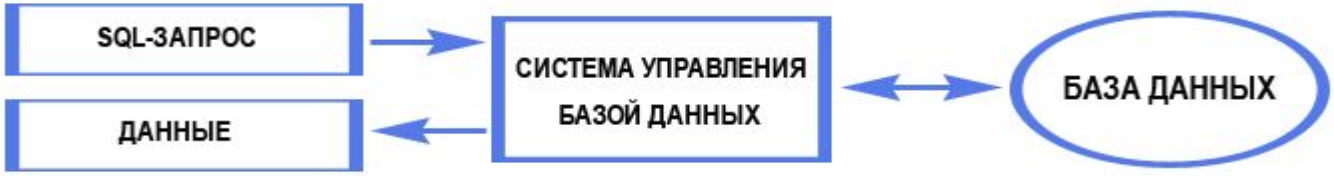
Работа с базой данных в языке JAVA

Схема работы с БД



База данных - набор сведений, хранящихся некоторым упорядоченным способом.

Система управления базами данных - это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных.

SQL - язык структурированных запросов, основной задачей которого является предоставление способа считывания и записи информации в базу данных.

Реляционная БД

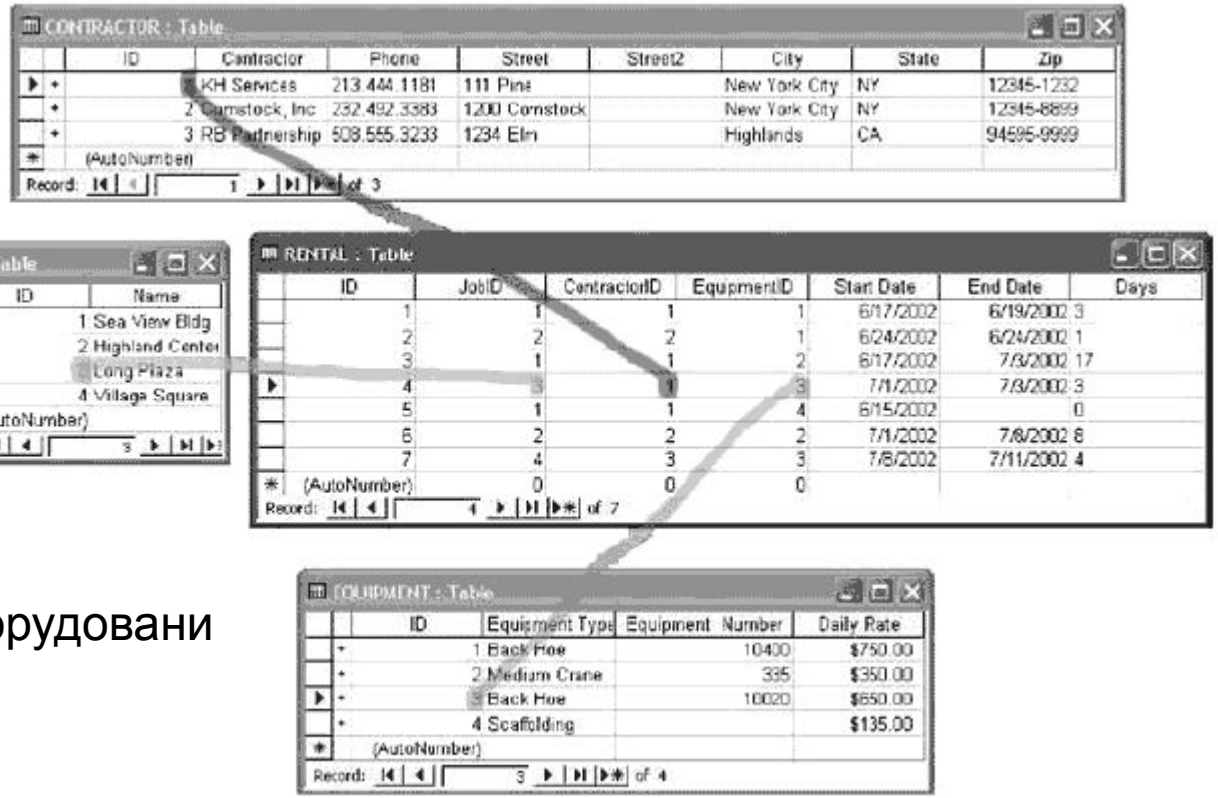
Реляционная База данных (database) – это совокупность связанных между собой таблиц.

Подрядчики

Проекты

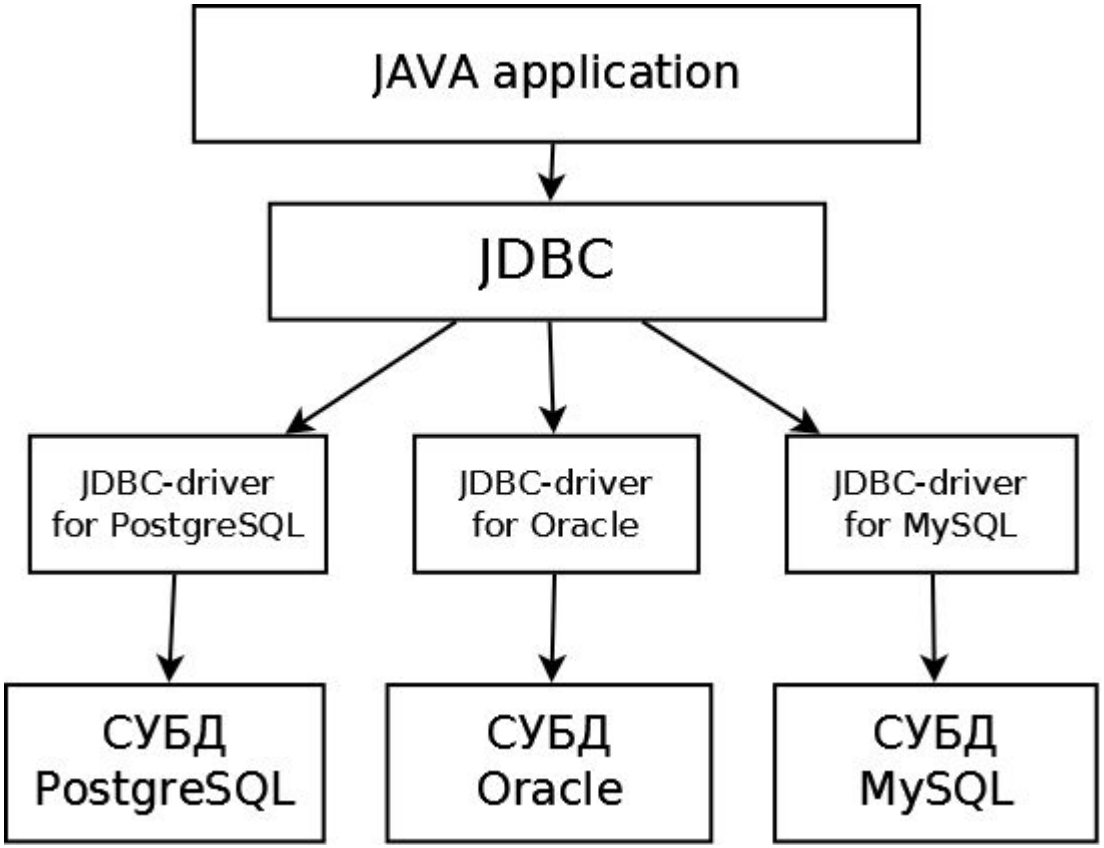
Прокат

Оборудование



Архитектура JDBC

JDBC(Java DataBase Connectivity) - это интерфейс для организации доступа Java-приложениям к базам данных.



Работа с БД MySQL в Java

Для того, чтобы начать работу с БД нужно установить сервер БД и драйвер. В нижеприведенных примерах будет рассматриваться работа с сервером баз данных MySQL.

Сервер можно взять отсюда: http://dev.mysql.com/downloads.

Скачать к нему драйвер можно здесь: http://dev.mysql.com/downloads.

Пакет java.sql - содержит классы и интерфейсы работы с БД:

DriverManager - управление JDBC-драйверами;

Connection - выбирает методы для создания сеанса связи с базой данных;

Statement - методы для передачи SQL-запроса базе данных;

ResultSet - методы для обработки результата обращения к базе данных;

DatabaseMetaData - устанавливает методы для получения сведений о базе данных, с которой работает

Основные типы данных, используемые в базе данных MySQL

Целые числа

TINYINT диапазон от -128 до 127

SMALLINT Диапазон от -32 768 до 32 767

MEDIUMINT Диапазон от -8 388 608 до 8 388 607

INT Диапазон от -2 147 483 648 до 2 147 483 647

BIGINT Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

Дробные числа

ИмяТипа[(length, decimals)] [UNSIGNED] Здесь

length - количество знакомест (ширина поля), в которых будет размещено дробное число при его передаче.

decimals - количество знаков после десятичной точки, которые будут учитываться.

UNSIGNED - задает беззнаковые числа.

FLOAT Число с плавающей точкой небольшой точности.

DOUBLE (REAL) Число с плавающей точкой двойной точности.

DECIMAL (NUMERIC) Дробное число, хранящееся в виде строки.

Строки

VARCHAR не более 255 символов.

TEXT не более 65 535 символов.

MEDIUMTEXT не более 16 777 215 символов.

LONGTEXT не более 4 294 967 295 символов.

Бинарные данные

Бинарные данные - это почти то же самое, что и данные в формате TEXT, но только при поиске в них учитывается регистр символов.

TINYBLOB не более 255 символов.

BLOBM не более 65 535 символов.

MEDIUMBLOB не более 16 777 215 символов.

LONGBLOB не более 4 294 967 295 символов.

Дата и время

DATE Дата в формате ГГГГ-ММ-ДД

TIME Время в формате ЧЧ:ММ:СС

DATETIME Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС

Последовательность работы с БД

- Загрузка драйвера и установка соединения с БД
- Создание БД (одноразово)
- Создание структуры таблиц (одноразово)
- Запись в БД
- Чтение из БД

Загрузка драйвера и установка соединения с БД

Загрузка драйвера осуществляется с помощью метода Class.forName()
Class.forName("com.mysql.jdbc.Driver");

Соединение с БД выполняет метод getConnection()

Connection con = DriverManager.getConnection(url, name, password);

url = "jdbc:mysql://localhost/mysql";

По умолчанию для MySQL имя - root, пароль - пустая строка.

Пример загрузки драйвера и установки соединения с БД

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class CreatingConnection {
    public static void main(String[] args) {
    try { Class.forName("com.mysql.jdbc.Driver");
    System.out.println(«Драйвер загружен успешно!");
//у MySQL обязательно есть системная база, к ней и будем создавать соединение.
    String url = "jdbc:mysql://localhost/mysql";
    String name = "root";
    String password = "";
    try { Connection con = DriverManager.getConnection(url, name, password);
    System.out.println("Соединение установлено.");
    con.close();
    System.out.println("Соединение разорвано."); }
    catch (SQLException e) { e.printStackTrace(); }  }
    catch (ClassNotFoundException e) { e.printStackTrace(); }
 }}
```

Создание БД

Создание базы данным состоит из следующих этапов:

1. Подключение к базе данных
2. Создание sql запроса
3. Выполнение sql запроса

Формат запроса:

CREATE DATABASE [IF NOT EXISTS] database_name [CHARACTER SET charset] [COLLATE collation]

database_name - название создаваемой базы данных

IF NOT EXISTS - указывает что создать базу данных следует только в том случае если её еще не существует. Если попытаться создать базу данных, которая уже существует, не указав параметр *IF NOT EXISTS*, то результатом выполнения запроса будет ошибка.

CHARACTER SET charset - указывает какая кодировка будет использоваться в создаваемой базе данных. Если не указывать этот параметр база данных будет создана в кодировке используемой по умолчанию.

COLLATE collation - указывает порядок сортировки при выборе данных из БД для заданного типа *CHARACTER SET*. Значения допустимые для параметра *collation* зависят от выбранной кодировки.

Пример:

CREATE DATABASE **bookstore** CHARACTER SET **utf8** COLLATE **utf8_general_ci**

http://www.aroundweb.ru/mysql/sql_language.htm

Выполнение запроса на создание БД

1. Создание объекта для передачи SQL-запросов (S**tatement)**

Statement s = connection.createStatement();

2. Выполнение SQL-запроса

ResultSet rs = s.executeQuery(createDatabaseQyery);

Возвращает таблицу

int m = s.executeUpdate(createDatabaseQyery);

возвращает количество измененных строк

Пример создания БД

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class CreatingDatabase { //Так мы создаем базу данных:
private final static String createDatabaseQyery = "CREATE DATABASE bookstore CHARACTER SET utf8
    COLLATE utf8_general_ci";
public static void main(String[] args) {
Connection connection = null;
Statement statement = null;
try { //Загружаем драйвер и подключаемся к БД
Class.forName("com.mysql.jdbc.Driver");
String url = "jdbc:mysql://localhost/mysql";
connection = DriverManager.getConnection(url, "root", "");
statement = connection.createStatement();
//Обратите внимание, что создаем базу с помощью executeUpdate().
    statement.executeUpdate(createDatabaseQyery); }
catch (Exception e) { e.printStackTrace(); }
finally { //закрываем теперь все
if (statement != null) { try { statement.close(); }
catch (SQLException e) { e.printStackTrace(); } }
if (connection != null) { try { connection.close(); }
catch (SQLException e) { e.printStackTrace(); } } } }}
```

Создание таблицы БД

Формат запроса:

CREATE TABLE [IF NOT EXISTS] table_name (create_definition)

IF NOT EXISTS - чтобы не возникала ошибка, если указанная таблица уже существует

table_name - название создаваемой таблицы

create_definition - описывает структуру таблицы (названия и типы полей, ключи, индексы и т.д.)

Формат строки с описанием полей таблицы:

field_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT] [PRIMARY KEY] [INDEX] [UNIQUE] [FULLTEXT]

field_name - задает название поля

type - задает тип данных для поля

не обязательные параметры:

NOT NULL | NULL - указывает на допустимость значения NULL для данного поля.

DEFAULT default_value - задает значение по умолчанию равное значению *default_value* для данного поля.

AUTO_INCREMENT - указывает что при каждом добавлении новой записи в таблицу значение для поля с типом *AUTO_INCREMENT* будет увеличиваться на единицу. Параметром *AUTO_INCREMENT* могут обладать только поля с целочисленным типом данных но не больше одного поля с параметром *AUTO_INCREMENT* на таблицу.

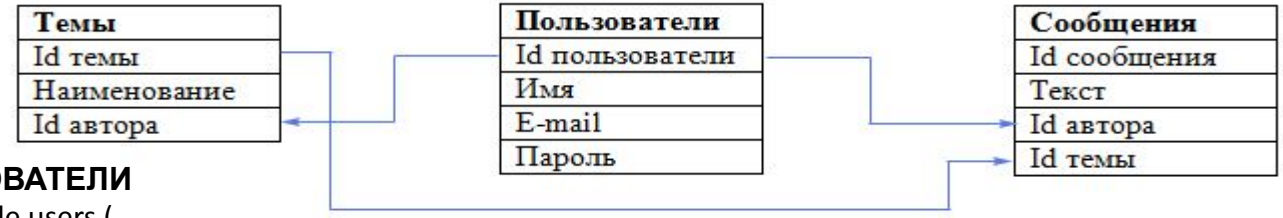
PRIMARY KEY - первичный ключ таблицы. Первичным ключом могут быть значения как одно поля, так и нескольких. Значения первичного ключа должны быть уникальны, это позволяет однозначно идентифицировать каждую запись в таблице. Поиск данных по первичному ключу происходит гораздо быстрее чем по другим полям.

INDEX - указывает на то что данное поле будет иметь индекс. Поиск по полям с индексом происходит быстрее чем по полям без индекса. Использование индексов увеличивает размер базы данных.

UNIQUE - указывает на то что все значения данного поля будут уникальными. Попытка записать не уникальное значение в данное поле будет приводить к ошибке. Поиск по уникальным поля происходит быстрее чем для полей с неуникальными данными.

FULLTEXT - указывает на то что к данным хранящимся в данном поле будет возможно применить полнотекстовый поиск.

Пример описания SQL-запроса на создание таблицы



ПОЛЬЗОВАТЕЛИ

```
create table users (
        id_user int (10) AUTO_INCREMENT,
        name varchar(20) NOT NULL,
        email varchar(50) NOT NULL,
        password varchar(15) NOT NULL,
        PRIMARY KEY (id_user)
        );
```

ТЕМЫ

```
        create table topics (
        id_topic int (10) AUTO_INCREMENT,
        topic_name varchar(100) NOT NULL,
        id_author int (10) NOT NULL,
        PRIMARY KEY (id_topic),
        FOREIGN KEY (id_author) REFERENCES users (id_user)
        );
```

СООБЩЕНИЯ

```
create table posts (
        id_post int (10) AUTO_INCREMENT,
        message text NOT NULL,
        id_author int (10) NOT NULL,
        id_topic int (10) NOT NULL,
        PRIMARY KEY (id_post),
        FOREIGN KEY (id_author) REFERENCES users (id_user),
        FOREIGN KEY (id_topic) REFERENCES topics (id_topic)
        );
```

FOREIGN KEY (имя_столбца_которое_является_внешним_ключом) REFERENCES имя_таблицы_родителя (имя_столбца_родителя);

Пример создания таблицы БД

Ключ	Название книги	Комментарии	Цена	Автор

```
public class CreatingTable {
private final static String createTableQuery = "CREATE TABLE `books` (" + " `id` int(11) NOT NULL auto_increment," + "
      `title` varchar(50) default NULL," + " `comment` varchar(100) default NULL," + " `price` double default NULL," + "
      `author` varchar(50) default NULL," + " PRIMARY KEY (`id`)" + ") ENGINE=InnoDB DEFAULT CHARSET=utf8;";
// задается движок БД и кодировка по умолчанию
public static void main(String[] args) {
Connection connection = null; Statement statement = null;
try { Class.forName("com.mysql.jdbc.Driver");
//Подключаемся к новосозданной базе.
String url = "jdbc:mysql://localhost/bookstore" + "?autoReconnect=true&useUnicode=true&characterEncoding=utf8";
connection = DriverManager.getConnection(url, "root", "");
statement = connection.createStatement();
statement.executeUpdate(createTableQuery); }
catch (Exception e) { e.printStackTrace(); }
finally { //позакрываем теперь все
 if (statement != null) { try {
statement.close(); }
catch (SQLException e) { e.printStackTrace(); } }
if (connection != null) {
try { connection.close(); }
catch (SQLException e) { e.printStackTrace(); } } } }}
```

Запрос на запись и удаление данных

1. Занесение данных во все поля таблицы:

INSERT INTO имя_таблицы VALUES ('значение_первого_столбца','значение_второго_столбца', ..., 'значение_последнего_столбца');

“INSERT INTO books VALUES(‘1’, ‘Евгений Онегин’, ‘Роман’, ‘300’,’А.С.Пушкин’), (‘2’,’Идиот’,’ Роман’, ‘400’,’Ф.М.Достоевский’)”

2. Занесение данных в некоторые поля таблицы:

INSERT INTO имя_таблицы ('имя_столбца', 'имя_столбца')
VALUES ('значение_первого_столбца','значение_второго_столбца');

“INSERT INTO books
(id, title, author) VALUES (‘3’,’Муму’,’И.С.Тургенев’)”

3. Очистка все таблицы

DELETE from имя таблицы

DELETE from books

4. Удаление строк таблицы по условию

DELETE from имя таблицы WHERE условие

DELETE from books WHERE id=1

Запрос на выборку данных из таблицы

SELECT *имя_столбца, ...*

FROM *имя_табл, ...*

[WHERE *условие*] условие для "отсеивания" не нужных записей

[GROUP BY *имя_столбца, ...*] группировка полученных результатов по какому-нибудь столбцу

[ORDER BY *имя_столбца, ...*] сортировка результатов ответа

[HAVING *условие*]; используется для фильтрации результата GROUP BY по заданным условиям, но только на другой стадии формирования ответа.

1. Выборка всех столбцов таблицы

select * from books

2. Выборка отдельных столбцов таблицы

Select **title, author, price** from books

3. Выборка по условию

select * from books WHERE **price < 500**

Группировка результатов запроса

Таблица **workers**

id	name	age	salary
1	Дима	23	100
2	Петя	23	200
3	Вася	23	300
4	Коля	24	1000
5	Иван	24	2000
6	Кирилл	25	1000

```
SELECT age, SUM(salary) as sum FROM workers
WHERE id>=2 GROUP BY age
```

age возраст	sum сумма
23	500
24	3000
25	1000

```
SELECT age, MAX(salary) as max FROM workers GROUP BY age
```

age возраст	max максимальная зарплата
23	300
24	2000
25	1000

Обработка результата запроса

Результат выполнения запроса **SELECT** формируется в объекте ResultSet (таблица с названиями столбцов). Курсор указывает на текущую строку таблицы. Для перемещения курсора используется методы **next** (возвращает false , если больше нет строк в ResultSet объекте) и **absolute**(номер строки).

Для чтения ячейки таблицы используются методы Get:

```
String title = rs.getString("title");
String title = rs.getString(2);
double price = rs.getDouble("price");
```

Пример работы с БД

```
public BookStore() {
String url = "jdbc:mysql://localhost/bookstore" +
"?autoReconnect=true&useUnicode=true&characterEncoding=utf8";
String name = "root";
String password = "";
try {
con = DriverManager.getConnection(url, name, password);
System.out.println("Connected.");
Statement st = con.createStatement();
String query1 = "INSERT INTO books VALUES  ('1','Евгений Онегин','роман','300','А.С.Пушкин')," +
"( '2' , 'Идиот' , 'Роман' , '400' , 'Ф.М.Достоевский' )";
String query3 = "INSERT INTO books (id,title,author)VALUES ('3','Муму','И.С.Тургенев')";
String query2 = "DELETE from books";
String query = "select * from books";
int rs2 = st.executeUpdate(query2);
int rs1 = st.executeUpdate(query1);
int rs3 = st.executeUpdate(query3);
System.out.println(rs1);
ResultSet rs = st.executeQuery(query);
printResults(rs);
System.out.println("Disconnected.");
con.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
```

Продолжение примера

```
private void printResults(ResultSet rs) throws SQLException {
String author, title, comment;
double price;
while (rs.next()) {
author = rs.getString("author");
title = rs.getString("title");
comment = rs.getString("comment");
price = rs.getDouble("price");
System.out.println("*******************************");
System.out.println("Author: " + author);
System.out.println("Title: " + title);
System.out.println("Price: " + price);
System.out.println("comment: " + comment);
System.out.println("*******************************");
}
}

public static void main(String[] args) {
try {
Class.forName("com.mysql.jdbc.Driver");
System.out.println("Driver loading success!");
} catch (ClassNotFoundException e) {
e.printStackTrace();
}
BookStore bookStore = new BookStore();
}
}
```