkbox"/> SURNAME	VARCHAR(256)
<input type="checkbox"/> OLDNAME	VARCHAR(256)
<input type="checkbox"/> REG_DATE	DATE

Data Generation Type

- ☒ Generate\_randomly
- ☐ Get from another table
- ☐ Get from list

Date Constraints

Min Date: 01.10.2011

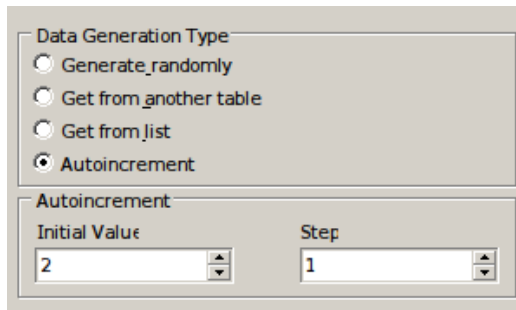
Max Date: 15.11.2016

☐ Include Time

Рис. 4: Интерфейс test data generator

Предоставляется выбор таблицы, количество записей для генерации, выбор полей для которых будут генерироваться значения, а также сам тип генерации этих значений.

Для первичных ключей где значения не должны повторяться имеется тип генерации autoincrement. При использовании 'get from list', значение для записи будет выбираться случайным образом из заданного пользователем списка.



Data Generation Type

☐ Generate randomly

☐ Get from another table

☐ Get from list

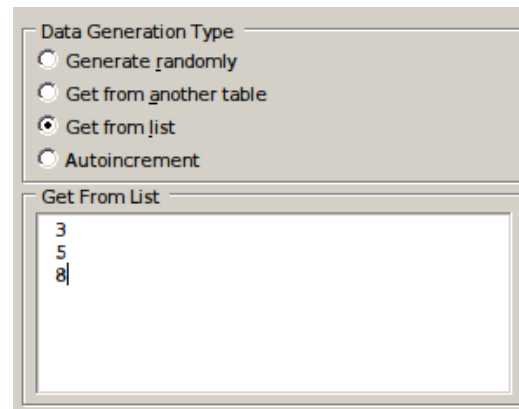
☒ Autoincrement

Autoincrement

Initial Value: 2

Step: 1

Рис. 5: Autoincrement



Data Generation Type

☐ Generate randomly

☐ Get from another table

☒ Get from list

☐ Autoincrement

Get From List

3  
5  
8

Рис. 6: Get from list

В данном случае значения 3, 5, 8 соответствовали категориям из таблицы Type.

TYPE_ID	P_ID	NAME
1	<null>	Комплектующие для ПК
2	1	Видеокарта
3	2	Мобильная видеокарта
4	1	Процессор
5	4	Серверный
6	<null>	ПК и Ноутбуки
7	6	Ноутбук
8	7	Игровые

Рис. 7: Содержимое таблицы Type

По завершению процесса генерации, выводится сообщение об успешной работе.

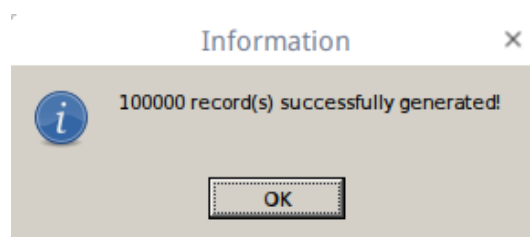


Рис. 8: Сообщение после генерации



USER_ID	NAME	SURNAME	OLDNAME	REG_DATE
99 788	dq lv	BZV3i	'j96*k7}	16.03.2012 00:00
99 789	;!Be75XA	4[23p	k-j[St1	15.10.2015 00:00
99 790	h7	W/uq	,QA{(lA;^1	24.09.2014 00:00
99 791	3rp#5	b	d8KkZ)!e Y	23.12.2011 00:00
99 792	@\}gu	[.Q:FXG(W/	`RooL;DLG	25.03.2016 00:00
99 793	xqjstqwSEG	y'{N2"X:	(.Hid3	22.07.2016 00:00
99 794	)x;AY	Ta3 ^Wuk	rw.26NvSO	28.02.2014 00:00
99 795	S	AsN)xEX	2X4#37M	19.10.2013 00:00
99 796	v/}	FjfdWByiQ	)d9	02.07.2012 00:00
99 797	u!MFK=<Q	KE1]XbU[ [	c\#E7_O#	30.11.2013 00:00
99 798	nfY,'m8 /v	>h{s	w	16.05.2012 00:00
99 799	DHB,(_I	c";Za! iX	= ZQ,[<\	19.10.2013 00:00
99 800	MJQidS	(Xf-	j	15.10.2014 00:00
99 801	!3}`#b1d1\	)B@	A>	06.10.2016 00:00
99 802	DP*KEX[J	S^e['V\Sm	R2+=+A/i	31.10.2014 00:00
99 803	NHMD^!5L	<Q/	Cy4	12.04.2015 00:00
99 804	13	&p8c*7/	&	08.05.2016 00:00
99 805	tt)Mo	Po	&Vo?(o	24.01.2016 00:00
99 806	tV	= r+2ppb	*R f5 IPJ.	27.07.2015 00:00
99 807	S`*\ZrZ	R]NRKF(`<Q	@9N4)yU7	18.01.2016 00:00
99 808	X	T h{	[x"d<Al	14.05.2012 00:00
99 809	L/dB(mD.	eprv	m/O,v	02.01.2015 00:00
99 810	Pt	`X6z+l^ez{	,mbp1eYa.G	24.11.2013 00:00
99 811	o4Z<mLq	*	DR^	24.08.2013 00:00
99 812	JJ;3	po{	V	17.12.2012 00:00
99 813	1OK1X~t P	Xl.	=]!Ps2S	04.01.2016 00:00
99 814	[&z/EDN	e;<ck`k(C	L	09.12.2013 00:00

Рис. 9: Содержимое таблицы My\_user после генерации

## Вывод

В ходе данной работы я познакомился с языком определения данных DDL. С помощью его я смог определять, изменять и удалять какие-либо структуры данных. Диалект данного языка понятен и удобен для написания каких-либо команд. В случае если результат выполнения скрипта не соответствовал ожиданиям, то могут возникнуть непредвиденные трудности.

Так-же я познакомился с СУБД IBEExpert, графический интерфейс которой, позволяет работать сразу с несколькими БД. Имеется возможность создать ER - диаграмма БД для наглядной проверки структур таблиц и зависимостей между ними. Так-же она имеет встроенные инструменты для генерации случайных записей в таблицы. Все это заметно облегчает работу с БД.