Министерство образования Республики Беларусь

Полоцкий государственный университет

**Методические**

**указания**

к лабораторным работам

по курсу «Базы данных»

для студентов специальности   
1-40 02 01 «Вычислительные машины, системы и сети»

Новополоцк 2020

## Введение

Данное методическое пособие предназначено для студентов специальности "Вычислительные машины, системы и сети" по курсу "Базы данных" (6 семестр) для выполнения лабораторных работ.

Для правильного выполнения работ необходимы знания теоретического материала по данному курсу, языков программирования (C++, PHP) и навыки работы в средах Borland C++ Builder и Microsoft Visual Studio. Также для успешного выполнения задания необходимо изучение краткой теории к соответствующей лабораторной работе.

В отчет по выполненной лабораторной работе входят: листинг программы, задание с указанием варианта, краткое описание алгоритма решения задачи. Для защиты работы необходимо ответить на ряд вопросов по программе, алгоритму решения и применяемой технологии.

## Лабораторная работа №1

#### ER-модель

**Цель работы:** освоить принципы построения модели СУЩНОСТЬ-СВЯЗЬ в нотации Питера Чена.

**Краткая теория:**

**ER-модель** – это диаграмма, описывающая основные сущности предметной области и связи между ними. Под связью понимается любой вид отношения (включение, ассоциации, принадлежности и т.д.). Важные сущности предметной области выявлябтся методом интервью с представителями бизнеса.



**Петер Пин-Шен Чен** - американский ученый в области информатики, предложивший в 1976 году ER-модель данных (модель сущность-связь). Родился в Тайчжуне (Тайвань), получил степень бакалавра в области электротехники в 1968 году в Национальном университете Тайваня и степень Ph.D в области компьютерных наук/прикладной математики в Гарвардском университете в 1973 году. С 1974 по 1978 годы работал в Массачусетском технологическом институте, в 1978—1984 — профессор Калифорниийского университета в Лос-Анджелесе. С 1983 года — профессор компьютерных наук в Университете штата Луизиана.

Работы профессора Чена являются краеугольным камнем разработки программного обеспечения, в частности, в области [CASE](https://ru.wikipedia.org/wiki/CASE) — средств.

Модель ER повлияла на разработку большинства основных CASE-инструментов, включая ERWIN компании [Computer Associates](https://ru.wikipedia.org/wiki/CA_Technologies), Designer/2000 корпорации [Oracle](https://ru.wikipedia.org/wiki/Oracle), PowerDesigner компании [Sybase](https://ru.wikipedia.org/wiki/Sybase) и даже [Microsoft Visio](https://ru.wikipedia.org/wiki/Microsoft_Visio), а также стандарт [IDEF1X](https://ru.wikipedia.org/wiki/IDEF1).

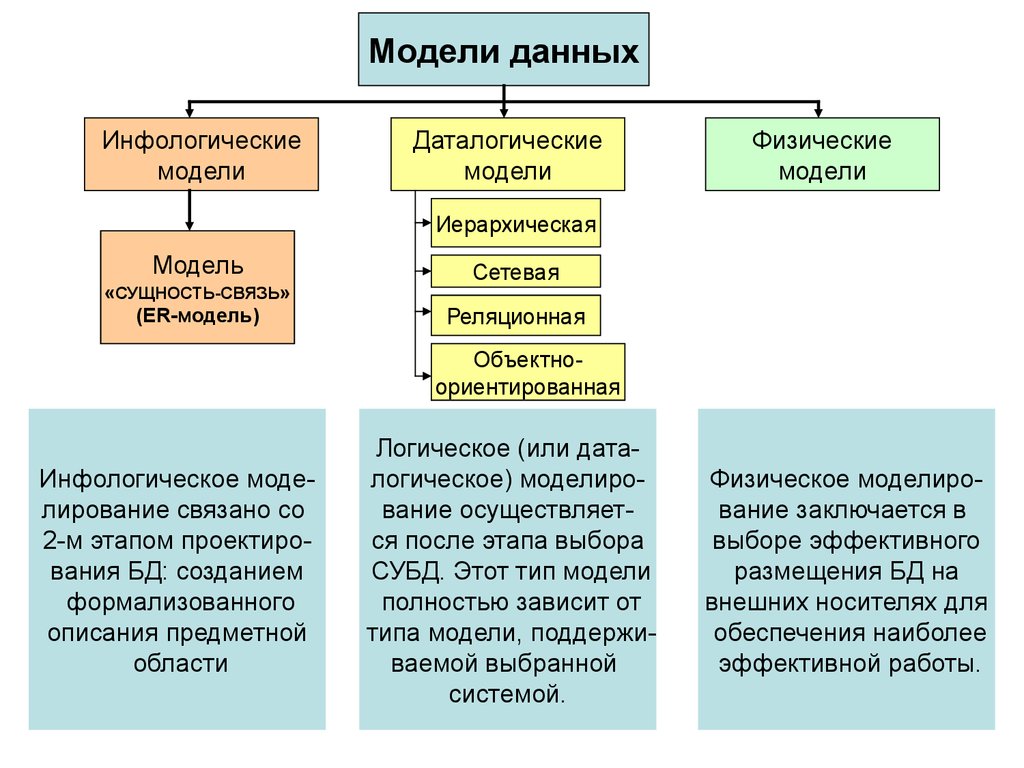


Рис. 1 - Классификация информационных моделей

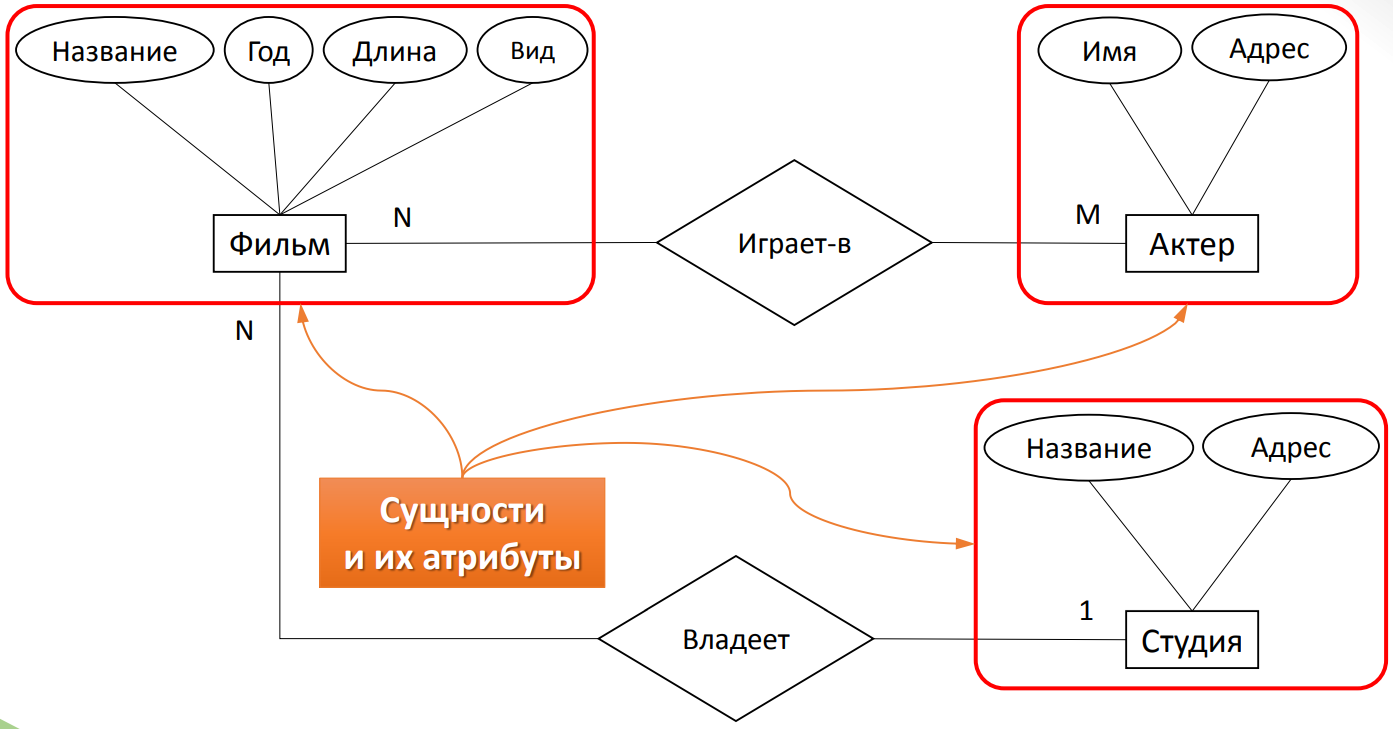


Рис 2. - Изображение сущностей и атрибутов на диаграмме.

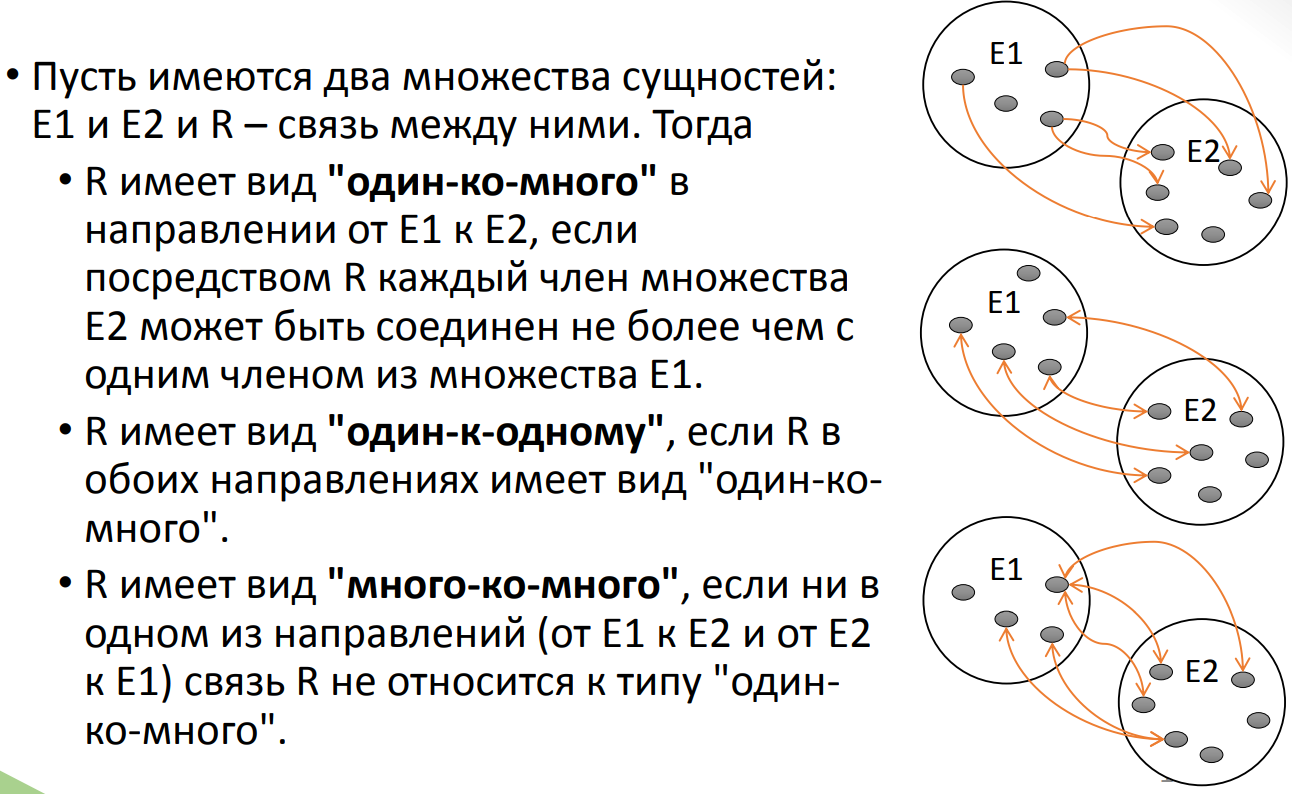


Рис. 3 - Виды бинарных связей

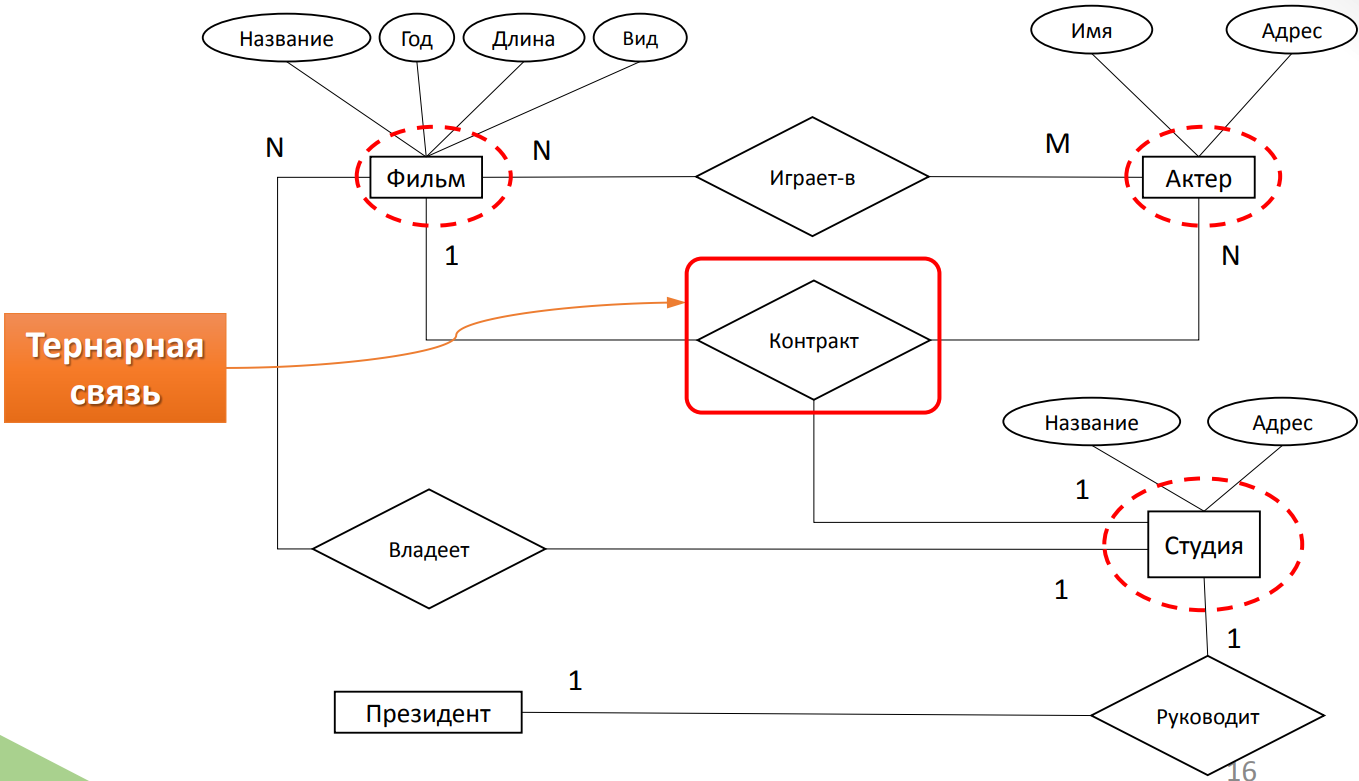


Рис. 4 - Тернарные связи

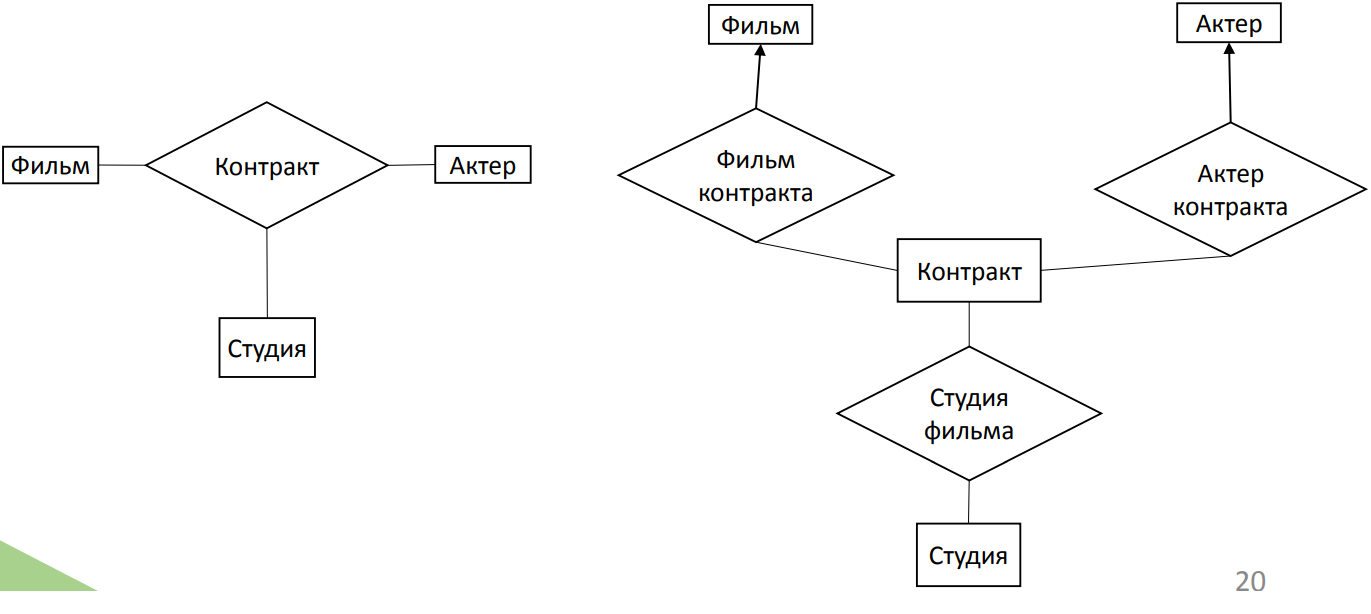


Рис. 5 – Преобразование связей

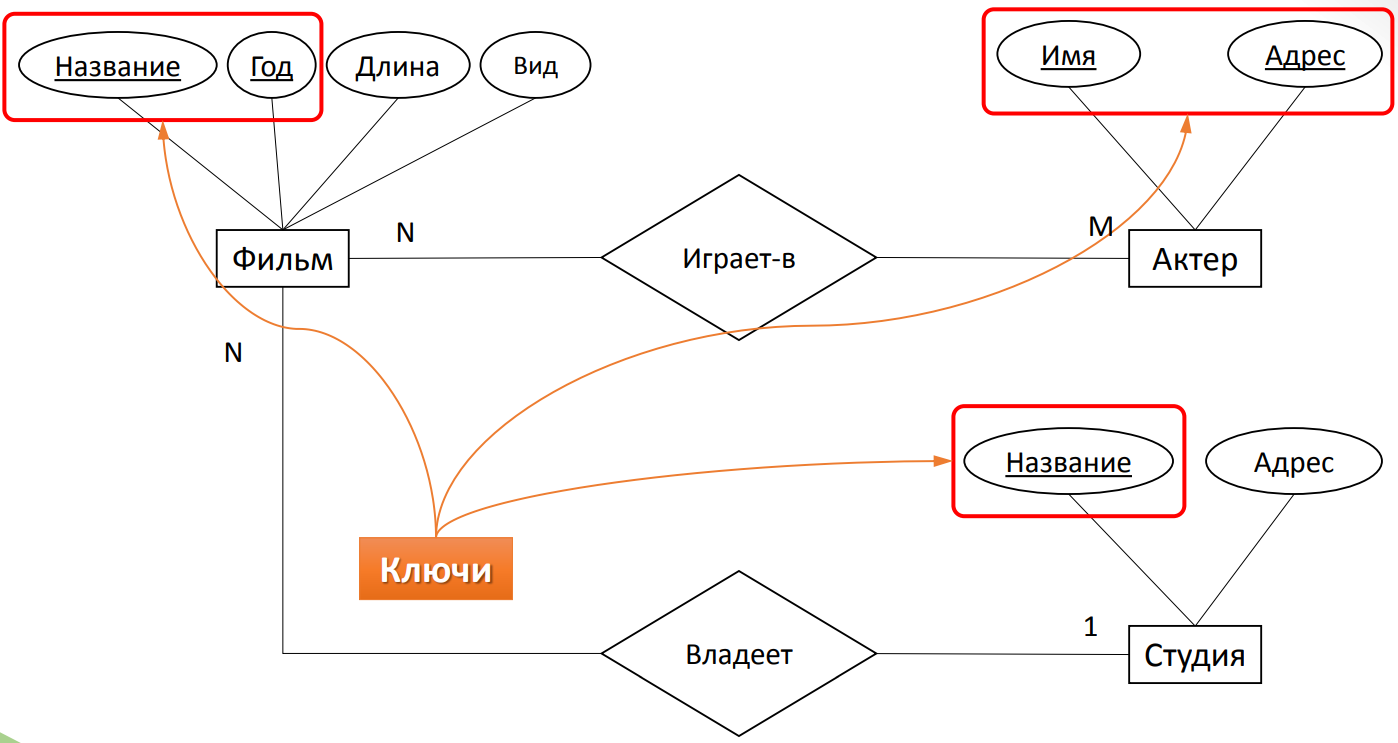


Рис. 6 – Отображение ключевых аттрибутов

**Задания:**

Задача состоит в построении информационной модели предметной среды в соответствии с темой, выданной преподавателем, в виде ER-модели в нотации Питера Чена.

##### Порядок выполнения работы:

1. Сформировать описание бизнес-процессов и выявить основные сущности учета.
2. Сформировать ER-модель.
3. Осуществить преобразование N-арных отношений в бинарные.
4. Результат загрузить в Google classroom.

Таблица 1 ─ Варианты заданий на лабораторную работу №1

|  |  |
| --- | --- |
| № варианта | Тема |
| 1 | Библиотека |
| 2 | Аптека |
| 3 | Спортивный клуб |
| 4 | Электронный магазин (рыболовных / строительных / …) товаров |
| 5 | Компьютерная игра MMORPG |
| 6 | Портал музыки |
| 7 | Архив конструкторской документации |
| 9 | Лодочная станция |
| 10 | Пункт проката |
| 11 | Музей |
| 12 | Картинная галлерея |
| 13 | Склад |
| 14 | Компьютерный клуб |
| 15 | Спортивные соревнования |

##### Контрольные вопросы.

1. Опишите алгоритм решения задачи.
2. Какие сущности Вы использовали?
3. Какие отношения были выявлены?
4. От чего зависит качество модели?
5. Какие преобразования модели Вы выполнили?

## Лабораторная работа №2

#### Физическая модель базы данных

**Цель работы:** освоить правила построения физической модели БД по построенной ранее ER-модели.

**Краткая теория:**

##### Современные системы управления базами данных (СУБД) придерживаются реляционной модели данных, предложенной Эдгаром Коддом.

**Эдгар Франк Кодд –** появился на свет 23 августа 1923 года в Портленде (английского графства Дорсет) в многодетной семье. Кодд сумел поступить и успешно окончить Оксфордский университет. Служил пилотом в ВВС Великобритании (в период Второй мировой войны). с 1949 года, когда он переезжает в США и приступает к работе математиком-программистом в IBM.

Он принимает участие в разработке [Selective Sequence Electronic Calculator](http://en.wikipedia.org/wiki/IBM_SSEC) — первого лампового компьютера IBM. Первым серьезным проектом талантливого математика в этой компании становится разработка системы мультипрограммирования, делающей возможным одновременное выполнение нескольких задач, а предназначается это для [IBM STRETCH](http://en.wikipedia.org/wiki/IBM_7030_Stretch) – нового экспериментального компьютера.

IBM существенно повлияла и на личную жизнь Кодда, ведь именно там он познакомился со своей будущей супругой Шэрон, ставшей впоследствии не только матерью его четверых детей, но его верной соратницей и продолжательницей его идей.

Получив в Мичиганском университете (магистратуру которого Эдгар Кодд окончил в качестве стипендиата от IBM) степень доктора по информатике и вычислительной технике, он уже через два года приступает к работе в Исследовательской лаборатории (сегодня — Альмаденском Исследовательском центре) IBM, расположенном в знаменитой Кремниевой долине.

Именно там находит реализацию гениальная (и, однозначно, революционная) идея Кодда по созданию [Реляционной модели организации данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

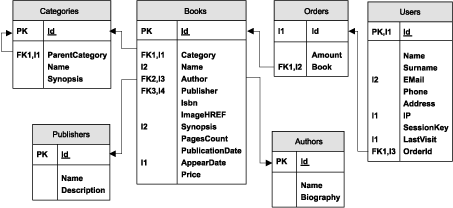


Рис. 1 – Пример физической модели данных

Физическая модель БД в обязательном порядке содержит информацию об отношениях (таблицах) БД, атрибутах отношений и их типах, связях, построенных на определении первичного и внешних ключей. Обратите внимание, что не должно быть связей 1 к 1 и многие ко многим. Также могут быть указаны какие поля требуют построения индексов.

**Основные типовые ошибки проектирования**

* Циклы в схеме БД
* Особенности отношения 1-1
* Применение суррогатных ключей
* Контроль типов

Рассмотрим на примерах:

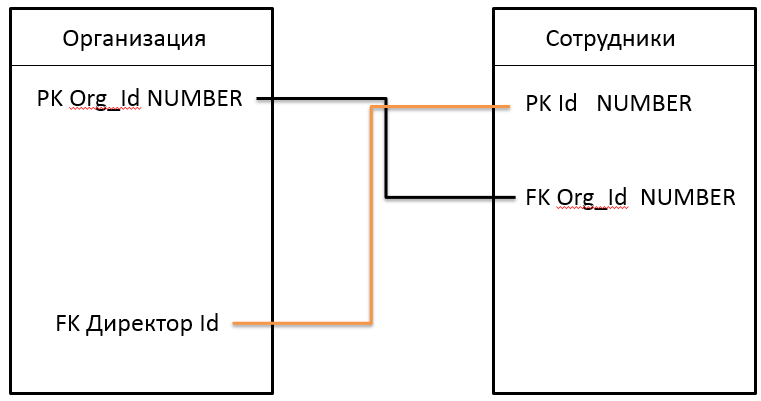


Рис. 2 - Пример цикла с невозможностью вставки записи

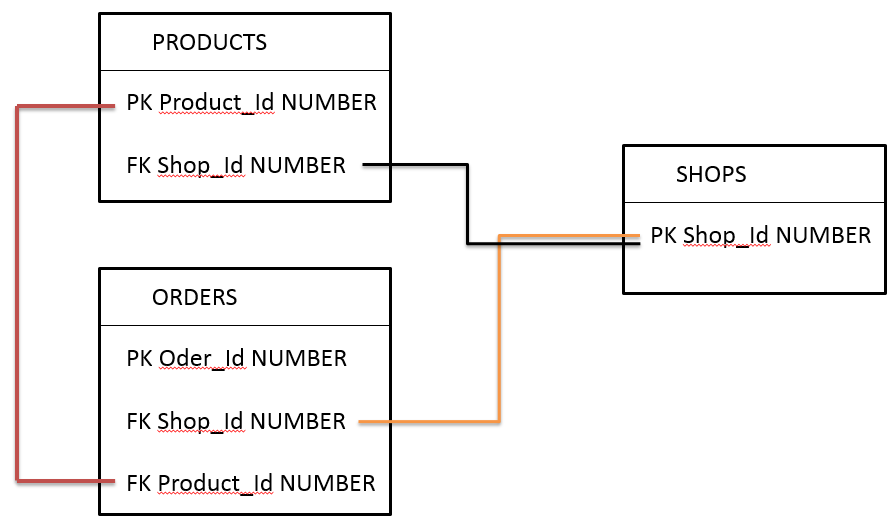


Рис. 3 - Пример цикла с двусмысленностью

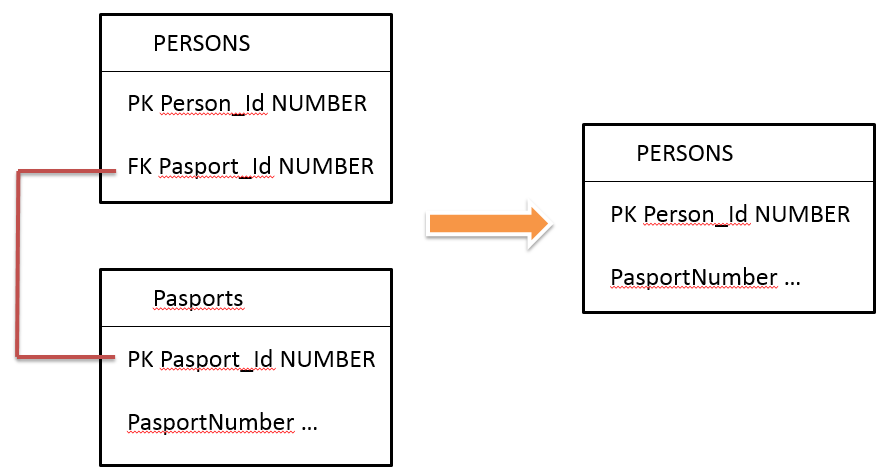


Рис. 4 - Связь один к одному

**Задание:**

Используя свободные инструменты построения схем (draw.io или аналоги) разработайте физическую модель БД.

##### Порядок выполнения работы:

1. Создайте отношения (таблицы) физической модели.
2. Устраните связи 1х1 и MxN. Последний тип устраняется введением таблицы-посредника.
3. Определите ключи, если целесообразно – введите суррогатные.
4. Нарисуйте связи и устраните циклы.
5. Проверьте схему БД на соответствие 1, 2, 3 нормальным формам.
6. Результат загрузить в Google classroom.

##### Контрольные вопросы.

1. Опишите алгоритм построения физической модели.
2. Какие сущности Вы превратили в таблицы БД?
3. Какие первичные ключи Вы определили?
4. Какие связи были реализованы и как Вы устранили наличие связей типа 1х1 MxN?
5. Нормализована ли полученная Вами схема БД?

## Лабораторная работа №3

#### Перенос физической модели в СУБД Oracle XE

**Цель работы:** освоить работу с БД Oracle XE. Осуществить перенос БД в Oracle. Наполнить БД тестовыми данными.

**Краткая теория:**

После установки и входа в Oracle под системным пользователем (sys с паролем, введенным во время инсталляции СУБД) необходимо создать своего пользователя и перевойти в систему под ним.

1. Затем необходимо перейти во вкладку управления таблицами.

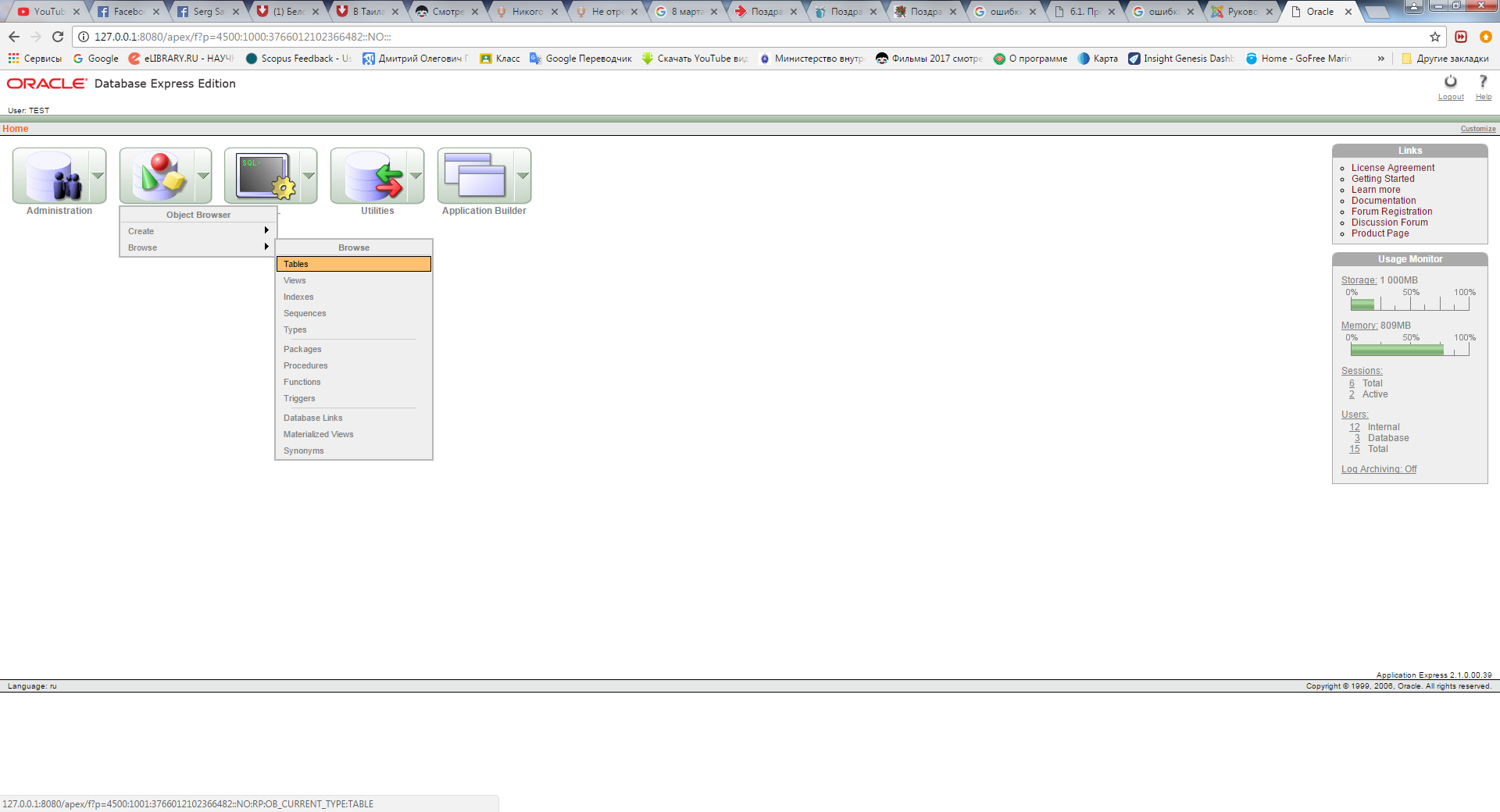


Рис. 1 – Управление таблицами

1. Необходимо создать все таблицы БД, начиная с таблиц-справочников, поскольку создать связи (внешние ключи) на несуществующие таблицы невозможно.

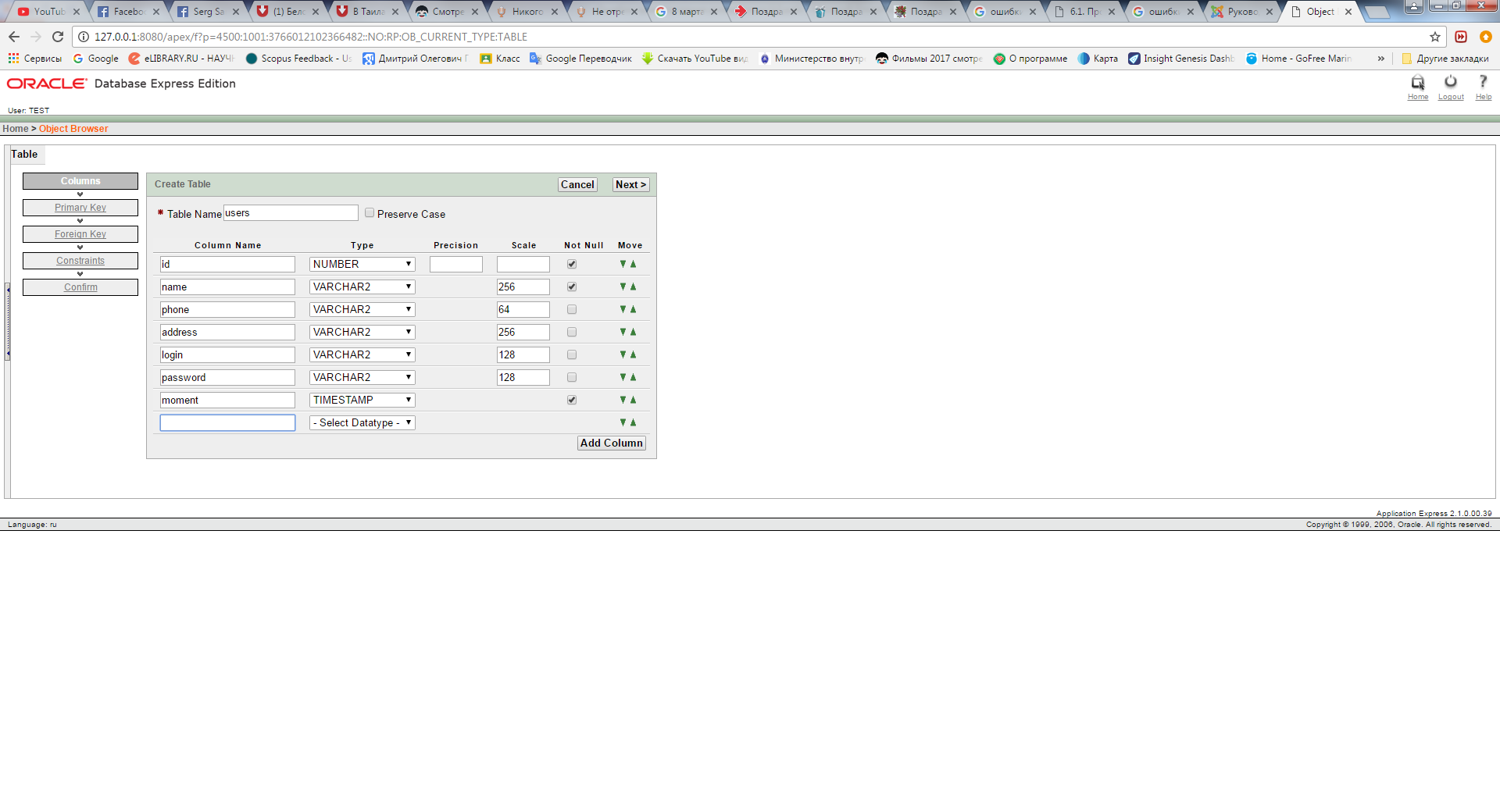


Рис. 2 – Создание таблицы

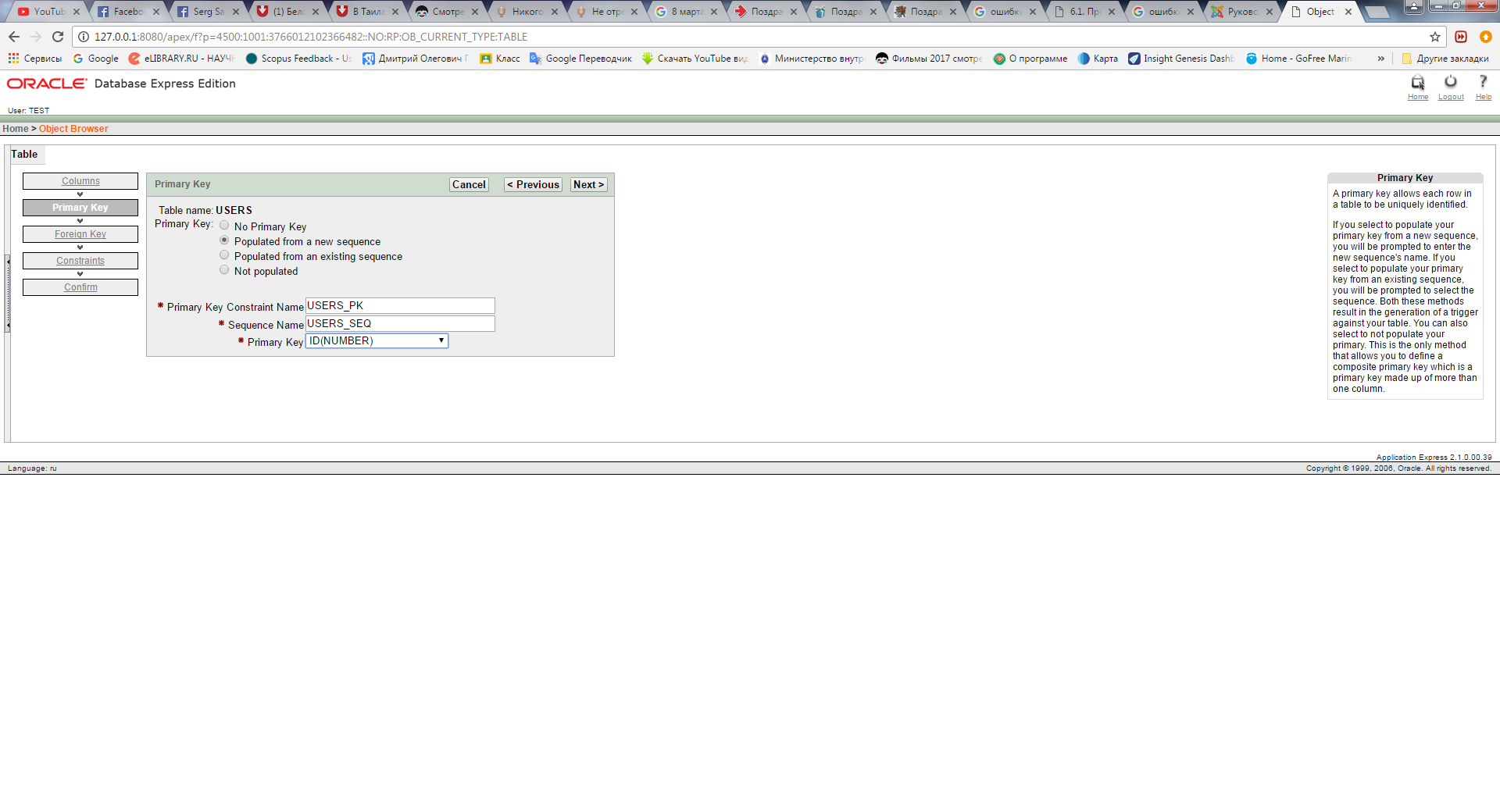


Рис. 3 – Создание первичного ключа как автоинкремент-поля из новой последовательности

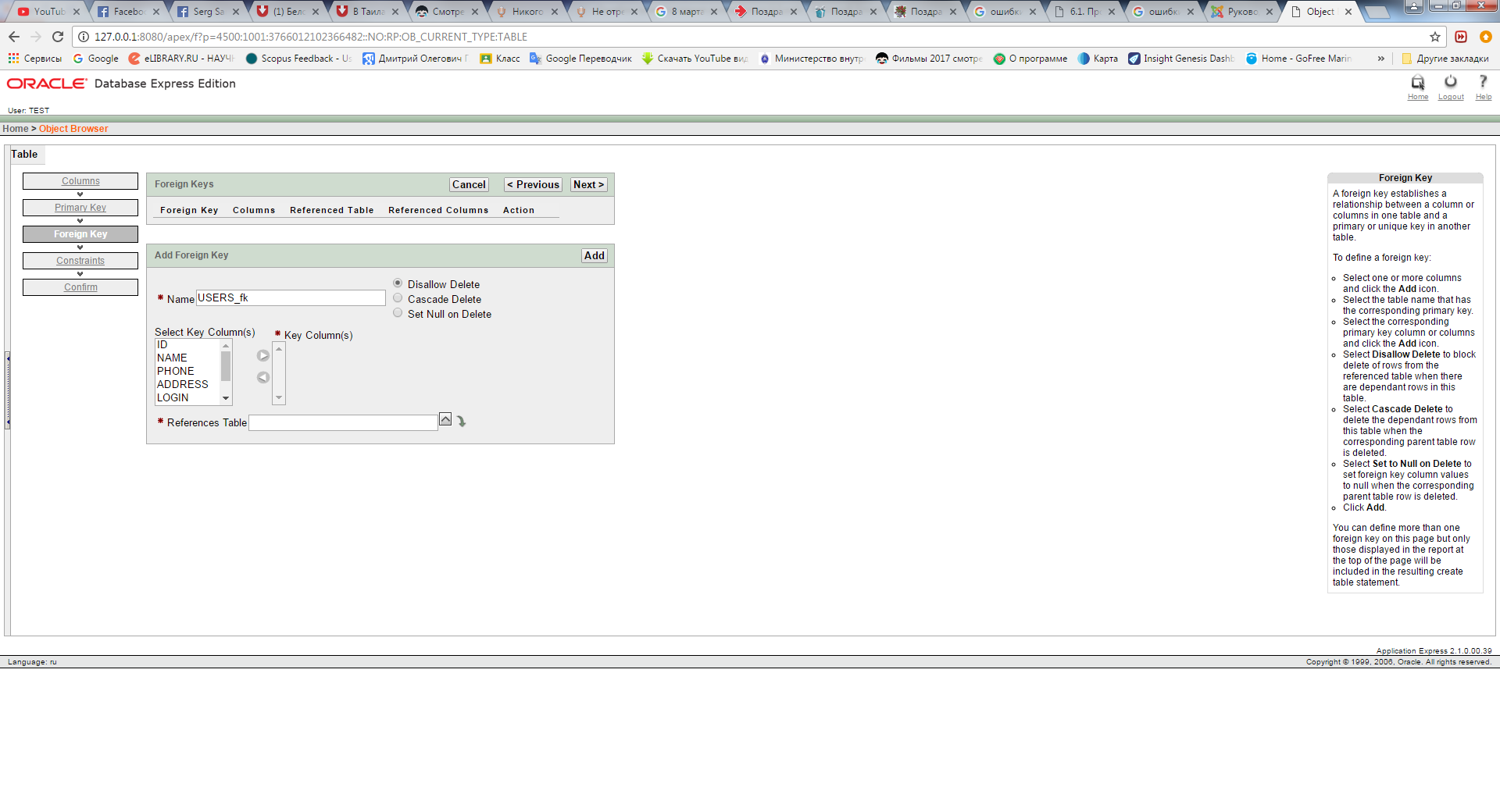


Рис. 4 – Пропуск внешних ключей

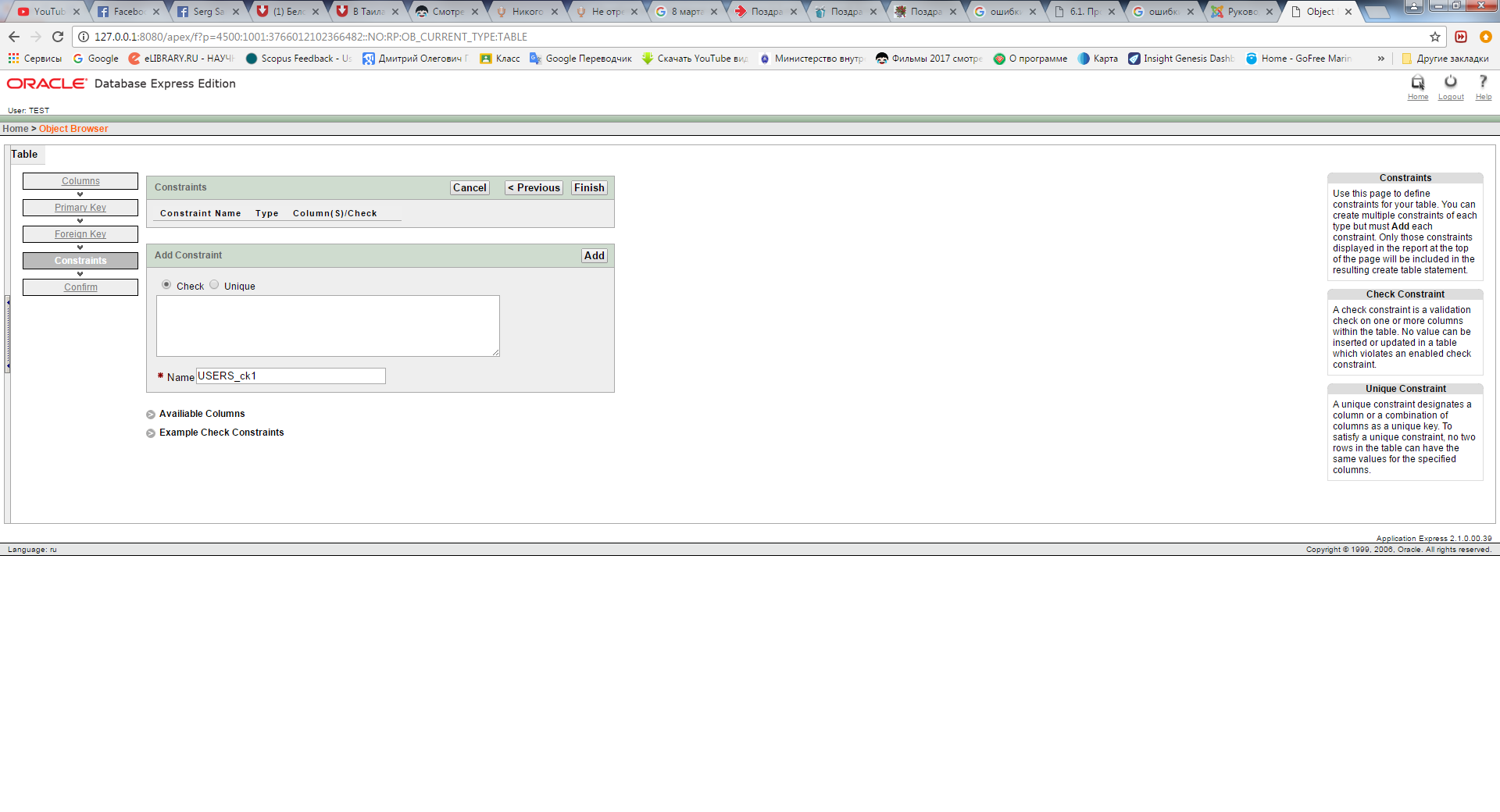


Рис. 5 – Создание ограничений

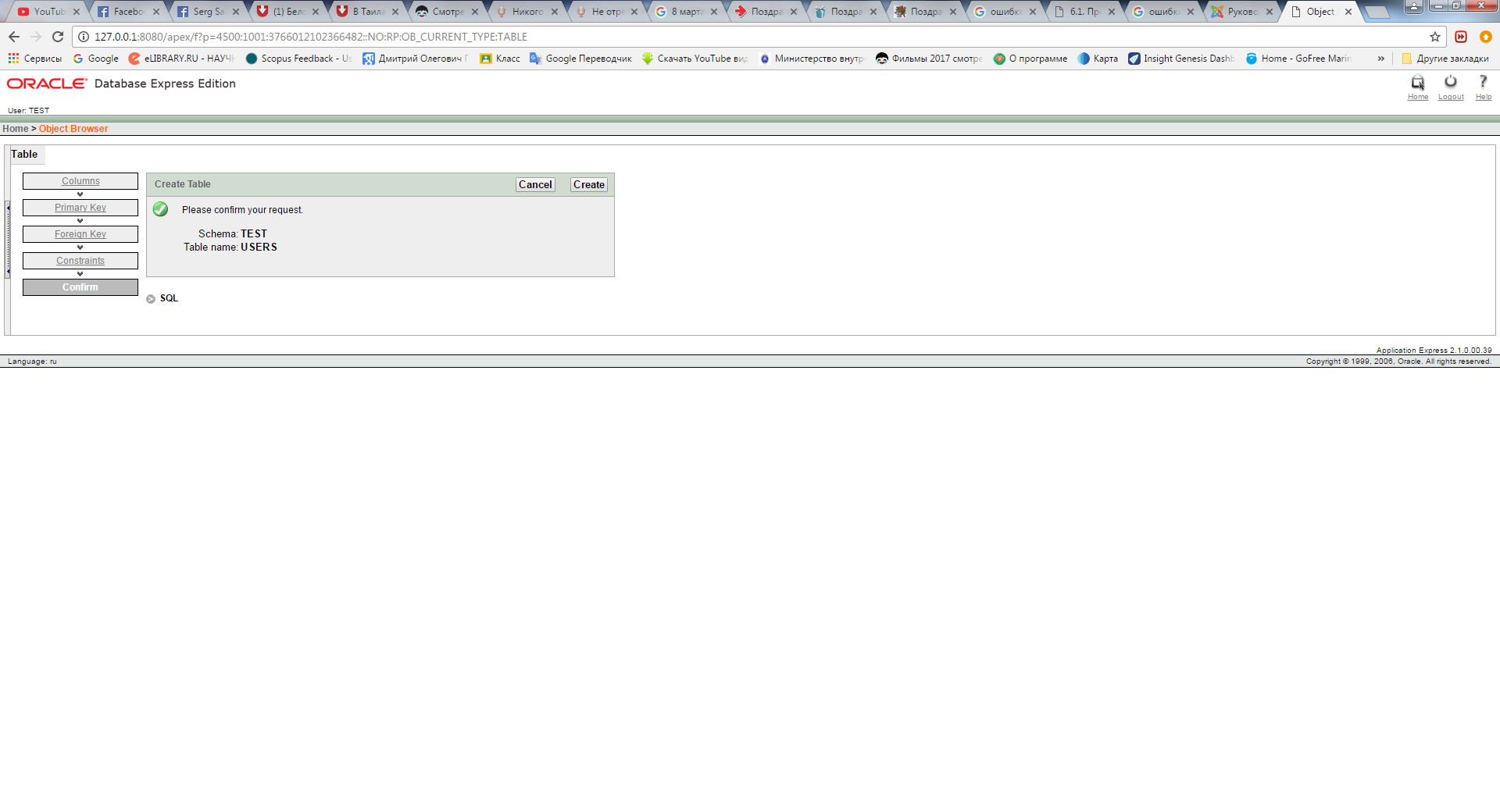


Рис. 6 – Подтвеждение действия

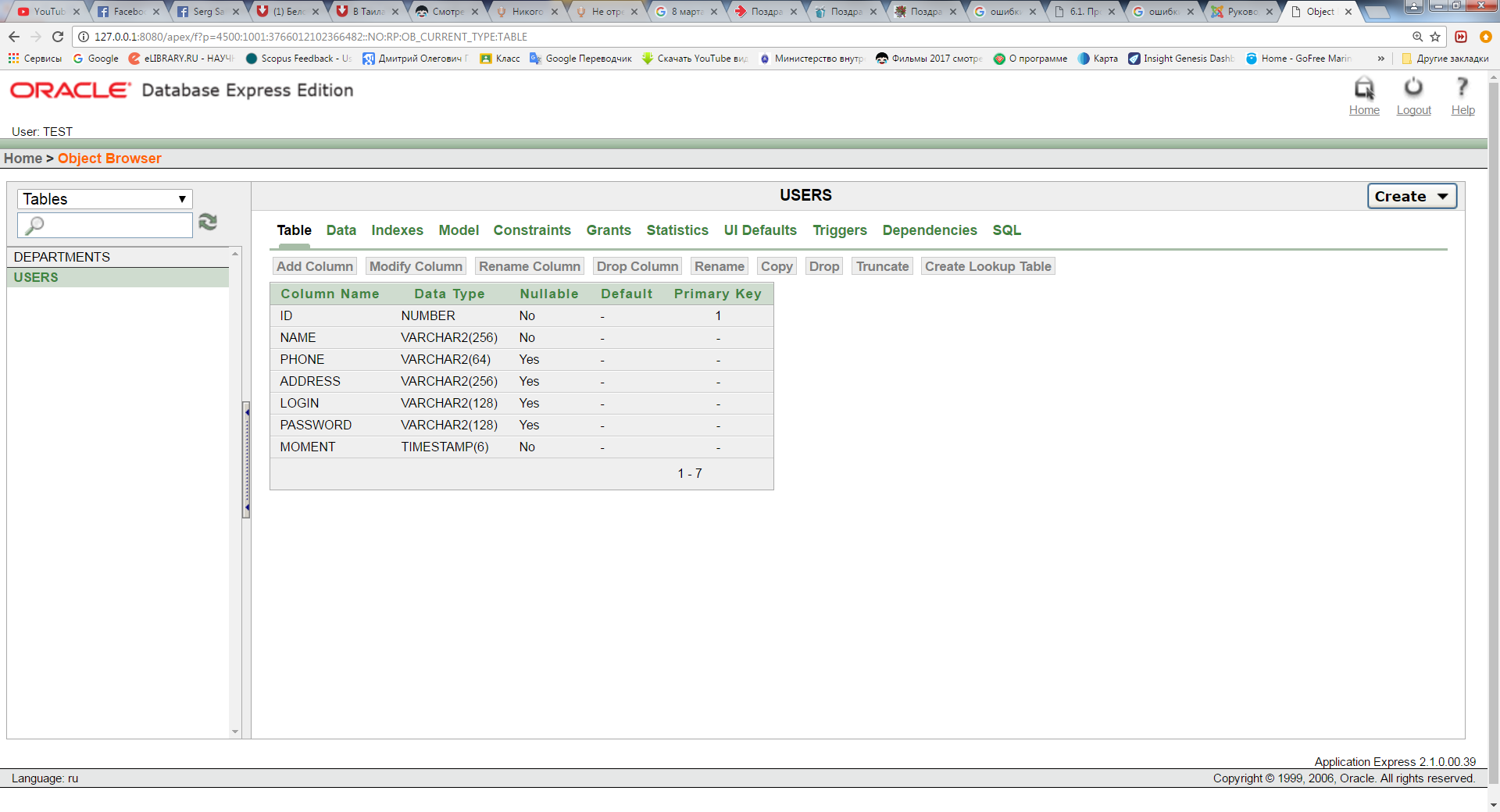


Рис. 7 – Результат создания таблицы

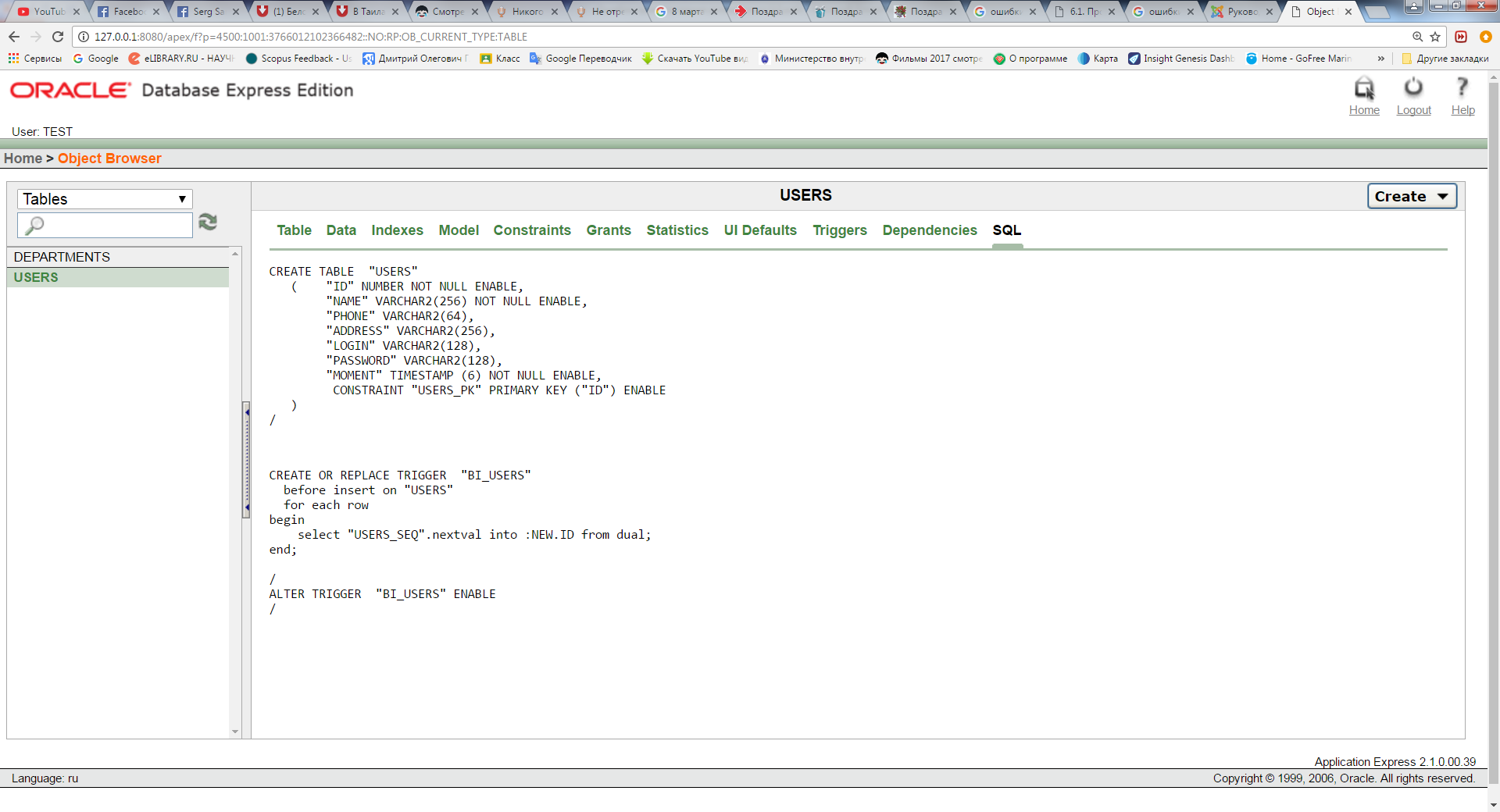


Рис. 8 – SQL представление таблицы

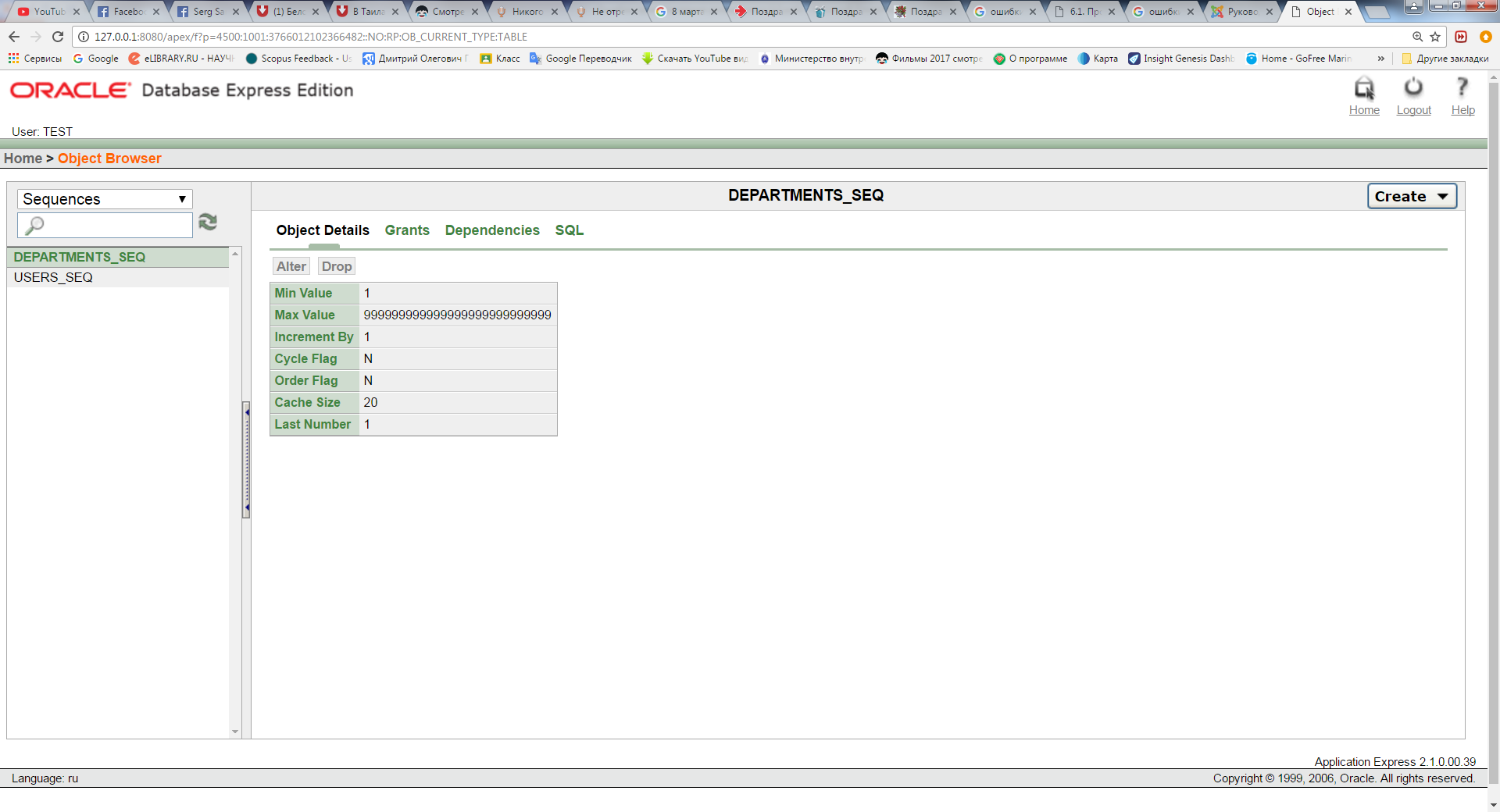


Рис. 9 – Созданная новая последовательность для автоинкремент поля

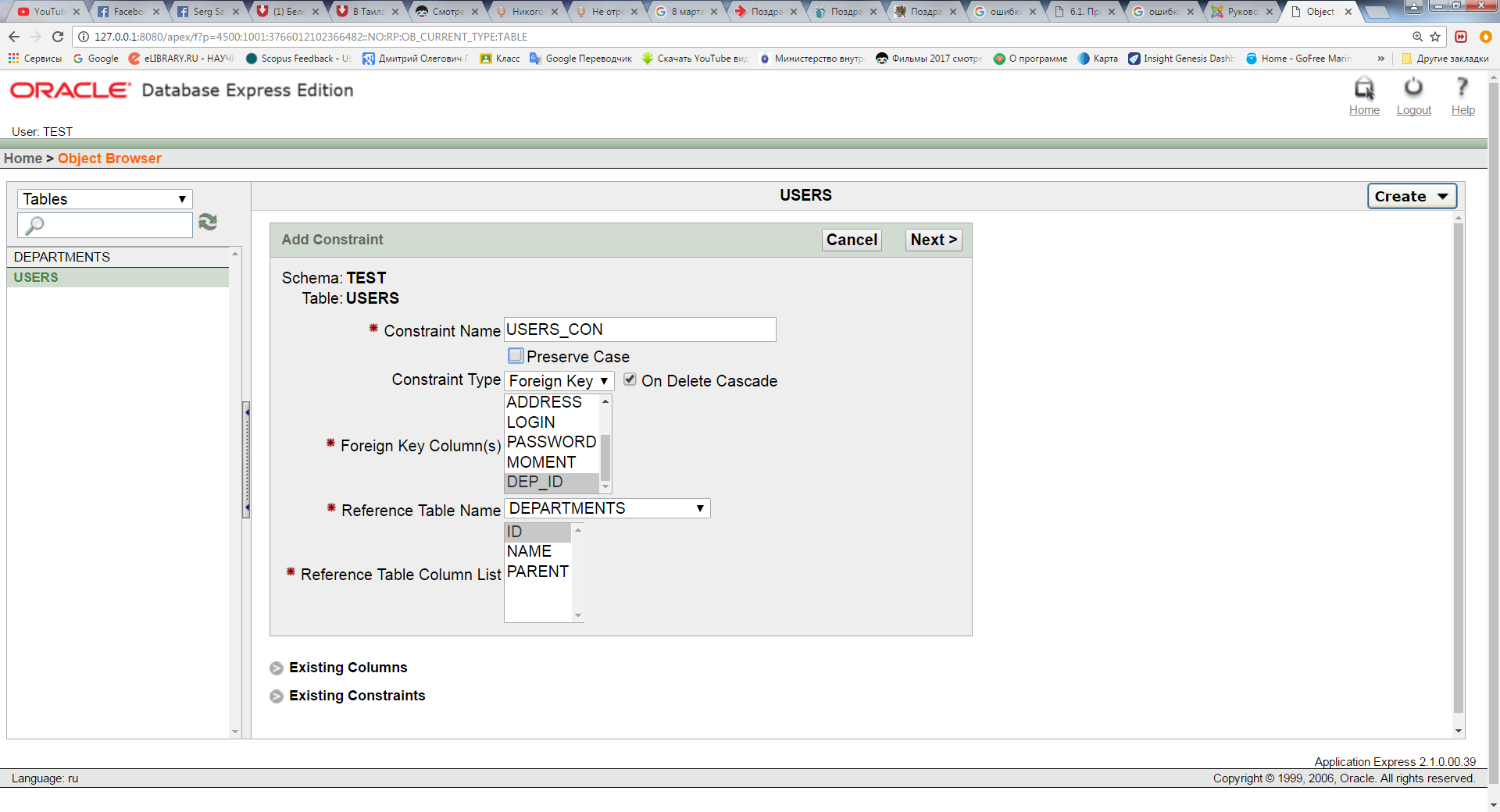


Рис. 10 – Создание внешних ключей

Таким образом, необходимо получить скрипт создания схемы БД, который для каждой таблицы содержит следующие инструкции:

1. Инструкцию CREATE TABLE с определением первичного и всех внешних ключей в виде ограничений (constraint);
2. Инструкцию создания последовательности CREATE SEQUENCE
3. Инструкцию создания триггера CREATE TRIGGER типа BEFORE INSERT для заполнения автоинкремент полей;
4. Инструкцию ALTER TRIGER для его разблокирования;
5. Инструкции создания индексов при необходимости.

##### Порядок выполнения работы:

1. Установить и запустить OracleXE.
2. Создать пользователя.
3. Войти под созданным пользователем.
4. Создать в OracleXE все необходимые таблицы и связи.
5. Заполнить БД тестовыми данными.
6. Скрипт создания БД и наполнения ее тестовыми данными, а также скриншоты Oracle с готовой схемой скопировать в Google Classroom.

##### Контрольные вопросы.

1. Что такое физическая модель данных реляционной модели?
2. Какие типы данных есть в Oracle?
3. Какие новые объекты БД вы создали?
4. Какие индексы вы определили?
5. В каком порядке Вы создавали таблицы, почему?

## Лабораторная работа №4

#### Построение сложных запросов SQL к БД

**Цель работы:** освоить работу с запросами SQL. Научиться строить сложные SQL запросы.

**Краткая теория:**

**SQL** (Structured Query Language) — [декларативный](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), применяемый для создания, модификации и управления данными в [реляционной базе данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), управляемой соответствующей [системой управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Является, прежде всего, [информационно-логическим языком](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), предназначенным для описания, изменения и извлечения данных, хранимых в [реляционных базах данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). SQL считается [языком программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в общем случае (без ряда современных расширений) не является [тьюринг-полным](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83), но вместе с тем стандарт языка спецификацией [SQL/PSM](https://ru.wikipedia.org/wiki/SQL/PSM) предусматривает возможность его [процедурных расширений](https://ru.wikipedia.org/wiki/SQL#%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D1%8B%D0%B5_%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D1%8F).

Изначально SQL был основным способом работы пользователя с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

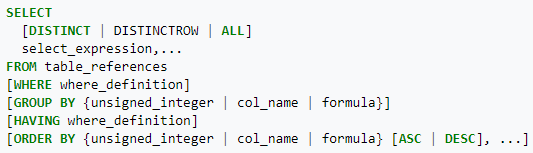
При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

**SELECT** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) select — «выбрать») — оператор запроса ([DML](https://ru.wikipedia.org/wiki/DML)/[DQL](https://ru.wikipedia.org/w/index.php?title=DQL&action=edit&redlink=1)) в языке [SQL](https://ru.wikipedia.org/wiki/SQL), возвращающий набор данных (выборку) из [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Оператор возвращает ноль или более строк. Список возвращаемых столбцов задается в части оператора, называемой предложением SELECT. Поскольку SQL является декларативным языком, запрос SELECT определяет лишь требования к возвращаемому набору данных, но не является точной инструкцией по их вычислению. [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94) транслирует запрос SELECT во внутренний план исполнения («query plan»), который может различаться даже для синтаксически одинаковых запросов и от конкретной СУБД.

Оператор SELECT состоит из нескольких предложений (разделов):

* SELECT определяет список возвращаемых столбцов (как существующих, так и вычисляемых), их имена, ограничения на уникальность строк в возвращаемом наборе, ограничения на количество строк в возвращаемом наборе;
* FROM задаёт табличное выражение, которое определяет базовый набор данных для применения операций, определяемых в других предложениях оператора;
* WHERE задает ограничение на строки табличного выражения из предложения FROM;
* GROUP BY объединяет ряды, имеющие одинаковое свойство с применением агрегатных функций
* HAVING выбирает среди групп, определённых параметром GROUP BY
* ORDER BY задает критерии сортировки строк; отсортированные строки передаются в точку вызова.



**GROUP BY** — необязательный параметр операторa SELECT, для группировки строк по результатам агрегатных функций (MAX, SUM, AVG, …).

Необходимо, чтобы в SELECT были заданы только требуемые в выходном потоке столбцы, перечисленные в GROUP BY и/или агрегированные значения. Распространённая ошибка — указание в SELECT столбца, пропущенного в GROUP BY.

**HAVING** — необязательный (опциональный) параметр оператора SELECT для указания условия на результат агрегатных функций (MAX, SUM, AVG, …).

HAVING <условия> аналогичен WHERE <условия> за исключением того, что строки отбираются не по значениям столбцов, а строятся из значений столбцов, указанных в [GROUP BY](https://ru.wikipedia.org/wiki/GROUP_BY_(SQL)), и значений агрегатных функций, вычисленных для каждой группы, образованной GROUP BY.

Необходимо, чтобы в SELECT были заданы только требуемые в выходном потоке столбцы, перечисленные в GROUP BY и/или агрегированные значения. Распространённая ошибка — указание в SELECT столбца, пропущенного в GROUP BY.

Если параметр GROUP BY в SELECT не задан, HAVING применяется к «группе» всех строк таблицы, полностью дублируя [WHERE](https://ru.wikipedia.org/wiki/WHERE) (допускается не во всех реализациях стандарта [SQL](https://ru.wikipedia.org/wiki/SQL)).

**ORDER BY** — необязательный (опциональный) параметр операторов SELECT и [UNION](https://ru.wikipedia.org/wiki/Union_(SQL)), который означает что операторы SELECT, [UNION](https://ru.wikipedia.org/wiki/Union_(SQL)) возвращают набор строк, отсортированных по значениям одного или более столбцов. Его можно применять как к числовым столбцам, так и к строковым. В последнем случае, сортировка будет происходить [по алфавиту](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA).

Использование предложения ORDER BY является единственным способом отсортировать результирующий набор строк. Без этого предложения СУБД может вернуть строки в любом порядке. Если упорядочение необходимо, ORDER BY должен присутствовать в SELECT, UNION.

Сортировка может производиться как по возрастанию, так и по убыванию значений.

Параметр ASC (по умолчанию) устанавливает порядок сортировки по возрастанию, от меньших значений к большим.

Параметр DESC устанавливает порядок сортировки по убыванию, от больших значений к меньшим.

**JOIN** — оператор языка [SQL](https://ru.wikipedia.org/wiki/SQL), который является реализацией [операции соединения](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_%D1%81%D0%BE%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_(%D0%A1%D0%A3%D0%91%D0%94)) [реляционной алгебры](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0). Входит в предложение [FROM](https://ru.wikipedia.org/w/index.php?title=From_(SQL)&action=edit&redlink=1) операторов [SELECT](https://ru.wikipedia.org/wiki/Select_(SQL)), [UPDATE](https://ru.wikipedia.org/wiki/Update_(SQL)) и [DELETE](https://ru.wikipedia.org/wiki/Delete_(SQL)).

Операция соединения, как и другие [бинарные операции](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F), предназначена для обеспечения выборки данных из двух таблиц и включения этих данных в один результирующий набор. Отличительными особенностями операции соединения являются следующие:

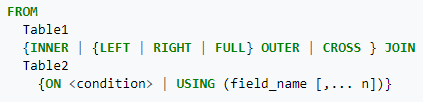
в схему таблицы-результата входят столбцы обеих исходных таблиц (таблиц-[операндов](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4)), то есть схема результата является «сцеплением» схем операндов;

каждая строка таблицы-результата является «сцеплением» строки из одной таблицы-операнда со строкой второй таблицы-операнда.

Определение того, какие именно исходные строки войдут в результат и в каких сочетаниях, зависит от типа операции соединения и от явно заданного условия соединения. Условие соединения, то есть условие сопоставления строк исходных таблиц друг с другом, представляет собой [логическое](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BA%D0%B0) выражение ([предикат](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%B8%D0%BA%D0%B0%D1%82)).

При необходимости соединения не двух, а нескольких таблиц, операция соединения применяется несколько раз (последовательно).

SQL-операция JOIN является реализацией операции соединения реляционной алгебры только в некотором приближении, поскольку в реляционной модели данных соединение выполняется над отношениями, которые являются [множествами](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE), а в SQL — над таблицами, которые являются [мультимножествами](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Результаты операций тоже, в общем случае, различны: в реляционной алгебре результат соединения даёт отношение (множество), а в SQL — таблицу (мультимножество).



Для тестовой БД <https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all> изучите ее структуру и рассмотрите примеры запросов:

1. Найдите самый дорогой товар (Product)

SELECT \* FROM [Products] order by price desc

Cote de Blaye = 263.5

2. Определите поставщика (Supplier) этого товар

SELECT \* FROM Products natural join Suppliers order by price desc

Aux joyeux ecclesiastiques

3.Сколько раз был продан этот товар

SELECT ProductName, sum(Quantity), min(Price) as lot FROM Products natural join Suppliers nutural join orders natural join orderdetails group by ProductName order by lot desc

239

4. Определите сколько было продано единиц самого продаваемого продукта?

SELECT sum(Quantity) as lot, p.productname FROM Orders o join orderDetails d on o.orderid = d.orderid join products p on p.productid = d.productid group by p.productname order by lot desc

458

5. Определите среднее число продаж товаров одного наименования?

SELECT avg(lot) FROM (SELECT count(\*) as lot, p.productname FROM Orders o join orderDetails d on o.orderid = d.orderid join products p on p.productid = d.productid group by p.productname order by lot desc)

= 165.4935064935065

count(\*) = 6.7272727272727275

6. Сколько единиц товаров продал самый успешный по доходу (по сумме заработанных денег) поставщик (Supplier) и кто он?

SELECT sum(Quantity), SupplierName, sum(Quantity \* Price) as lot FROM Suppliers s join products p on s.supplierid=p.supplierid join orderdetails d on d.productid=p.productid join orders o on o.orderid=d.orderid group by SupplierName order by lot desc

Aux joyeux ecclesiastiques = 505

7. Какой покупатель (Customer) потратил больше всего денег?

SELECT CustomerName, sum(Price\*Quantity) as sum FROM Customers c join orders o on o.CustomerID=c.CustomerID join OrderDetails d on d.OrderID=o.OrderID join products p on p.ProductID=d.ProductID group by CustomerName order by sum desc

Ernst Handel = 35631.21000000001

8. Какой продавец (Shipper) продал товаров большему количеству покупателей?

SELECT ShipperName, count(CustomerName) as lot FROM Shippers natural join products natural join orderdetails natural join orders natural join customers group by ShipperName order by lot desc

United Package

9. Скольким покупателям продал свою продукцию этот продавец (Shipper)?

188

10. На какую сумму продал этот продавец (Shipper) товаров?

SELECT ShipperName, count(CustomerName) as lot, sum(Quantity\*Price) FROM Shippers natural join products natural join orderdetails natural join orders natural join customers group by ShipperName order by lot desc

155152.03

**Задания:**

Задача состоит в построении нескольких сложных запросов SQL, формулировка которых дается преподавателем. Опираясь на знания языка SQL, встроенных функций и инструментов реляционной алгебры, постройте запросы и выведите результат запроса по Вашей БД с тестовым наполнением.

##### Порядок выполнения работы:

1. Изучите приведенные примеры запросов по тестовой БД.
2. Изучите поставленные Вам запросы и продумайте подходы к их реализации.
3. Отработайте фрагменты запросов (подзапросы) и убедитесь в их корректности.
4. Постройте итоговые запросы.
5. Оформите отчет.
6. Загрузите отчет в Google Classroom.

##### Контрольные вопросы.

1. Что такое SQL?
2. Какие функции работы со строками есть в SQL?
3. Какие функции работы с датами есть в SQL?
4. Виды join-ов в SQL.
5. Реализация теоретико множественных операции в SQL.
6. Как выполнити операцию переименования?
7. Как Вы использовали инструменты группировки и сортировки данных?

## Лабораторная работа №5

#### Разработка прикладного ПО для работы с БД

**Цель работы:** освоить работу на C++ с популярными библиотеками, обеспечивающими взаимодействие с БД. Создать программу, предоставляющую удобный пользовательский интерфейс по работе со спроектированной БД.

**Краткая теория:**

Рассмотрим процесс создания приложения на С++ Builder 6.0 для работы с БД.

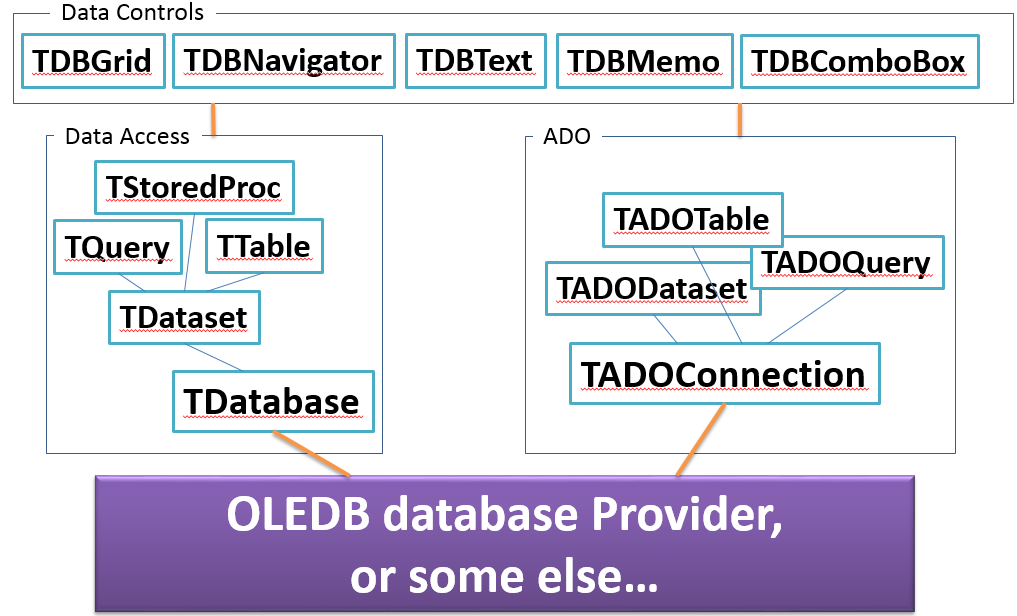


Рис. 1 – Структура классов C++ Builder для работы с БД

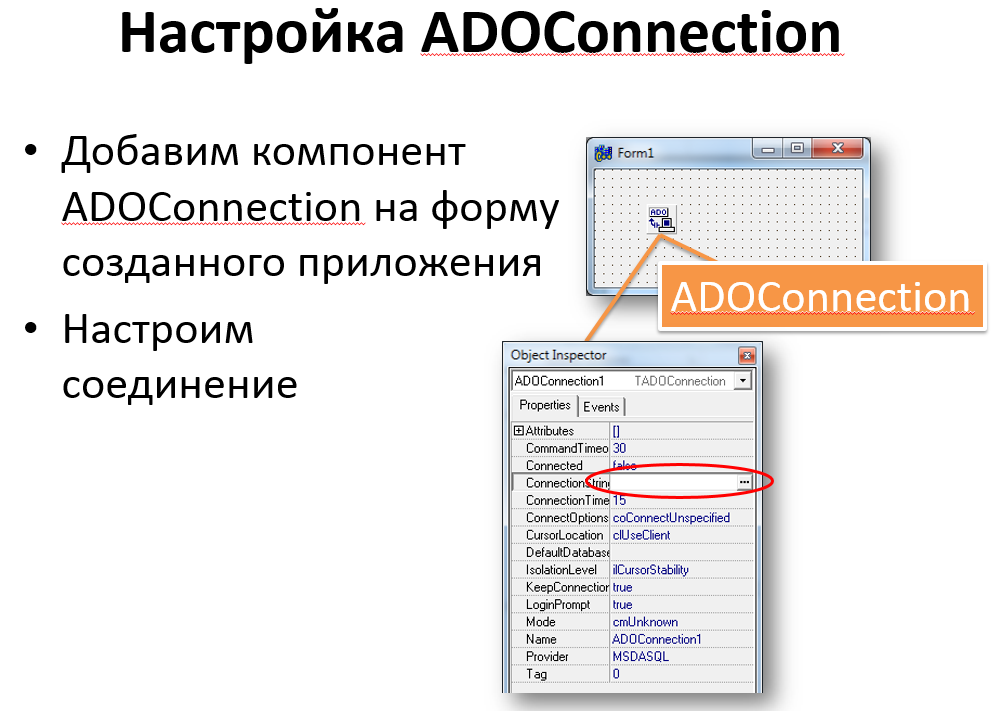


Рис. 2 – Настройка соединения с БД с использованием ADOConnection

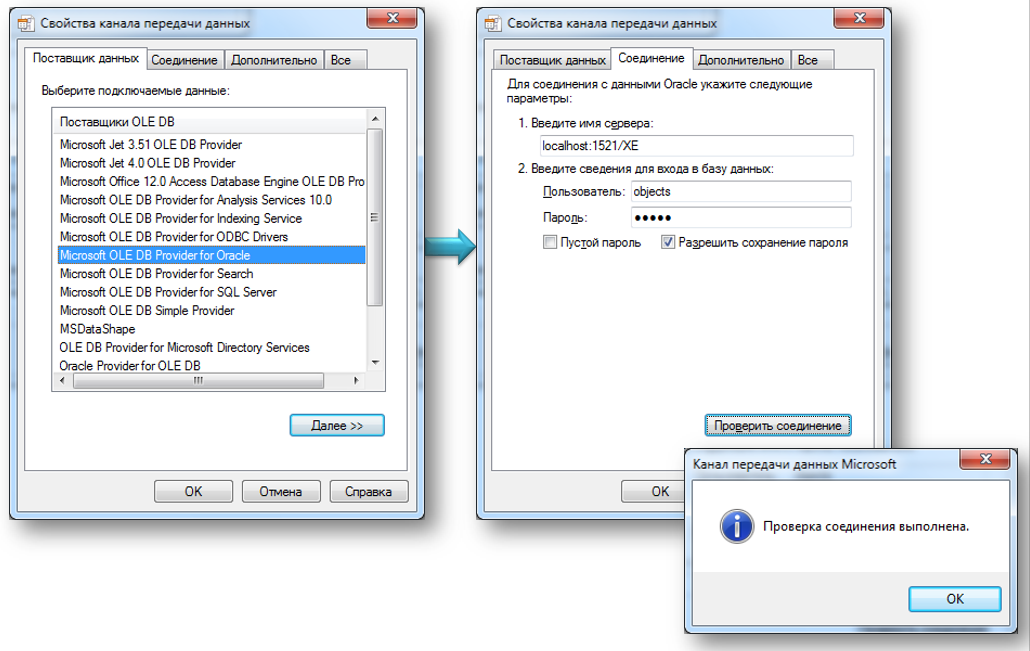


Рис. 3 – Выбор поставщика данных, проверка соединения

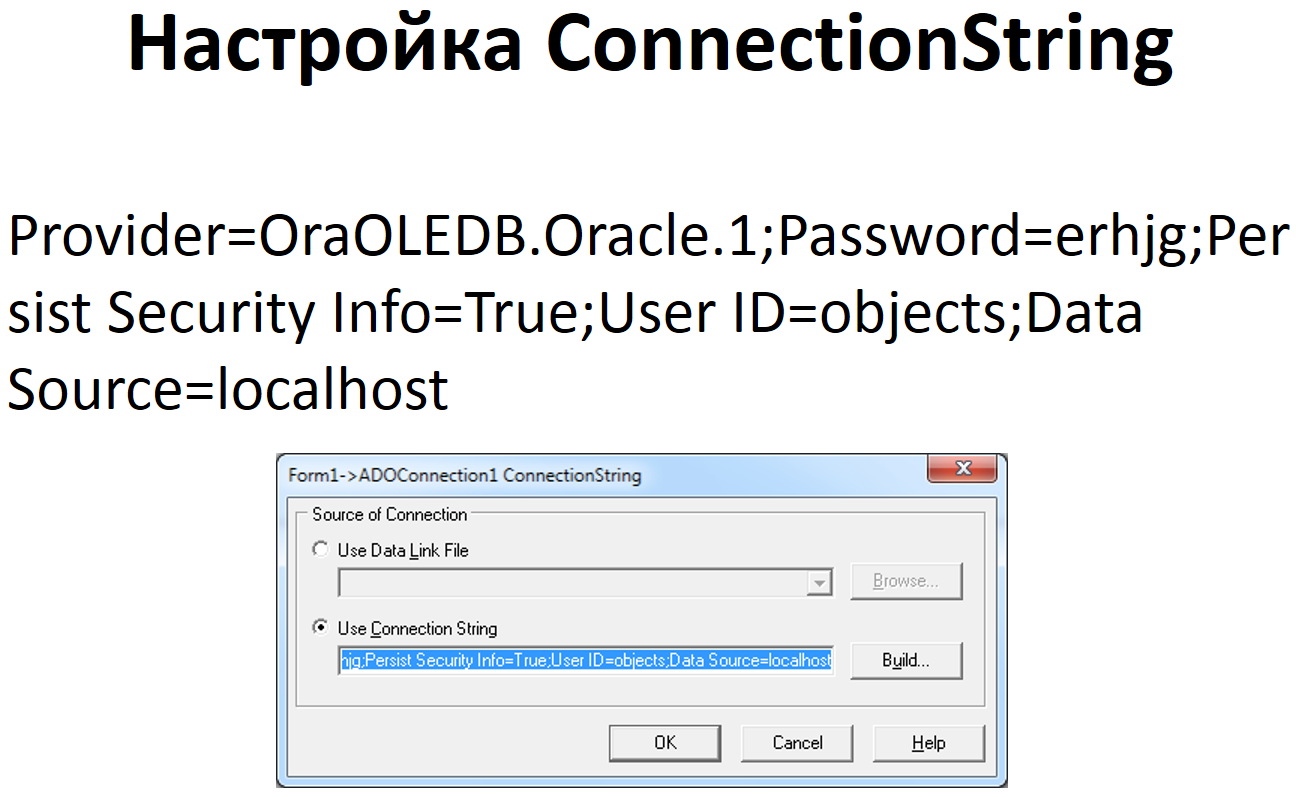


Рис. 4 – Строка соединения с БД ConnectionString

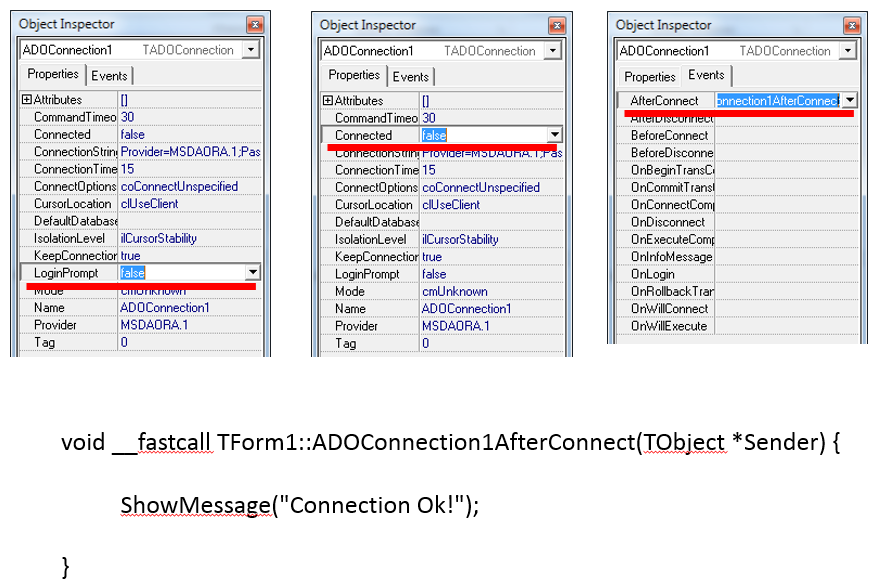


Рис. 5 – Настройка подключения и обработка события успешного подключения к БД

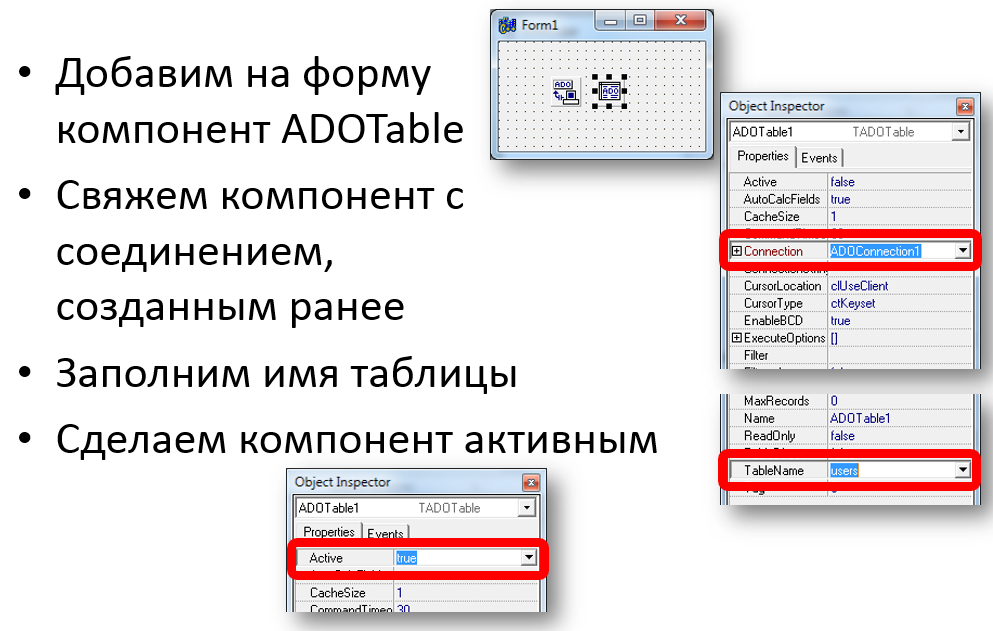


Рис. 6 – Добавление компонента ADOTable и его настройка

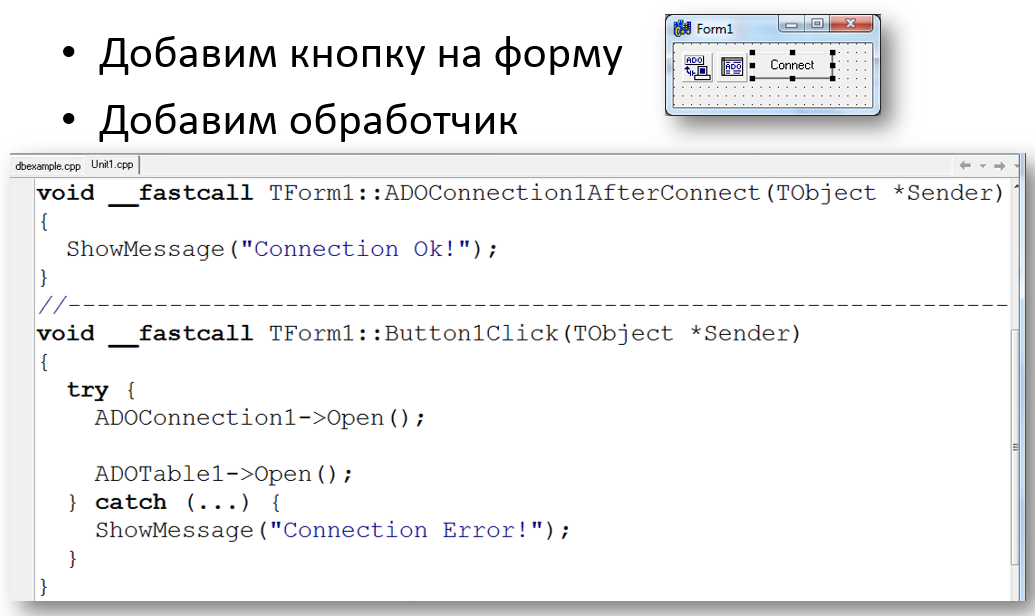


Рис. 7 – Программное подключение к БД

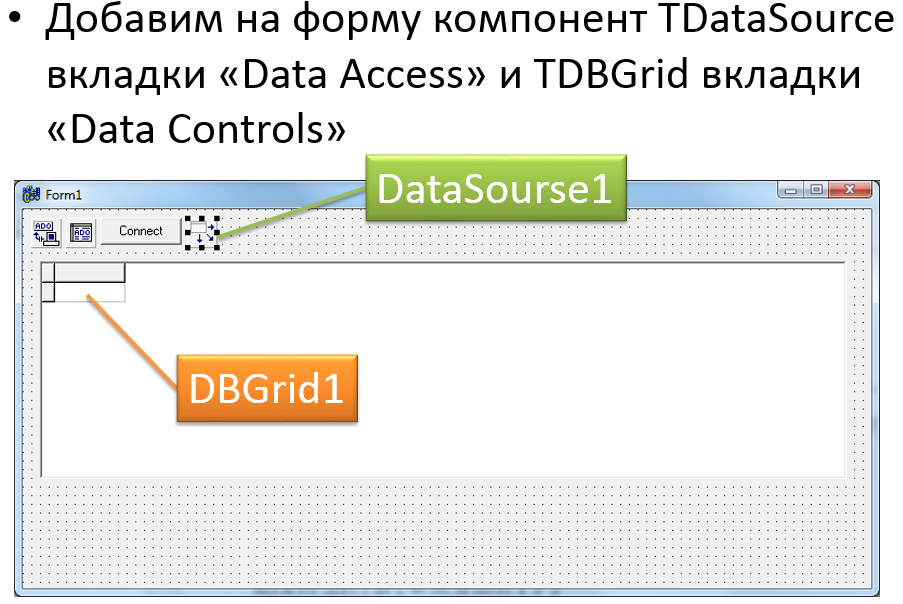


Рис. 8 – Визуальное представление таблицы

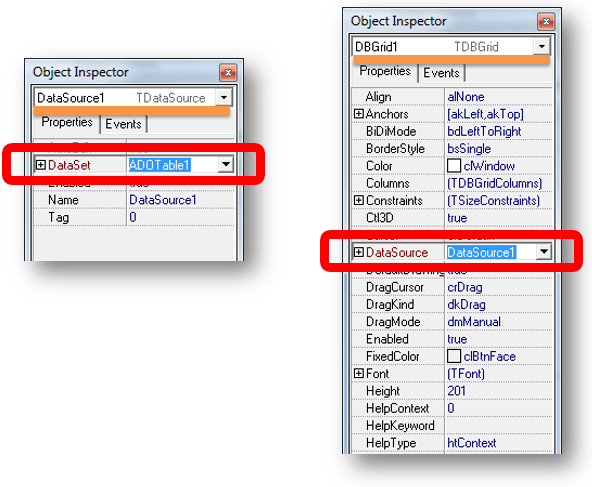


Рис. 9 – Настройка добавленных компонентов

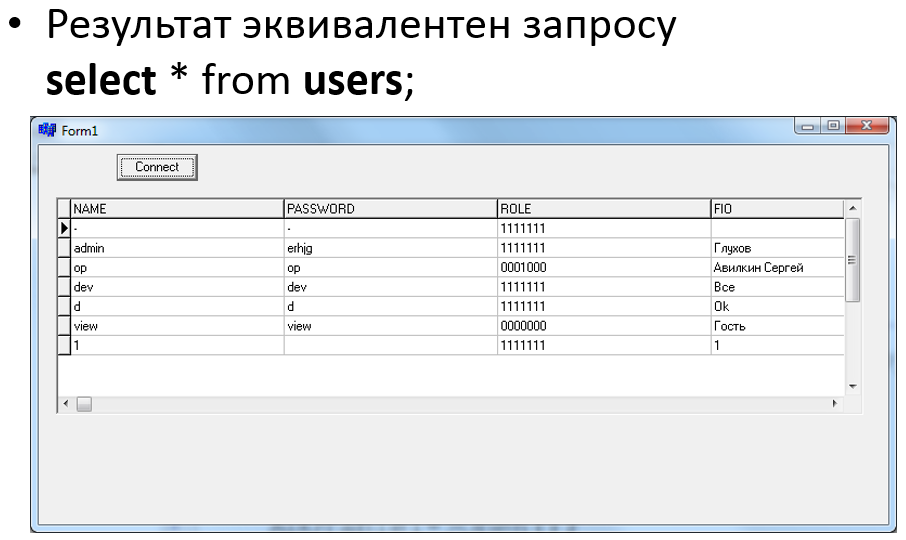


Рис. 10 – Результат отображения

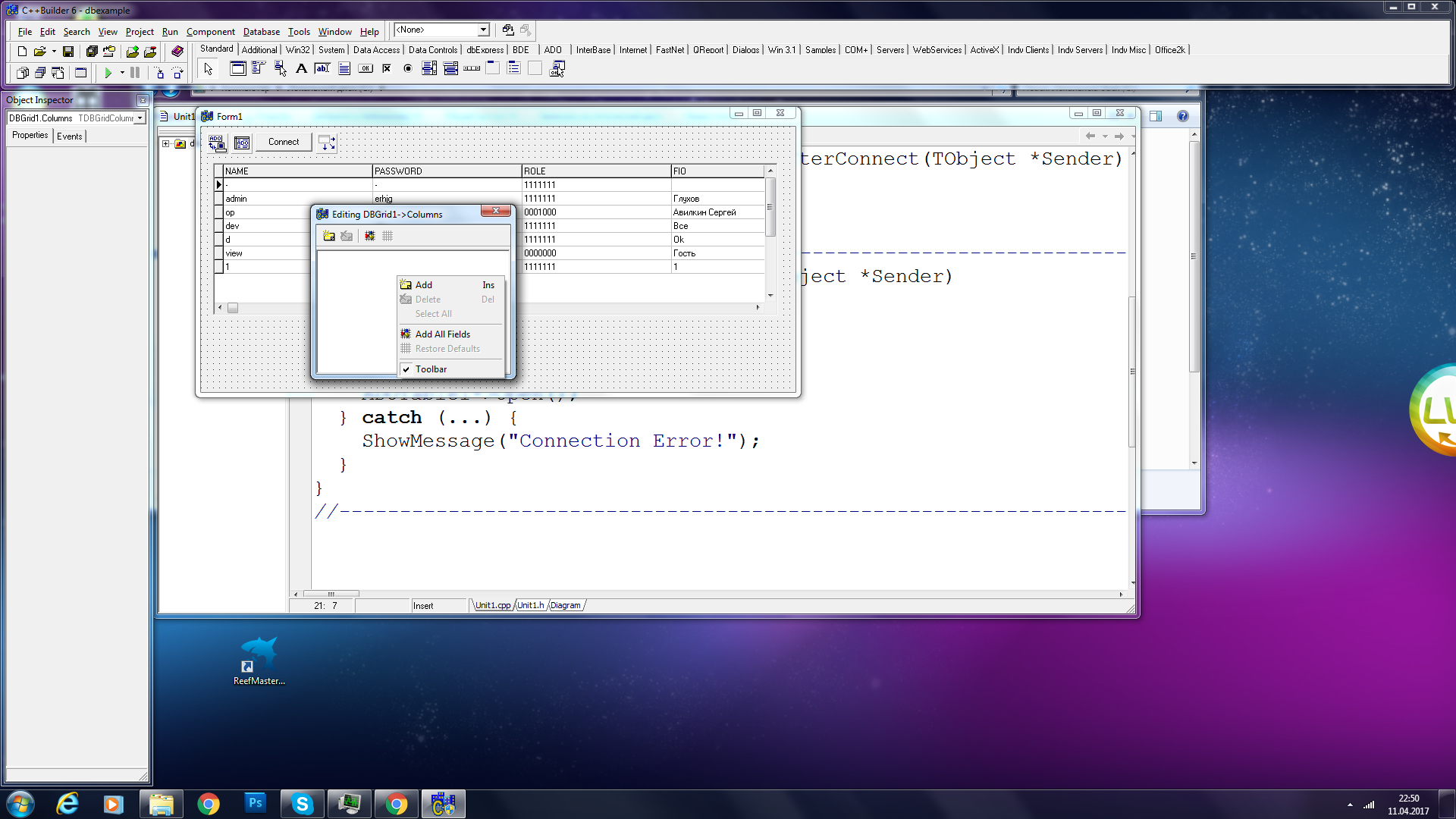


Рис. 11 – Настройка заголовка

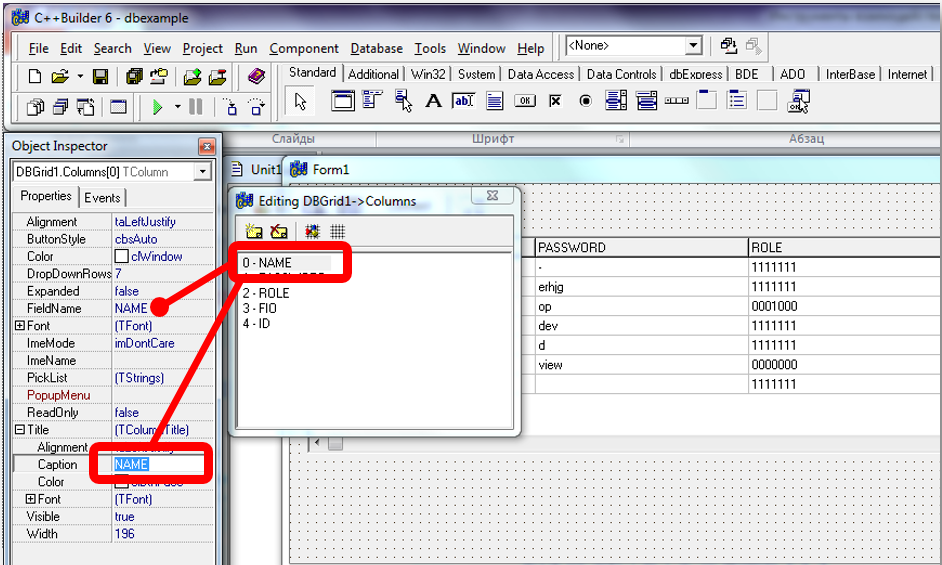


Рис. 12 – Параметры столбцов



Рис. 13 – Стандартный навигатор

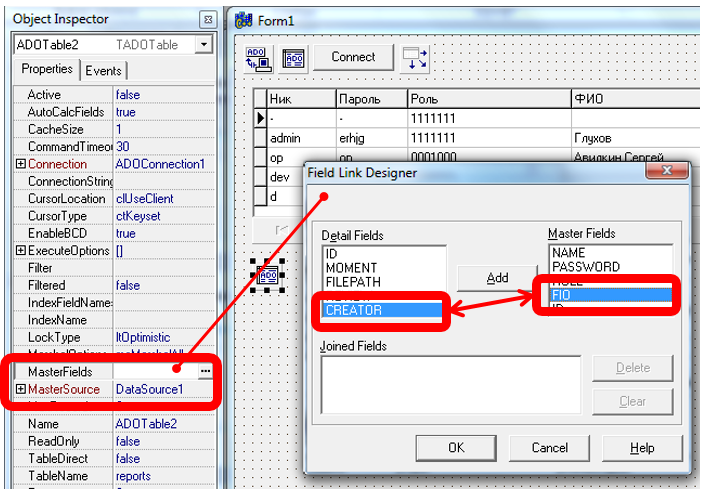


Рис. 14 – Связанные таблицы



Рис. 15 – Результат связывания

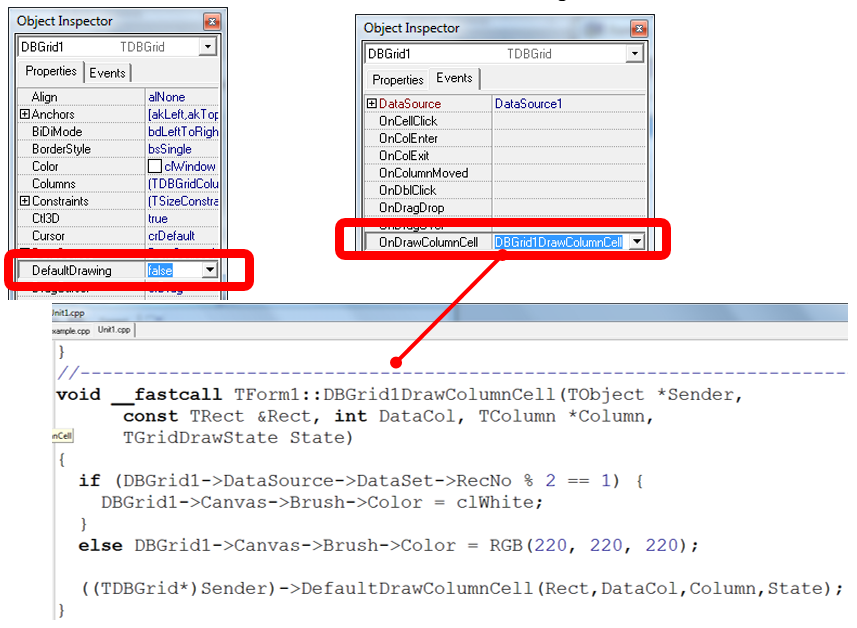


Рис. 16 – Пользовательская отрисовка полей

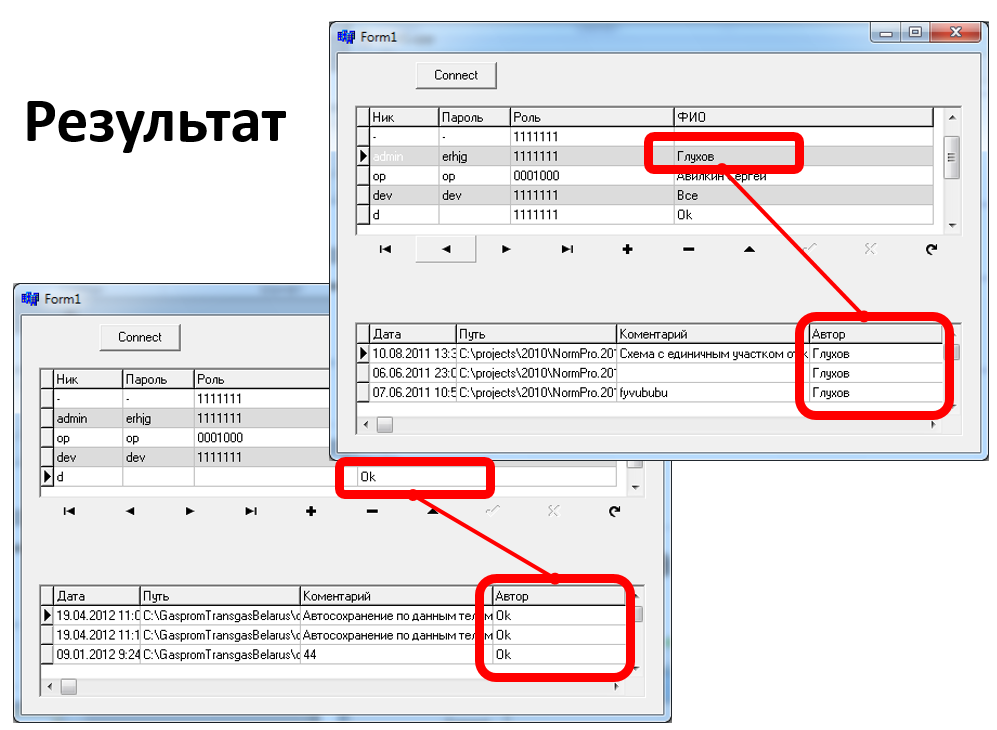
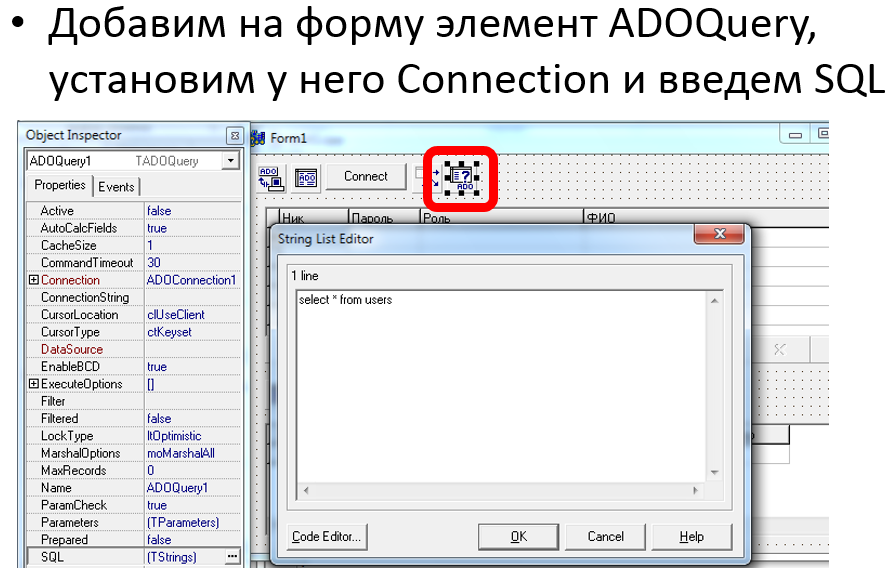
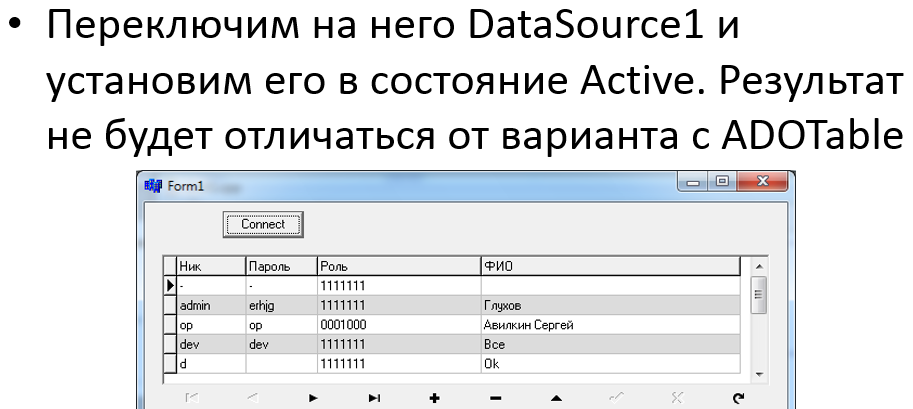


Рис. 17 – Результат отрисовки с чередованием цвета фона записи





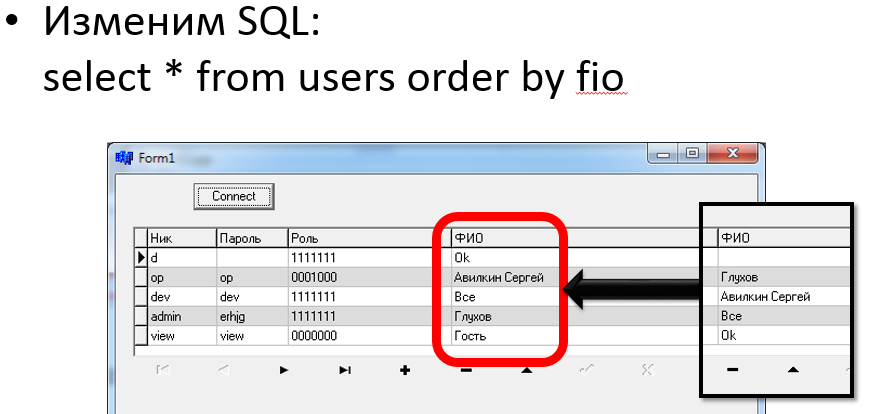


Рис. 18 – Запросы



Рис. 19 – Реализация фильтров

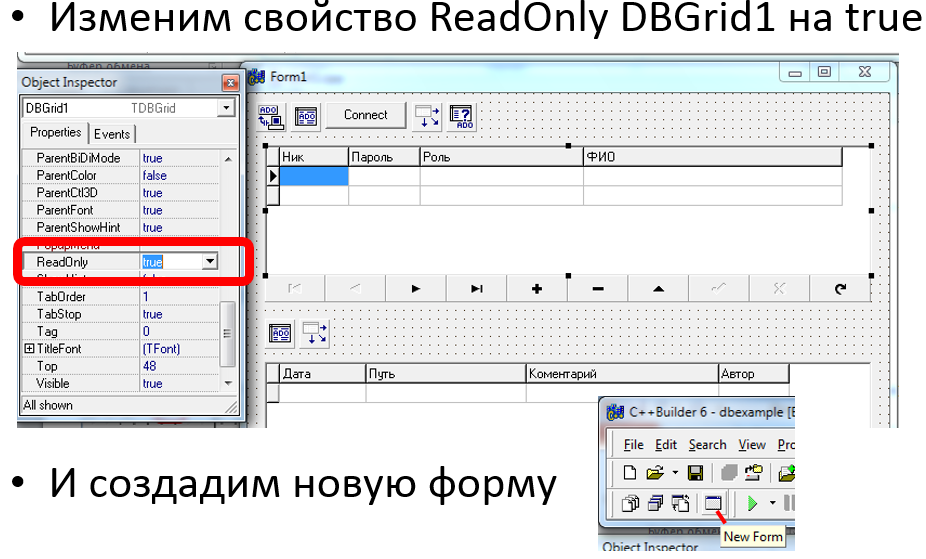
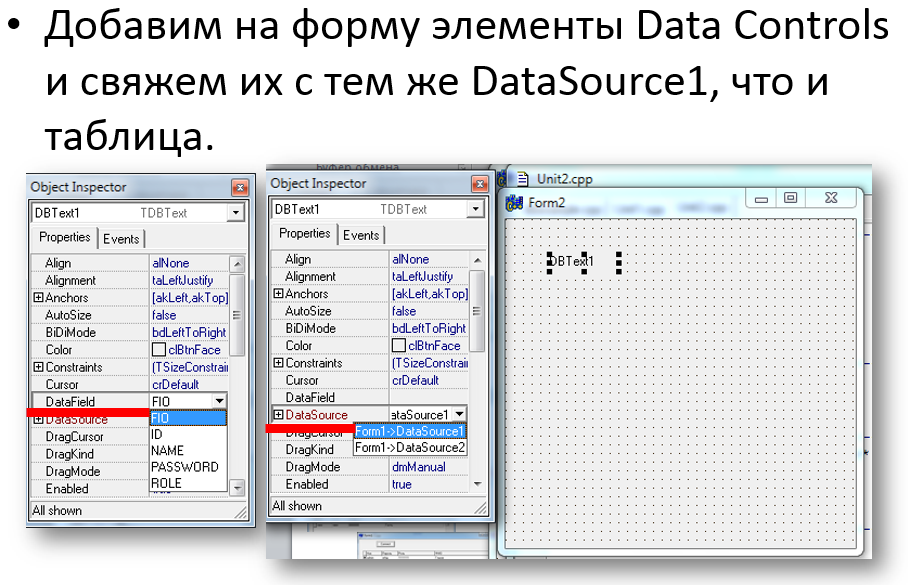


Рис. 20 – Создание форм



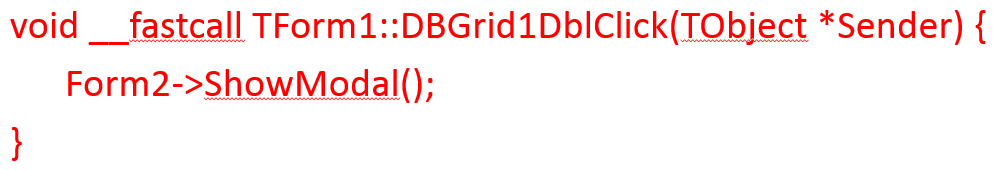


Рис. 21 Добавим элементы на форму и пропишем событие открытия   
формы по лвойному клику на таблице DBGrid1

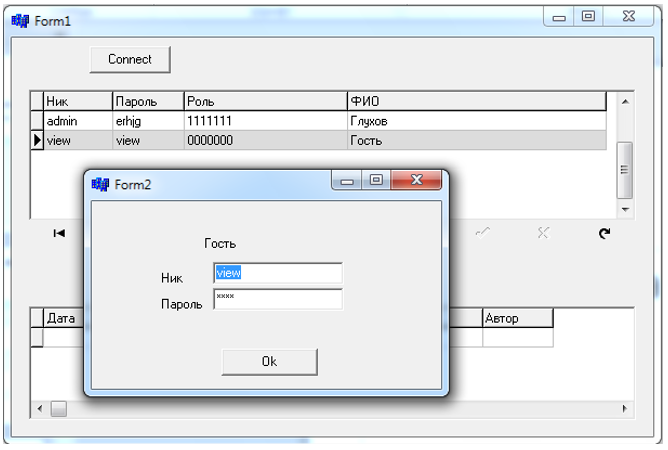


Рис. 22 – Результат работы с формой

**Задания:**

Задача состоит в разработке и программировании программного обеспечения для взаимодействия с сервером баз данных на котором размещается БД студента. По согласованию с преподавателем студент может выбрать средство разработки, или реализовать web-интерфейс доступа, выбрать технологический стек.

1. Создать приложение, настроить соединение с БД
2. Реализовать управление данными всех таблиц справочников и основных таблиц
3. Реализовать таблицу с результатом сложного запроса, выданного преподавателем

##### Порядок выполнения работы:

1. Запустить Borland C++ Builder, создать приложение.
2. Настроить соединение с БД и проверить соединение.
3. Построить интерфейс связанных таблиц.
4. Построить форму.
5. Протестировать основные функции приложения.

##### Контрольные вопросы.

1. Что такое ADO, ODBC, OLEDB, JDBC, BDE?
2. Структура ConnectionString.
3. Классы применяются для работы с БД на языке С++?
4. Какие компоненты необходимо использовать для работы с объектами БД?
5. Какие присутствуют возможности настройки компонентов?

## Литература

1. Рудикова, Л.В. Базы данных: разработка приложений. – СПб.: БХВ-Петербург, 2006. – 487 с.: ил. – См. также эл. копию. – ISBN 5-94157-805-9: 16124-00.
2. Хомоненко, А.Д. Базы данных: учебник для вузов / под ред. А.Д. Хомоненко. – СПб.: КОРОНА принт, 2004. – 736 с. – Рек. учеб.-метод. объединением по образованию в обл. автоматики, электроники, микроэлектроники и радиотехники при обуч. по технич. и экон. спец. – ISBN 5-7931-0284-1: 28416-00.
3. Карпова, Т.С. Базы данных: Модели, разработка, реализация: Учебник. – СПб.: Питер, 2001. – 303с.: ил. – См. также эл. копию. – ISBN 5-272-00278-4 : 5268-00.
4. Oracle SQL Library / <https://docs.oracle.com/en/database/oracle/application-express/18.2/aemig/about-PLSQL-libraries.html#GUID-27B30E6A-C86A-4D45-8E7F-A315445AE091>
5. w3school SQL Tutorial / <https://www.w3schools.com/sql/>