Java Lecture

DB. JDBC

Saint Petersburg, 2015

T · · Systems

Agenda

- **Database**
- Normalization
- Transactions
- JDBC
  - Connection
  - Statement
  - ResultSet

Database

- Model
  - Иерархическая (Hierarchical)
  - Сетевая (Network)
  - Реляционная (Relational)
  - Объектно-Реляционная (Object-relational)
  - XML

Database Requirements

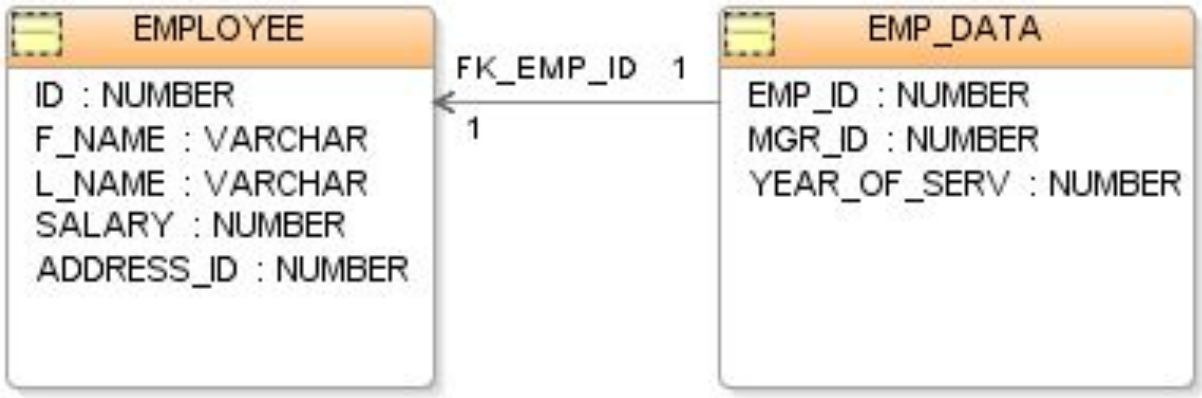
- Data structure
  - Data modeling
  - DDL (CREATE, DROP, ALTER)
- Manipulate data
  - DML (SELECT, INSERT, UPDATE, DELETE)
  - QL (e.g. SQL)
- Data protection

- Availability
- Performance
- Isolation
- Recovery
- Backup and restore

T · · Systems

Relational DB

- Э. Кодд 1970
- Relations – Table; Tuple – Row; Attribute - Column
- Constrains
  - PK
  - FK
- Нормализация
  - 1NF – 3NF
  - BCNF (Бойса - Кодда)
  - 4NF
  - 5NF
  - DKNF (Доменно-ключевая)



T-Systems

Agenda

- Database
- **Normalization**
- Transactions
- JDBC
  - Connection
  - Statement
  - ResultSet

1 NF

- Отношение находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов. (с)

Сотрудник	Номер телефона
Иванов И. И.	283-56-82 390-57-34
Петров П. П.	708-62-34

Сотрудник	Номер телефона
Иванов И. И.	283-56-82
Иванов И. И.	390-57-34
Петров П. П.	708-62-34

T · · Systems

2 NF

- Отношение находится в первой нормальной форме
- Любое неключевое поле полностью зависит от ключа

Сотрудник	Должность	Зарплата	Наличие компьютера
Гришин	Кладовщик	20000	Нет
Васильев	Программист	40000	Есть
Иванов	Кладовщик	25000	Нет

2 NF

- Отношение находится в первой нормальной форме
- Любое неключевое поле полностью зависит от ключа

Сотрудник	Должность	Зарплата	Наличие компьютера
Гришин	Кладовщик	20000	Нет
Васильев	Программист	40000	Есть
Иванов	Кладовщик	25000	Нет

Сотрудник	Должность	Зарплата
Гришин	Кладовщик	20000
Васильев	Программист	40000
Иванов	Кладовщик	25000

Должность	Наличие компьютера
Кладовщик	Нет
Программист	Есть

3 NF

- Отношение находится во второй нормальной форме
- Нет неключевых полей зависящих от значения других неключевых полей

Сотрудник	Отдел	Телефон
Гришин	Бухгалтерия	11-22-33
Васильев	Бухгалтерия	11-22-33
Петров	Снабжение	44-55-66

3 NF

- Отношение находится во второй нормальной форме
- Нет неключевых полей зависящих от значения других неключевых полей

Сотрудник	Отдел	Телефон
Гришин	Бухгалтерия	11-22-33
Васильев	Бухгалтерия	11-22-33
Петров	Снабжение	44-55-66

Отдел	Телефон
Бухгалтерия	11-22-33
Снабжение	44-55-66

Сотрудник	Отдел
Гришин	Бухгалтерия
Васильев	Бухгалтерия
Петров	Снабжение

T · · Systems

Agenda

- Database
- Normalization
- **Transactions**
- JDBC
  - Connection
  - Statement
  - ResultSet

Transactions

ACID

- Atomicity
  - в контексте транзакции либо выполняются все действия, либо не выполняется ни одно из них. Либо происходит *commit* (фиксация), либо *rollback* (откат).
- Consistency
  - системные ресурсы должны пребывать в целостном и непротиворечивом состоянии как до начала транзакции, так и после ее окончания.

Transactions

- Isolation
  - промежуточные результаты транзакции должны быть закрыты для доступа со стороны любой другой действующей транзакции до момента фиксации.
- Durability
  - результат выполнения завершенной транзакции не должен быть утрачен ни при каких условиях.

Isolation problems

- Потерянное обновление (lost update):

Транзакция 1	Транзакция 2
`SELECT f2 FROM tbl1 WHERE f1=1;`	`SELECT f2 FROM tbl1 WHERE f1=1;`
`UPDATE tbl1 SET f2=20 WHERE f1=1;`	
	`UPDATE tbl1 SET f2=25 WHERE f1=1;`

- «Грязное» чтение (dirty read) — чтение данных, которые были записаны откаченной транзакцией:

Транзакция 1	Транзакция 2
`SELECT f2 FROM tbl1 WHERE f1=1;`	
`UPDATE tbl1 SET f2=f2+1 WHERE f1=1;`	
	`SELECT f2 FROM tbl1 WHERE f1=1;`
`ROLLBACK WORK;`	

Isolation problems

- Неповторяющееся чтение (non-repeatable read);

Транзакция 1	Транзакция 2
`SELECT f2 FROM tbl1 WHERE f1=1;`	`SELECT f2 FROM tbl1 WHERE f1=1;`
`UPDATE tbl1 SET f2=f2+1 WHERE f1=1;`	
`COMMIT;`	
	`SELECT f2 FROM tbl1 WHERE f1=1;`

- Фантомное чтение (phantom reads).

Транзакция 1	Транзакция 2
	`SELECT SUM(f2) FROM tbl1;`
`INSERT INTO tbl1 (f1,f2) VALUES (15,20);`	
	`SELECT SUM(f2) FROM tbl1;`

T · · Systems

Isolation levels

- Read uncommitted
- Read committed
- Repeatable read
- Serializable

Isolation levels

- **Read uncommitted**: разрешает грязные чтения, но без потери обновлений. Одна транзакция может не писать в строку, если другая незафиксированная транзакция уже записывает туда. Однако, любая транзакция может читать любые строки.

- **Read committed** : разрешает неповторяемые чтения, но не грязные чтения. Это может быть достигнуто с помощью мгновенных общих блокировок чтения и эксклюзивной блокировки записи. Однако, незафиксированные пишущие транзакции блокируют все другие транзакции на доступ к строке.

Isolation levels

- **Repeatable read** : не допускает ни неповторяемого чтения, ни грязного чтения. Фантомное чтение может произойти. Это может быть достигнуто с использованием общих блокировок на чтение и эксклюзивной блокировки на запись.

- **Serializable** : обеспечивает строгую изоляцию транзакций. Эмулирует последовательное выполнение операций, как если бы операция была выполнена одна за другой последовательно, а не параллельно..

Isolation levels

- **Выбор уровня изоляции зависит от конкретной задачи**

Уровень изоляции	Фантомная вставка	Неповторяющееся чтение	«Грязное» чтение	Потерянное обновление
SERIALIZABLE	+	+	+	+
REPEATABLE READ	–	+	+	+
READ COMMITTED	–	–	+	+
READ UNCOMMITTED	–	–	–	+

T · · Systems

Блокировки

- **Блокировка** - это установка метки на запись, что запись заблокирована для изменений.
- **Оптимистичная**

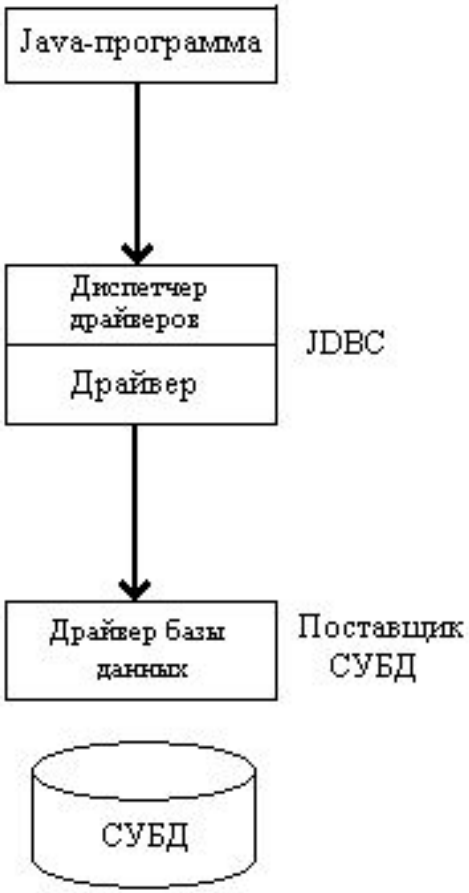
  Реальной блокировки не происходит. Для реализации оптимистичной блокировки часто используется версионирование данных - в таблицу добавляется колонка, которая хранит текущую версию.
- **Пессимистичная**

  Для записи ставится эксклюзивная блокировка на уровне базы данных, запрещая таким образом доступ к данным из других транзакций.
  - Блокировка при чтении
  - Блокировка при записи

Agenda

- Database
- Normalization
- Transactions
- **JDBC**
  - Connection
  - Statement
  - ResultSet

JDBC: Java Data Base Connectivity

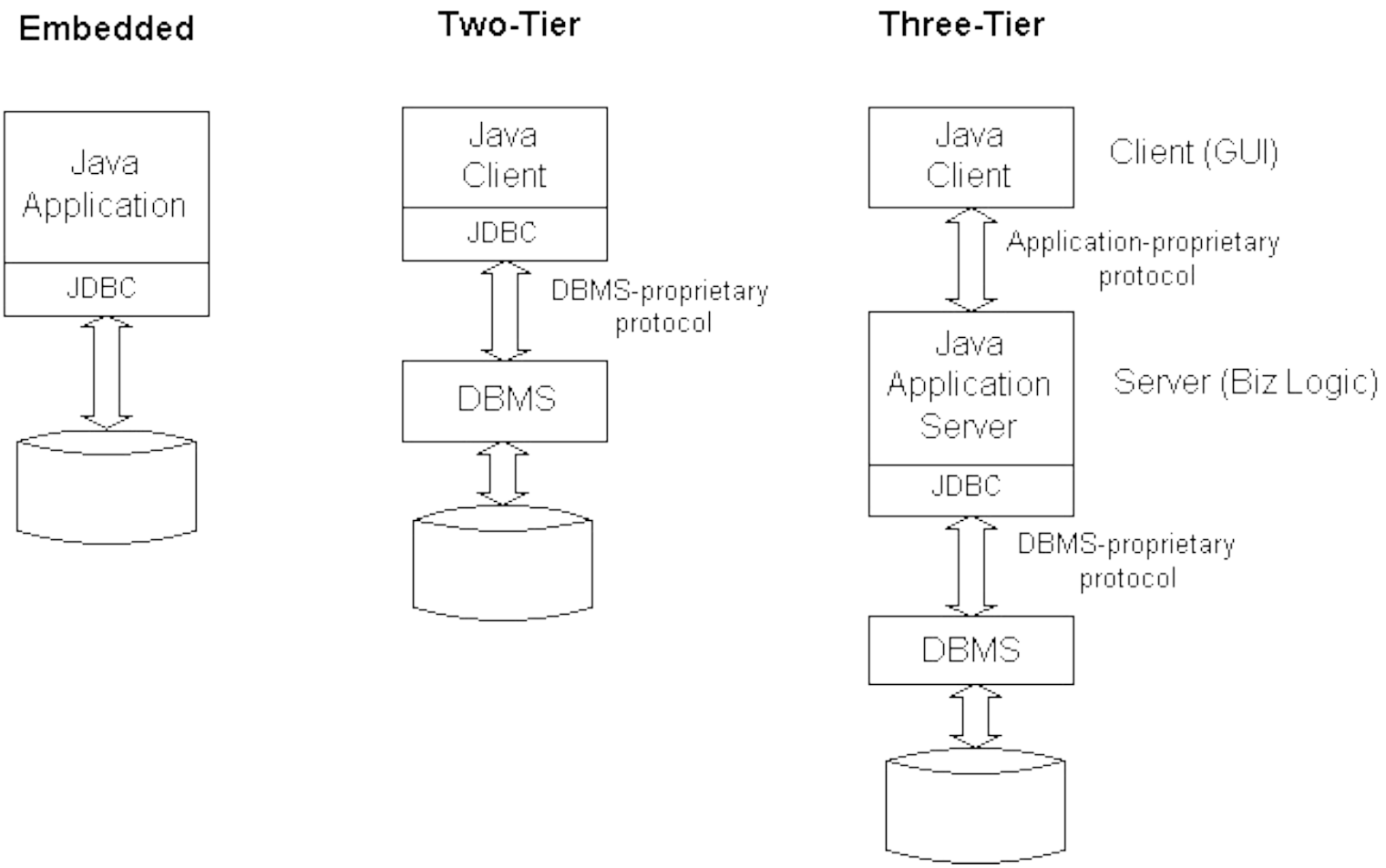


Платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE. (c)

API:

- Для разработки приложений
- Для разрбботки драйверов

JDBC в архитектурном разрезе



Agenda

- Database
- Normalization
- Transactions
- JDBC
  - **Connection**
  - Statement
  - ResultSet

Connection

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306","root", "secret");
Statement stmt = connection.createStatement();
CallableStatement cstmt = connection.prepareCall("?=call some()");
PreparedStatement pstmt = connection.prepareStatement("INSERT INTO ti_cf_roles (?,?,?)");
connection.setAutoCommit(false);
connection.commit();
connection.rollback();
```

- Основной интерфейс для работы с базой данных
- Является ограниченным невозобновляемым ресурсом

T · · Systems

Connection

Connection нужно открывать

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306","root", "secret");
if(!con.isClosed()){
    System.out.println("Connected to MySql");
}
con.close();
```

Строка соединения с базой: jdbc:<subprotocol>:<subname>

- jdbc:odbc:dsn_name;UID=your_uid;PWD=your_pwd
- jdbc:mysql://host_name:port/dbname
- jdbc:oracle:thin:@machine_name:port_number:instance_name

Subprotocol: oracle, mysql, odbc, firebird

T · · Systems

Connection

Connection нужно закрывать

```
Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery("SELECT A_ID, A_NAME FROM ti_cf_personal_data");
doSome(rs);
String firstResult = rs.getString(1);
System.out.println("First result was: " + firstResult);
connection.close();
```

T · · Systems

Connection

Connection нужно закрывать правильно

```
try {
    Statement stmt = connection.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT A_ID, A_NAME FROM ti_cf_personal_data");
    doSome(rs);
    String firstResult = rs.getString(1);
    System.out.println("First result was: " + firstResult);
} finally {
    connection.close();
}
```

T · · Systems

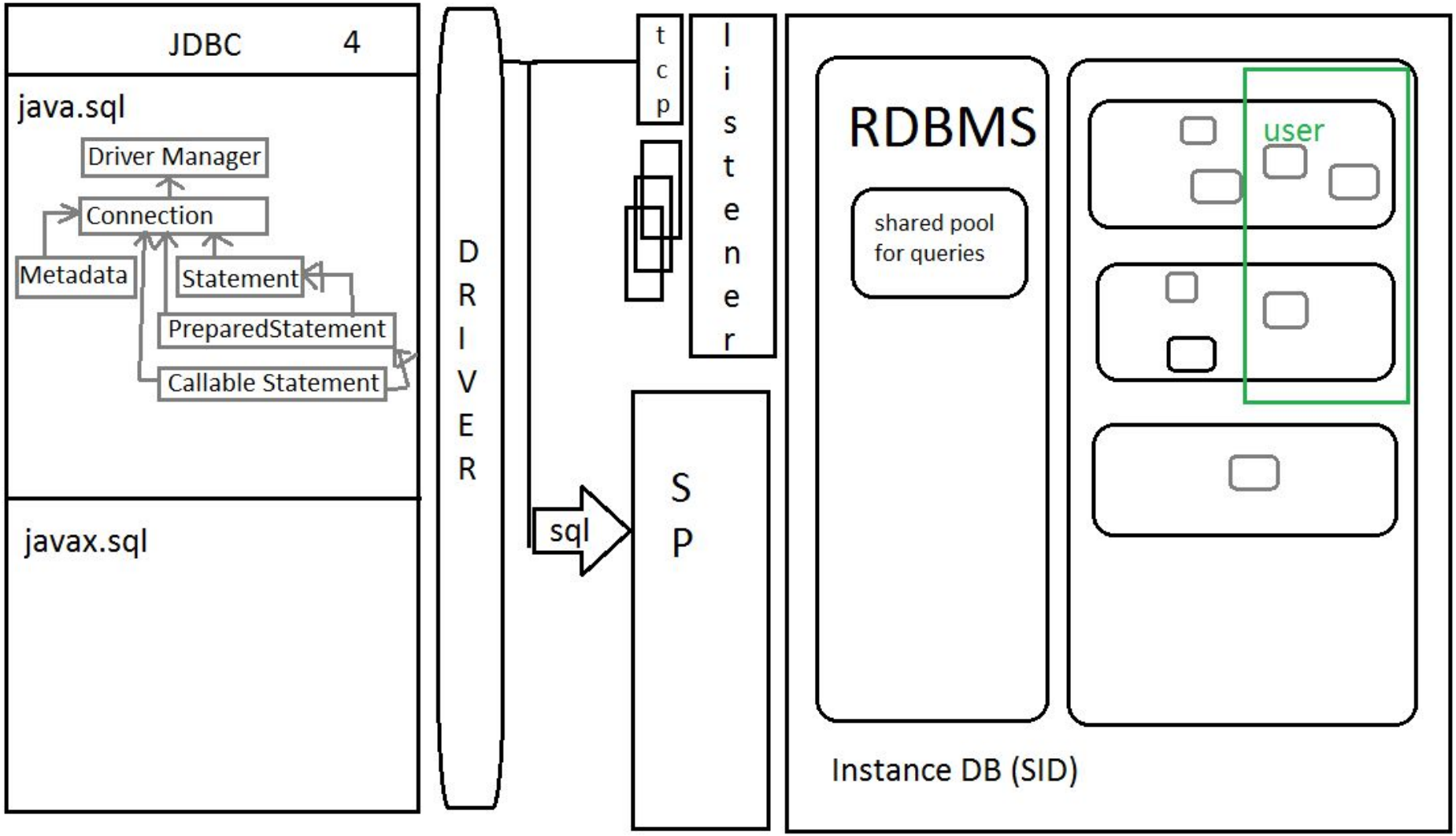
Connection Pool

- Cache of database connections
- Повышается performance – connections создаются сразу, а не по запросу
- Уменьшает время ожидания
- Обычно используется в web и enterprise приложениях
- Управляется application server'ом

Практика №1

- Скачать и установить mySQL server
- Создать схему со следующими сущностями
  - Passenger (id, name)
  - Train (id, seats, name)
  - Ticket (id, date, passenger, train)
- Создать проект с функциональностью подлючения к созданной БД

JDBC


JDBC 4
java.sql
Driver Manager
Connection
Metadata
Statement
PreparedStatement
Callable Statement
javax.sql
DRIVER
tcp
listener
sql
SP
RDBMS
shared pool
for queries
user
Instance DB (SID)

JDBC (JDBC implements…)

- **DDL** –data definition language (change metadata, see system's schema) **NO PARAMETERS**
  - Create
  - Drop
  - Alter
- **DCL** – Data control language **NO PARAMETERS**
  - Commit
  - Rollback
  - Set transaction
- **DML** – Data manipulation language **PARAMETERS (IN) NO RESULT**
  - Insert
  - Delete
  - Update
  - Merge (oracle has)
- **SELECT PARAMETERS (IN) RESULT**
  - +select for update (oracle, postgreSQL)
- **PL/SQL PARAMETERS (IN OUT) RESULT**
  - Begin…End
  - Functions
  - Procedures
  - Packages

T · · Systems

Agenda

- Database
- Normalization
- Transactions
- JDBC
  - Connection
  - **Statement**
  - ResultSet

Statements

- **Statement -** the object used for executing a static SQL statement and returning the results it produces
- **Prepared Statement -** an object that represents a precompiled SQL statement
- **Callable Statement -** the interface used to execute SQL stored procedures.

Statements

- **Statement – простое исполнение статических SQL**

```
Statement stmt = connection.createStatement();
boolean executed = stmt.execute("DROP TABLE ti_wt_sync");
int status = stmt.executeUpdate("UPDATE ti_wt_sync SET DATE=CURRENT_DATE()");
ResultSet rs = stmt.executeQuery("SELECT A_ID, A_NAME FROM ti_cf_personal_data");
stmt.addBatch("UPDATE ti_wt_sync SET DATE=CURRENT_DATE()");
stmt.addBatch("UPDATE ti_wt_sync SET lastId=1986740");
int[] statuses = stmt.executeBatch();
```

T · · Systems

PreparedStatement

- **PreparedStatement – исполнение скомпилированных запросов**

```
Statement stmt = connection.createStatement();
String query = "SELECT * FROM ti_users WHERE a_id="+userId;
stmt.executeQuery(query);
// versus
PreparedStatement findUserStmt = connection.prepareStatement("SELECT * FROM ti_users WHERE a_id=?");
findUserStmt.setLong(1, 5407);
ResultSet rs = findUserStmt.executeQuery();
```

- Выполняется быстрее, чем Statement
- Предохраняет от SQL Injection
- Подобный шаблон используется в Hibernate и JPA

CallableStatement

- **CallableStatement – исполнение хранимых процедур**

```
CallableStatement wtStmt = connection.prepareCall("?=call update_time(?,?,?,?)");
wtStmt.setLong(1, 5407);
//......
ResultSet result = wtStmt.executeQuery();
```

ResultSet

ResultSet: курсор текущего запроса

```
Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery("SELECT A_ID, A_NAME FROM ti_cf_personal_data");
while (rs.next()){
    rs.getString("A_NAME");
    rs.getString(2);
}
```

- Является мощным средством для итерации по набору данных

T · · Systems

ResultSetMetaData

Предоставляет мета данные запроса

```
Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery("SELECT A_ID, A_NAME FROM ti_cf_personal_data");
ResultSetMetaData rsmd = rs.getMetaData();
rsmd.getColumnCount();
rsmd.getColumnLabel(2);
```

Общие правила

- На 1 Statement – 1 ResultSet
- Открыл – закрой
- Statement можно (и даже нужно) использовать повторно
- В серьезных проектах использовать только PreparedStatement
- Помнить про транзакцию

Архитектура доступа к данным

- Смешивание SQL кода и реализации является в Java антипаттерном (недействительно в Индии ☺)
- Ни в коем случае не допускается вызов SQL из view или controller (MVC)
- Соединения лучше пуллировать
- А еще лучше использовать типовые решения

Практика №2

- Вывести все билеты, купленные пассажиром %username%, с помощью
  - Statement
  - PreparedStatement