Лекция 1

Преподаватель: Дмитрий Барашев

Организационное

Начальный уровень.
Слайдов нет! Но может что-то изменится.

Практос: реляционные БД(SQL)
Лектос: теория БД и внутреннее устройство.

Оценки.

По практике можно заработать 0 или -1 балл(по сути это зачет и незачет)
На лекциях можно заработать $[0 \dots 5]$. Экзамен. Письменный. Будет какие-то задачи. Какие-то легкие, какие-то нет.
Оценки суммируются.
Получить 5 баллов очень сложно. Со слов преподавателя.

Список литературы.

Крис Дейт “Введение в базы данных” теоритические аспекты
Джефри Ульман, Г. Гарсиа-Молина, Уидом “Базы данных полный курс” железо, алгоритмы
Википедия — отстой.

Введение

Что хотим с данными делать:
- уметь обрабатывать
- поиск сортировка

Хотим добавить заголовок. Смысл.
Метаинформация:
- тип данных.
- название,
- ограничения

Определение БД можно сформулировать следующим образом: интегрированое самодокументированное хранилище информации.

Но поговорим лучше о применении БД
Мы будем использовать PostgreSQL

С точки зрения данных:
Это набор файлов(а может и нет)
Они находятся в некоторой единой структуре(физическая организация)
С точки зрения кода:
prog.exe программа, управляющая данными. Обычно это называют СУБД.

Модели данных:
Объектно-Ориентированная
Традиционная(таблички)
Иерархические(JSON) и как продолжение этой модели — графовая

Что делает СУБД?
1. Управляет размещением данных на диске
2. Выполняет запросы
3. Предоставить высокоуровневый язык для манипуляции с БД
OQL(object query language), SQL, MongoDB(монгодибишный язык)
Нельзя просто так взять и перенести из одной БД в другую данные на SQL.
Стандарты есть, но всем пофиг
4. Разделение прав доступа.(управление ролями). Представление данных для разных ролей
5. Инструменты для бэкапов
6. Управление конкурентным доступом
7. Восстановление после сбоев
Т.е. СУБД делает много. Какие виды СУБД?

Архитектура информационных систем с БД
1. Встроенные БД.Например SQLite
2. Клиент-серверная архитектура. Но есть проблема: очень сложно управлять. то есть допустим заведуем БД в городе и чтобы
реализовать какую-то фичу, придется обновится клиента на всех клиентских компьютерах. Если в БД используется где-нибудь в
госструктурах, то можно стрелятся. Даже если есть автообновление, деятельность ВСЕЙ БД приходится на какое-то время
останавливать.
Поэтому люди задумались, чтобы код SQL жил где-нибудь поближе к ядру СУБД.
3. Многоуровневая архитектура. то есть СУБД <—> сервер приложений и уже к этому серверу направляется пользовательский ввод.
Поэтому в этом случае мы просто обновим сервер приложений. Эта система доминирует на данный момент в Web-приложениях.

Практика

Цель: получение некоторого законченного проекта.

Виды реляционных СУБД:
1. Oracle
2. IBM DB2
3. MS SQL Server
4. MySQL / MariaDB
5. PostgreSQL

Почта: dmitry@barashev.net

Тут нам показывают как работать с бд
sqlitebrowser — простой GUI для sqlite

Нужно разбиться на команды по 3-4 человека и делать задания самостоятельно. Задания по практике будут командными. НО за это не
будут даваться баллы.

Зачёт/незачёт будет ставится по контрольным, которые будут уже индивидуальными.
Поэтому общаться с друг другом стоит.

К следующему занятию нужно научиться работать с PostgreSQL из терминала.

Водяная лекция 2

Реляционная модель

В ООП есть свои кирпичики — классы.
В реляционной модели это таблицы или отношения.

Основной элемент: отношение(relation)

Домен := некоторое множество значений(бесконечное или конечное)
:= тип данных + некоторые ограничения на этот тип
В домене определены некоторые операции. Сравнение, + * - /

Примеры доменов: BOOL, INT, TEXT, INT > 0 и так далее.

Рассмотрим следующую функцию:
$f : D_1 \times \ldots \times D_n -> true, false$
D_i — домен
$\{D_i\}_{i=1}^n$

f — отношение на доменах $D_1, \ldots D_n$.

Есть и другое определение:
Отношение: схема + тело.
Схема отношения — множество атрибутов. то есть $S = (a_i, D_i)$, a_i — имя, D_i — домен. Атрибут это пара.
$a_i == a_j$ => атрибуты равны.
Тело отношения — множество картежей. Кортеж $t = (a_i, v_i)$ $a_1, \ldots a_n$ —

последовательность имен атрибутов в S.
$v_i \in D_i$

таблица№1
Это графическое представление.

Можно также изобразить тоже самое в виде JSON:

```
{
  “a1”: 1,
  “a2”: true,
  “a3”: “abc”,
  “a4”: 3.14
}
```

Если предикат f возращает истину, то строчка есть в таблице, иначе ее нет.
Одинаковые строчки мы запрещаем.

Равенство картежей
$t = t'$ если $V_{a_i}(t) = V_{a_i}(t') \forall a_i \in S$

Чего нет:
1. Нет одинаковых кортежей(на практике всем пофиг)
2. Нет упорядоченности кортежей
3. Нет упорядоченности атрибутов
4. В кортежах значений: атомарны

Аспекты модели данных

1. Структура
2. Операции
3. Ограничения

Простейшие ограничения.
1. Домен — ограничение множества значений
2. Ограничения на атрибут внутри одной таблицы

3. NULL или NOT NULL.

Потенциальный ключ.
Для некоторого отношения R подмножство $K \subseteq S$ называется потенциальным ключом, если известно, что в любом экземпляре(конкретный набор строчек, то есть схема с конкретным телом) этого отношения
1. Если в любом экземпляре R для t и t' из $V_k(t) = V_k(t') => t = t'$
разных кортежей с одинаковым ключом не бывает
2. (Неизбыточность) не существует $K'K$ для K' выполняется 1.
Если выполняется только 1. то это суперключ.

Рассмотрим таблицу Студенты
направление | №паспорта | ФИО | №Студ | билета | №Группы

№паспорта и № студ. билета могут рассматриваться как потенциальный ключ.
По этим штукам можно однозначно определить студента.
Суперключ: например №паспорта + ФИО

Неизбыточность нужна, чтобы моделировать правила предметной области точно. не было студентов с совпадающими номерами паспортов

Потенциальный ключ (candidate key)
Первичный ключ(primary key)

Практика

stepik.org/lesson/54670

И тут какие-то обычные SQL запросы. Я сижу на первом ряду и хоть немного что-то вижу, как люди на это смотрят вообще.