2011

SQL

Structured Query Language

Образовательный центр Парка высоких технологий

Проектирование баз данных

- Нормальные формы – требования предъявляемые к структуре таблиц в теории реляционных баз данных.
- Процесс приведения базы данных к виду, соответствующему нормальным формам называется нормализацией.
- Приведение таблиц к нормальным формам осуществляется за счет декомпозиции, то есть разбиения таблиц на несколько.

Первая нормальная форма

Первая нормальная форма требует, чтообы каждый элемент таблицы имел только одно значение. Т.е. был «атомарным»

Не соответствует

Сотрудник	Телефон
Петровский И.А.	2302342
Павлюк П.П.	2302344, 34563456

Соответствует

Сотрудник	Телефон
Петровский И. А.	2302342
Павлюк П.П.	2302344
Павлюк П.П.	34563456

Вторая нормальная форма

- Вторая нормальная форма требует, чтобы все поля зависели от первичного ключа, а не от его какой-то части.
- Ключ может состоять из нескольких полей. Если какие-либо данные в таблице зависят только от одного поля ключа, для нормализации их надо выносить в отдельную таблицу.

Третья нормальная форма

- Третья нормальная форма требует, чтобы каждый неключевой элемент таблицы зависел только от ключа, и ни от каких других элементов таблицы

Четвертая нормальная форма

- Четвертая нормальная форма требует, чтообы у данных не было нетривиальных многозначных зависимостей.
- Например: Из наличия в базе одной строки не должно следовать наличие другой.

Создание базы данных

- Для создания базы данных используется команда следующего вида:
- CREATE DATABASE имя_базы;
- Например:
- CREATE DATABASE ListExpenses;

Создание таблиц

- Общий вид запроса:
- `CREATE TABLE имя_таблицы(имя_поля тип, имя_поля тип.....);`
- Например:
- `CREATE TABLE expenses(`
- `num int,`
- `paydate date,`
- `receiver int,`
- `value dec`
- `);`

Типы данных

- Типы данных чаще всего используемые в таблицах:
- `INT` или `INTEGER` — целочисленные данные
- `DEC` или `DECIMAL` — десятичные дробные величины
- `CHAR` или `CHARACTER` — строковый тип данных с фиксированной длинной
- `VARCHAR` — строковые данные переменной

Добавление данных в таблицу

- Общий вид запроса
- `INSERT INTO имя_таблицы VALUES(значение1, значение2, значение3...)`
- Например
- `INSERT INTO expenses VALUES(1,'2011-5-10', 1,2000.0);`

Выборка из базы

- Общий вид
- `SELECT имена_полей FROM имя таблицы;`
- Например, для таблицы расходов:
- `SELECT num,paydate,value,receiver FROM expenses;`
- После SELECT перечисляются интересующие поля. Если все – указывается звездочка

Фильтрация запросов

- Если необходимо получить только данные удовлетворяющие определенному условию к запросу добавляется условие с помощью ключевого слова WHERE
- `SELECT * FROM expenses WHERE value >= 20000;`

Операторы условий

- В условиях могут использоваться операторы больше, меньше, больше либо равно, меньше либо равно, которые записываются соответственно: `<` , `>` , `<=` , `>=`
- Оператор не равно `<>`
- Также можно использовать логические операторы `and` и `or`

Упорядочивание результатов

- Можно выстраивать результаты по возрастанию либо убыванию по одному или нескольким полям. Для этого используется ключевое слово ORDER BY
- `SELECT * FROM expenses ORDER BY value;`
- Если необходимо использовать несколько полей, они перечисляются через запятую.

Запрос к нескольким таблицам

- Запрос к нескольким таблицам сразу записывается следующим образом:
- `SELECT paydate,value,name FROM expenses,receivers WHERE receiver=receivers.num;`
- Все используемые таблицы перечисляются в разделе FROM, плюс, если таблицы связаны между собой, связь надо поместить в виде условия в раздел WHERE. Если этого не сделать результат будет содержать все

Изменения данных таблицы

- Общий вид запроса:
- `UPDATE имя_таблицы SET имя_столбца=значение, имя_столбца=значение... WHERE имя_столбца=значение;`
- условие `WHERE` не является обязательным для этой команды, но если его пропустить, будут изменены все записи таблицы.

Удаление записей таблицы

- Общий вид запроса:
- `DELETE FROM имя_таблицы WHERE имя_столбца=значение;`
- Как и в случае обновления, если не использовать `WHERE` будут удалены все поля таблицы.

Удаление таблиц

- Общий вид команды
- `DROP TABLE имя_таблицы`
- Аналогично удаляется база данных целиком:
- `DROP DATABASE имя_базы`

Псевдонимы таблиц

- Псевдонимы или «алиасы» таблиц позволяют сократить запись имен таблиц.
- Select paydate,value,name FROM expences, reseivers **rs** WHERE receiver=**rs**.num

Отбор уникальных строк

- Select **distinct** value,name FROM expences, reseivers rs WHERE receiver=rs.num

Агрегатные функции

- Count()
- Функция осуществляющая подсчет строк
- Select count(*)from expenses
- В качестве аргумента можно также указать имя поля, в этом случае будет подсчитано количество непустых значений
- Count(distinct имя_поля) считает кол-во уникальных значений

Агрегатные функции

- min(имя_поля)
- Max(имя_поля)
- Вычисление максимального и минимального значений поля
- Sum(имя_поля)
- Вычисление суммы значений полей

Группировка

- Select paydate,value,name FROM expenses, receivers rs WHERE receiver=rs.num **group by name**

- Select count(*),name FROM expenses, receivers rs WHERE receiver=rs.num **group by name**

Ограничение количества строк

- Select paydate,value,name FROM expenses, receivers rs WHERE receiver=rs.num limit 0,5

Подзапросы

- Select paydate,value,name FROM expenses, receivers WHERE value=(select max(value) from expenses)

- Select name, (select count(*) from expenses where receiver=r.num) FROM receivers r

Основы JDBC

- JDBC – технология, позволяющая программе на Java взаимодействовать с СУБД. Обычно работа происходит в несколько стадий:
- Загрузка драйвера.
- Создание URL соединения
- Установка соединения
- Создание объекта запроса
- Выполнение запроса
- Получение результатов

Загрузка драйвера

- Как правило драйвер поставляется в виде jar-файла. Такой файл должен лежать в каталоге, присутсвующем в CLASSPATH
- Загрузка осуществляется следующим образом:
- ```
  try {
  Class.forName("com.mysql.jdbc.Driver");
  ```
- ```
  } catch (ClassNotFoundException cnfe) {
  ```
- ```
   System.out.println("Error loading driver:
  " +cnfe);
  ```
- ```
  }
  ```
- В данном случае подключается драйвер для работы с MySQL

Создание URL

- URL подключения может различаться в разных СУБД. Для MySQL он будет выглядеть так:
- `jdbc:mysql://адрес_сервера:порт/ имя_базы`
- Например:
- `jdbc:mysql://localhost:3306/ListExpenses`

Установка соединения

- Для установки соединения следует создать объект класса Connection с помощью метода getConnection класса DriverManager
- `Connection  myConnection;`
- `myConnection = DriverManager.getConnection( dbURL,username,password);`

Создание объекта запроса

- Для создания объекта запроса следует воспользоваться методом createStatement объекта класса Connection
- `Statement statement = myConnection.createStatement();`

Выполнение запроса

- Для выполнения запроса `SELECT` используется метод `executeQuery`:
- `statement.executeQuery(query);`
- Для запросов изменяющих данные или саму базу используется метод `executeUpdate`

Поучение данных

- Метод executeQuery возвращает объект класса `ResultSet`
- `ResultSet result = statement.executeQuery(query);`
- Для извлечения данных следует воспользоваться методом next для получения очередной записи, и `getString(номер_поля)` для получения значения конкретного поля прочитанной записи.
- Для нестроковых типов можно использовать аналогичные методы, например `getDouble`.

Прекомпилированные запросы

- Для прекомпилированных запросов создаются специальные запросы, в которых вместо меняющихся значений ставятся знаки вопросов:
- ```
  String template = "SELECT *
  from expenses where value > ?";
  ```
  ```

Прекомпилированные запросы

- Далее создается объект класса PreparedStatement
- `PreparedStatement pStatement = myConnection.prepareStatement( template);`
- Знаки вопроса заменяются значениями с помощью функций `setТип_значения`, например
- `pStatement.setFloat(1, 10000.0F);`