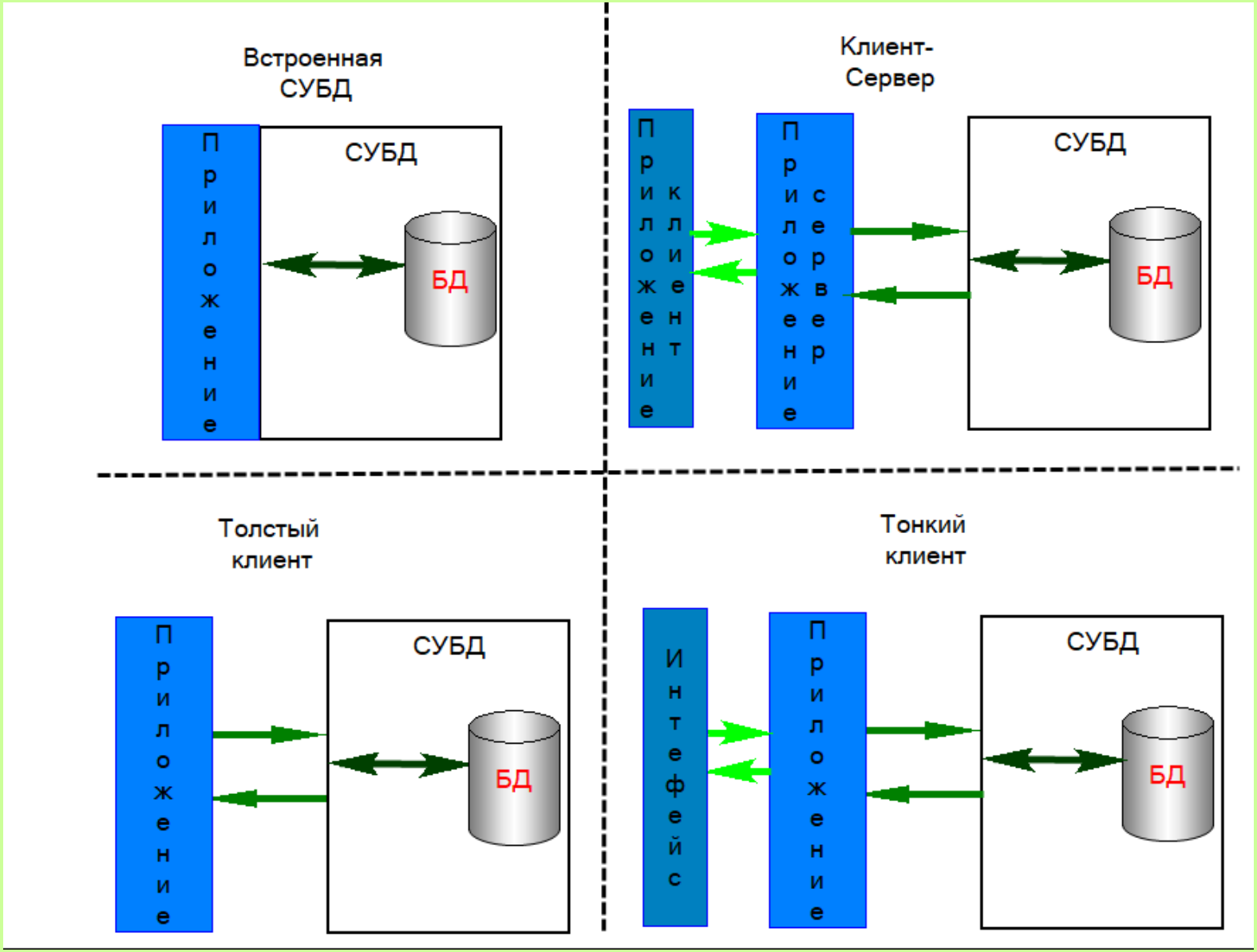
Роль баз данных в программных системах

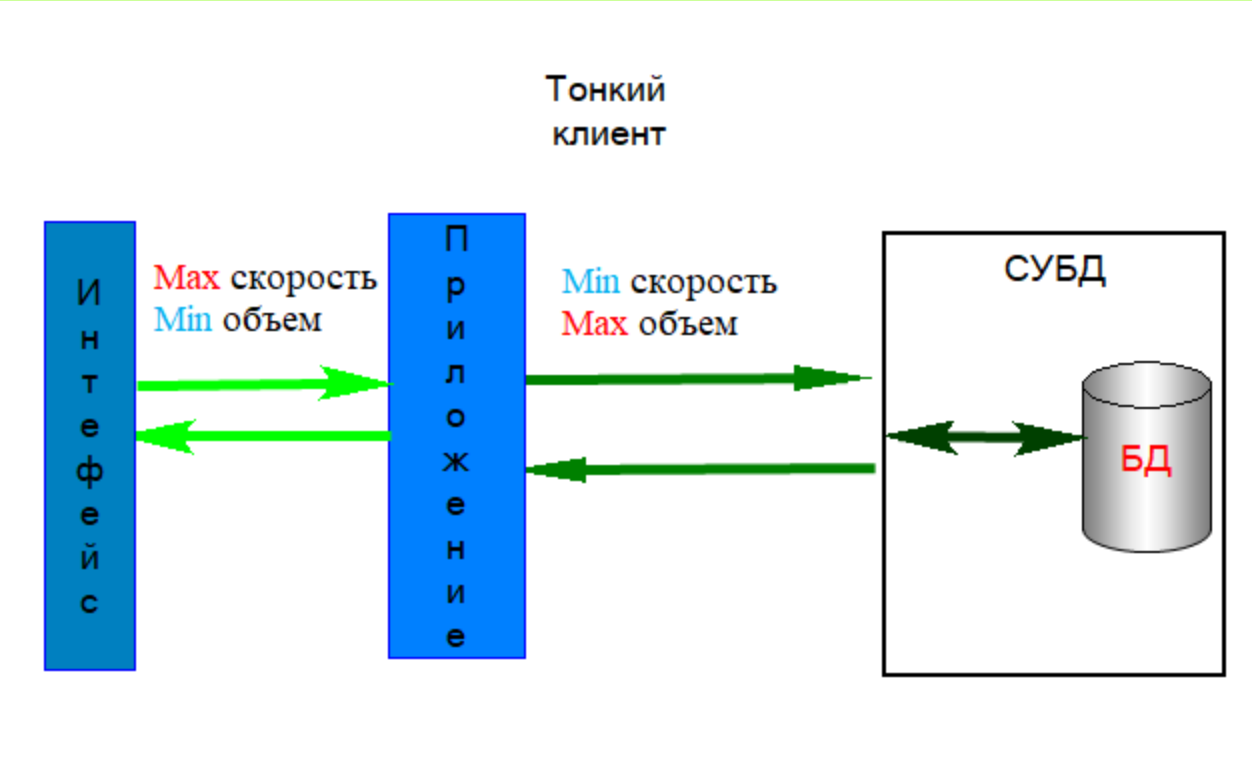
Способы взаимодействия базы данных и приложения

- Встроенная БД
- In-Memory (резидентная) БД
- Тонкий клиент/клиент-сервер
- Толстый клиент

БД,СУБД и приложение



Тонкие места взаимодействия



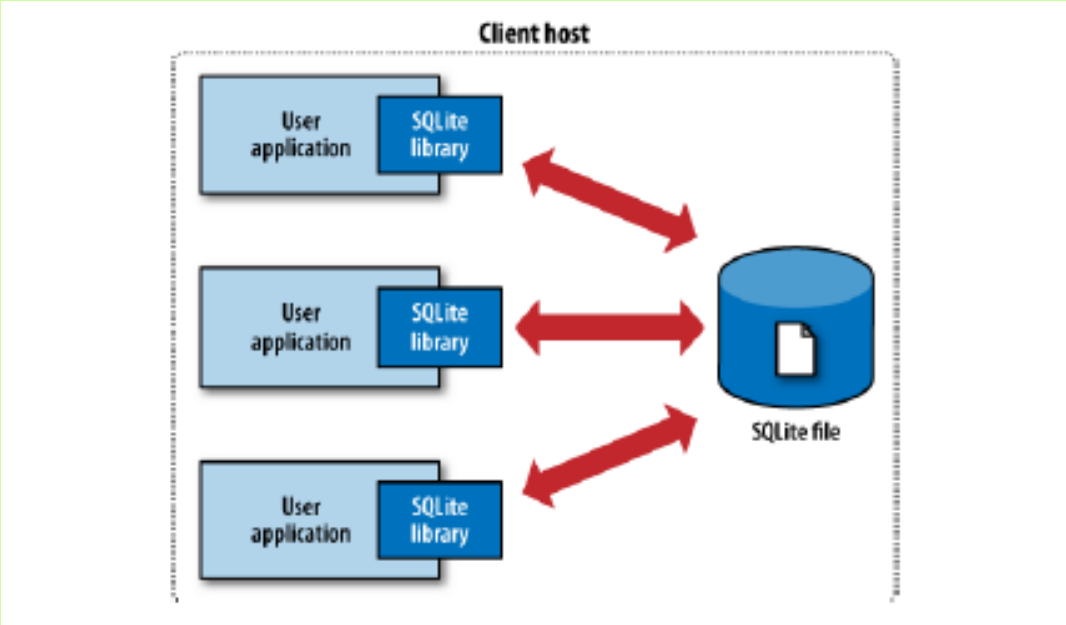
Встроенные БД (Emdedded)

- Встроенные системы баз данных - это системы управления базами данных (СУБД), встроенные или интегрированные в приложение, эффективно скрывающие или минимизирующие взаимодействие с базой данных конечными пользователями приложения.
- **SQLite**
- **SQL Server Compact**
- **RocksDB**
- **InterBase**
- **Firebird Embedded**
- **Apache Derby**
- InnoDB
- …

Встроенная БД (Emdedded)

- **Бессерверное**
- SQLite не требует отдельного серверного процесса или системы для работы. SQLite Библиотека обращается к своим файлам хранения напрямую.
- **Нулевая конфигурация**
- Отсутствие сервера означает отсутствие настройки. Создать экземпляр базы данных SQLite так же просто, как открытие файла.
- **Кросс-платформенная**
- Весь экземпляр базы данных находится в одном кроссплатформенном файле, не требующем администрация.
- **Автономный**
- Одна библиотека содержит всю систему баз данных, которая интегрируется непосредственно в хост-приложение.
- Малый шаг во время выполнения
- Сборка по умолчанию занимает меньше мегабайта кода и требует всего несколько мегабайт памяти. С некоторыми изменениями размер библиотеки и использование памяти могут быть значительно снижено

SQLite


Client host
User application
SQLite library
User application
SQLite library
User application
SQLite library
SQLite file

Где используются встроенные бд

- Встраиваемые устройства и интернет вещей
- Формат файла приложения
- Веб-сайты (низкий и средний трафик)
- Анализ данных
- Внутренние или временные базы данных
- Заглушка для корпоративной базы данных во время демонстраций или тестирования

In-Memory (резидентная) БД

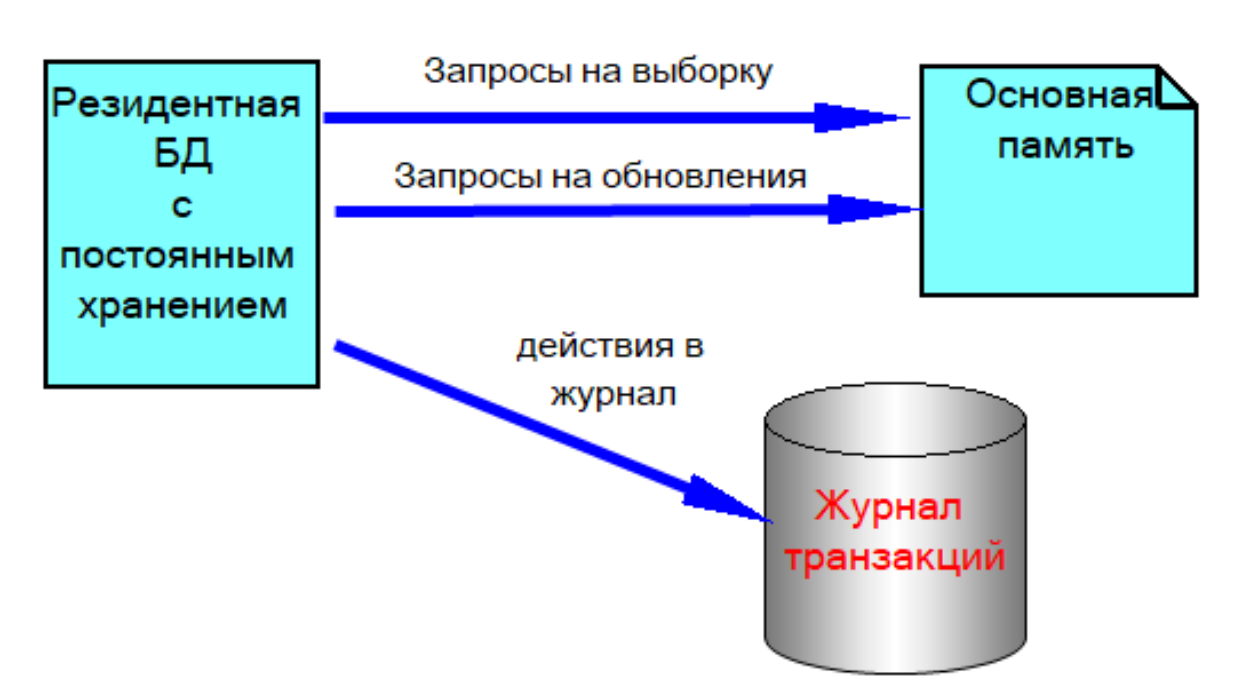
- Резидентная база данных - это тип специально созданной базы данных, которая в основном опирается на оперативную память для хранения данных, в отличие от баз данных, которые хранят данные на диске или твердотельных накопителях.
  Резидентные базы данных предназначены для достижения минимального времени отклика за счет исключения необходимости доступа к дискам. Поскольку все данные хранятся и управляются исключительно в основной памяти, существует риск их потери при сбое процесса или сервера. Базы данных в памяти могут сохранять данные на дисках, сохраняя каждую операцию в журнале или делая снимки.
- Резидентные базы данных идеально подходят для приложений, которые требуют времени отклика в микросекундах и могут иметь большие всплески трафика, приходящего в любое время, такие как игровые таблицы лидеров, хранилища сеансов и аналитика в реальном времени.

Резидентные БД использование

- Торги в реальном времени
- Таблицы лидеров компьютерных игр
- обслуживание оборудования для мобильной связи
- Кэширование

Как работают резидентные БД

- СУБД в оперативной памяти не обращаются к диску при обработке запросов, не изменяющих данные.
- СУБД в оперативной памяти все-таки обращаются к диску при обработке запросов, изменяющих данные, но работают с ним на максимальной скорости.



Три основных слоя приложения

Слой	Функция
Представление (*presentation*)	Предоставление услуг, отображение данных, обработка событий пользовательского интерфейса (щелчков кнопками мыши и нажатий клавиш), обслуживание запросов HTTP, поддержка функций командной строки и API пакетного выполнения
Домен (domain)	Бизнес-логика приложения
Источник данных (*data source*)	Обращение к базе данных, обмен сообщениями, управление транзакциями и т.д

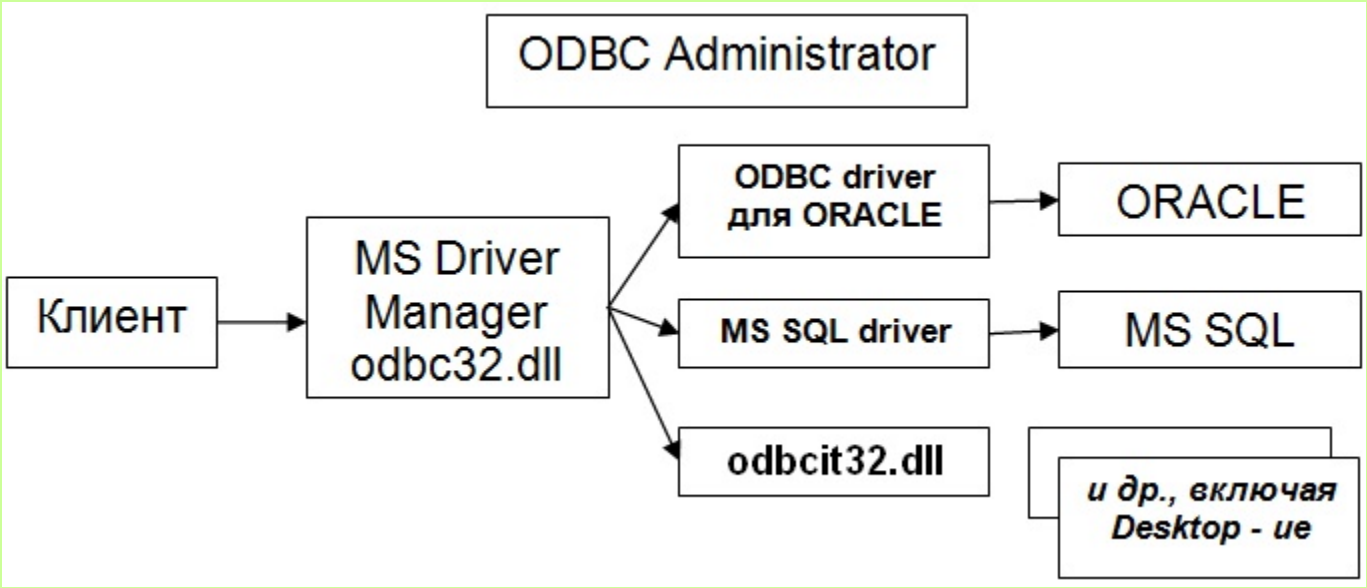
Способы доступа к данным/ подключения к базе

- ODBC
- JDBC
- OLE DB (Microsoft)
- ADO $\rightarrow$ADO.NET

- ORM

ODBC

- ODBC (англ. Open Database Connectivity) — это программный интерфейс (API) доступа к базам данных, разработанный фирмой Microsoft, в сотрудничестве с Simba Technologies на основе спецификаций Call Level Interface (CLI)



Использование ODBC в C# (.Net)

Настройки:

- В меню проект выберите команду Добавить ссылку.
- На вкладке .NET выберите Microsoft.Data.ODBC.dll.
- После сборки Microsoft.Data.ODBC.dll появится в списке Выбранные компоненты, нажмите кнопку ОК.
- Переключитесь в представление кода и добавьте следующий код сразу после других конструкций using :

```
using System.Data;
using Microsoft.Data.Odbc;
```

Использование ODBC в C# (.Net)

```
OdbcConnection cn; cmd=new OdbcCommand(MyString,cn);
cn.Open(); //Открытие подключения
cn.Close(); //Закрытие подключения
```

- OdbcCommand sampleCommand; // Переменная для выполнения запроса
- OdbcDataReader sampleReader = null; // Переменная для чтения полученных данных из базы
- int sampleCountColumns = 0;
  string MyString; MyString="Select * from Customers";
  cn= new OdbcConnection("Driver={SQL Server};Server=mySQLServer;UID=sa;
  PWD=myPassword;Database=Northwind;");

  sampleCommand = new OdbcCommand(MyString);
- sampleCommand.Connection =cn; // Стыкуем с открытым соединением
- sampleReader = sampleCommand.ExecuteReader(); // Выполняем запрос на чтение данных
- sampleCountColumns = sampleReader.FieldCount; // Количество столбцов, полученных в результате работы запроса
  while(sampleReader.Read())
- {
- for(int i = 0;i<sampleCountColumns;i++) {// Получаем значение конкретного столбца из полученной строки
- Console.Write("{0,10}",sampleReader[i]); }
- Console.WriteLine();
- }

Использование ODBC в C# (.Net)

- ```
  OdbcConnection cn;
  OdbcCommand cmd;
  string MyString;

  MyString="Select * from Customers";

  cn= new OdbcConnection("Driver={SQL
  Server};Server=mySQLServer;UID=sa;
  PWD=myPassword;Database=Northwind;");

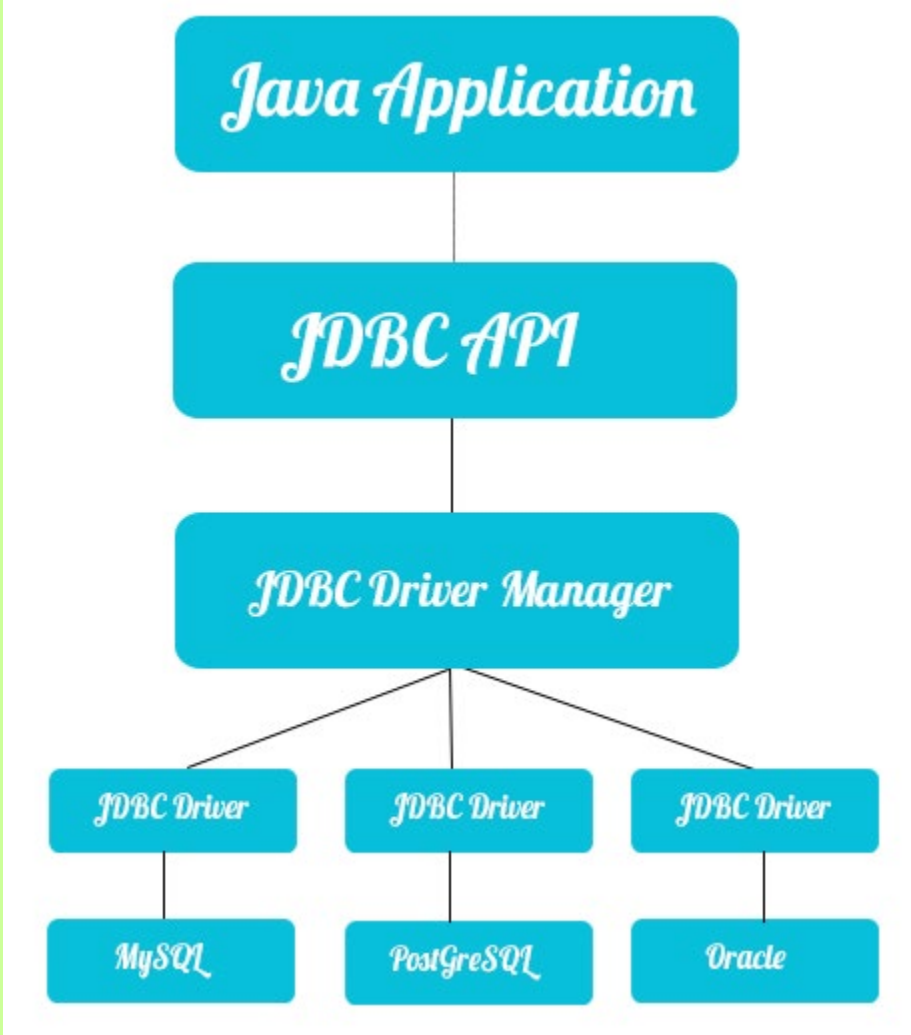
  cmd=new OdbcCommand(MyString,cn);
  cn.Open();

  MessageBox.Show("Connected");

  cn.Close();
  ```
  ```

JDBC

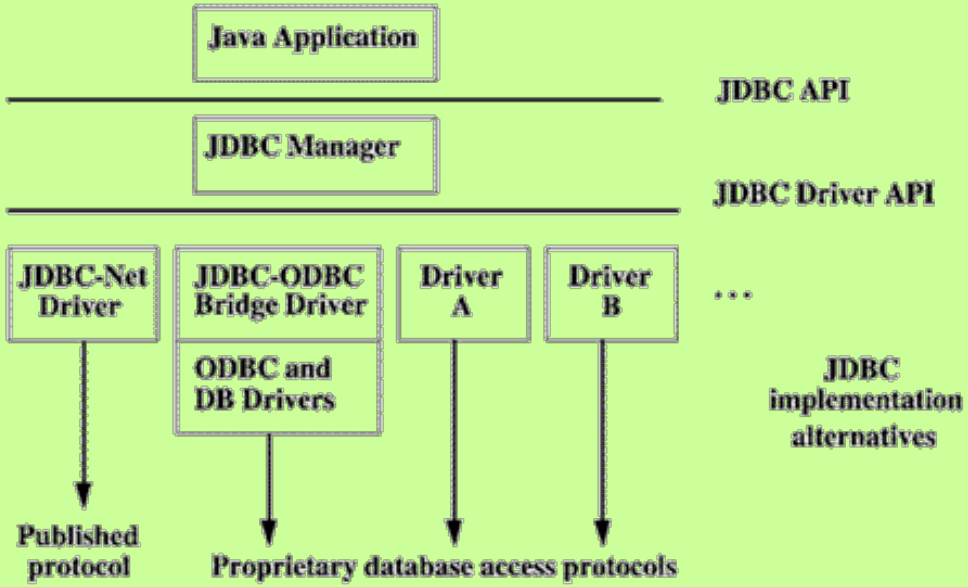
- **JDBC Driver** – (*Java DataBase Connectivity — соединение с базами данных на Java*) — платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.



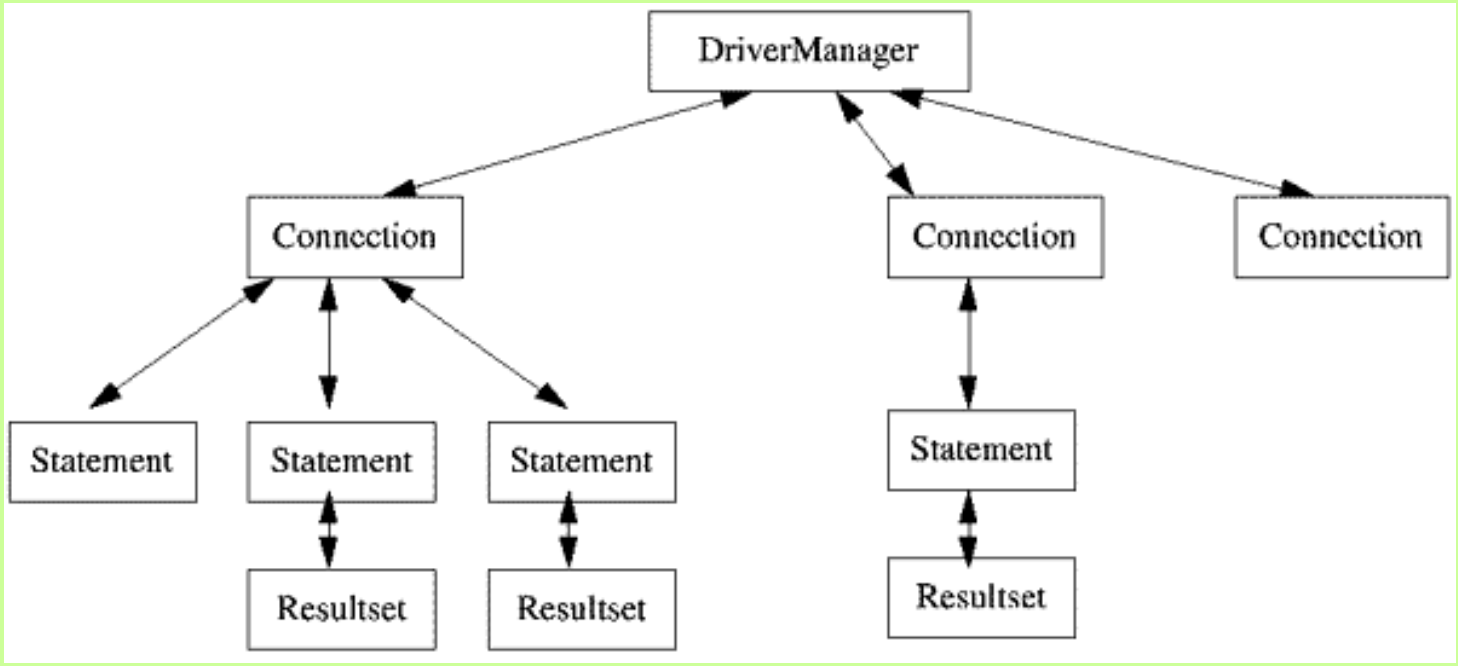
JDBC

- Стандарт JDBC первоначально разработанный, чтобы иметь доступ к БД по такой же архитектуре сначала включал в себя мост JDBC/ODBC Bridge, а дальше подключались ODBC драйвера. Но с течением времени научились делать и свои драйверы более прямые

- Уровни реализации JDBC API:



Основные интерфейсы JDBC



- http://www.nsc.ru/win/docs/db/sql/21.htm
- https://devcolibri.com/%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%B1%D0%B4-mysql-postgresql-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-jdbc-%D0%B4%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80%D0%B0/

JDBC пример

```
Connection con = DriverManager.getConnection ( "jdbc:wombat",
    "myloginname", "mypassword");
 // то же самое с использованием моста
// JDBC-ODBC
// Connection con =
// DriverManager.getConnection (
// "jdbc:odbc:wombat", "myloginname",
// "mypassword");
Statement stmt = con.createStatement();
ResultSet res = stmt.executeQuery("SELECT StringColumn,IntColumn
    FROM MyTable");
while (res.next())
{
String sCol = res.getString("StringColumn");
int iCol = res.getInt("IntColumn"); ... // обработка данных
}
```

ADO.NET

- ADO.NET (ActiveX Data Object для .**NET**) — это набор классов, предоставляющих службы доступа к данным программистам, которые используют платформу .NET Framework. ADO.NET имеет богатый набор компонентов для создания распределенных приложений, совместно использующих данные. Это неотъемлемая часть платформы .NET Framework, которая предоставляет доступ к реляционным данным, XML-данным и данным приложений.

Архитектура ADO.NET

Подключенные классы

Object	SQL Server	OLE DB	ODBC
Connection	SqlConnection	OleDbConnection	OdbcConnection
Command	SqlCommand	OleDbCommand	OdbcCommand
Data reader	SqlDataReader	OleDbDataReader	OdbcDataReader
Data adapter	SqlDataAdapter	OleDbDataAdapter	OdbcDataAdapter

Пример MySQL C# ADO.NET

```
using MySql.Data.MySqlClient;
private DataTable GetComments() {
DataTable dt = new DataTable();
MySqlConnectionStringBuilder mysqlCSB;
mysqlCSB = new MySqlConnectionStringBuilder();
mysqlCSB.Server = "127.0.0.1";
mysqlCSB.Database = "mytest";
mysqlCSB.UserID = "root"; mysqlCSB.Password = "123";
string queryString = @"SELECT comment_author, comment_date, comment_content FROM
      wp_comments WHERE comment_date >= CURDATE()";
 using (MySqlConnection con = new MySqlConnection())
{
con.ConnectionString = mysqlCSB.ConnectionString;
 MySqlCommand com = new MySqlCommand(queryString, con);
try {
con.Open();
using(MySqlDataReader dr = com.ExecuteReader())
{ if (dr.HasRows) { dt.Load(dr); }
}    }
 catch (Exception ex)
{ MessageBox.Show(ex.Message); } }
return dt;
 }
dataGridView1.DataSource = GetComments();
```

Пример 2 MySQL C# ADO.NET

```
using MySql.Data.MySqlClient;
private DataTable GetComments() {
DataTable dt = new DataTable();
MySqlConnectionStringBuilder mysqlCSB;
mysqlCSB = new MySqlConnectionStringBuilder();
mysqlCSB.Server = "127.0.0.1";
mysqlCSB.Database = "mytest";
mysqlCSB.UserID = "root"; mysqlCSB.Password = "123";
string queryString = @"SELECT comment_author, comment_date, comment_content FROM
      wp_comments WHERE comment_date >= CURDATE()";
 using (MySqlConnection con = new MySqlConnection())
{
con.ConnectionString = mysqlCSB.ConnectionString;
 MySqlCommand com = new MySqlCommand(queryString, con);
try {
con.Open();
using(MySqlDataReader reader  = com.ExecuteReader())
{while (reader.Read())
            {  Console.WriteLine("\t{0}\t{1}\t{2}",  reader[0], reader[1], reader[2]);        }
            reader.Close();
}    }
 catch (Exception ex) { MessageBox.Show(ex.Message); } }
return dt;
 }
```

ORM

- **ORM (Object-Relational Mapping)** – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.
- Библиотеки ORM существуют для самых разных языков программирования. В общих чертах, технология ORM позволяет проектировать работу с данными в терминах классов, а не таблиц данных. Она позволяет преобразовывать классы в данные, пригодные для хранения в базе данных, причем схему преобразования определяет сам разработчик. Кроме того, ORM предоставляет простой API- интерфейс для CRUD-операций над данными. Благодаря технологии ORM нет необходимости писать SQL-код для взаимодействия с локальной базой данных

Entity Framework

- **Entity Framework** — это инструмент, упрощающий сопоставление объектов в программном обеспечении с таблицами и столбцами реляционной базы данных. **Entity Framework** (EF) — это ORM-фреймворк с открытым исходным кодом для ADO.NET, который является частью .NET **Framework**.

Hibernate

- **JPA** – это технология, обеспечивающая объектно-реляционное отображение простых JAVA объектов и предоставляющая API для сохранения, получения и управления такими объектами.
- **Hibernate** — самая популярная реализация спецификации JPA (), предназначенная для решения задач объектно-реляционного отображения (ORM). Распространяется свободно на условиях GNU Lesser General Public License.

ORM (Object-Relational Mapping)

ORM

- **ORM (Object-Relational Mapping)** – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.
- Библиотеки ORM существуют для самых разных языков программирования. В общих чертах, технология ORM позволяет проектировать работу с данными в терминах классов, а не таблиц данных. Она позволяет преобразовывать классы в данные, пригодные для хранения в базе данных, причем схему преобразования определяет сам разработчик. Кроме того, ORM предоставляет простой API- интерфейс для CRUD-операций над данными. Благодаря технологии ORM нет необходимости писать SQL-код для взаимодействия с локальной базой данных

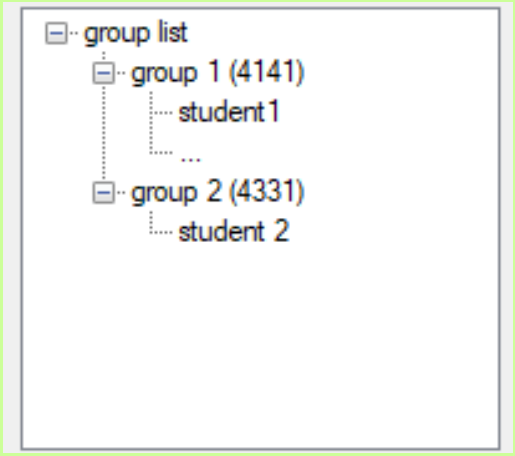
Entity Framework

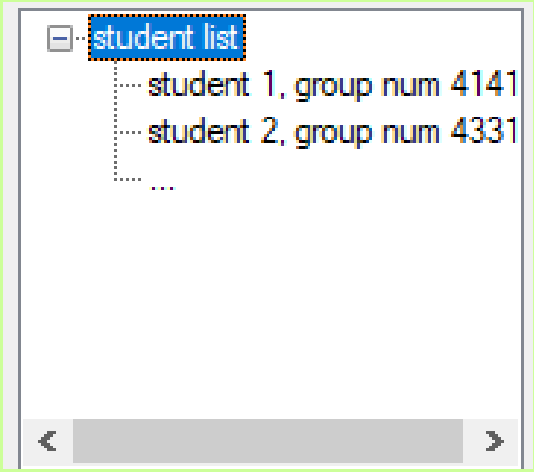
- **Entity Framework** — это инструмент, упрощающий сопоставление объектов в программном обеспечении с таблицами и столбцами реляционной базы данных. **Entity Framework** (EF) — это ORM-фреймворк с открытым исходным кодом для ADO.NET, который является частью .NET **Framework**.

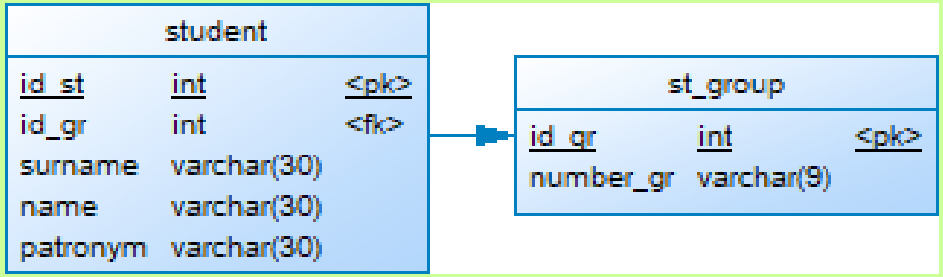
Способы взаимодействия Entity Framework с БД

- **Database first**: Entity Framework создает набор классов, которые отражают модель конкретной базы данных
- **Model first**: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.
- **Code first**: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы

ООП и Реляционная БД







33