

Технологии программирования

Часть 5

Направление:

Информатика и вычислительная техника
Инфокоммуникационные технологии и системы связи

Ст. преподаватель
кафедры ПИВТ
Петрова О.Б.

2017 год

Разработка приложения, взаимодействующего с базой данных

- Основные понятия теории баз данных.
- Конструирование реляционной базы данных.
- Конструирование запросов на языке SQL.
- Средства библиотеки Qt для работы с базами данных.

Основные понятия теории баз данных

- **База данных** — именованная совокупность данных, отображающая состояние изучаемых объектов (предметов, явлений и т.д.).
- **Предметная область** — изучаемая совокупность логически связанных объектов.
- **Актуальность** базы данных — постоянное изменение, пополнение данных в соответствии с изменением состояния изучаемых объектов и наших знаний о них.
- **СУБД** — система управления базами данных — совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования баз данных.

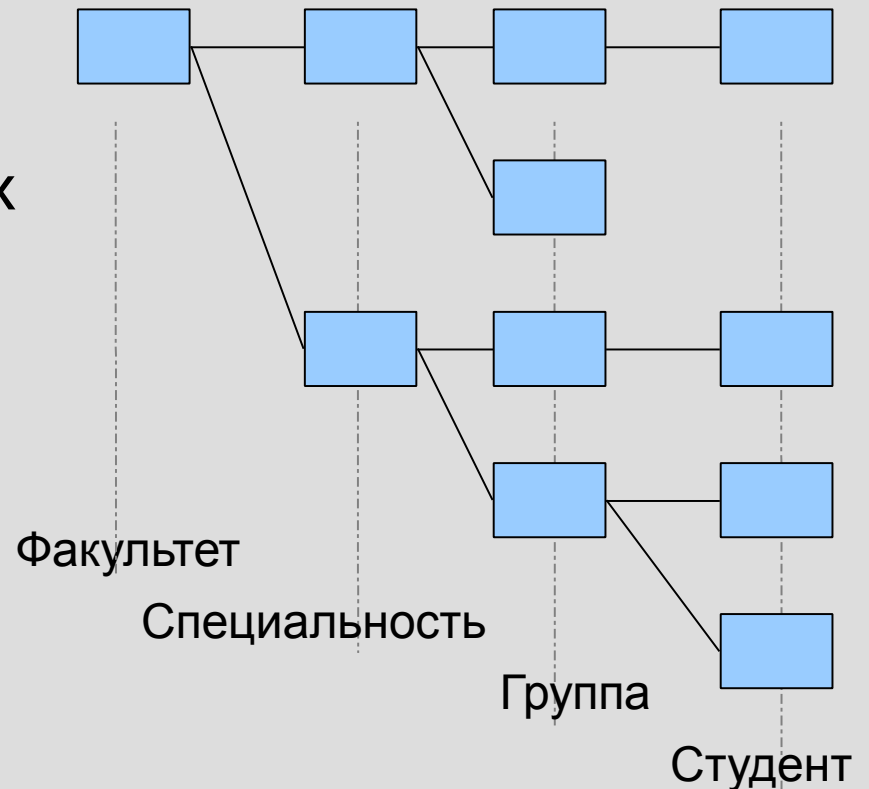
Преимущества использования баз данных

- Многократное использование данных.
- Простота и легкость использования.
- Гибкость использования.
- Быстрота обработки запросов пользователей.
- Разграничение прав пользователей.
- Функционирование в условиях вычислительных сетей.
- Контроль за целостностью данных.
- Восстановление данных после сбоя.
- Средства администрирования и оптимизации работы системы.

Модели данных

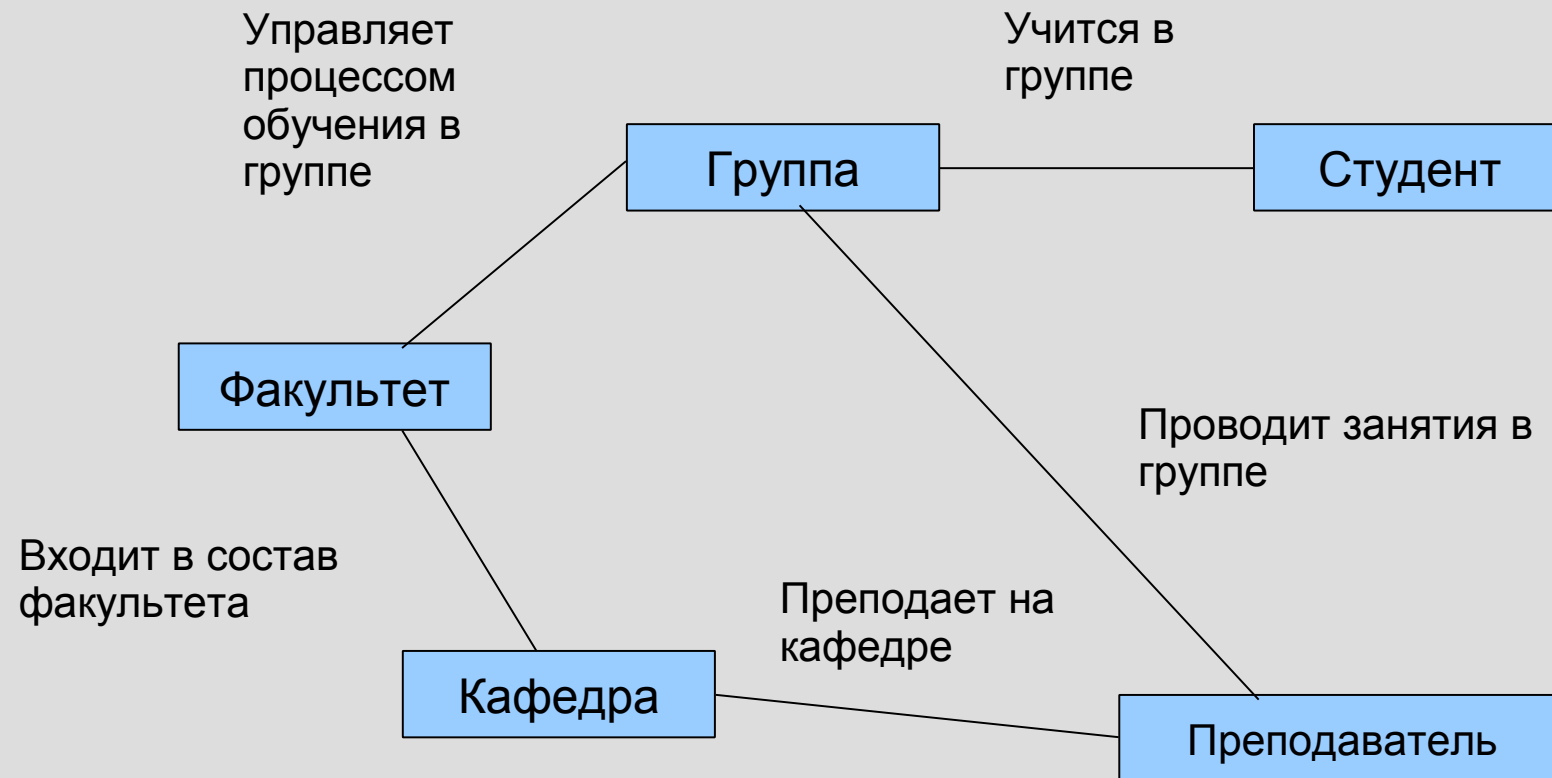
- Иерархические (IMS, IBM, 1968)
- Сетевые (ИСУБД Cronos Pro, Россия)
- Реляционные

Иерархическая модель данных
Структура данных: дерево



Сетевая модель данных

Для связи элементов данных использует указатели
Структура данных: граф



Реляционная модель данных

Предложена: Едгар Кодд, 1970 год.

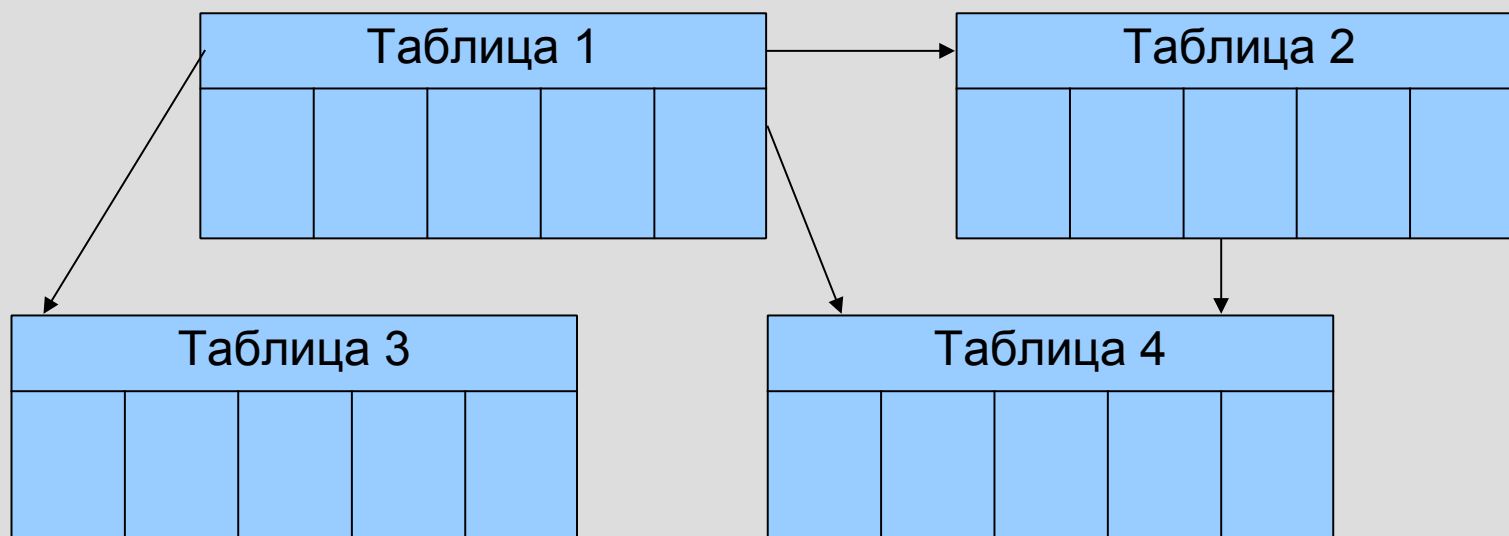
Логическая модель, использует математическую теорию отношений (relation).

Математический аппарат: теория множеств, логика первого порядка (логика предикатов), реляционная алгебра.

Модель не зависит от физической организации данных.

Модель не использует указатели.

Отношение — таблица данных.



Примеры реляционных СУБД

- Oracle
- Microsoft SQL
- MySQL
- PostgreSQL
- SQLite

Языковые средства:

- SQL (Structured Query Language).

Интерфейс пользователя:

- Web-приложение.

Основные понятия реляционных баз данных

- Тип данных.
- Домен.
- Отношение.
- Атрибут отношения.
- Кортеж.
- Первичный ключ.
- Внешний ключ.

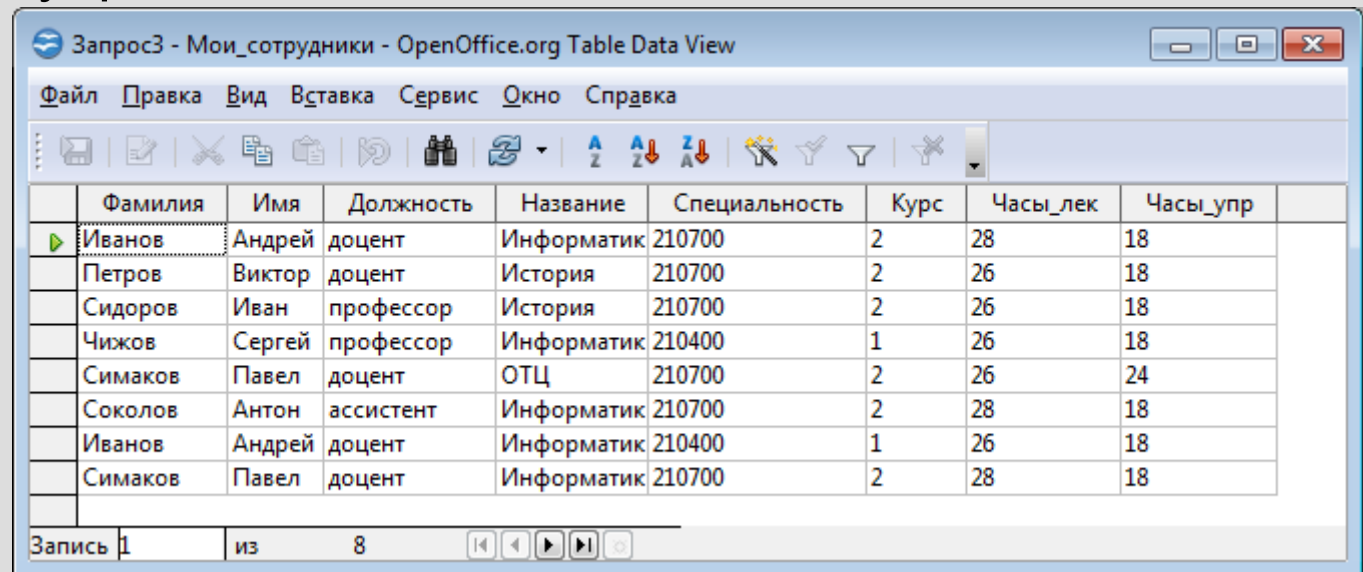
Типы данных	Целое	Строка		Целое	
Домены	Номер	Фамилии	Группы	Деньги	
Атрибуты	Номер личного дела	Фамилия	Группа	Стипендия	Надбавка
Кортежи	12345	Розов	ИКТ-123	1000	300
	12357	Симонов	ИКТ-173	1000	0
	12378	Михайлов	ИКТ-144	0	0

Ключ

Отношение

Конструирование реляционной базы данных

- Дано: информация о преподавателях
 - Фамилия преподавателя.
 - Имя преподавателя.
 - Должность преподавателя.
 - Предмет.
 - Специальность.
 - Курс.
 - Кол-во часов лекций.
 - Кол-во часов упражнений.
- 1 вариант структуры данных:



Запрос3 - Мои_сотрудники - OpenOffice.org Table Data View

Файл Правка Вид Вставка Сервис Окно Справка

	Фамилия	Имя	Должность	Название	Специальность	Курс	Часы_лек	Часы_упр
▶	Иванов	Андрей	доцент	Информатик	210700	2	28	18
	Петров	Виктор	доцент	История	210700	2	26	18
	Сидоров	Иван	профессор	История	210700	2	26	18
	Чижов	Сергей	профессор	Информатик	210400	1	26	18
	Симаков	Павел	доцент	ОТЦ	210700	2	26	24
	Соколов	Антон	ассистент	Информатик	210700	2	28	18
	Иванов	Андрей	доцент	Информатик	210400	1	26	18
	Симаков	Павел	доцент	Информатик	210700	2	28	18

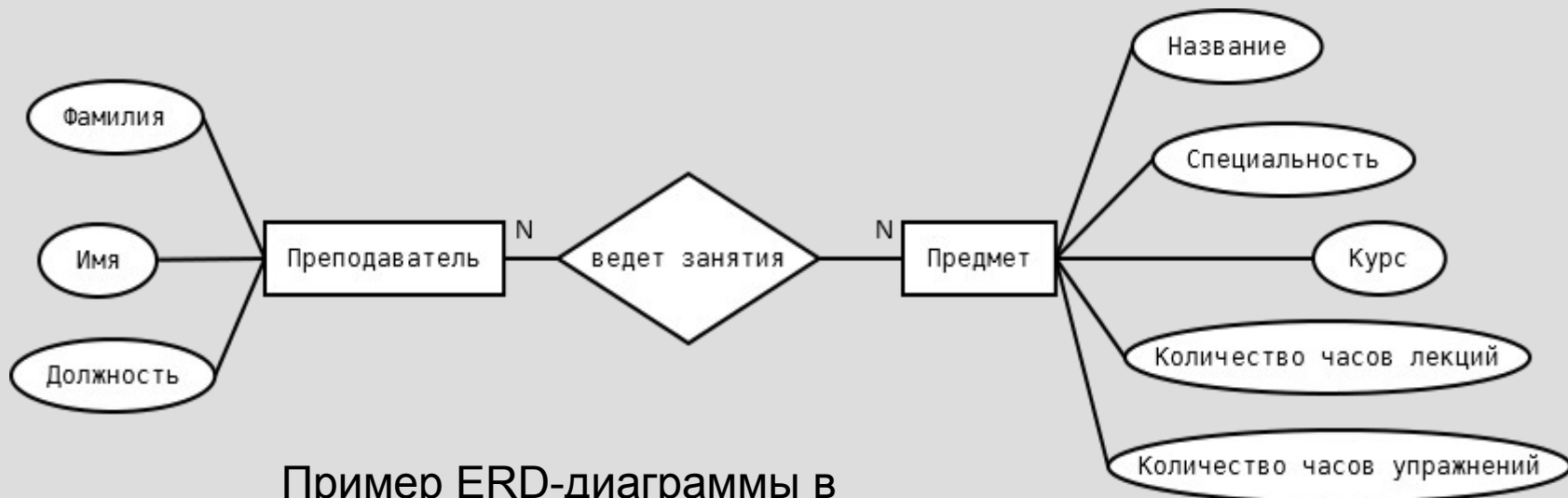
Запись 1 из 8

Этапы проектирования базы данных

- Разработка инфологической модели
- Разработка даталогической модели
- Разработка физической модели

1. Разработка инфологической модели

- Анализ предметной области: выявление объектов и их атрибутов (объектная декомпозиция)
 - Преподаватель (фамилия, имя, должность);
 - Предмет (название предмета, специальность, курс, кол-во часов лекций, упражнений).
- Выявление связей между объектами: построение ER-модели (Entity-Relation), не зависящей от СУБД.



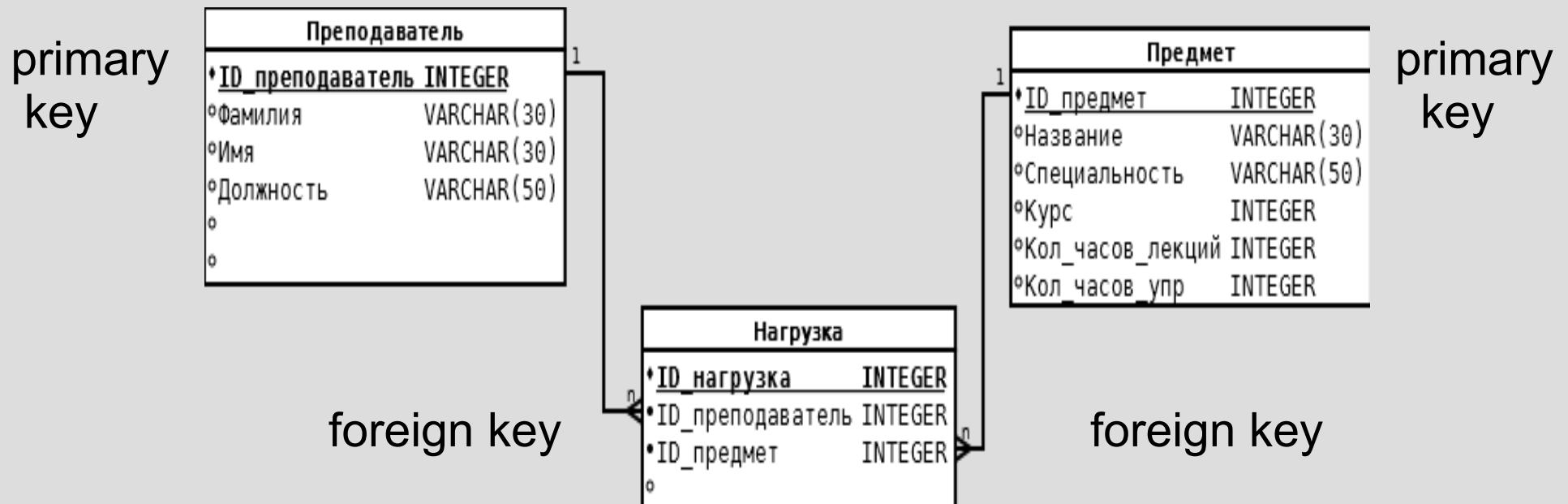
Пример ERD-диаграммы в нотации Чена

Разработка даталогической модели

- Логическое проектирование данных.
- Выбор СУБД определяет модель данных.
- Описание данных — в терминах выбранной модели.
- Реляционная модель: сущность — таблица, имя сущности — имя таблицы.
- Атрибут — столбец таблицы.
- Уникальный идентификатор — первичный ключ.
- Связи «много-к-одному» и «один-к-одному» становятся внешними ключами.
- Связь «много-ко-многим» превращается в таблицу с двумя связями «много-к-одному».

Пример даталогической модели

ID_преподавателя и ID_предмета — ключевые поля



Пример ERD-диаграммы в нотации Баркера

Физическая модель данных

- Определяет способ размещения данных на носителях (устройствах внешней памяти):
 - структура записи в файле данных,
 - количество файлов данных,
 - местоположение файлов данных.
- Определяет способ и средства организации эффективного доступа к данным:
 - способы адресации и методы поиска записей в файлах.

Средства управления данными

- Запросы
 - запросы по образцу
 - SQL-запросы
- Формы
- Отчеты

Конструирование запросов на языке SQL

- Язык SQL:
 - Structured Query Language.
 - Первый стандарт 1989 года.
 - Текущий стандарт ISO/IEC 9075-2016 (1-14).
 - Логический (декларативный) язык.
 - Использует исчисление кортежей (раздел реляционной алгебры).
 - Содержит
 - средства управления таблицами,
 - средства отбора данных,
 - средства модификации данных.

Команды SQL

- CREATE — создать
- UPDATE - обновить
- INSERT - вставить
- SELECT - выбрать
- DELETE — удалить

Предложения, используемые в командах:

- WHERE — где (условие отбора)
- FROM — откуда (источник данных — таблица)
- INTO — куда

Типы данных:

- INTEGER – целое число,
- FLOAT - вещественное число,
- VARCHAR(n) — строка длиной n символов.

Примеры запросов

```
CREATE TABLE Преподаватель (ID_преподаватель  
INTEGER PRIMARY KEY, Фамилия  
VARCHAR(30), Имя VARCHAR(30), Должность  
VARCHAR(50));
```

```
INSERT INTO Преподаватель (ID_преподаватель,  
Фамилия, Имя, Должность) VALUES (1, "Валова",  
"Анна", "ассистент");
```

```
DELETE FROM Преподаватель WHERE Фамилия =  
"Валова";
```

```
UPDATE Преподаватель SET Должность="Доцент"  
WHERE Фамилия= "Валова";
```

Запросы на выборку

```
SELECT * FROM Преподаватель WHERE  
Имя="Анна";
```

```
SELECT * FROM Преподаватель WHERE  
ID_преподаватель>4;
```

```
SELECT * FROM Преподаватель WHERE  
ID_преподаватель>4 AND ID_преподаватель<10;
```

```
SELECT * FROM Преподаватель WHERE  
Имя="Анна" OR Имя="Елена";
```

Оператор LIKE

- Стандарт SQL:
SELECT Фамилия, Имя, Должность FROM
Преподаватель WHERE Фамилия LIKE "%ро-";
- Windows:
SELECT Фамилия, Имя, Должность FROM
Преподаватели WHERE Фамилия LIKE "*ро?";
- Ответ: Петров, Сидоров, Крот, Рой.
- Не будет отобрано: Сидорова

INNER JOIN

- связывает две таблицы: левую и правую, при этом в запросе не участвуют строки из правой таблицы, не имеющие продолжение в левой, и наоборот.
- `SELECT <список полей из правой и левой таблицы>`
- `FROM левая_таблица INNER JOIN правая_таблица`
- `ON <условия связывания> WHERE <условия отбора>;`

Запрос по двум таблицам

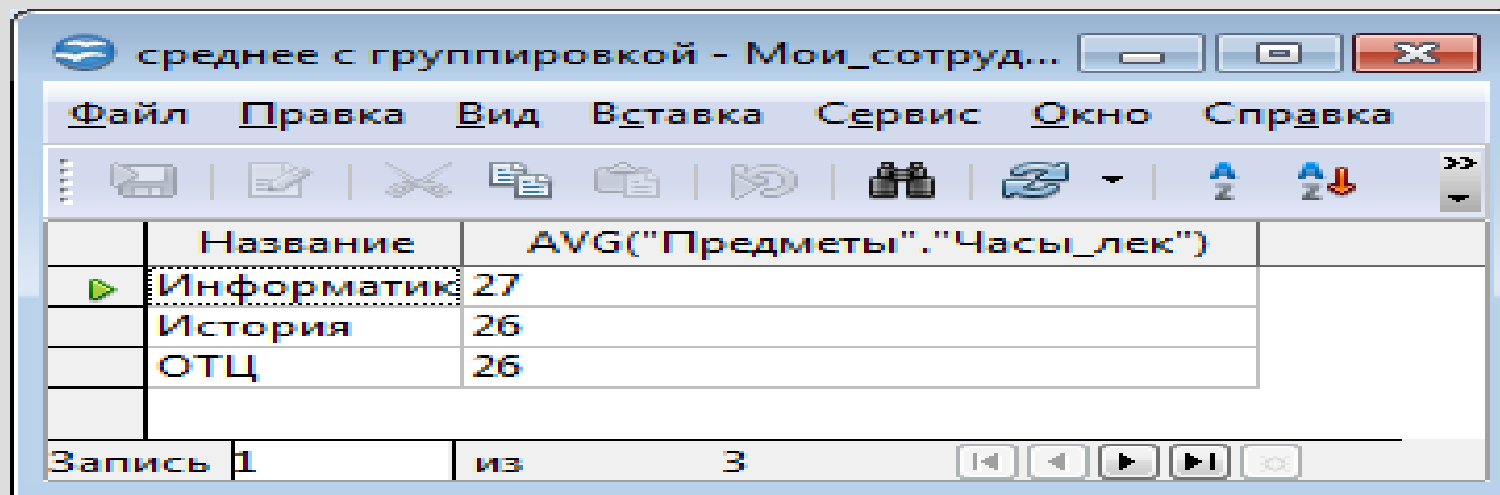
- `SELECT Преподаватели.Фамилия,
Нагрузка.ID_предмет FROM
Преподаватели INNER JOIN Нагрузка ON
Преподаватели.ID_преподаватель=Нагруз
ка.ID_преподаватель WHERE
Нагрузка.ID_предмет>1 AND
Нагрузка.ID_предмет<5;`

Статистическая обработка в запросе

- Статистические функции (функции агрегирования): count, sum, min, max, avg (среднее арифметическое).
- `SELECT AVG(Часы_лекций) FROM Предметы;`
- Ответ: среднее арифметическое по всей таблице.

Группировка в запросе

- Предложение GROUP BY
- SELECT AVG(Часы_лекций) FROM Предметы GROUP BY Название;
- SELECT Название, AVG(Часы_лекций) FROM Предметы GROUP BY Название;



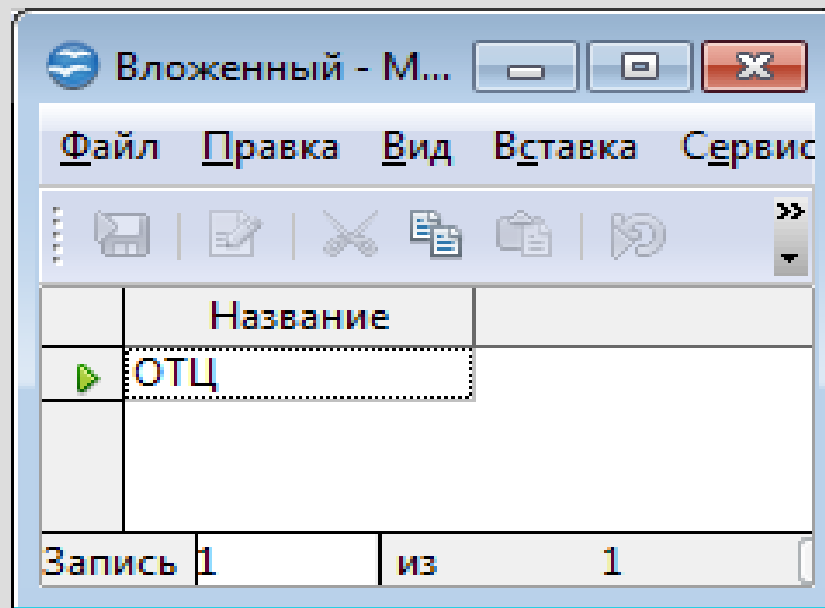
The screenshot shows a window titled "среднее с группировкой - Мои_сотруд...". The window contains a table with the following data:

Название	AVG("Предметы"."Часы_лек")
Информатик	27
История	26
ОТЦ	26

At the bottom of the window, there is a status bar showing "Запись 1 из 3" and navigation icons.

Вложенный запрос

- `SELECT Название FROM Предмет WHERE Часы_упражнений =(SELECT MAX(Часы_упражнений) FROM Предмет);`



Вложенный - М...

Файл Правка Вид Вставка Сервис

	Название
▶	ОТЦ

Запись 1 из 1

Средства библиотеки Qt для работы с базами данных

- драйвера в соответствии с выбранной СУБД,
- класс QSqlDatabase для связи приложения и базы данных,
- класс QSqlQuery - для выполнения инструкций SQL,
- класс QSqlTableModel — для представления одной таблицы базы данных в приложении, редактируемая модель таблицы,
- класс QTableView для вывода результатов запросов в окно.

Подключение библиотечных средств к проекту

- подключить заголовочный файл QtSql
- добавить информацию о компоновке приложения с библиотекой в pro-файле проекта:

```
QT += core gui sql
```

Драйвера баз данных

QMYSQL — MySQL,
QODBC — Microsoft SQL Server,
QPSQL — PostgreSQL,
QSQLITE2 — SQLite версия 2,
QSQLITE — SQLite версия 3.

Класс QSqlDatabase

Методы:

```
QSqlDatabase addDatabase ( const QString & type, const QString &  
    connectionName =QLatin1String( defaultConnection ) );
```

```
void setDatabaseName ( const QString & name );
```

```
void setHostName ( const QString & host );
```

```
void setPassword ( const QString & password );
```

```
void setPort ( int port );
```

```
void setUsername ( const QString & name );
```

```
QString databaseName () const;
```

```
bool open ();
```

```
void close ();
```

Класс QSqlQuery

```
bool exec ( const QString & query );  
const QSqlResult * result () const;  
bool first ();  
bool last ();  
bool next ();  
bool previous ();  
int size () const;  
void clear ();
```

Перебор записей в выборке по запросу

```
QSqlQuery query;  
query.exec("SELECT name, salary FROM  
employee WHERE salary > 50000");  
  
while (query.next())  
{  
    QString name = query.value(0).toString();  
    int salary = query.value(1).toInt();  
    // другие действия  
}
```


Класс QSqlTableModel

```
QSqlTableModel ( QObject * parent = 0, QSqlDatabase db =  
    QSqlDatabase() );  
virtual void setEditStrategy ( EditStrategy strategy );  
enum EditStrategy { OnFieldChange, OnRowChange,  
    OnManualSubmit }  
EditStrategy editStrategy () const;  
virtual void setTable ( const QString & tableName );  
virtual void setSort ( int column, Qt::SortOrder order );  
virtual bool select ();  
virtual void setFilter ( const QString & filter );  
QString filter () const;
```

Класс `QTableView`

Методы:

```
QTableView ( QWidget * parent = 0 );
```

```
void setColumnWidth ( int column, int width );
```

```
void setRowHeight ( int row, int height );
```

```
int columnWidth ( int column ) const;
```

```
int rowHeight ( int row ) const;
```

```
void setGridStyle ( Qt::PenStyle style );
```

```
Qt::PenStyle gridStyle () const;
```

Типы линий в Qt

enum Qt::PenStyle:

Qt::NoPen	0	нет линии.
Qt::SolidLine	1	сплошная линия.
Qt::DashLine	2	черточки, разделенные несколькими пикселями.
Qt::DotLine	3	точки, разделенные несколькими пикселями.
Qt::DashDotLine	4	штрих-пунктирная линия.
Qt::DashDotDotLine	5	одна черточка, две точки, одна черточка, две точки.

Создание объектов, взаимодействующих с базой данных

Создание объекта базы данных и его подключение к базе данных:

```
QSqlDatabase m_db =  
    QSqlDatabase::addDatabase("QSQLITE");
```

Создание объекта для выдачи запросов и соединение его с базой данных:

```
QSqlQuery* query = new QSqlQuery(m_db);
```

Создание редактируемой модели таблицы:

```
QSqlTableModel* model = new  
    QSqlTableModel(this, m_db);
```

Выдача запроса базе данных

Отсылка запроса:

```
query->exec("INSERT INTO Person (number, name, salary) VALUES (1,'Ann', 25)")
```

При повторном использовании объекта query его следует очистить:

```
query->clear();
```

Связывание таблицы и модели таблицы:

```
model->setTable("Person");
```

Получение результата запроса:

```
model->select();
```

Фильтрация в запросе:

```
model->setFilter("Year>2000"); model->select();
```

Вывод результатов запроса из модели на экран в таблицу:

```
ui->tableView->setModel(model);
```

где ui – идентификатор формы, tableView — таблица в форме.