Методические указания к выполнению курсовой работы

Части 1 и 2

Предметно-ориентированные информационные системы

Н.Н. Иванова
01.03.2010

Оглавление

Введение

В основе всех информационных систем и систем управления предприятием лежат базы данных или хранилища знаний. Поэтому так важно уметь проанализировать процессы, происходящие на предприятии, выделить предметную область, разделить ее на соответствующие фрагменты, смоделировать процессы и спроектировать соответствующую базу данных, на которую в дальнейшем ляжет весь функционал управления производством.

Рассмотрим упрощенный процесс моделирования и разработки информационной системы на примере элементарной ИС «Разработка информационной автоматизированной системы для учета абонементов в районной библиотеке».

Часть первая.

1. Описание предприятия
2. Описание предметной области
3. Выделение фрагментов (объектов)
4. Создание таблиц
5. Создание связей
6. Заполнение таблиц данными

Описание предприятия.

Муниципальное учреждение культуры Районная Библиотека.

Учредитель — Отдел культуры Администрации района.

Работает на основе Устава, принятого 01 января 2008 года. ИНН № 1111111111.

В состав библиотеки входят 2 отдела:

- Отдел обслуживания населения
- Отдел комплектования и обработки

В отделе обслуживания населения работает два подразделения: читальный зал и абонемент. Фонд отдела — около 50 тысяч экземпляров. К услугам посетителей в читальном зале имеется разнообразная периодическая литература.

Отдел комплектования и обработки является структурным подразделением районной библиотеки. Основная функция отдела – комплектование библиотечного фонда, ведение учета новых поступлений и их распределение по отделам библиотеки, ведение учета пришедших в негодность и требующих списания изданий. Сотрудники отдела обрабатывают поступления, составляют библиографическое описание в соответствии с требуемыми государственными стандартами, проводят подписку на периодические издания, исключают из учетных документов отобранные к списанию издания.

Контактная информация.

Юридический и фактический адрес:
222222
Энская область
Энский район
г. Энск
ул. Калинина дом 1.

Телефон:
(12345) 6-78-90

Цели и задачи.

В данной библиотеке необходимо автоматизировать работу по абонементам.

Определение предметной области и деление ее на фрагменты.

Сначала надо определиться с предметной областью. Итак, что собой представляет абонемент? Человек приходит в библиотеку, его там регистрируют, выдают читательский билет, после чего выдают интересующие книги, которые он приносит обратно через некоторое время. Вот кажется и все. НО! Нам необходимо построить такую базу данных, в которой будет храниться вся история работы, и с помощью которой будут строиться различные аналитические отчеты. Для начала выделяем большие явные объекты, встречающиеся в описанном процессе.

Это **читатель**, **сотрудник**, **книга**. У каждого объекта есть свой набор атрибутов или характеристик. В случае **читателя** это будет его фамилия, имя, отчество, паспортные данные, номер читательского билета, пол, дата регистрации в библиотеке. Обращаем внимание на то, что сочетание полей фамилия, имя, отчество и паспортные данные будут уникальным сочетанием.

Таким образом, мы не позволяем появляться избыточным данным в данной таблице. Оговорим, что в данном примере мы не рассматриваем ситуацию, когда человек закрывал свой читательский билет, а потом регистрировался снова. В качестве самостоятельного задания подумайте, как можно реализовать такую ситуацию.

У **книги** атрибутами являются название, автор, издательство, год издания, количество страниц. Уникальным сочетанием здесь являются название и автор.

Сотрудник характеризуется фамилией, именем, отчеством, паспортными данными, полом, датой приема на работу, датой увольнения. Уникальность – фамилия, имя, отчество и паспортные данные.

В объекте **книга** не все так просто, как это может показаться на первый взгляд. Например, книга Красная Шапочка издавалась в 1975, 1989 и 1991 годах в разных издательствах и выходила разным количеством страниц. Если всю эту информацию хранить в одном объекте **книга**, то в будущем мы получим таблицу, не состоящую в 3НФ. Поэтому, если есть одна сущность и у нее бывает несколько воплощений, то храниться они будут в разных объектах. Следовательно, в результате размышлений мы приходим к выводу, что нам необходим дополнительный объект, которому дадим название **характеристики книги.**

Теперь необходимо описать связи между данными объектами. Когда читатель берет и отдает книгу, мы эту информацию должны куда-то записать, так как учет литературы является одной из основных функций библиотеки. Для данной операции создаем еще один объект, который так и будет называться – **библиотека**. В нем будет храниться информация о читателе, который взял книгу, о сотруднике, выдавшем ее, о самой книге, а также дата получения на руки и дата сдачи. В примечании можно указать состояние книги до и после сдачи. Уникальным сочетанием в данном случае будут номер читателя, номер книги и дата получения на руки. Таким ограничением мы исключаем ситуацию, когда в один день читателю выдают одну и ту же книгу несколько раз.

Связи между объектами осуществляются с помощью первичных (Primary Key, PK) и внешних (Foreign Key, FK) ключей. Уникальность обозначается буквой U.

Под первичным ключом мы будем понимать уникальный номер записи в таблице, а под внешним – ссылку на первичный ключ в другой таблице.

Итак, у нас получились следующие объекты с атрибутами:

- **Читатель**
  - Уникальный номер читателя (НЕ номер читательского билета!), PK
  - Фамилия, U
  - Имя, U
  - Отчество, U
  - Паспортные данные, U
  - Номер читательского билета
  - Пол
  - Дата регистрации в библиотеке
- **Сотрудник**
  - Уникальный номер сотрудника, PK
  - Фамилия, U
  - Имя, U
  - Отчество, U
  - Паспортные данные, U
  - Пол
  - Дата приема на работу
  - Дата увольнения
- **Книга**
  - Уникальный номер книги, PK
  - Название, U
  - Автор, U
- **Характеристики книги**
  - Уникальный номер характеристики книги, PK
  - Номер книги, чья характеристика, FK
  - Издательство
  - Год издания
  - Количество страниц
- **Библиотека**
  - Уникальный номер записи в этой таблице, PK
  - Номер читателя, FK1, U
  - Номер характеристики книги, FK2, U
  - Номер сотрудника, FK3

- Дата получения книги, U
- Дата сдачи книги
- Примечание

Создание и оформление схемы базы данных.

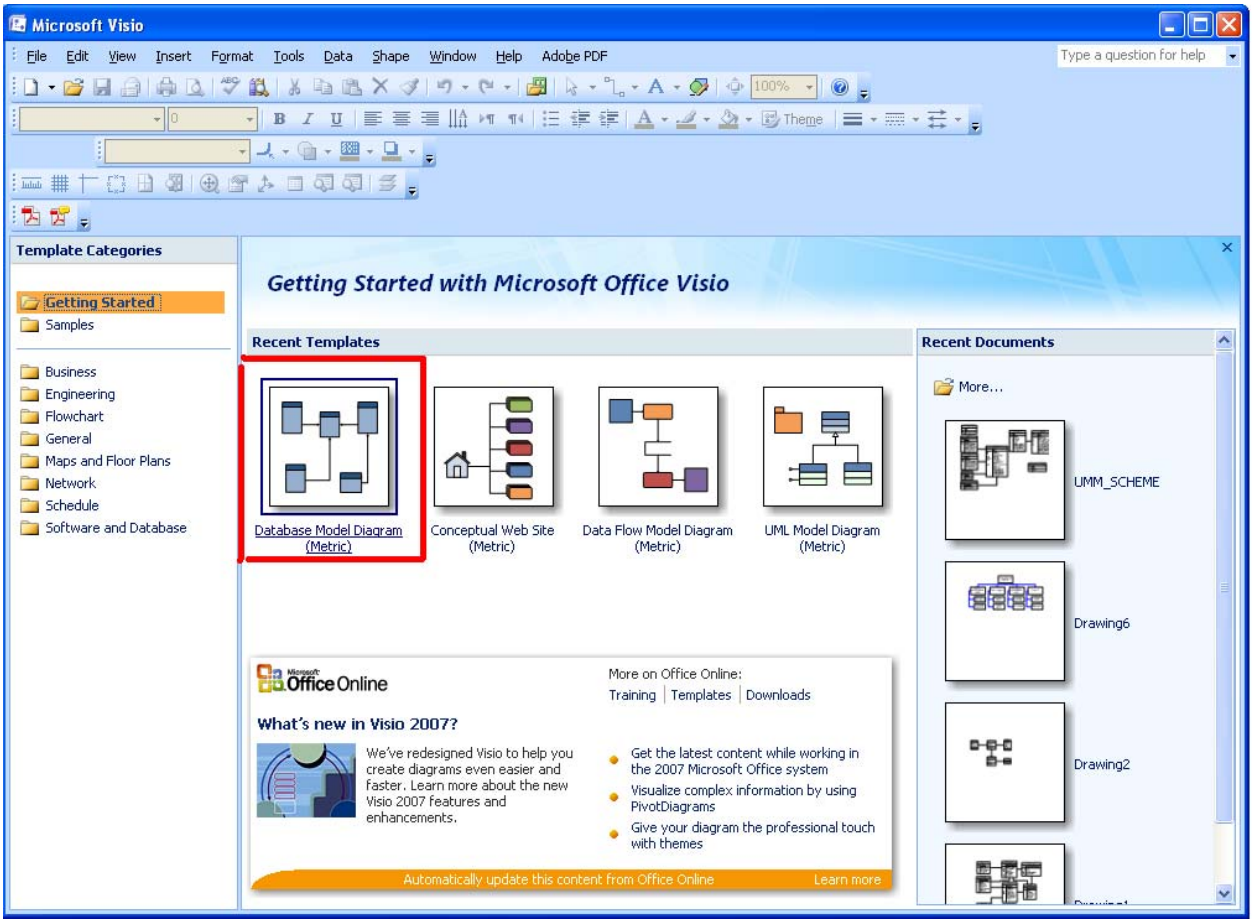
Основные правила названий объектов:

Названия объектов должны быть написаны только латиницей, без знаков препинания и пробелов, в качестве разделителя допустимо использовать только символ подчеркивания “_”, в названиях недопустимо использовать служебные слова.

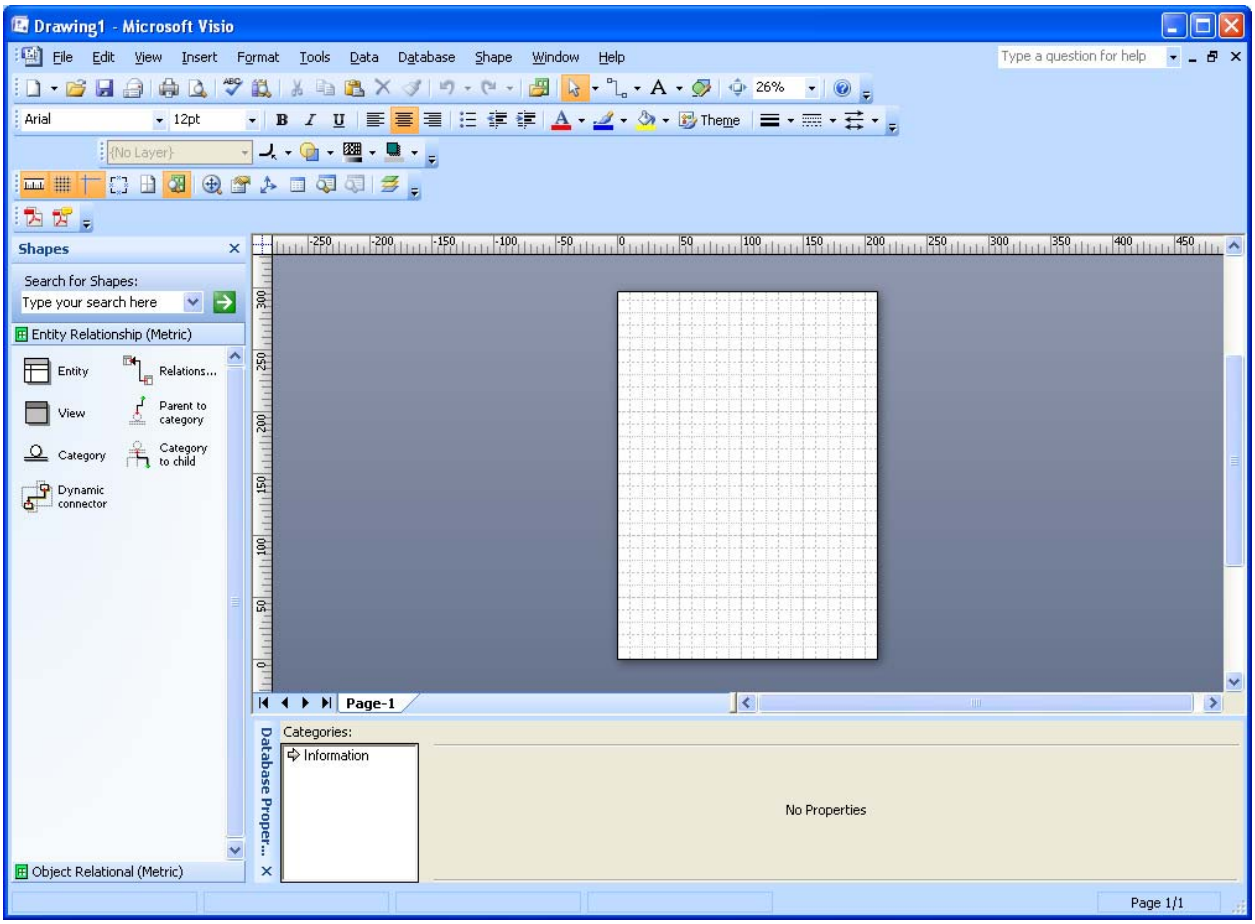
Каждый объект должен иметь свое уникальное название.

Поэтому, переименуем наши объекты в **Reader**, **Sotrudnik**, **Book**, **Char_Book**, **Library** соответственно.

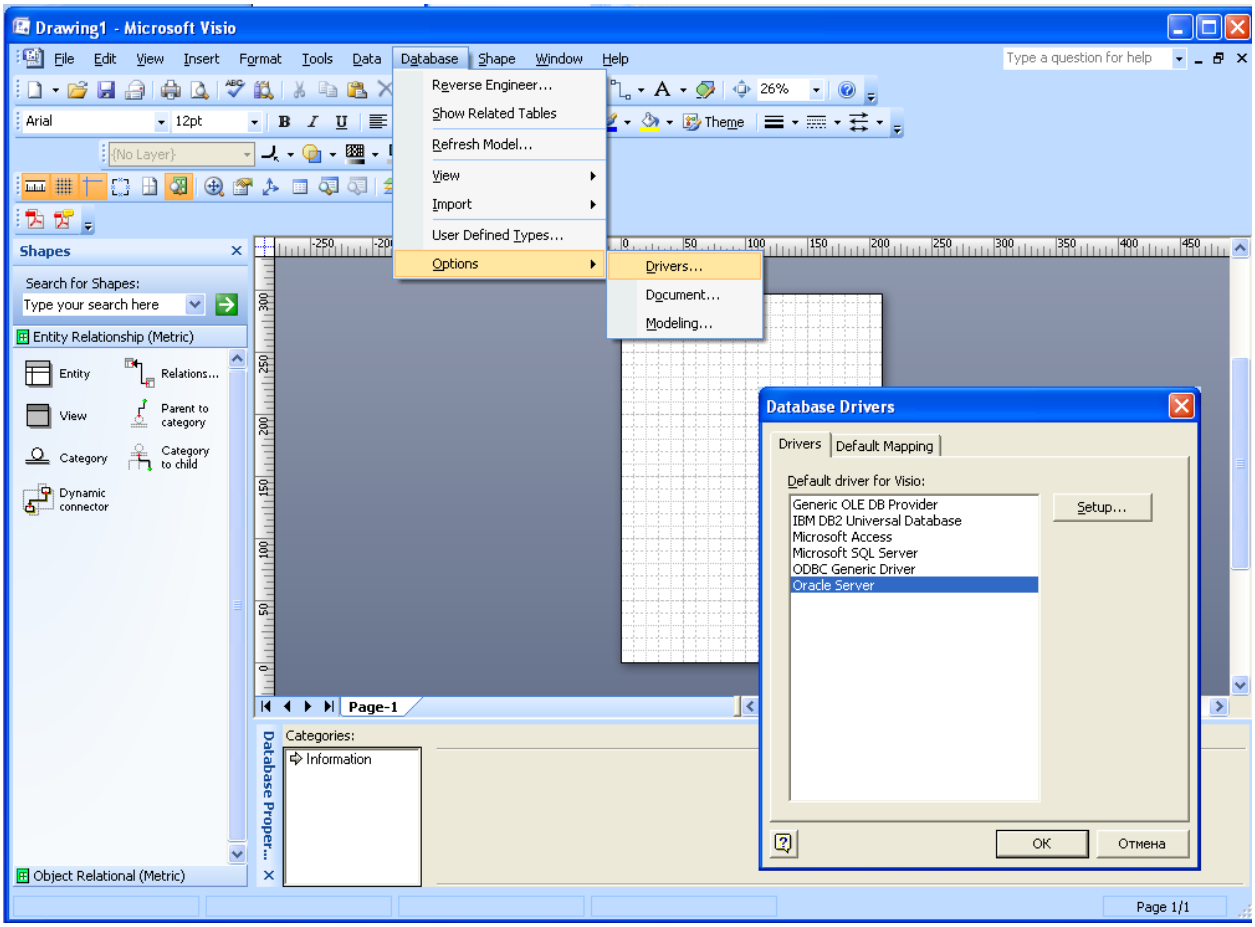
Для создания схемы базы данных необходимо запустить MS Visio, и выбрать пункт Database Model Diagram.



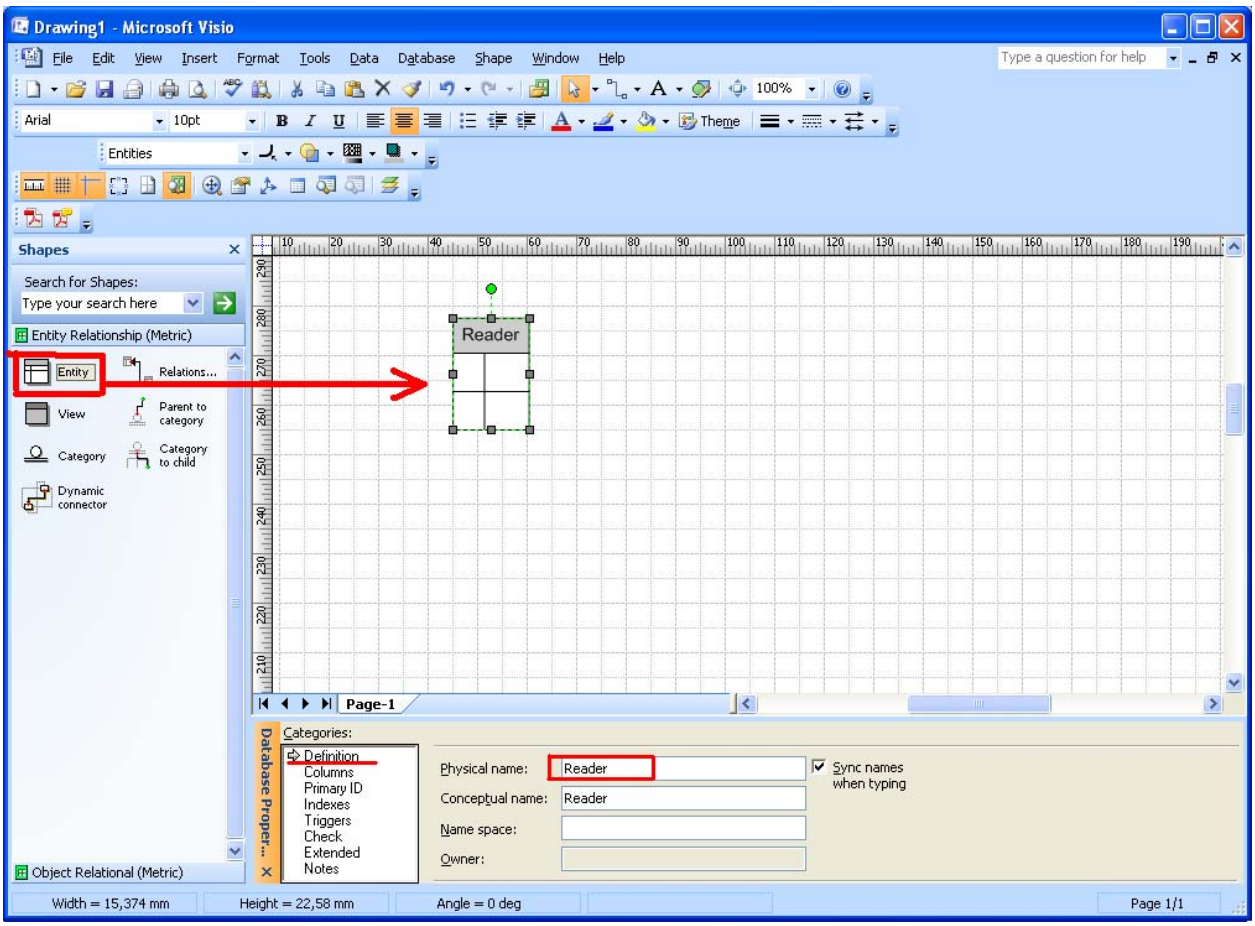
После выбора данного пункта получается такое окошко:



В центре располагается рабочая область, слева – панель объектов, снизу – параметры выбранных объектов. Перед началом работы в меню необходимо подключить характеристики нашей СУБД, а именно ORACLE Server (см. рисунок).



Для создания объекта **таблица** с панели инструментов надо перетащить таблицу (Entity), и в нижнем окне начать ее заполнение. Начинаем работать с таблицей **Reader**. В первой закладке объявляется имя таблицы.



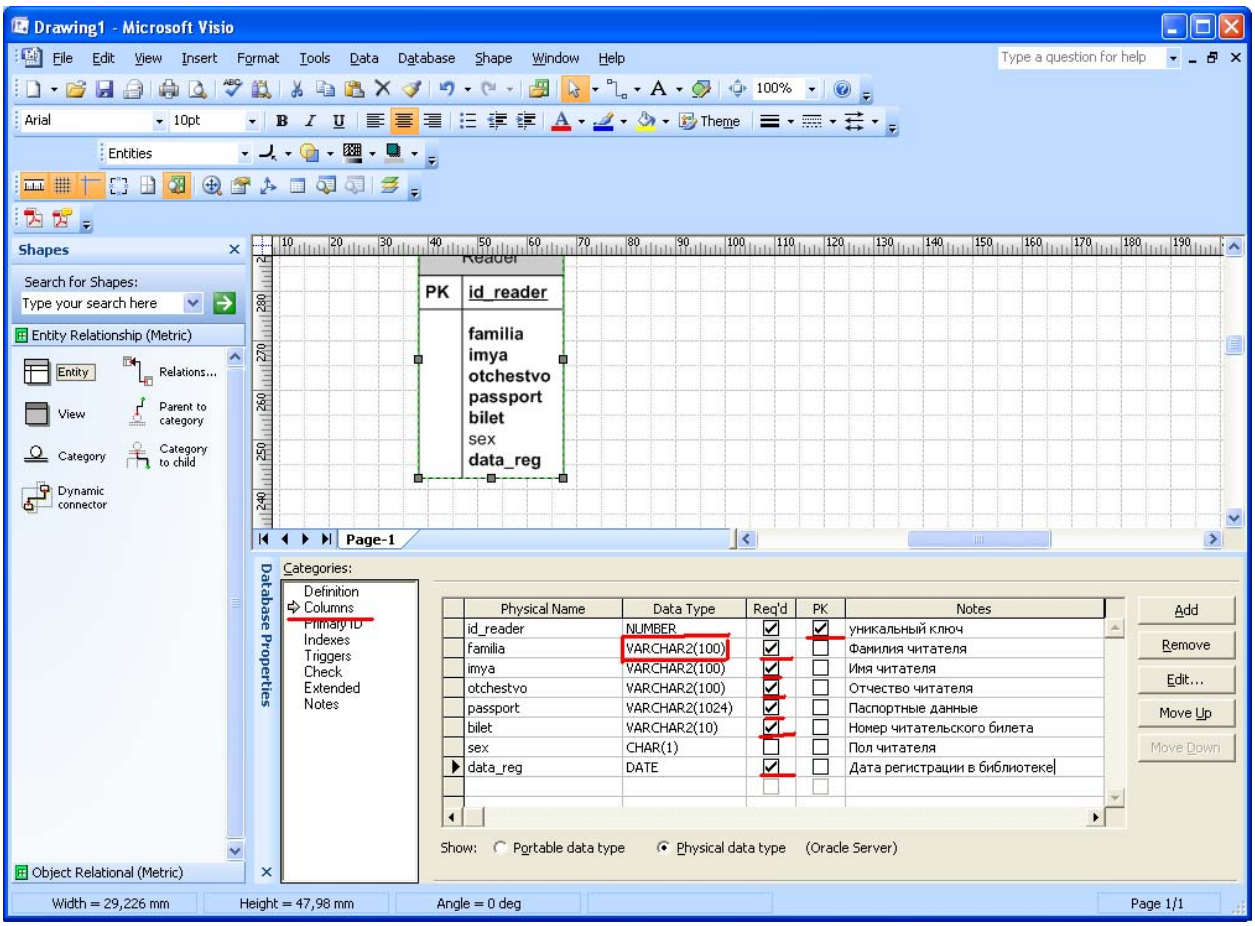
Во второй закладке заполняются названия полей (синонимами являются такие понятия, как атрибуты или столбцы).

Из курса баз данных вспоминаем, что у каждого поля должен быть свой тип данных.

В нашей работе мы будем использовать следующие типы:

- NUMBER для числовых данных.
- DATE для дат.
- VARCHAR2(n) для строковых значений. Здесь n – максимальная длина строки, которая будет храниться в данном поле.
- CHAR(n) для символьных значений.

В столбце с первичным ключом и в уникальных столбцах ставятся соответствующие галочки (см. рисунок), а в последнем столбце – русскоязычные комментарии.



В таблицы заносим все поля, кроме тех, что являются внешними ключами. Кроме того, в третьем столбце ставим галочки в том случае, если в них всегда есть значения, то есть эти столбцы никогда не бывают пустыми.

Аналогичную работу проделываем с остальными таблицами и получаем в общей сложности 5 таблиц со следующими описаниями:

Sotrudnik

Physical Name	Data Type	Req'd	PK	Notes
id_sotr	NUMBER	☑	☑	уникальный номер
familia	VARCHAR2(100)	☑	☐	Фамилия сотрудника
imya	VARCHAR2(100)	☑	☐	Имя сотрудника
otchestvo	VARCHAR2(100)	☑	☐	Отчество сотрудника
passport	VARCHAR2(1024)	☑	☐	Паспортные данные
sex	CHAR(1)	☐	☐	пол
date_hire	DATE	☑	☐	Дата приема на работу
date_uvol	DATE	☐	☐	Дата увольнения
		☐	☐	

Book

Physical Name	Data Type	Req'd	PK	Notes
id_book	NUMBER	☑	☑	Уникальный номер книги
name	VARCHAR2(500)	☑	☐	Название книги
author	VARCHAR2(500)	☑	☐	Автор книги

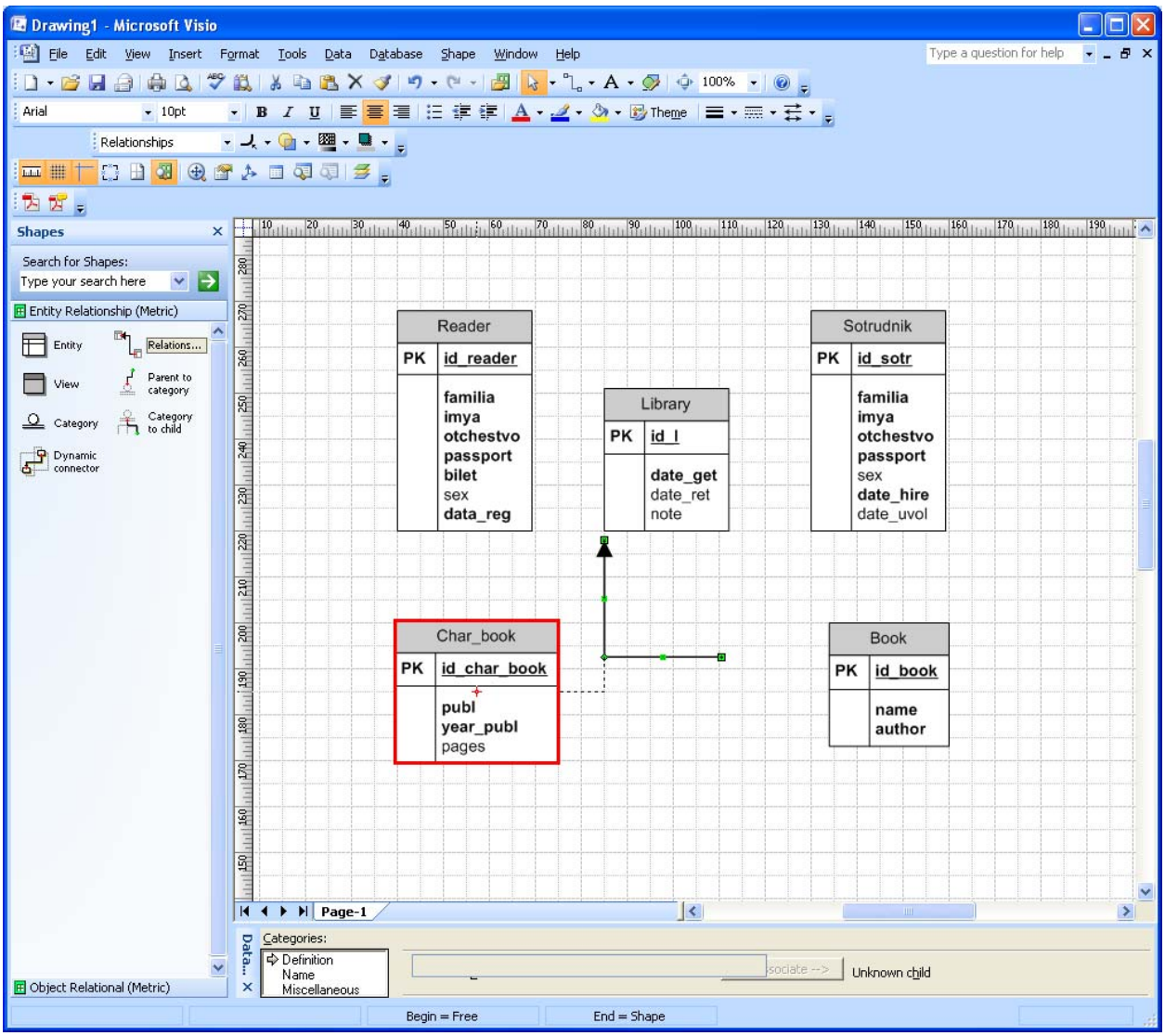
Char_book

	Physical Name	Data Type	Req'd	PK	Notes
▶	id_char_book	NUMBER	☑	☑	Уникальный номер характеристики
	publ	VARCHAR2(200)	☑	☐	Издательство
	year_publ	NUMBER	☑	☐	Год издания
	pages	NUMBER	☐	☐	Количество страниц

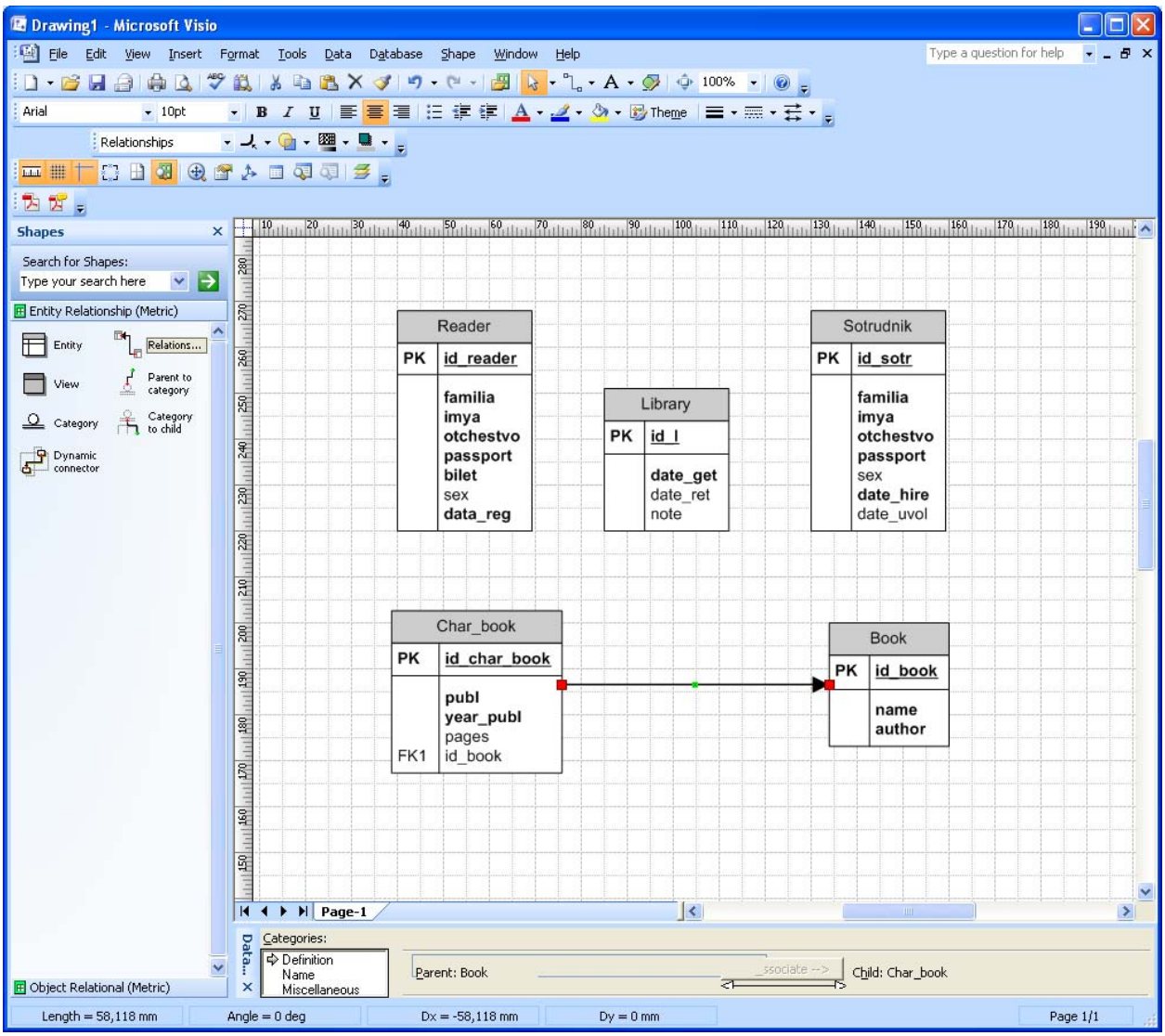
Library

	Physical Name	Data Type	Req'd	PK	Notes
▶	id_l	NUMBER	☑	☑	id_l identifies Library
	date_get	DATE	☑	☐	date_get is of Library
	date_ret	DATE	☐	☐	date_ret is of Library
	note	VARCHAR2(500)	☐	☐	note is of Library

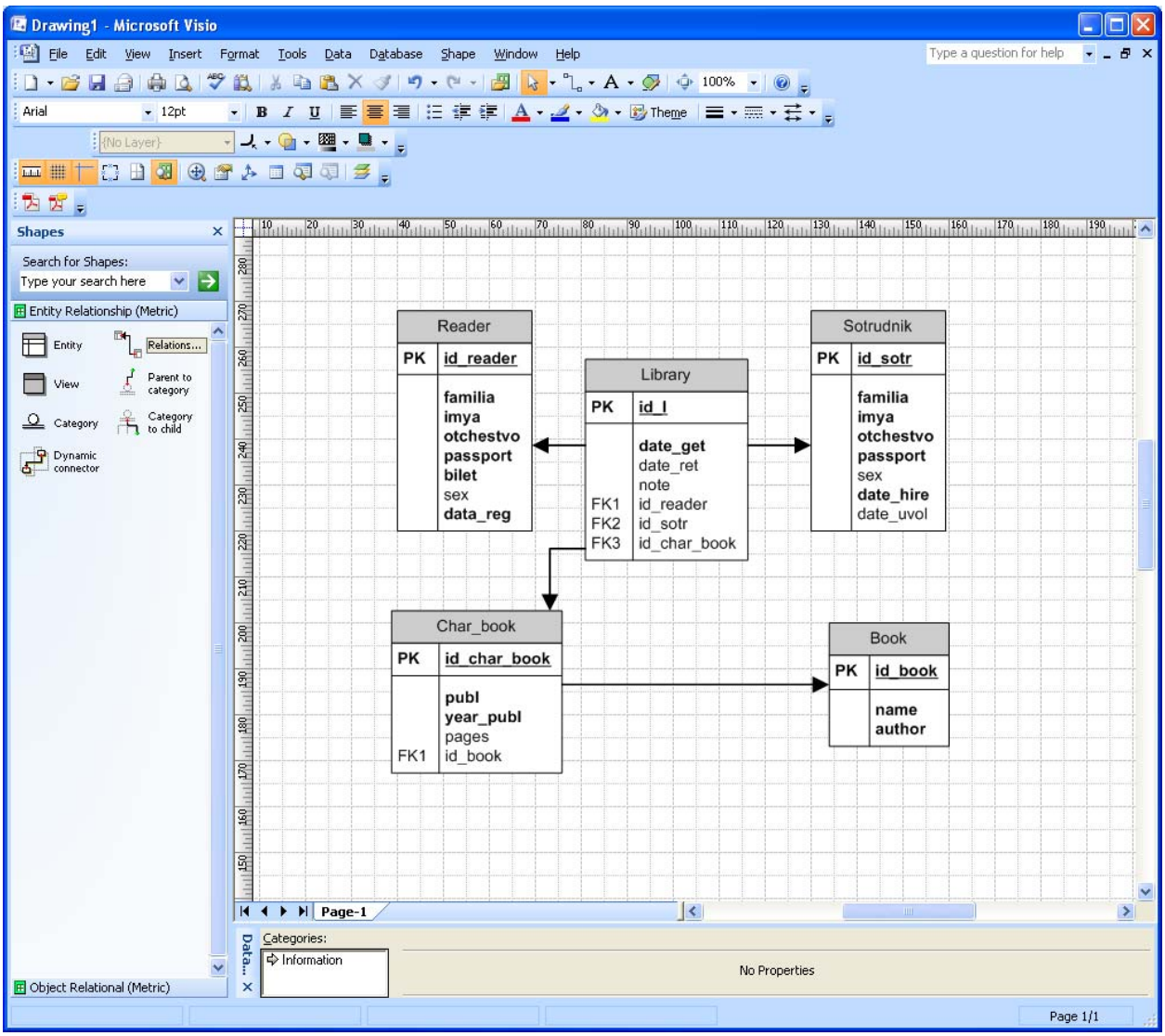
Осталось соединить таблицы связями. Для этого перетаскиваем с панели объектов стрелочку (Relationship) на рабочую область. Для начала соединим таблицы **Book** и **Char_Book**. Начало стрелки цепляем мышью и тащим к центру таблицы **Char_Book** до тех пор, пока границы таблицы не окрасятся в красный цвет.



Отпускаем стрелку. Теперь берем за конец стрелки и аналогичным образом направляем ее к таблице **Book.**

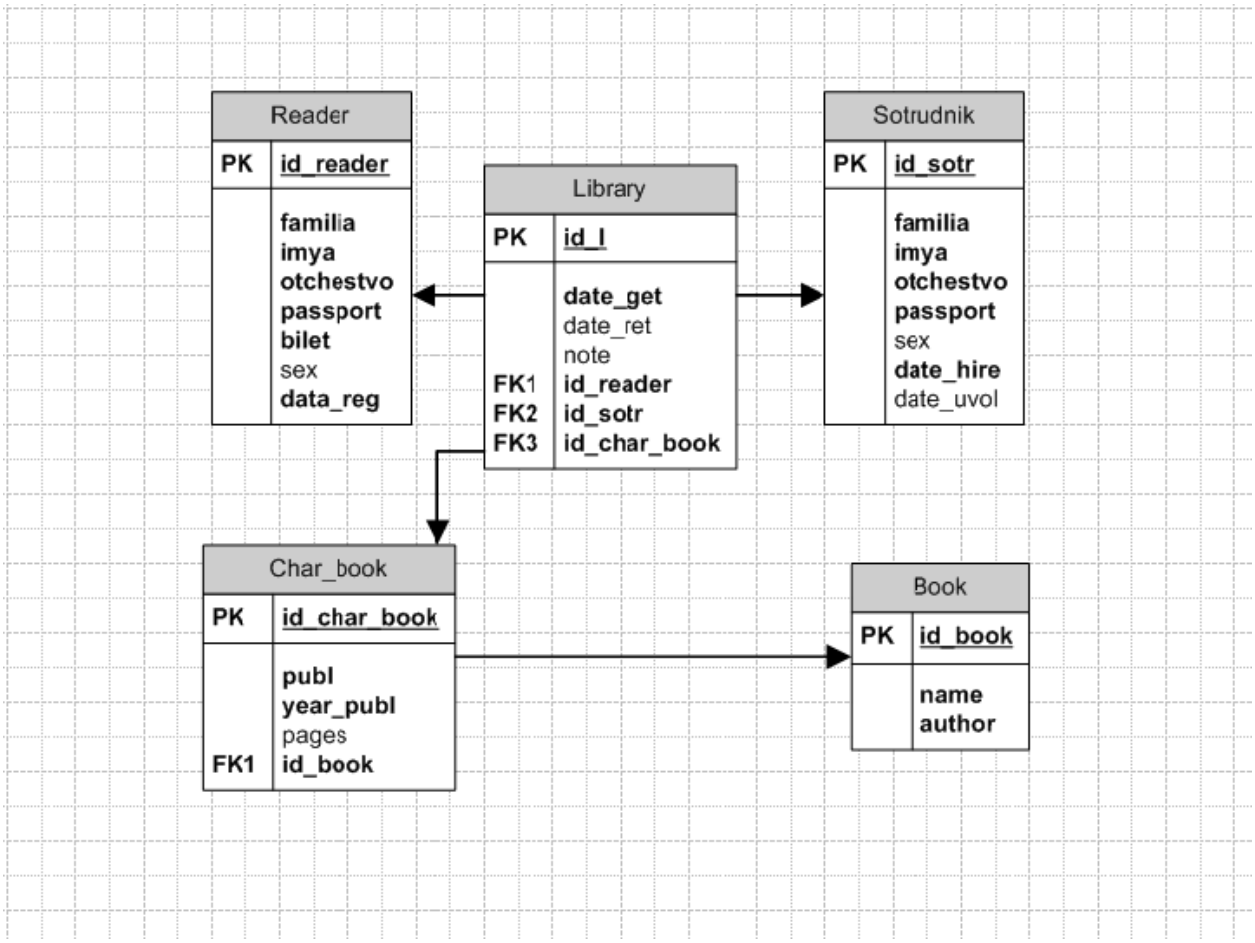


Если Вы сделали все правильно, то внешние ключи появятся автоматически. Аналогично свяжем таблицу Library с Reader, Char_book и Sotrudnik:



В данной схеме надо сделать несколько окончательных изменений, а именно обозначить внешние ключи ненулевыми, так как в данном случае они все должны быть не пустыми при заполнении.

В результате получаем готовую схему базы данных.



Задания для самостоятельного решения

- Добавить возможность учета жанров литературы (одна книга может принадлежать к нескольким видам жанра)
- Расширить существующую схему следующими информационными надстройками:
  1) Реализовать штатное расписание сотрудников библиотеки
  2) Автоматизировать работу Отдела комплектования и обработки

Внимание!!!

Курсовая работа должна состоять из не менее, чем из 8 таблиц!

Если брать рассмотренный пример в рамках курсовой работы, то схема на оценку должна состоять из всех 3-х частей (абонемент, штатное расписание и комплектование).

Физическое создание спроектированных объектов

Данный процесс осуществляется непосредственно в редакторе SQL.

Мы рассмотрим создание таблиц и сопутствующих объектов с помощью визуального редактора TOra. TOra - расшифровывается как "Toolkit For Oracle" и представляет из себя IDE для таких СУБД как Oracle, MySQL, PostgreSQL. На данный момент является одной из трех полноценных Oracle IDE под Linux (остальные две - это "SQL Developer" от Oracle и "Aqua Data Studio" от AquaFold). Концептуальное отличие - язык программирования исходного кода: "SQL Developer", "Aqua Data Studio" - Java, "TOra" - C++.

На следующем рисунке показан фрагмент интерфейса программы TOra. Белое поле (SQL Editor) – это рабочая область, где пишутся различные запросы.

Для того, чтобы открыть еще одну рабочую область, надо нажать на кнопочку . Для того, чтобы выполнить написанный скрипт, необходимо на панели инструментов нажать на , причем курсор должен находиться в середине того запроса, который выполняется.

Основы языка SQL

В данном разделе очень кратко описаны основные операции SQL, знания которых достаточно для написания первой части курсовой работы.

Курсивом выделены те атрибуты и объекты, названия которым студент дает самостоятельно.

Для того, чтобы создать таблицу, необходимо воспользоваться оператором CREATE, который имеет следующий синтаксис:

create table *my_table* (
my_column1 type1,
my_column2 type2,
…
);

Каждый оператор языка SQL заканчивается символом «;».

Внешние ключи добавляются к уже созданным таблицам с использованием оператора ALTER TABLE, который позволяет делать изменения в уже существующей структуре таблицы. Внешние ключи создаются как ограничение (Constraint) на значения столбца. Ограничения тоже являются объектами, их названия не повторяются и задаются самим пользователем (*FK_my_constraint1*).

Alter table *my_table* add(
Constraint *FK_my_constraint1* foreign key (*my_column1*) references *my_table2(my_column),*
…
);
Рассмотрим это ограничение подробно:
Constraint указывает на то, что будет создано ограничение с названием *FK_my_constraint1*, которое будет относиться к столбцу *my_column1* таблицы *my_table* и будет являться внешним ключом foreign key. После ключевого слова references указывается название таблицы *my_table2*, на которую ссылаемся, и в скобках ее первичный ключ *my_column*.

Для того, чтобы при внесении данных в таблицу первичный ключ переключался на следующее значение автоматически, необходимо создать объект «последовательность».

Create sequence *my_sequence1_seq* increment by 1 start with 1;

Последовательности создаются ко всем таблицам, где есть первичный ключ.

Для внесения данных в созданные таблицы используется оператор INSERT:

Insert into *my_table* (*my_column1, my_column2, …*) values *(‘my_value1’, ‘my_value2’, …*);

Строковые значения и числа с плавающей точкой заключаются в одинарные кавычки. Оператор INSERT является транзакцией, следовательно, после каждого его употребления, необходимо подтвердить произведенные изменения, нажав на COMMIT  .

Скрипты создания объектов информационной системы на языке SQL

В редакторе SQL Editor начнем создание наших таблиц с таблицы **Reader**.

```
create table reader (
id_reader number,
familia varchar2(100) not null,
imya varchar2(100) not null,
otchestvo varchar2(100) not null,
passport varchar2(1024) not null,
bilet varchar2(10) not null,
sex char(1),
data_reg date not null,
constraint PK_reader primary key (id_reader),
constraint U_reader unique(familia, imya, otchestvo, passport),
constraint CK_reader check( sex in (‘м’, ‘ж’))
);
```

Данный скрипт требует некоторых пояснений:
В строке *constraint PK_reader primary key (id_reader)* задается первичный ключ. Его название всегда начинается с *PK* и продолжается названием таблицы.
Название следующего ограничения говорит о том, что это уникальность, а после ключевого слова unique в скобках через запятую перечисляется совокупность, которая будет уникальна.
Последнее ограничение позволяет на этапе ввода проверять данные.

Аналогичным образом пишем скрипты для создания оставшихся таблиц.

```
create table sotrudnik (
id_sotr number,
familia varchar2(100) not null,
imya varchar2(100) not null,
otchestvo varchar2(100) not null,
passport varchar2(1024) not null,
sex char(1),
date_hire date not null,
date_uvoln date,
```

```
constraint PK_sotrudnik primary key (id_sotr),
constraint U_sotrudnik unique(familia, imya, otchestvo, passport),
constraint CK_sotrudnik check( sex in (‘м’, ‘ж’))
);

create table book (
id_book number,
name varchar2(500) not null,
author varchar2(500) not null,
constraint PK_book primary key (id_book),
constraint U_sotrudnik unique(name, author)
);

create table char_book (
id_char_book number,
publ varchar2(200) not null,
year_publ number not null,
pages number,
id_book number not null,
constraint PK_char_book primary key (id_char_book)
);

create table library (
id_l number,
date_get date not null,
date_ret date,
note varchar2(500),
id_reader number not null,
id_sotr number not null,
id_char_book number not null,
constraint PK_library primary key (id_l),
constraint U_library unique(id_reader, id_char_book, date_get)
);
```

После создания таблиц переходим к созданию связей между ними.
Обращаем внимание на тот факт, что поля, которые являются внешними ключами, создаются в таблицах наравне с остальными атрибутами.

Добавляем в структуру таблиц, в которых должны быть внешние ключи, соответствующие изменения.

```
alter table char_book add(
constraint FK_book foreign key (id_book) references book (id_book)
);
```

```
alter table library add(
constraint FK_reader foreign key (id_reader) references reader (id_reader),
constraint FK_sotrudnik foreign key (id_sotr) references sotrudnik (id_sotr),
constraint FK_char_book foreign key (id_book) references char_book(id_char_book)
);
```

Создаем к каждой таблице последовательности для генерации значений первичных ключей.

```
Create sequence reader_seq increment with 1 start with 1;
Create sequence sotrudnik_seq increment with 1 start with 1;
Create sequence book_seq increment with 1 start with 1;
Create sequence char_book_seq increment with 1 start with 1;
Create sequence library_seq increment with 1 start with 1;
```

Наконец, осталось внести в таблицы по 3 записи для проверки ее работоспособности.

Начинаем вносить данные с тех таблиц, в которых нет внешних ключей.

Таблица **Reader**:

```
insert into reader (id_reader, familia, imya, otchestvo, passport, bilet, sex, data_reg) values (reader_seq.nextval, ‘Иванов’, ‘Иван’, ‘Иванович’, ’22 67 111111 ОВД района города Москвы’, ‘1/01’, ‘м’, to_date(’01.09.2009’,’dd.mm.yyyy’));

insert into reader (id_reader, familia, imya, otchestvo, passport, bilet, sex, data_reg) values (reader_seq.nextval, ‘Петров’, ‘Михаил’, ‘Иванович’, ’11 34 222222 ОВД района города Москвы’, ‘2/01’, ‘м’, to_date(’10.10.2009’,’dd.mm.yyyy’));

insert into reader (id_reader, familia, imya, otchestvo, passport, bilet, sex, data_reg) values (reader_seq.nextval, ‘Белкин’, ‘Андрей’, ‘Анатольевич’, ’34 22 333333 ОВД района города Москвы’, ‘5/01’, ‘м’, to_date(’02.10.2009’,’dd.mm.yyyy’));
```

После внесения данных необходимо подтвердить произведенные изменения, нажав на кнопку commit  на панели инструментов программы TOra.

Таблица **Sotrudnik**:

insert into sotrudnik (id_sotr, familia, imya, otchestvo, passport, sex, date_hire) values (sotrudnik_seq.nextval, ‘Музин’, ‘Иван’, ‘Иванович’, ’35 67 777777 ОВД района города Протвино’, ‘м’, to_date(’01.03.1998’,’dd.mm.yyyy’));
insert into sotrudnik (id_sotr, familia, imya, otchestvo, passport, sex, date_hire) values (sotrudnik_seq.nextval, ‘Ласкова’, ‘Елена’, ‘Дмитриевна’, ’44 22 444444 ОВД района города Москва’, ‘ж’, to_date(’18.08.2001’,’dd.mm.yyyy’));

insert into sotrudnik (id_sotr, familia, imya, otchestvo, passport, sex, date_hire) values (sotrudnik_seq.nextval, ‘Иванова’, ‘Елена’, ‘Викторовна’, ’11 14 555555 ОВД района города Москва’, ‘ж’, to_date(’02.04.1991’,’dd.mm.yyyy’));

Таблица **Book**:

insert into book (id_book, name, author) values (book_seq.nextval, ‘Краткий курс высшей математики’,’В.А. Кудрявцев, Б.П. Демидович’);

insert into book (id_book, name, author) values (book_seq.nextval, ‘Сборник сочинений’,’А.С. Пушкин’);

insert into book (id_book, name, author) values (book_seq.nextval, ‘Мастер и Маргарита’,’М. Булгаков’);

В следующую таблицу вносим данные в предположении, что в библиотеке есть книга по высшей математике 3-х разных годов издания.

Таблица **Char_book**:

insert into char_book (id_char_book, publ, year_publ, pages, id_book) values (char_book_seq.nextval, ‘Наука’, 1975, 687, 1);

insert into char_book (id_char_book, publ, year_publ, pages, id_book) values (char_book_seq.nextval, ‘Наука’, 1986, 673, 1);

insert into char_book (id_char_book, publ, year_publ, pages, id_book) values (char_book_seq.nextval, ‘Наука’, 1989, 656, 1);

insert into char_book (id_char_book, publ, year_publ, pages, id_book) values (char_book_seq.nextval, ‘Эксмо’, 1954, 321, 2);

insert into char_book (id_char_book, publ, year_publ, pages, id_book) values (char_book_seq.nextval, ‘Эксмо’, 2004, 167, 3);

Таблица **Library**:

```
insert into library (id_l, date_get, id_reader, id_sotr, id_char_book) values (library_seq.nextval, to_date('14.02.2010','dd.mm.yyyy'), 3, 3, 1);
insert into library (id_l, date_get, note, id_reader, id_sotr, id_char_book) values (library_seq.nextval, to_date('14.02.2010','dd.mm.yyyy'), 'пятно на обложке', 3, 3, 5);

insert into library (id_l, date_get, id_reader, id_sotr, id_char_book) values (library_seq.nextval, to_date('18.02.2010','dd.mm.yyyy'), 1, 3, 4);
```

Напоминаем, что после внесения данных необходимо подтвердить произведенные изменения, нажав на кнопку commit  на панели инструментов программы TOra.

Часть вторая.

Запросы SQL.

Запросы – это основное средство для получения информации из базы данных.

Базовый синтаксис оператора SELECT выглядит следующим образом:

```
SELECT [DISTINCT] список_столбцов
FROM из_каких_таблиц
WHERE по_какому_условию
ORDER BY какая_сортировка;
```

> ***Пример:***
>
> Вывести фамилию, имя и отчество всех читателей из таблицы **Reader:**
>
> ```
> SELECT familia, imya, otchestvo
>  FROM reader;
> ```

Для того, чтобы вывести записи без повторений, используется ключевое слово DISTINCT

> ***Пример:***
>
> Вывести список различных фамилий, встречающихся в таблице **Reader:**
>
> ```
> SELECT DISTINCT familia
> FROM reader;
> ```

В том случае, если из таблицы нужно вывести все строки и все столбцы, то вместо списка столбцов можно просто указать символ *.

> ***Пример:***
>
> SELECT *
>
> FROM reader;

ORDER BY.

Раздел ORDER BY используется для сортировки результата по возрастанию либо по убыванию. Сортировка по возрастанию осуществляется по умолчанию.

> ***Пример:***
>
> Вывести список всех читателей отсортированных по фамилии по алфавиту.
>
> SELECT id_reader, familia, imya, otchestvo
>
> FROM reader
>
> ORDER BY familia;

Если нужно отсортировать записи в порядке убывания, то используется ключевое слово **DESC**.

> ***Пример:***
>
> Вывести список всех читателей отсортированных по фамилии в обратном порядке.
>
> SELECT id_reader, familia, imya, otchestvo
>
> FROM reader
>
> ORDER BY familia DESC;

Псевдонимы.

Выводимое имя столбца можно поменять, используя псевдонимы. Для этого достаточно просто написать имя псевдонима сразу после названия столбца или использовать ключевое слово AS (оба синтаксиса полностью равнозначны). Если в псевдониме используются пробелы или какие-то служебные символы, то такой псевдоним нужно поместить в двойные кавычки.

Пример:

Вывести список всех читателей, отсортированных по фамилии.

```
SELECT id_reader AS “номер читателя”, familia AS “фамилия”, imya AS “имя”, otchestvo AS “отчество”
FROM reader
ORDER BY familia;
```

Аналогичным образом можно давать псевдонимы таблицам и обращаться в запросе к их полям через псевдонимы. Это делается для упрощения кода в сложных запросах, где используется большое количество таблиц и полей, которые надо из них извлечь. Перепишем предыдущий пример с использованием псевдонимов таблиц. Псевдоним таблицы пишется сразу после названия таблицы в разделе FROM. Обращение к полям в запросе происходит через точку в виде **псевдоним_таблицы.название_поля**

Пример:

Вывести список всех читателей, отсортированных по фамилии.

```
SELECT r.id_reader AS “номер читателя”, r.familia AS “фамилия”, r.imya AS “имя”, r.otchestvo AS “отчество”
FROM reader r
ORDER BY r.familia;
```

Псевдонимы таблиц нам понадобятся, когда мы будем изучать запросы из нескольких таблиц в конце данного материала.

В запросах можно использовать операторы – это синтаксические конструкции языка, предназначенные для проведения определенных действий. Используются следующие арифметические операторы:

+ -- сложение

- -- вычитание

* -- умножение

/ -- деление

Оператор конкатенации.

Несколько значений можно вывести как одно, используя оператор конкатенации – слияние строк. Обозначается он двумя вертикальными палочками ||.

Пример:

Написать запрос, который выводит одним значением ФИО читателя

```
SELECT familia ||' '||imya ||' '|| otchestvo AS "ФИО"
FROM reader;
```

WHERE.

Раздел WHERE используется в том случае, когда необходимо указать условия запроса. Для этого используются простые операторы сравнения:

= Равно

< Меньше

> Больше

<= Меньше или равно

>= Больше или равно

!= Не равно

Пример:

Выбрать всех читателей по фамилии «Иванов»

```
SELECT *
FROM reader
WHERE familia = 'Иванов';
```

Логические операторы.

Несколько условий соединяются друг с другом логическими операторами.

- AND – возвращает истину, когда оба условия истинны.
- OR – возвращает истину, когда хотя-бы одно условие истинно.
- NOT – отрицание условия.

Если в одном разделе WHERE используется сразу несколько различных логических операторов, то порядок их выполнения регулируется круглыми скобками (как и в математике).

Пример:

Вывести все значения из таблицы читателей, где имя читателя Андрей, а фамилия Иванов или Петров.

```
SELECT *

FROM reader

WHERE imya = ‘Андрей’ AND (familia = ‘Иванов’ OR familia = ‘Петров’);
```

Наивысший приоритет у операции отрицания – она всегда выполняется первой. Затем идет AND, и самый низкий приоритет у OR.

Работа с пустыми значениями.

Если в какой-то ячейке в таблице базы данных нет значения, то считается, что в этой ячейке находится специальное значение NULL. NULL — это не 0 и не пустая строка. NULL считается специальным значением, для которого существуют специальные приемы работы.

Применить специальные операторы сравнения для значений типа NULL не получится: NULL никогда не равен другому значению NULL. Поэтому для работы с ним предусмотрены два специальных условия:

- **IS NULL** — это условие вернет истину, если проверяемое значение равно NULL;
- **IS NOT NULL** — это условие вернет истину, если проверяемое значение не равно NULL.
-

Пример:

Выбрать id читателей, id книг и даты получения книг из таблицы library, если дата возврата книги пустая (то есть книга находится на руках)

```
SELECT id_reader “Номер читателя”, id_book “Номер книги”, date_get “Дата получения”
FROM library
WHERE date_ret IS NULL;
```

Оператор LIKE.

Очень часто в работе возникает необходимость провести поиск по набору символов в любом месте столбца — например, для поиска всех записей с названиями товаров, содержащими какое-либо слово, или для поиска какого-либо слова как в единственном, так и во множественном числе. Для этой цели используется оператор LIKE.

Особенностью оператора LIKE является то, что он может включать в условие специальные подстановочные символы (метасимволы). Для этого оператора предусмотрено только два подстановочных символа:

- % — представляет любую последовательность из нуля или более символов. При этом значение '%' никогда не будет равно NULL (для проверки таких значений используется IS NULL);
- _ — представляет любой одиночный символ.

Пример:

Выбрать названия всех книг, которые содержат фрагмент ‘электро’

```
SELECT name
FROM book
WHERE name LIKE ‘%электро%’;
```

Пример:

Выбрать фамилии всех читателей, которые заканчиваются на ‘вич’

```
SELECT familia
FROM reader
WHERE familia LIKE ‘%вич’;
```

Оператор BETWEEN.

Иногда необходимо вернуть все значения, которые попадают в какой-либо диапазон. Для этой цели используется специальный оператор BETWEEN.

Пример:

Выбрать номера всех книг, которые были выпущены в период 1998-2010гг.

```
SELECT id_book
FROM char_book
WHERE year_publ BETWEEN 1998 AND 2010;
```

Сложные запросы. Выборка из нескольких таблиц.

Практически во всех серьезных отчетах требуется запрашивать данные из различных таблиц. Выборка данных из нескольких таблиц осуществляется следующим образом:

Пример:

Выбрать фамилии читателей, номер читательского билета, дату получения книги, которые брали книгу под названием «Математика» и еще не вернули в алфавитном порядке.

Пояснение: В данной задаче информация о читателе берется из таблицы reader, дата получения и сдачи книги – из таблицы library, а название – из таблицы book. Между этими таблицами есть связь в виде первичных и внешних ключей (значение внешнего ключа в одной таблице равно значению первичного в другой – см. связи между таблицами в схеме).

На данную выборку накладывается условие, что date_reg должно быть пустым (так как книгу еще не вернули, то дата возврата пустая), а также существует ограничение по названию – оно жестко прописано в условии задачи, поэтому оператор LIKE здесь не используется.

Обращение к полям таблиц будем делать с использованием псевдонимов таблиц. Все участвующие в запросе таблицы перечисляются через запятую в разделе FROM.

Обратите внимание на тот факт, что таблицы library и book связаны друг с другом не напрямую, а через промежуточную таблицу char_book. Поэтому,

связующую таблицу также надо указывать в перечне, несмотря на то, что явно в запросе из нее данные не выбираются.

Связи между таблицами перечисляются в разделе WHERE в виде приравнивания первичного ключа из одной таблицы и соответствующего ему внешнего ключа из другой таблицы (это лучше делать глядя на схему, чтобы не запутаться).

Все условия связываются друг с другом логическим оператором AND. Теперь собираем запрос.

```
SELECT  r.familia AS “Фамилия читателя”, r.bilet AS “Номер билета”,
        lib.date_get AS “ Дата получения книги”
FROM reader r, library lib, book b, char_book cb
WHERE r.id_reader = lib.id_reader and
      lib.id_char_book = cb.id_char_book and
      cb.id_book = b.id_book and
      b.name = ‘Математика’ and
      lib.date_reg IS NULL;
```

Задание на вторую часть курсовой работы:

1. По аналогии с каждым разделом второй части (кроме последнего) построить запросы к любым 4-м таблицам из Вашей схемы базы данных (к каждой из 4-х таблиц весь набор запросов, кроме последнего). Данный процесс осуществляется в программе TOra (см. первую часть Методических указаний).
2. По аналогии с последним разделом второй части сделать один сложный запрос, который будет делать сводный отчет из не менее, чем 5-и таблиц из Вашей схемы базы данных.
3. Каждый запрос должен сопровождаться скриншотом с его результатами.
4. Каждый запрос должен сопровождаться осмысленным описанием (как задания в примерах). Особое внимание обратите на описание последнего, сложного запроса.
5. Оформляете в конец того же файла, где писали первую часть курсовой работы.