Казанский федеральный УНИВЕРСИТЕТ

Казанский федеральный УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА
информационных технологий
и интеллектуальных систем

04. Работа с СУБД

Лекции по информатике
для студентов второго курса Высшей школы ИТИС КФУ
2020

Ференец Александр Андреевич
старший преподаватель кафедры программной инженерии

С использованием материалов
к. т. н., доцента кафедры программной инженерии Абрамского М.М.

aferenets@it.kfu.ru

V. 1.0

БАЗА ДАННЫХ. Определение

База данных – ?

БАЗА ДАННЫХ. Определение

База данных — это хранимая упорядоченная структурированная информация.

- Определённая структура или способ упорядочивания
- Хранение: кратковременное или длительное

БД != СУБД (потом поясним)

БАЗА ДАННЫХ. Определение

База данных — это хранимый упорядоченный набор структурированной информации.

- Определённая структура или способ упорядочивания
- Хранение: кратковременное или длительное

Является ли БД?

1. XML-документ
2. HTML-документ
3. Файл без расширения, в котором лежит JSON/YAML/ini-содержимое
4. Массив/коллекция в программе
5. “Открытый” FileOutputStream
6. /dev/null
7. Флешка
8. Глиняная табличка
9. ОЗУ

ТИПЫ БД. Классификация по модели данных

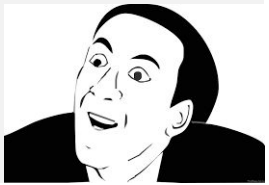
1. Ключ-значение
2. Иерархическая
3. Объектная (≈документная)
4. Реляционная
5. …

Можно эмулировать одну модель данных на основе другой.

Не зависит от способа хранения!

ТИПЫ БД. По распределённости

1. Централизованная
2. ~~Нецентрализованная~~ Распределённая



ТИПЫ БД. Распределённые БД

a. Неоднородная (heterogeneous distributed). Управляют несколько СУБД.
b. Однородная (homogeneous distributed). Управляет одна СУБД.
c. Фрагментированная (partitioned database). Вертикально или горизонтально.
d. Тиражированная (replicated database). Репликация!

ТИПЫ БД. Тиражированная БД

Репликация – процесс копирования определённого объекта.

Репликации равны? Варианты:

1. В любой момент времени одно и то же. При изменении одного узла, сразу меняется и другой.
2. Копирование при определённых условиях (например, раз в неделю).

Зачем нужны?

1. Ускорение доступа в другой системе (программной или физической).
2. Бекап.
3. …

Типы БД. Какие ещё интересности?

- Временная (Temporal)
- Пространственная (Spatial)
- Циклическая (Round-robin)

Типы БД. Какие ещё интересности?

- ❑ Формат хранения
- ❑ Скорость работы
- ❑ Механизмы репликации
- ❑ Интерфейс доступа

- ★ БД магазина с множеством вариаций товара, регионами
- ★ Базы сообщений/логов (мнооооого однородных данных)
- ★ Специфические данные (элементы карт)

Множество задач с иногда нетривиальными решениями. Отсюда специальность – ***Архитектор баз данных.***

ПРОБЛЕМАТИКА БД. О чём должен думать архитектор БД?

1. Функционал. Способность чтения, записи, изменения, удаления (CRUD) и другие операции (поиск, проверка, триггеры).
2. Целостность БД (integrity). Типы:
    i. Соответствие множеству значений;
    ii. Соответствие отношению.
3. Абстрагированность.

ТИПЫ БД. Big Data

А что такое “МНОГО”?

Volume – **V**elocity – **V**ariety

1 Гб – 1 Тб – 1Тб – 1Йиб
10 тыс. записей – 1 млн записей – 1 млрд записей

Изображение: https://eu.alienwarearena.com/ucf/show/1917995/boards/in-game-media-2/Image/starcraft-zerg-rush

ТИПЫ БД. Big Data

СУБД. Определение

Система Управления Базами Данных – набор инструментов, позволяющих совершать операции с базой данных. **Database Management System (DBMS)**

Типы соответствуют типам БД. То есть ключ-значение, реляционная СУБД и т.д.

СУБД. Ответственность и форматы

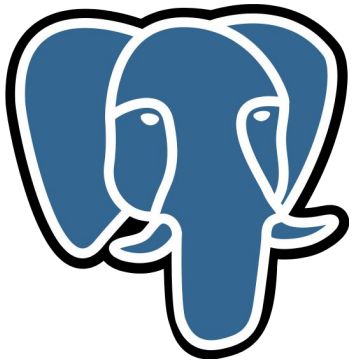
1. Create Read Update Delete данных
2. Оптимизация доступа (кэш, репликация и проч.)
3. Предоставление доступа
4. Бэкапы
5. Журналирование операций
6. “Удобные” интерфейсы: языки запросов, графические и программные интерфейсы

- Графический/консольный интерфейс
- Библиотека
- Сетевой интерфейс

а по какому протоколу общаются клиент и сервер?

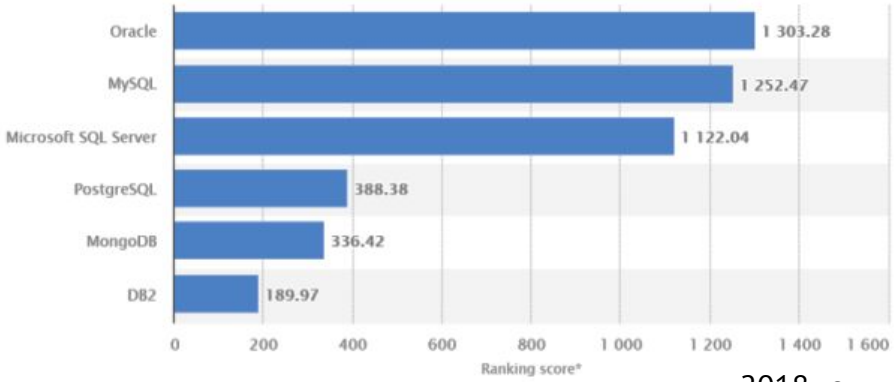
СУБД. Популярные

MySQL



mongoDB

SQLite



2018 год

СУБД. Клиенты

- Веб-приложение
- Встроенный в ИДЕ
- Прочие системные

СУБД != Клиент
СУБД

РЕЛЯЦИОННЫЕ БД. Введение

1969, Кодд

«БД – набор таблиц, связанных друг с другом»
"Relation" – отношение

Очень популярна! Логична для типичного способа мышления (субъективно).

Дисклеймер

Просьба не придираться к «ошибкам проектирования», они введены для упрощения. Проектировать БД хорошо вы учитесь на соответствующем предмете.

«использование имени группы в качестве ключа некорректно!»
«а можно вообще говоря было сделать еще одну таблицу»
«вообще-то ФИО – это три поля, а не одно»

РЕЛЯЦИОННЫЕ БД. Таблицы

- Таблица, отношение, реляция
- Кортеж, строка, объект
- Столбец, поле, атрибут

Student(s)

ID	Студент	Группа
12323	Еникеева	11-801
564325	Иванов	11-805
1234243	Хайруллина	11-802
23424	Шафеева	11-803
34535	Елин	11-804

PRIMARY KEY

- Уникальный набор полей, по которым можно однозначно идентифицировать строку
    - Части PK могут быть неуникальными.
      Но обычно ключ состоит из одного и уникального поля.

- Очень часто – одно целочисленное поле ID, специально для этого заведенное. С числом удобнее работать и оно занимает относительно немного места.
- Может быть целым числом, дробным числом, меткой времени, строкой и т.д.

- НЕ ПОРЯДКОВЫЙ НОМЕР! Но может генерироваться автоматически и последовательно.

Примеры?

FOREIGN KEY

- Набор полей, по которым можно однозначно идентифицировать ДРУГУЮ строчку в ДРУГОЙ таблице
- Фактически: *ссылка (копия) на primary key в другой таблице*

Примеры?

Теория:

PK должен быть выбран из полей таблицы.

страна + номер паспорта + дата рождения

Индустрия:

РЕЛЯЦИОННЫЕ БД. Теория и практика ключей

1. Таблицы с Foreign Key будут хранить копию всех полей из Primary Key
2. Надо помнить перечень полей в SQL-запросах (для JOIN, например)
3. Как быть с URL? /user?country=RU&passport=123456&birthdate=398791231

- Когда решаем задачки на нормальные формы – да, там PK – набор полей
- Но когда делаем приложение – PK – уникальное целочисленное поле

РЕЛЯЦИОННЫЕ БД. PK в web

vk.com/id692473

music.yandex.ru/album/1144155/track/10570957

www.mvideo.ru/products/televizor-samsung-ue55ru7470u-10021197

www.youtube.com/watch?v=dQw4w9WgXcQ

РЕЛЯЦИОННЫЕ БД. Пример таблиц с FK

Students

Студент (PK)	Группа (FK)
Нуриева	11-802
Сафин	11-802
Ибрагимова	11-803
Шигабутдинов	11-805
Берендакова	11-804

Groups

Группа (PK)	Куратор
11-802	…
11-803	…
11-804	…
11-805	…

РЕЛЯЦИОННЫЕ БД. Проблемы. Избыточность

Classes

Студент	Предмет	Группа	Факультет	Начало
Давлетгареева	Информатика	11-801	ИТИС	8:30
Азин	Мат.анализ	11-802	ИТИС	8:30
Гимазов	Мат.анализ	11-805	ИТИС	11:50
Гимазов	Информатика	11-805	ИТИС	10:10
Амирханова	Алгебра	11-807	ИТИС	10:10

- Дублируются связи
  - "студент-группа",
  - "группа-факультет"
- Предметы связаны с группами, их не нужно писать для каждого студента
- … !

РЕЛЯЦИОННЫЕ БД. Проблемы. Избыточность

- Корректность, логичность, адекватность БД
  - "Если расформировывают группу, куда девать связанных с ней студентов?"
  - Если я составлю новую строчку из значений, которые есть в других строках, будет ли она корректной?

Соблюдение целостности –
ключевое преимущество реляционных БД!

РЕЛЯЦИОННЫЕ БД. Нормальные формы

- Нужны для устранения наиболее частых ошибок проектирования БД (ради ее целостности).
- Один из самых популярных вопросов на собеседованиях.
- Разберем некоторые из них (остальные на курсе БД).

РЕЛЯЦИОННЫЕ БД. 1НФ

- В одной ячейке одно значение
- 1НФ требует атомарности данных

Хотя на практике не всегда

РЕЛЯЦИОННЫЕ БД. 1НФ. Пример

Не соответствует:

Студент	Предмет
Гильмуллина	Информатика, АиСД
Билалов	АиСД

РЕЛЯЦИОННЫЕ БД. 1НФ. Пример

Не соответствует:

Студент	Предмет
Гильмуллина	Информатика, АиСД
Билалов	АиСД

Соответствует:

Студент	Предмет
Гильмуллина	Информатика
Гильмуллина	АиСД
Билалов	АиСД

РЕЛЯЦИОННЫЕ БД. Функциональная зависимость

B зависит от A (A определяет B) (A -> B)

«Если два объекта совпадают по A, то они совпадают и по B»

Пример:

Если у студентов одна и та же группа, то у них совпадает и факультет. Из того, что запись характеризуется определённой группой следует, что она характеризуется определённым факультетом.

Значит, есть функциональная зависимость
«Группа -> Факультет»

РЕЛЯЦИОННЫЕ БД. 2НФ

1НФ
+
Каждый неключевой атрибут неприводимо зависит от потенциального ключа

В переводе на студенческий-лекционный:
"Не должно быть такого, чтобы какое-нибудь поле зависело однозначно только от части ключа"

Это и называется "неприводимость"

РЕЛЯЦИОННЫЕ БД. 2НФ. Пример

Не соответствует:

Успеваемости (кстати, придумайте название получше)

Студент	Предмет	Группа	Оценка
Коньков	Алгебра	11-801	95
Коньков	Информатика	11-801	100
Мирзаянова	Информатика	11-807	100
Мирзаянова	АиСД	11-807	97

РЕЛЯЦИОННЫЕ БД. 2НФ. Пример

Не соответствует:

Успеваемости (кстати, придумайте название получше)

Студент	Предмет	Группа	Оценка
Коньков	Алгебра	11-801	95
Коньков	Информатика	11-801	100
Мирзаянова	Информатика	11-807	100
Мирзаянова	АиСД	11-807	97

Группа зависит только от студента.

РЕЛЯЦИОННЫЕ БД. Подзаголовок

Соответствует:

Студент	Предмет	Оценка
Коньков	Алгебра	95
Коньков	Информатика	100
Мирзаянова	Информатика	100
Мирзаянова	АиСД	97

Студент	Группа
Коньков	11-801
Мирзаянова	11-807

РЕЛЯЦИОННЫЕ БД. 2НФ и в целом о зависимостях

Если в ключе только одно поле,
зависимости «только от части ключа» точно не будет.

РЕЛЯЦИОННЫЕ БД. Транзитивность

- A < B, B < C – значит, A < C
- A = B, B = C – значит A = C

Пример нетранзитивного отношения?

РЕЛЯЦИОННЫЕ БД. 3НФ

2НФ

+

Ни один неключевой атрибут не находится в транзитивной функциональной зависимости от потенциального ключа.

РЕЛЯЦИОННЫЕ БД. 3НФ. Пример

Не соответствует:

Студент	Группа	Факультет
Хаялеева	11-804	ИТИС
Харисов	11-805	ИТИС
Ермаков	11-804	ИТИС
Абрамский	953а	ВМК

РЕЛЯЦИОННЫЕ БД. 3НФ. Пример

Соответствует:

Студент	Группа
Хаялеева	11-804
Харисов	11-805
Ермаков	11-804
Абрамский	953а

Группа	Факультет
11-804	ИТИС
11-805	ИТИС
09-502	ВМК

Усиленная 3НФ

Простым языком:

Для любой зависимости X --> Y:
X должен быть потенциальным ключом.

РЕЛЯЦИОННЫЕ БД. НФ Бойса-Кодда. Пример

Не соответствует:

Аудитория	Этаж	Предмет	Время
1409	14	Информатика	8:30
1310	13	К/В	8:30
1408	14	Информатика	8:30
1409	14	ВССиТ	10:10

Почему?

Structured Query Language

Декларативный

А что такое “императивный”?

Основан на исчислении кортежей

Разные СУБД предлагают свои диалекты.

Хорошей идеей будет при изучении СУБД, узнавать, является ли конструкция общей или введена в диалекте.

SQL. Введение

- **Data Definition Language (DDL)**
  CREATE, ALTER, DROP
- **Data Manipulation Language (DML)**
  SELECT, INSERT, UPDATE, DELETE
- **Data Control Language (DCL)**
  GRANT, REVOKE
- **Transaction Control Language(TCL)**
  COMMIT, ROLLBACK

SQL. Примеры


```
CREATE TABLE "STUDENTS" (
   "ID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
   "NAME" VARCHAR(30) NOT NULL,
   "GROUP_NAME" VARCHAR(10)
);

...

SELECT NAME
FROM STUDENTS
WHERE GROUP_NAME = '11-801';
```

СУБД и Java. Универсальность?

Как соединить приложение и СУБД?
СУБД разные, неужели придется писать приложение для каждой?

- Подключиться к БД
  - в postgres: \connect database_name
  - в mysql: use database_name
- Сетевые протоколы разные.

А как же Java!? Write once, run everywhere!

Java DataBase Connectivity

java.sql.* + драйвер

Посредником между БД (СУБД) и приложением является драйвер. *Шаблон проектирования?*

Будьте аккуратны при выборе Драйвера! Не верьте Idea =)

JDBC. Первые строчки кода


```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.cj.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
            // The newInstance() call is a work around for some
            // broken Java implementations

            Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

Пример с https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-usagenotes-connect-drivermanager.html#connector-j-examples-connection-drivermanager

JDBC. Первые строчки кода


```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

Connection conn = null;
...
try {
    conn =
       DriverManager.getConnection("jdbc:mysql://localhost/test?" +
                                   "user=minty&password=greatsqldb");

    // Do something with the Connection

   ...
} catch (SQLException ex) {
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
```

Пример с https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-usagenotes-connect-drivermanager.html#connector-j-examples-connection-drivermanager

Объект для выполнения запроса

Узнаем его у Connection:

```
Statement stmt = conn.createStatement();
```

Для выполнения запросов вызываем методы

`executeUpdate` – для insert, create, update и DDL – всего того, что не возвращает данные из БД.

`executeQuery` – для select.

`execute` – для остального.

Результат executeQuery

Коллекция строк таблицы (но не связанная с Collections)

Нужные методы:

`boolean next()` – перемещает курсор на очередную доступную строку результата

`int getInt(…), String getString(…), Date getDate(…)`– для текущей строки вернуть значение в столбце по его имени или порядковому номеру (пишется вместо многоточий)

Какой шаблон проектирования?

JDBC. Statement, ResultSet. Пример


```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(
      "SELECT NAME FROM STUDENTS");
while (rs.next()) {
     System.out.println(rs.getString("NAME"));
}
stmt.close();
```

Классический сценарий обработки данных в веб-приложении с БД:

1. Пользователь вводит данные в форму

2. Запрос с параметрами приходит на сервер, где параметры используют:
String city = request.getParameter("city");

3. Очень часто параметры подставляются в SQL-запросы к БД:

```
stmt.exequteQuery("select * from products where city='" + city + "';");
```

В city передадут не город, а что-нибудь более интересненькое?

Злоумышленник (хакер) знает, что данные форм часто попадают напрямую в SQL-запрос.
«Я передам не значение, а кусочек SQL-кода, чтобы при подстановке его в запрос ошибки не было, но запрос делал бы то, что я хочу.

SQL-инъекция – вставка SQL-кода через формы/параметры для выполнения SQL-действия, необходимого злоумышленнику.

SQL. SQL Injection

Очевидно, где-то в приложении школы была строчка

```
"select * from students where name='" + name + "';"
```

Студента звали:

```
Роберт'); DROP TABLE students; --"
```

А, нет, там же скобка, значит не select, а insert:

```
"insert into students("name") values('" + name + "');"
```

При подстановке «имени» студента получим:

```
"insert into students("name") values('" + "Роберт'); DROP TABLE
students; --" + "';"
```

т.е.

```
insert into students("name") values('Роберт'); DROP TABLE students; --';
```

Защита от SQL Injection:

1. Проверка/удаление SQL-кода из всех параметров. Глупо, неуниверсально.
2. Экранирование управляющих конструкций.

Защита от SQL Injection:

1. Проверка/удаление SQL-кода из всех параметров. Глупо, неуниверсально.
2. Экранирование управляющих конструкций.

PreparedStatement – подготовленные выражения – параметризованные запросы, куда мы отдельно вставляем параметры, которые безопасно обрабатываются.

```
PreparedStatement ps = conn.prepareStatement( "insert into
students(\"name\") values(?)");

// Параметры нумеруются с 1
ps.setString(1, name); //setInt, setDate, ...
ps.executeUpdate();
```

Не только Java! В других языках аналогичный подход!

JDBC. Минусы библиотеки

1. SQLException,
2. Примитивность обработки,
3. Отсутствия понимания связи FK – PK в терминах ссылок (а зря).

Фреймворки, решающие эту проблему, мы разбираем в следующем семестре.

См. Библиотеку Spring JDBC Template

А ещё Hibernate ORM

Архитектура кода. Подбираемся к упрощению


```
CREATE TABLE "STUDENTS" (
    "ID" INTEGER NOT NULL PRIMARY KEY …,
    "NAME" VARCHAR(30) NOT NULL,
    "GROUP_NAME" VARCHAR(10)
);
```

≈

```
class Student {
     long id;
    String name;
    String group_name;
}
```

Архитектура кода. Подбираемся к упрощению


```
class Group {
    String name;
}

class Student {
    String name;
    Group group;
}
```

```
CREATE TABLE "GROUPS"
    "ID" INTEGER NOT NULL PRIMARY
KEY …,
    "NAME" VARCHAR(10) NOT NULL,
);

CREATE TABLE "STUDENTS" (
    "ID" INTEGER NOT NULL PRIMARY
KEY …,
    "NAME" VARCHAR(30) NOT NULL,
    "GROUP_ID" INTEGER REFERENCES
"GROUPS"(ID)
);
```

Архитектура кода. ORM

Object-relational mapping

- JPA, Hibernate,
- Django models,
- Ruby on Rails Active Records etc.
- Symfony ORM

```java
@Entity
@Table(name = "book")
public class Book{

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "id", unique = true, nullable = false)
  private int id;

  @Column(columnDefinition="text")
  private String description;

  @Column(unique = true)
  private String isbn;

  @ManyToOne(fetch = FetchType.EAGER)
  @JoinColumn(name = "publishing_house", nullable = false)
  private PublishingHouse publishingHouse;

  public Book(){}
  public Book(int id, String name) {
    this.id = id;
    this.name = name;
  }
```

Архитектура кода. ORM

Relational DB	ORM
Объявление таблицы	Объявление класса
Строка таблицы	Объект класса
Таблица students	Параметризованная коллекция объектов Collection <Student> students
Столбец, поле	Атрибут, поле

Архитектура кода. Выполнение операций с данными

Active Record

```
User u = new User(...);
u.save();

User u2 = User.find(4221);
u.getName();
```

VS

Data Mapper

```
User u = new User(...);
em.save(u);


User u2 = em.find(4221,
                  User.getClass());
u.getName();
```

em = Entity Manager