

# Проектирование баз данных

## Design of databases

Копп Андрей Михайлович

Andrii M. Kopp

Ассистент кафедры ПИИТУ

Assistant Lecturer of the Department of SEMIT

[kopp93@gmail.com](mailto:kopp93@gmail.com) @andriikopp

# Программа курса / Course program

- 1 Знакомство с MySQL / Introduction to MySQL
- 2 Проектирование и реализация БД / DB design and implementation
- 3 Работа с данными при помощи SQL / Data manipulation using SQL
- 4 Реализация бизнес-логики / Business logic implementation
- 5 Целостность данных, транзакции, права пользователей / Data integrity, transactions, user privileges
- 6 Построение ПО для работы с БД / Database application design

# Контрольные точки / Checkpoints

1	Модуль 1 / Part 1	(50)
	45% КП / Course project	
	1 – 4 ЛР / Lab. Works	(30)
	Индивидуальное задание / Personal assignment	(10)
	Контрольная работа / Test	(10)
2	Модуль 2 / Part 2	(50)
	80% КП / Course project	
	5 – 9 ЛР / Lab. works	(30)
	Индивидуальное задание / Personal assignment	(10)
	Контрольная работа / Test	(10)
3	Курсовой проект / Course project	(100)

# 1 Знакомство с MySQL

## 1 Introduction to MySQL

- 1.1 Почему MySQL? / Why MySQL?
- 1.2 Клиент-серверная архитектура / Client-server architecture
- 1.3 Установка MySQL и сопутствующих компонентов / Install MySQL and required components
- 1.4 Начало работы с MySQL / Starting work with MySQL
- 1.5 Резервное копирование и восстановление / Backups and recovery

## 2 Проектирование и реализация БД

### 2 DB design and implementation

2.1 Процесс проектирования БД / DB design process

2.2 Создание и использование БД / Creating and using DB

2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

2.4 Типы таблиц в MySQL / Table types in MySQL

# 3 Работа с данными при помощи SQL

## 3 Data manipulation using SQL

- 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data
- 3.2 Создание запросов на выборку / Creating select queries
- 3.3 Условия выборки / Conditions
- 3.4 Сортировка / Sorting
- 3.5 Ограничение выборки / Limit
- 3.6 Группировка записей / Group
- 3.7 Использование функций / Using functions
- 3.8 Операторы / Operators
- 3.9 Переменные / Variables
- 3.10 Временные таблицы / Temporary tables

# 4 Реализация бизнес-логики

## 4 Business logic implementation

4.1 Создание и использование представлений / Creating and using views

4.2 Создание и использование хранимых процедур / Creating and using stored procedures

4.3 Создание и использование триггеров / Creating and using triggers

- 5 Целостность данных, транзакции, права пользователей
- 5 Data integrity, transactions, user privileges
- 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms  
NO ACTION, CASCADE, SET NULL, SET DEFAULT
- 5.2 Механизм транзакций / Transactional mechanism  
ACID, уровни изоляции / isolation levels
- 5.3 Управление правами пользователей / Manage user privileges



# 6 Построение ПО для работы с БД

## 6 Database application design

### 6.1 Технология JDBC / JDBC technology

Построение Java приложения для работы с БД / Build Java application to work with DB

### 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Построение PHP приложения для работы с БД / Build PHP application to work with DB

# Курсовой проект / Course project

- 1 Знакомство с предметной областью / Study the domain
- 2 Обзор существующих решений / Review the existing solutions
- 3 Разработка бизнес-правил и модели данных / Design business rules and data model
- 4 Обзор и выбор средств разработки / Review and select development tools
- 5 Создание БД / Create the DB
- 6 Заполнение БД данными / Populate DB with data
- 7 Разработка приложения / Develop the application
- 8 Тестирование ПО / Test the software
- 9 Подготовка и защита / Preparation and defense

- 1 Знакомство с MySQL
- 1 Introduction to MySQL

# 1.1 Почему MySQL / Why MySQL?

- Свободно-распространяемая система управления базами данных
- Free database management system
- Клиент-серверная архитектура
- Client-server architecture

# 1.1 Почему MySQL / Why MySQL?

- Открытый исходный код (MariaDB)
- Open-Source (MariaDB)
- Кроссплатформенная система
- Cross-platform system
- Программные интерфейсы
- Application Programming Interfaces (APIs)

# 1.1 Почему MySQL / Why MySQL?

- Многопоточность
- Multithreading
- Многопользовательский доступ
- Concurrent access
- Быстродействие
- Speed
- Масштабируемость
- Scalability

# 1.1 Почему MySQL / Why MySQL?

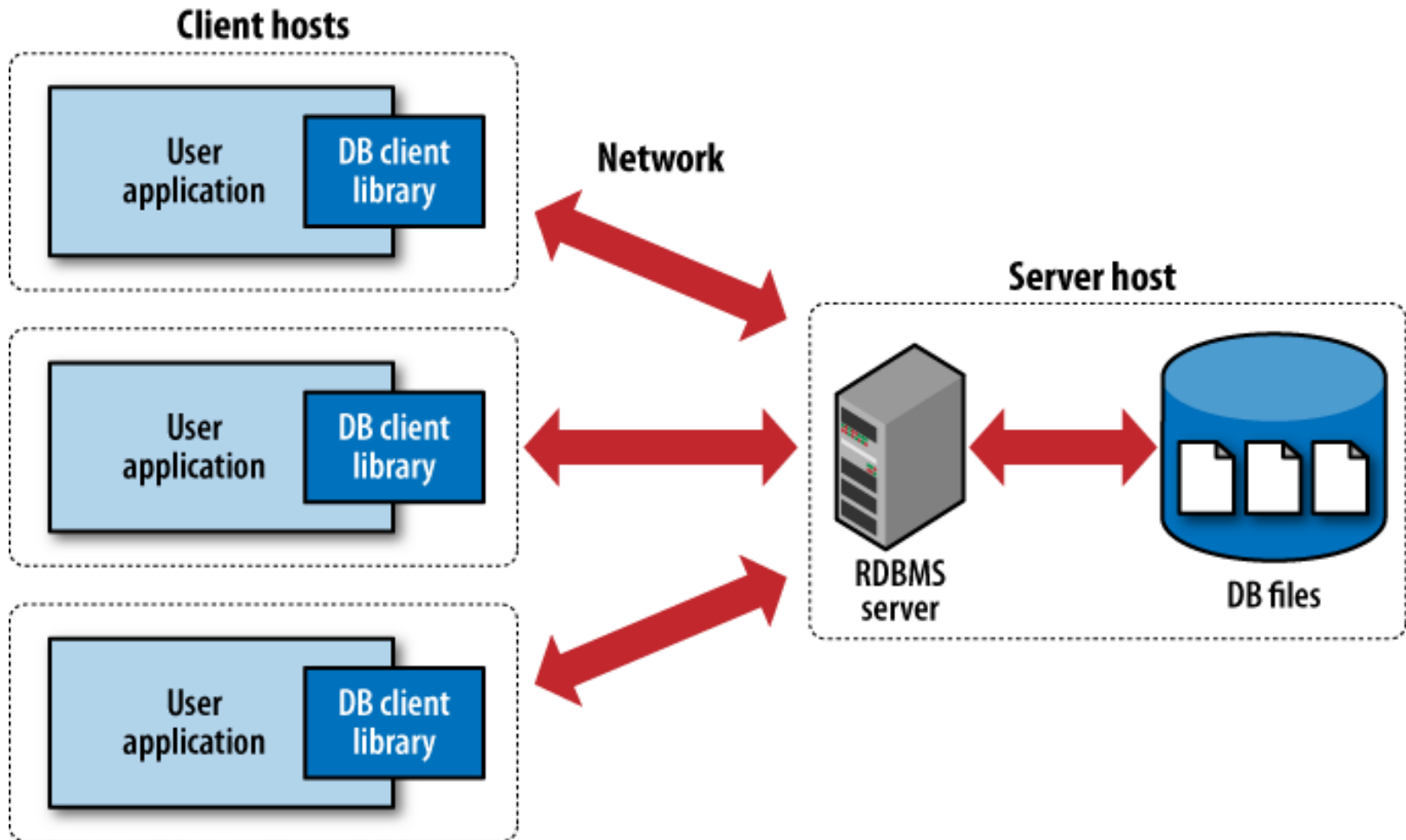
- Обеспечение безопасности и разграничение доступа на основе системы привилегий
- Security and access management based on the privileges system

# 1.1 Почему MySQL / Why MySQL?

- Реляционная система управления базами данных (СУБД)
- Relational database management system (DBMS)
- Базовые понятия теории реляционных баз данных (БД)
- Basic concepts of the theory of relational databases (DB)



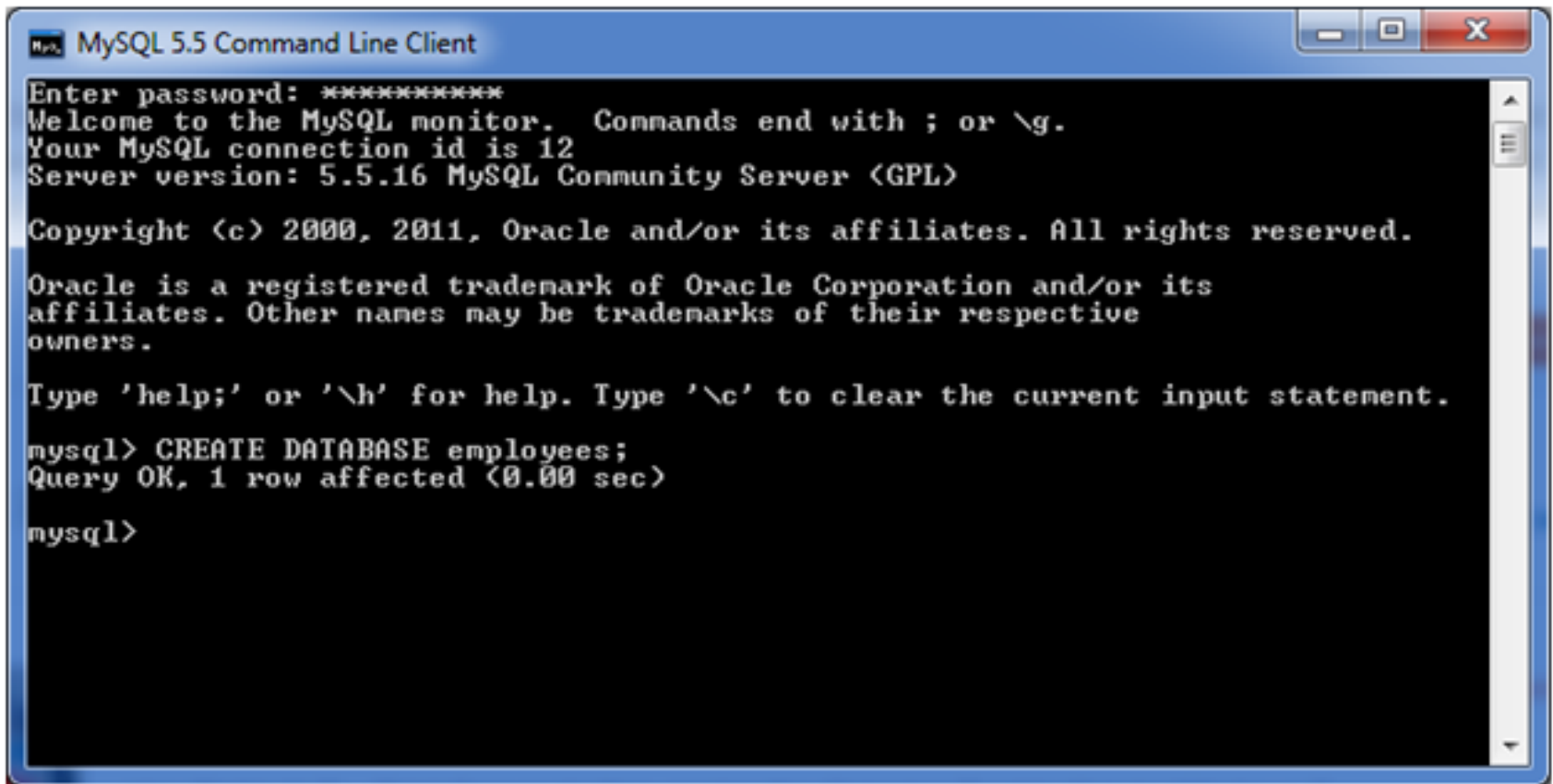
## 1.2 Клиент-серверная архитектура / Client-server architecture



## 1.2 Клиент-серверная архитектура / Client-server architecture

- К серверу MySQL могут подключаться различные клиентские приложения, в том числе и с удаленных устройств
- There are many MySQL clients available including remote devices
- Стандартный клиент MySQL представлен в виде приложения командной строки
- The standard MySQL client is the command-line application

## 1.2 Клиент-серверная архитектура / Client-server architecture



```
MySQL 5.5 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE employees;
Query OK, 1 row affected (0.00 sec)

mysql>
```

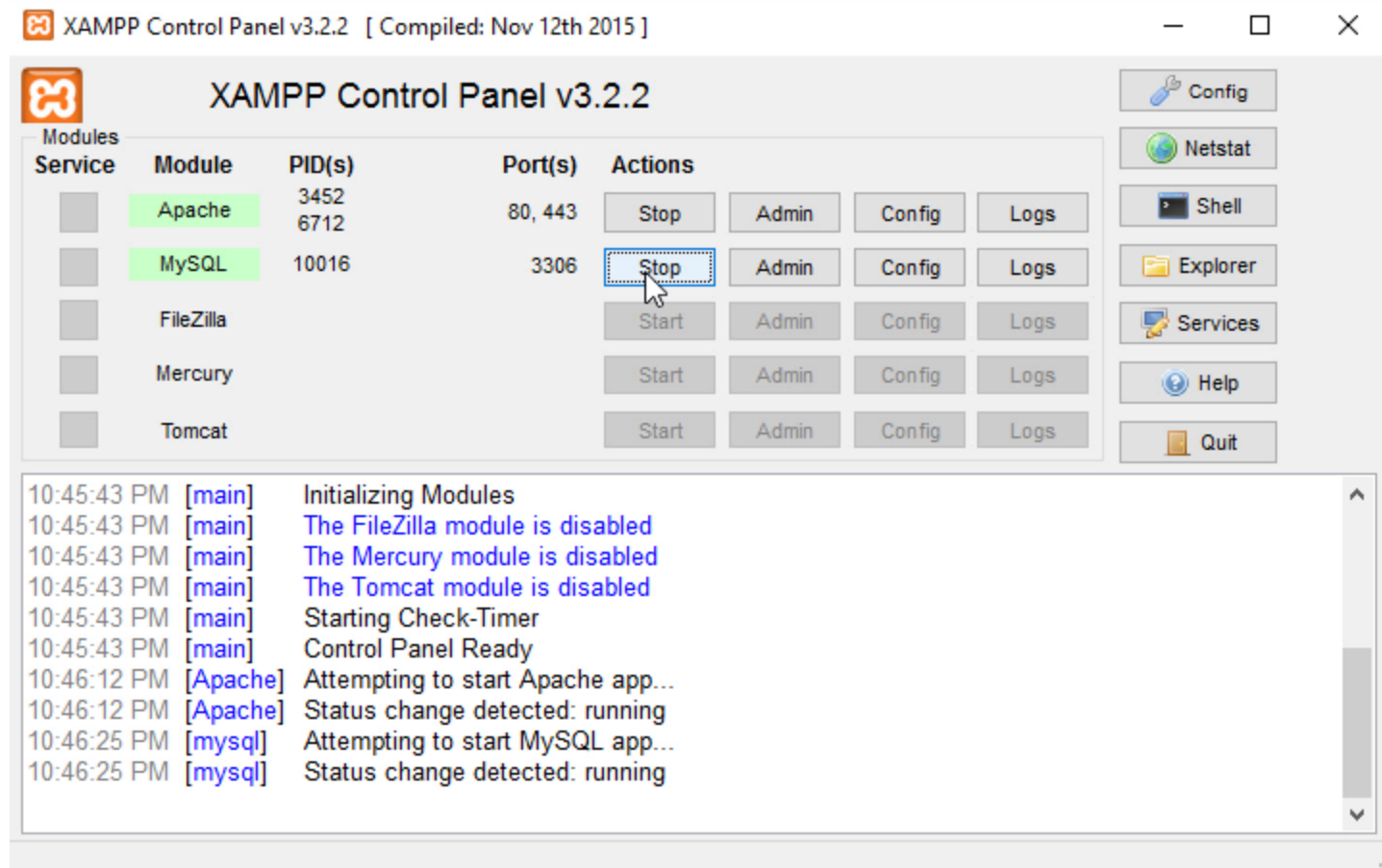
## 1.3 Установка MySQL и сопутствующих компонентов / Install MySQL and required components

- Установка при помощи инсталляционного пакета с графическим интерфейсом
- System-wide installation using a graphical installation package
- Установка без инсталляции и изменений в системе
- Local installation using a “no-install” package

## 1.3 Установка MySQL и сопутствующих компонентов / Install MySQL and required components

- Установка с помощью интегрированного пакета XAMPP
- System-wide installation using the XAMPP integrated package
- Второй способ удобен при отсутствии прав администратора
- Second option is handy when you do not have administrator privileges

# 1.3 Установка MySQL и сопутствующих компонентов / Install MySQL and required components

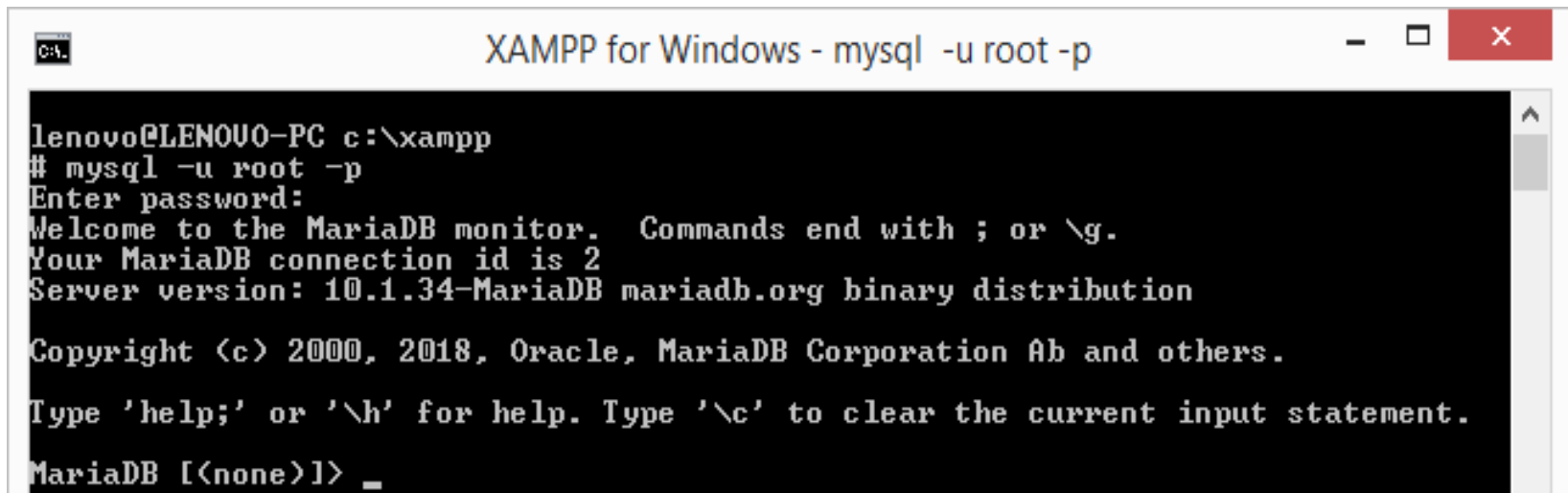


## 1.3 Установка MySQL и сопутствующих компонентов / Install MySQL and required components

- X (Linux, macOS/OS X, Solaris, Windows)
- Apache
  - свободный веб-сервер
  - free web-server
- MySQL
- PHP
- Perl

## 1.3 Установка MySQL и сопутствующих компонентов / Install MySQL and required components

C:\xampp\mysql\bin > mysql -u root -p



```
lenovo@LENOVO-PC c:\xampp
# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.34-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> _
```

XAMPP Control Panel > Shell > mysql -u root -p



# 1.4 Начало работы с MySQL / Starting work with MySQL

- SHOW DATABASES;

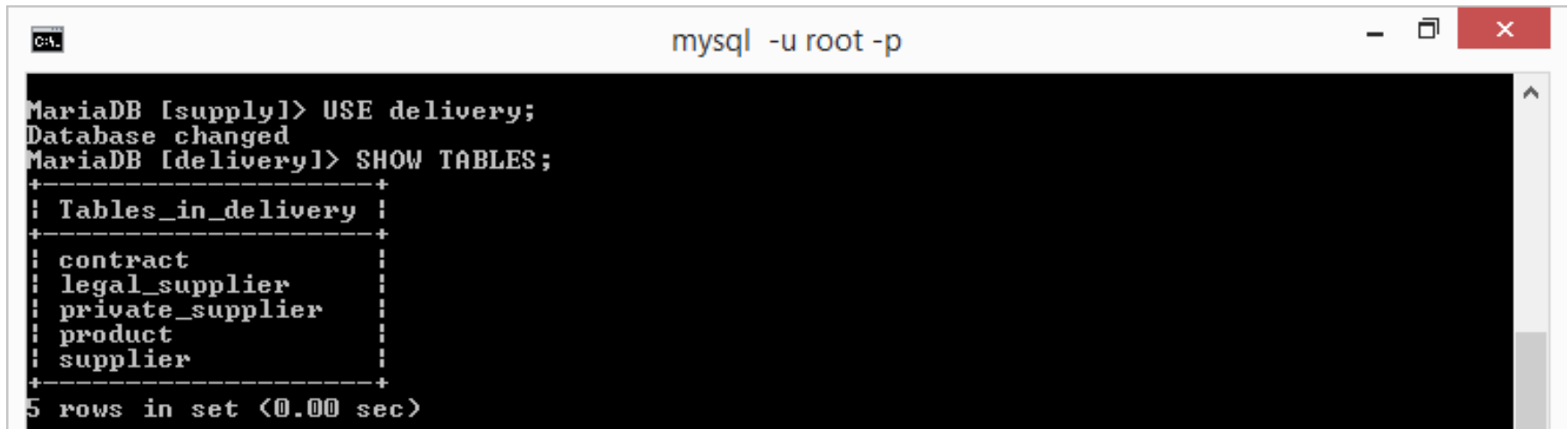


The screenshot shows a terminal window titled "mysql -u root -p". The prompt is "MariaDB [(none)]>". The command "SHOW DATABASES;" has been entered. The output is a table with one column, "Database", listing ten databases: delivery, information\_schema, itdb, medical\_card, mysql, performance\_schema, phpmyadmin, supply, supply\_1, and test. The output is formatted with dashed lines around the table. At the bottom, it says "10 rows in set (0.00 sec)".

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| delivery |
| information_schema |
| itdb      |
| medical_card |
| mysql     |
| performance_schema |
| phpmyadmin |
| supply    |
| supply_1  |
| test      |
+-----+
10 rows in set (0.00 sec)
```

## 1.4 Начало работы с MySQL / Starting work with MySQL

- USE <table\_name>;
- SHOW TABLES;



The screenshot shows a terminal window titled "mysql -u root -p". The prompt is "MariaDB [supply]>". The user enters "USE delivery;", and the prompt changes to "MariaDB [delivery]>". The user then enters "SHOW TABLES;". The output is a table with one row: "Tables\_in\_delivery". The table lists five tables: "contract", "legal\_supplier", "private\_supplier", "product", and "supplier". The output ends with "5 rows in set (0.00 sec)".

```
mysql -u root -p
MariaDB [supply]> USE delivery;
Database changed
MariaDB [delivery]> SHOW TABLES;
+-----+
| Tables_in_delivery |
+-----+
| contract            |
| legal_supplier      |
| private_supplier    |
| product             |
| supplier            |
+-----+
5 rows in set (0.00 sec)
```

## 1.5 Резервное копирование и восстановление / Backups and recovery

- Выгрузка базы данных в виде SQL выражений
- Dumping a database as SQL statements
- Используется утилита mysqldump
- The mysqldump utility is using

## 1.5 Резервное копирование и восстановление / Backups and recovery

```
C:\xampp\mysql\bin > mysqldump
```

```
--user=<user_name>
```

```
--password=<user_password>
```

```
--result-file=<file_name>
```

```
<db_name>
```

```
> mysqldump --user=root --result-file=supply.sql  
supply
```

# 1.5 Резервное копирование и восстановление / Backups and recovery

supply.sql

```
19  -- Table structure for table `contract`
20  --
21
22  DROP TABLE IF EXISTS `contract`;
23  /*!40101 SET @saved_cs_client      = @@character_set_client */;
24  /*!40101 SET character_set_client = utf8 */;
25  CREATE TABLE `contract` (
26    `contract_number` int(11) NOT NULL AUTO_INCREMENT,
27    `contract_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
28    `supplier_id` int(11) NOT NULL,
29    `contract_note` varchar(100) DEFAULT NULL,
30    PRIMARY KEY (`contract_number`),
31    KEY `contract_ibfk_1` (`supplier_id`),
32    CONSTRAINT `contract_ibfk_1` FOREIGN KEY (`supplier_id`) REFERENCES `supplier` (`supplier_id`)
33  ) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=latin1;
34  /*!40101 SET character_set_client = @saved_cs_client */;
35
36  --
37  -- Dumping data for table `contract`
38  --
39
40  LOCK TABLES `contract` WRITE;
41  /*!40000 ALTER TABLE `contract` DISABLE KEYS */;
42  INSERT INTO `contract` VALUES (1,'2018-08-31 21:00:00',1,'Order 34 on 30.08.2018'),(2,'2018-09-09
43  /*!40000 ALTER TABLE `contract` ENABLE KEYS */;
44  UNLOCK TABLES;
```

## 1.5 Резервное копирование и восстановление / Backups and recovery

- Файл резервного копирования не содержит выражения CREATE DATABASE и USE
- Backup file does not contain CREATE DATABASE and USE statements
- DROP DATABASE <existing\_db>;
- CREATE DATABASE <recovered\_db>;
- USE <recovered\_db>;

## 1.5 Резервное копирование и восстановление / Backups and recovery

```
MariaDB [(none)]> CREATE DATABASE rec_supply;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> USE rec_supply;
Database changed
MariaDB [rec_supply]> SOURCE supply.sql;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

# 1.5 Резервное копирование и восстановление / Backups and recovery

'rec\_supply' database

```
MariaDB [rec_supply]> SHOW TABLES;
+-----+
| Tables_in_rec_supply |
+-----+
| contract              |
| contract_supplier     |
| supplied              |
| supplier              |
| supplier_info         |
| supplier_org          |
| supplier_person       |
+-----+
7 rows in set (0.00 sec)

MariaDB [rec_supply]> SELECT * FROM supplier_org;
+-----+-----+
| supplier_id | supplier_org_name |
+-----+-----+
|          2 | Interfruit Ltd.   |
|          4 | Transservice LLC  |
+-----+-----+
2 rows in set (0.00 sec)
```



## 2 Проектирование и реализация БД

### 2 DB design and implementation

## 2.1 Процесс проектирования БД / DB design process

- Невозможно реализовать БД, не потратив достаточно времени и усилий на проектирование
- It is impossible to implement a new DB without dedicating adequate time and effort to the design

## 2.1 Процесс проектирования БД / DB design process

- Хорошо продуманный дизайн позволит расширять имеющееся решение, а не ломать и начинать работу с нуля
- Good design allows you to extend the original solution without having to pull everything down and start from scratch

## 2.1 Процесс проектирования БД / DB design process

1. Анализ предметной области / Domain analysis
2. Разработка системы бизнес-правил / Business rules system development
3. Разработка моделей данных / Data models development
4. Реализация базы данных / Database implementation

## 2.1 Процесс проектирования БД / DB design process

1. Анализ предметной области / Domain analysis
  - Выделить и описать основные бизнес-процессы
  - Identify and describe core business processes

## 2.1 Процесс проектирования БД / DB design process

1. Анализ предметной области / Domain analysis
  - Бизнес-процесс – структурированный набор действий, использующих материалы и/или информацию для производства продукции или услуг, имеющих ценность для потребителя
  - Business process is a structured sequence of activities that takes materials and/or information and produces products or services valuable for a customer

## 2.1 Процесс проектирования БД / DB design process

1. Анализ предметной области / Domain analysis

Поставка продукции / Product supply

- 1 Формирование заказа / Order creation

1.1 ...

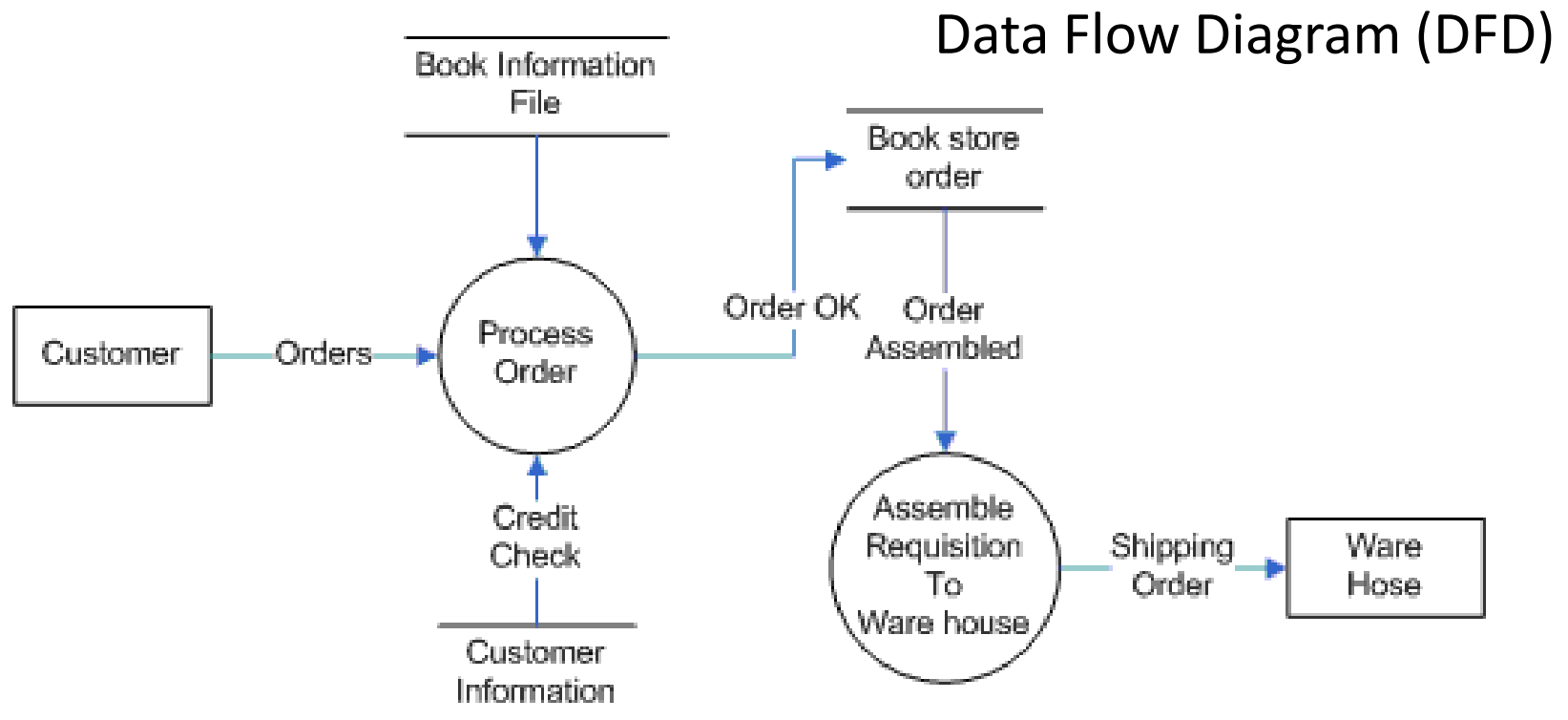
- 2 Согласование с поставщиком / Supplier approval

- 3 Доставка продукции / Product delivery

- 4 Прием продукции / Product reception

## 2.1 Процесс проектирования БД / DB design process

### 1. Анализ предметной области / Domain analysis





## 2.1 Процесс проектирования БД / DB design process

### 2. Разработка системы бизнес-правил / Business rules system development

Формализованные результаты обследования предметной области

Formalized results of domain analysis

Определяют структуру информационных объектов БД

Define the structure of DB informational objects

## 2.1 Процесс проектирования БД / DB design process

### 2. Разработка системы бизнес-правил / Business rules system development

- Факты / Facts
- Ограничения / Restrictions
- Активаторы операций / Operation triggers
- Выводы / Conclusions

## 2.1 Процесс проектирования БД / DB design process

### 2. Разработка системы бизнес-правил / Business rules system development

#### Факт / Fact

Каждый поставщик характеризуется названием и адресом. Для уникальной идентификации поставщиков используется код.

Each supplier is described by the name and address. Unique code is used to identify suppliers.

## 2.1 Процесс проектирования БД / DB design process

### 2. Разработка системы бизнес-правил / Business rules system development

#### Ограничение / Restriction

Поставщик как субъект предпринимательской деятельности может быть или юридическим, или физическим лицом

Supplier as a business entity can be either a legal entity or an individual entrepreneur

## 2.1 Процесс проектирования БД / DB design process

### 2. Разработка системы бизнес-правил / Business rules system development

Активатор операции / Operation trigger

Ввод данных о поставщике как о физ. лице  
запрещен, если уже хранятся данные о нем  
как о юр. лице

Entering information about the supplier as an  
individual entrepreneur is prohibited if data  
about it as a legal entity is already stored

## 2.1 Процесс проектирования БД / DB design process

### 2. Разработка системы бизнес-правил / Business rules system development

#### Вывод / Conclusion

Если поставщик не отправил заказ в течение 5 календарных дней с момента получения заказа, заказ считается невыполненным

If supplier has not sent an order within 5 calendar days from the moment of receiving the order, the order is considered as unfulfilled

## 2.1 Процесс проектирования БД / DB design process

### 3. Разработка моделей данных / Data models development

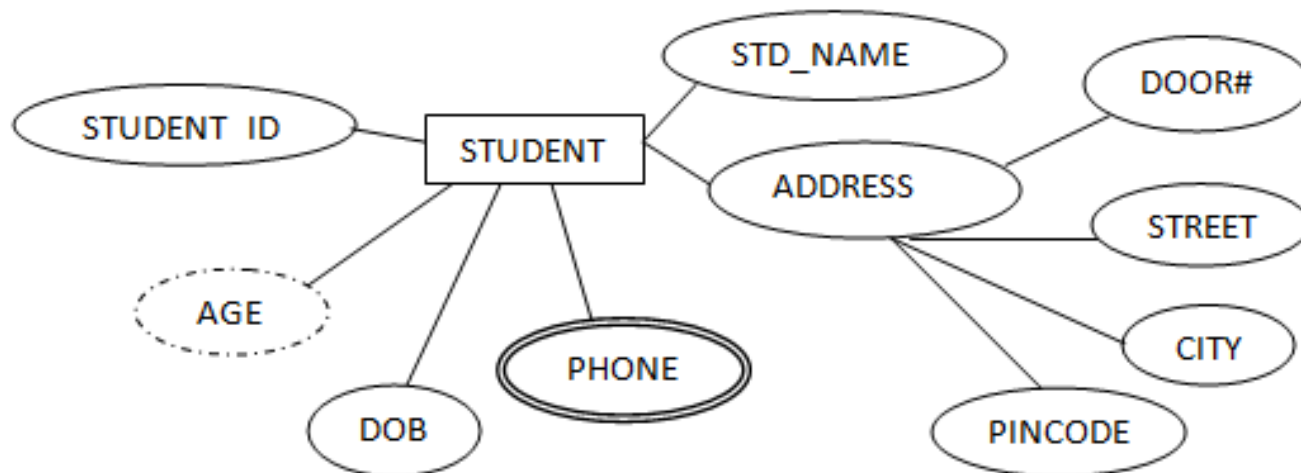
- Концептуальная МД / Conceptual DM
- Логическая МД / Logical DM
- Физическая МД / Physical DM

## 2.1 Процесс проектирования БД / DB design process

### 3. Разработка моделей данных / Data models development

- Концептуальная МД / Conceptual DM

ER модель



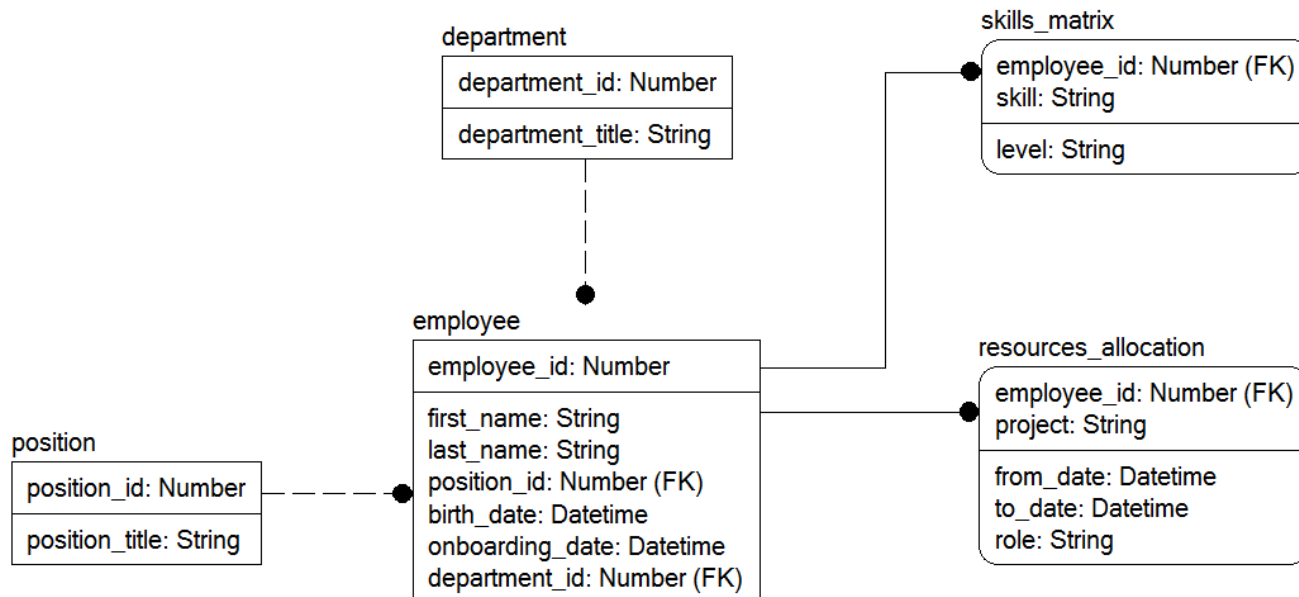


## 2.1 Процесс проектирования БД / DB design process

### 3. Разработка моделей данных / Data models development

- Логическая МД / Logical DM

IDEF1X модель



## 2.1 Процесс проектирования БД / DB design process

### 3. Разработка моделей данных / Data models development

- Физическая МД / Physical DM

Должна учитывать особенности конкретной СУБД

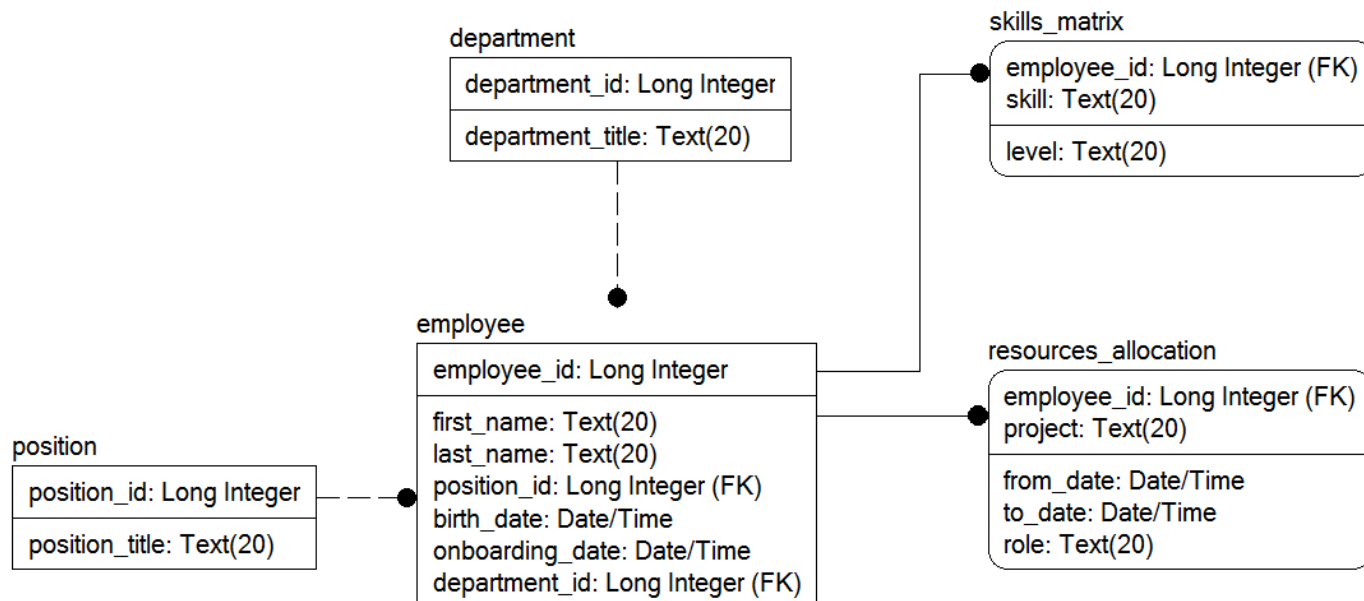
It should be designed according to features of the chosen DBMS

## 2.1 Процесс проектирования БД / DB design process

### 3. Разработка моделей данных / Data models development

- Физическая МД / Physical DM

#### IDEF1X модель



## 2.1 Процесс проектирования БД / DB design process

### 4. Реализация базы данных / Database implementation

Детальное табличное описание каждой  
реляционной таблицы и представления БД

Detail tabular description of each relational  
table and view of a DB

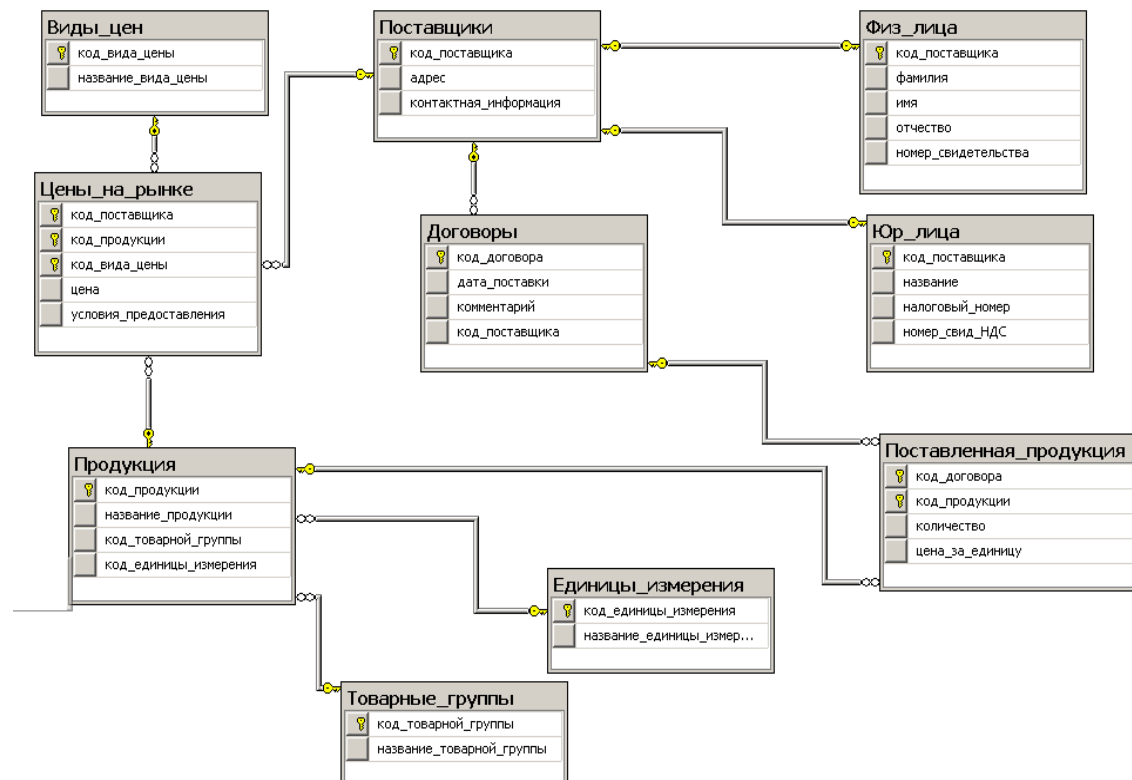
## 2.1 Процесс проектирования БД / DB design process

### 4. Реализация базы данных / Database implementation

Ключ		Ім'я поля	Тип даних	Розмір поля	Опис
PK	FK	НомерДоговора	Чисельний	Ціле	номер договору
	FK	КодПродукции	Чисельний	Ціле	код продукції
		Количество	Чисельний	Ціле	кількість одиниць продукції
		ЦенаЗаЕдиницу	Чисельний	Одинарное з плаваючою крапкою	ціна за одиницю продукції

# 2.1 Процесс проектирования БД / DB design process

## 4. Реализация базы данных / Database implementation



## 2.2 Создание и использование БД / Creating and using DB

... еще немного о клиент-серверной  
архитектуре

... a little more about the client-server  
architecture

- Традиционная архитектура (2-х уровневая)
- Traditional architecture (2 tier)

## 2.2 Создание и использование БД / Creating and using DB

Клиент – программа представления данных и реализация бизнес-логики

Client – a software used to represent data and implement business logic

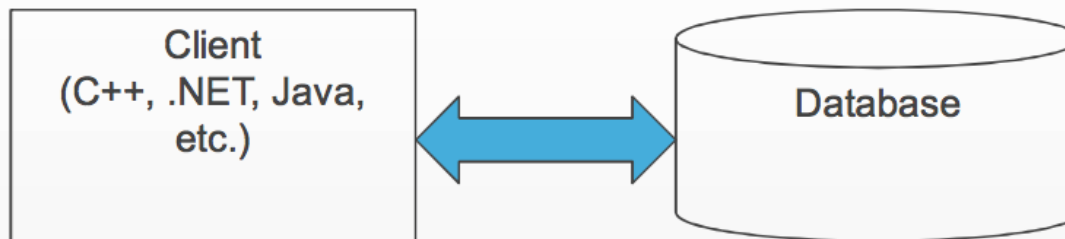
Сервер – программа управления данными и сами данные

Server – a software used to manage data and data itself

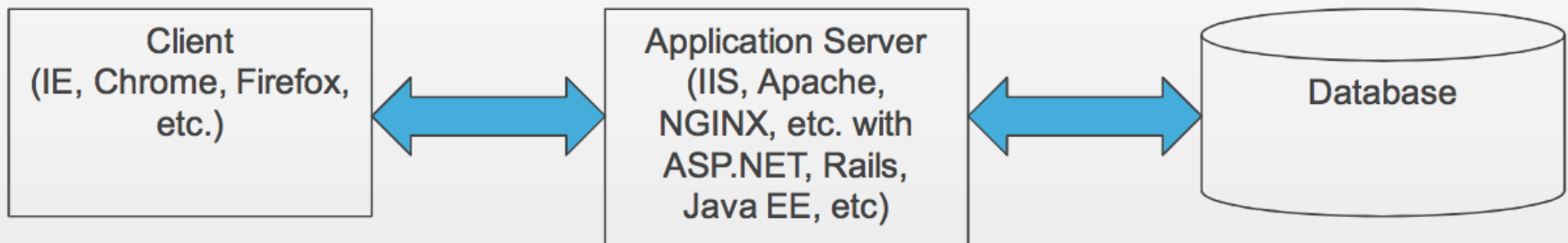


## 2.2 Создание и использование БД / Creating and using DB

2-Tiered Architecture



3-Tiered Architecture



## 2.2 Создание и использование БД / Creating and using DB

Трехуровневая архитектура

3-Tiered architecture

- Уровень представления – представление данных
- Presentation layer – data representation
- Уровень приложения – бизнес-логика
- Application layer – business logic
- Уровень базы данных – управление данными и сами данные
- Database layer – data management and data

## 2.2 Создание и использование БД / Creating and using DB

Уровень представления / Presentation layer:

```
<div class="col-4">
  <ul class="list-group">
    <li class="list-group-item active">Contracts</li>
    <?php foreach ($service->getAllContracts() as $contract) { ?>
      <li class="list-group-item">
        <a href="contracts.php?details=<?=$contract->getNumber() ?>">
          #<?=$contract->getNumber() ?>, <?=$contract->getAgreed() ?>,
        </a>
      </li>
    <?php } ?>
  </ul>
</div>
```

## 2.2 Создание и использование БД / Creating and using DB

Уровень приложения / Application layer:

```
class ContractService
{
    private $repository;

    public function __construct(ContractRepositoryInterface $repository)
    {
        $this->repository = $repository;
    }

    public function getAllContracts()
    {
        return $this->repository->getContractList();
    }
}
```

## 2.2 Создание и использование БД / Creating and using DB

Уровень базы данных / Database layer:

```
public function getContractList()
{
    $conn = MySQLConnectionUtil::getConnection();
    $contracts = array();

    $query = 'SELECT number, agreed, supplier.name, title, note
             FROM contract INNER JOIN supplier ON contract.supplier = supplier.id';
    $result = mysqli_query($conn, $query);
```

 number	agreed	 supplier	title	note
1	1999-09-01	2	Contract 1	Invoice 34from 8/30/99
2	1999-09-10	2	Contract 2	Invoice 08-78 from 8/28/99
3	1999-09-10	4	Contract 3	Invoice 08-78 from8/28/99
4	1999-09-23	4	Contract 4	Order 56from 8/28/99
5	1999-09-24	3	Contract 5	Invoice 74from 9/11/99
6	1999-10-01	2	Contract 6	Invoice 9-12from 9/28/99
7	1999-10-02	3	Contract 7	Invoice 85from 9/21/99

## 2.2 Создание и использование БД / Creating and using DB

```
CREATE DATABASE <db_name>;
```

```
CREATE DATABASE supply;
```

- При создании БД можно указать нужную кодировку и/или правило сравнения символьных значений
- When creating a DB it is possible to configure the character set and/or collation rule for char values

## 2.2 Создание и использование БД / Creating and using DB

SHOW CHARACTER SET;

SHOW COLLATION;

Команды для просмотра списков  
используемых в MySQL кодировок и  
правил сравнения символьных значений

Commands used to view lists of used in MySQL  
character sets and collation rules

## 2.2 Создание и использование БД / Creating and using DB

```
mysql -u root -p
MariaDB [(none)]> SHOW CHARACTER SET;
```

Charset	Description	Default collation	Maxlen
big5	Big5 Traditional Chinese	big5_chinese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
cp850	DOS West European	cp850_general_ci	1
hp8	HP West European	hp8_english_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
latin1	cp1252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
swe7	7bit Swedish	swe7_swedish_ci	1
ascii	US ASCII	ascii_general_ci	1
ujis	EUC-JP Japanese	ujis_japanese_ci	3
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
euckr	EUC-KR Korean	euckr_korean_ci	2
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
greek	ISO 8859-7 Greek	greek_general_ci	1
cp1250	Windows Central European	cp1250_general_ci	1
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
armscii8	ARMSCII-8 Armenian	armscii8_general_ci	1
utf8	UTF-8 Unicode	utf8_general_ci	3
ucs2	UCS-2 Unicode	ucs2_general_ci	2
cp866	DOS Russian	cp866_general_ci	1
keybcs2	DOS Kamenicky Czech-Slovak	keybcs2_general_ci	1
macce	Mac Central European	macce_general_ci	1
macroman	Mac West European	macroman_general_ci	1
cp852	DOS Central European	cp852_general_ci	1
latin7	ISO 8859-13 Baltic	latin7_general_ci	1
utf8mb4	UTF-8 Unicode	utf8mb4_general_ci	4
cp1251	Windows Cyrillic	cp1251_general_ci	1
utf16	UTF-16 Unicode	utf16_general_ci	4
utf16le	UTF-16LE Unicode	utf16le_general_ci	4
cp1256	Windows Arabic	cp1256_general_ci	1
cp1257	Windows Baltic	cp1257_general_ci	1
utf32	UTF-32 Unicode	utf32_general_ci	4
binary	Binary pseudo charset	binary	1
geostd8	GEOSTD8 Georgian	geostd8_general_ci	1
cp932	SJIS for Windows Japanese	cp932_japanese_ci	2
eucjpms	UJIS for Windows Japanese	eucjpms_japanese_ci	3

```
40 rows in set (0.00 sec)
```



## 2.2 Создание и использование БД / Creating and using DB

mysql -u root -p

MariaDB [(none)]> SHOW COLLATION;

Collation	Charset	Id	Default	Compiled	Sortlen
big5_chinese_ci	big5	1	Yes	Yes	1
big5_bin	big5	84		Yes	1
dec8_swedish_ci	dec8	3	Yes	Yes	1
dec8_bin	dec8	69		Yes	1
cp850_general_ci	cp850	4	Yes	Yes	1
cp850_bin	cp850	80		Yes	1
hp8_english_ci	hp8	6	Yes	Yes	1
hp8_bin	hp8	72		Yes	1
koi8r_general_ci	koi8r	7	Yes	Yes	1
koi8r_bin	koi8r	74		Yes	1
latin1_german1_ci	latin1	5		Yes	1
latin1_swedish_ci	latin1	8	Yes	Yes	1
latin1_danish_ci	latin1	15		Yes	1
latin1_german2_ci	latin1	31		Yes	2
latin1_bin	latin1	47		Yes	1
latin1_general_ci	latin1	48		Yes	1
latin1_general_cs	latin1	49		Yes	1
latin1_spanish_ci	latin1	94		Yes	1
latin2_czech_cs	latin2	2		Yes	4
latin2_general_ci	latin2	9	Yes	Yes	1
latin2_hungarian_ci	latin2	21		Yes	1
latin2_croatian_ci	latin2	27		Yes	1
latin2_bin	latin2	77		Yes	1
swe7_swedish_ci	swe7	10	Yes	Yes	1
swe7_bin	swe7	82		Yes	1
ascii_general_ci	ascii	11	Yes	Yes	1
ascii_bin	ascii	65		Yes	1
ujis_japanese_ci	ujis	12	Yes	Yes	1
ujis_bin	ujis	91		Yes	1
sjis_japanese_ci	sjis	13	Yes	Yes	1
sjis_bin	sjis	88		Yes	1
hebrew_general_ci	hebrew	16	Yes	Yes	1
hebrew_bin	hebrew	71		Yes	1
tis620_thai_ci	tis620	18	Yes	Yes	4
tis620_bin	tis620	89		Yes	1
euckr_korean_ci	euckr	19	Yes	Yes	1
euckr_bin	euckr	85		Yes	1
koi8u_general_ci	koi8u	22	Yes	Yes	1
koi8u_bin	koi8u	75		Yes	1

## 2.2 Создание и использование БД / Creating and using DB

Если нужно загрузить данные, находящиеся в кодировке CP-1251 ...

If it is required to upload data in CP-1251 character set ...

```
CREATE DATABASE <db_name>  
CHARACTER SET cp1251 COLLATE  
cp1251_general_ci;
```

## 2.2 Создание и использование БД / Creating and using DB

- `_ci` = “case insensitive” при сравнении и сортировке не учитывается регистр
- `_cs` = “case sensitive” регистр учитывается
- `_bin` = “binary” сравнение и сортировка по числовым кодам символов

`utf8mb4` рекомендуется использовать вместо `utf8` с MySQL 5.5.3

`utf8mb4` is recommended to use instead of `utf8` starting with MySQL 5.5.3

## 2.2 Создание и использование БД / Creating and using DB

- Юникод – стандарт кодирования, представления и обработки текстов, включающий в себя знаки почти всех письменных языков мира
- UTF-8 – кодировка, позволяющая компактно хранить и передавать символы Юникода
- Unicode is a standard for the consistent encoding, representation, and handling of text represented in most of the world's writing systems
- UTF-8 is a character set for compact storing and transferring Unicode characters

## 2.2 Создание и использование БД / Creating and using DB

```
ALTER DATABASE <db_name>  
CHARACTER SET <character_set_name>  
COLLATE <collation_name>;
```

Изменение кодировки и/или правила сравнения  
для БД затронет только вновь создаваемые  
таблицы

Changes of character set and/or collation rule will  
affect only the new created tables

## 2.2 Создание и использование БД / Creating and using DB

```
CREATE DATABASE supply  
CHARACTER SET utf8mb4  
COLLATE utf8mb4_unicode_ci;
```

Для корректного отображения юникода:

Use it to represent unicode correctly:

```
SET NAMES utf8mb4 COLLATE utf8mb4_unicode_ci;
```

## 2.2 Создание и использование БД / Creating and using DB

Для удаления БД используется команда

```
DROP DATABASE <db_name>;
```

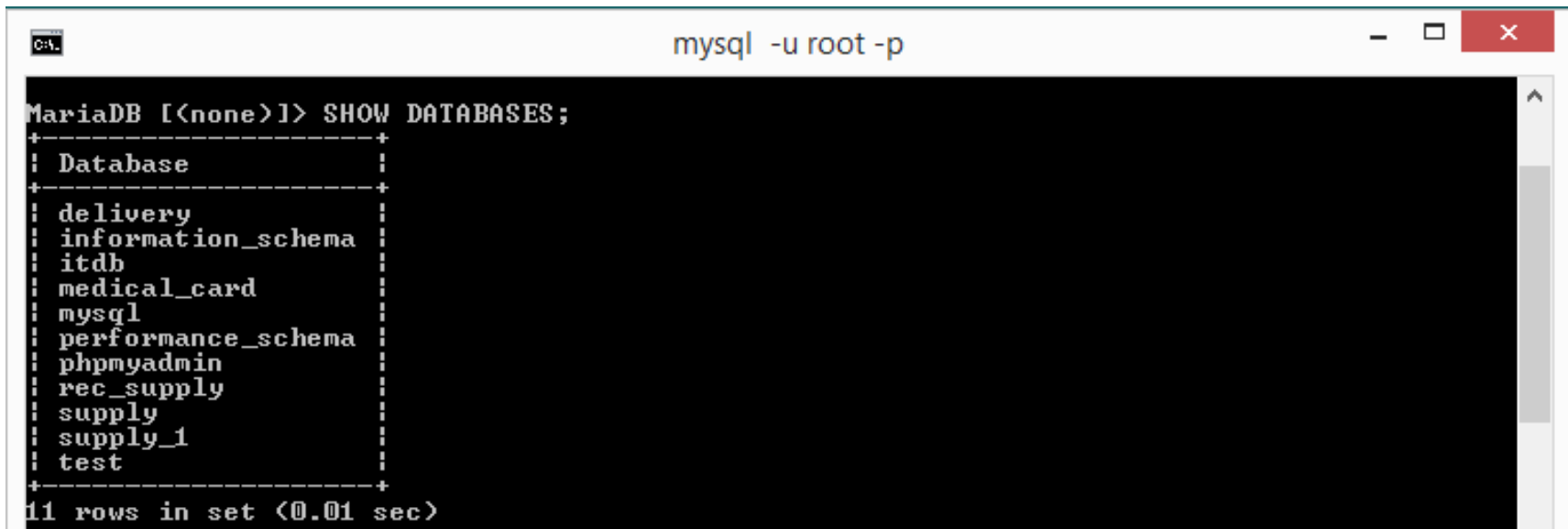
This statement is used to remove existing DB

Рекомендуется создать резервную копию БД  
перед удалением

It is recommended to backup the DB before delete  
operation

## 2.2 Создание и использование БД / Creating and using DB

- information\_schema
- mysql
- test



The screenshot shows a terminal window titled "mysql -u root -p". Inside the terminal, the command "MariaDB [(none)]> SHOW DATABASES;" has been executed. The output is a table with one column named "Database" and 11 rows of database names. The databases listed are: delivery, information\_schema, itdb, medical\_card, mysql, performance\_schema, phpmyadmin, rec\_supply, supply, supply\_1, and test. At the bottom of the terminal, it says "11 rows in set (0.01 sec)".

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| delivery |
| information_schema |
| itdb      |
| medical_card |
| mysql     |
| performance_schema |
| phpmyadmin |
| rec_supply |
| supply    |
| supply_1  |
| test      |
+-----+
11 rows in set (0.01 sec)
```



## 2.2 Создание и использование БД / Creating and using DB

- **information\_schema**

Информационная БД о всех остальных БД и их структуре  
Informational DB about other DBs and their structure

- **mysql**

Служебная БД, хранит сведения о пользователях, правах доступа и т.д.

Service DB that stores information about users, access privileges etc.

- **test**

Пустая БД для экспериментов / Empty DB for experiments

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
CREATE TABLE <table name>
```

```
(
```

```
<field name 1> <field type 1> [<field properties 1>],
```

```
<field name 2> <field type 2> [<field properties 2>],
```

```
...
```

```
[<keys and indexes information>]
```

```
)
```

```
[<optional properties>;
```

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

Многие из параметров задавать не обязательно  
A lot of parameters are optional

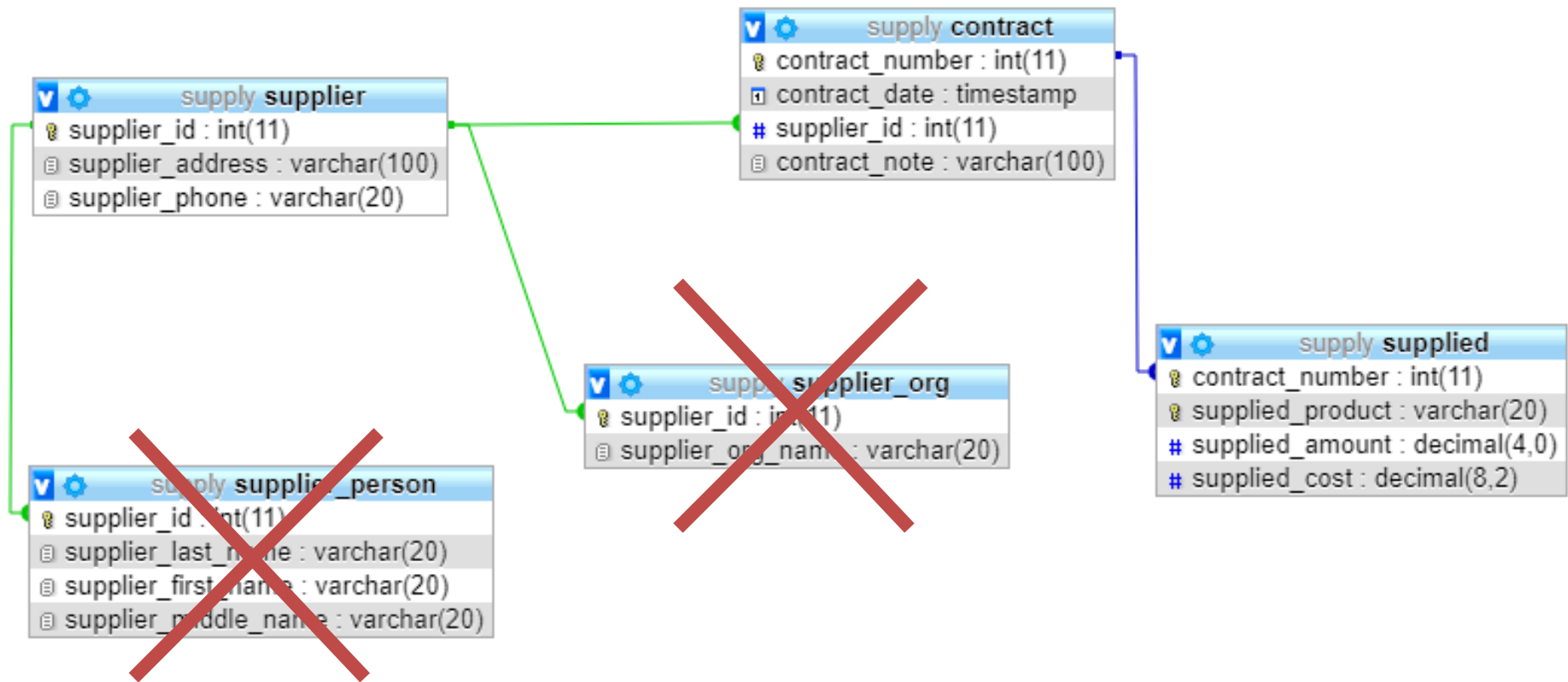
Необходимые параметры / Required parameters:

- имя таблицы / table name;
- имена и типы столбцов / columns names and types.

Остальные параметры используются при необходимости

The rest of parameters are used if necessary

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

Таблицы базы данных / Database tables:

- supplier
- ~~supplier\_person~~
- ~~supplier\_org~~
- contract
- supplied

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
CREATE TABLE supplier (  
    supplier_id int NOT NULL,  
    supplier_address varchar(100) NOT NULL,  
    supplier_phone varchar(20) NOT NULL,  
    PRIMARY KEY (supplier_id)  
) ENGINE=InnoDB;
```

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

supplier\_id int NOT NULL

**supplier\_id SERIAL**

SERIAL = BIGINT UNSIGNED NOT NULL  
AUTO\_INCREMENT UNIQUE

BIGINT – большие целые / big integers

UNSIGNED – положительные числа / positive  
numbers

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**NOT NULL UNIQUE** – автоматически контролируется наличие неопределенных и повторяющихся значений / automatically controlled presence of null and duplicate values

**AUTO\_INCREMENT** – в столбец автоматически вносится очередной порядковый номер, если значение не указано / the next number is automatically entered into the column, if no value is specified



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

NULL – константа, указывающая на отсутствие значения / constant shows the absence of value

NULL != "", NULL != 0

1 = 1 => TRUE

NULL = NULL => NULL

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**supplier\_address varchar(100) NOT NULL,  
supplier\_phone varchar(20) NOT NULL,**

VARCHAR

поскольку эти столбцы будут содержать  
символьные значения

since these columns will contain character  
values

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **PRIMARY KEY (supplier\_id)**

Указывает на то, что столбец supplier\_id будет первичным ключом таблицы

Indicates that the supplier\_id column will be a primary key of the table

**PRIMARY KEY (<col\_1>, <col\_2>)**

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **ENGINE InnoDB**

Опциональный параметр задает тип таблицы

Optional parameter defines the table type

InnoDB обеспечивает ссылочную целостность между таблицами

InnoDB provides referential integrity between the tables

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

ENGINE InnoDB **CHARACTER SET utf8mb4;**

Определяет кодировку для данных в таблице  
Defines character set for data in the table

Кодировка по умолчанию назначается для всех столбцов таблицы

The default character set is defined for all table columns

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
CREATE TABLE contract (  
    contract_number int NOT NULL AUTO_INCREMENT,  
    contract_date timestamp NOT NULL,  
    supplier_id int NOT NULL,  
    contract_note varchar(100),  
    PRIMARY KEY (contract_number),  
    FOREIGN KEY (supplier_id) REFERENCES  
        supplier(supplier_id)  
) ENGINE=InnoDB;
```

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

contract\_note varchar(100)

**contract\_note TEXT**

TEXT удобно использовать вместо VARCHAR, если столбец будет содержать длинные значения

TEXT is convenient to use instead of VARCHAR if the column will contain long values

VARCHAR – 65 535 bytes / table

TEXT – нельзя использовать как FK / can not be used as the FK

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**contract\_date timestamp NOT NULL**

TIMESTAMP

“YYYY-MM-DD HH:MM:SS”

4-байтное целое = кол-во секунд, прошедших с  
полуночи 1 января 1970

amount of seconds passed since the midnight of January  
1st 1970


NOT NULL по умолчанию / by default

NOW() – текущее значение / current value



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
CREATE TABLE test (  
    id SERIAL,  
    tstmp TIMESTAMP  
) ENGINE INNODB;
```

 id	tstmp
1	2019-02-08 15:39:13

```
INSERT INTO test (id) VALUES (1);
```

tstmp TIMESTAMP **DEFAULT CURRENT\_TIMESTAMP**

задается только при добавлении строки (без свойства –  
задается и при изменении)

specified only when a record is inserted (without property it is  
specified also when a record is updated)

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**FOREIGN KEY (supplier\_id) REFERENCES  
supplier(supplier\_id)**

Столбец `supplier_id` содержит номера поставщиков из  
таблицы `supplier`

The column `supplier_id` contains numbers of suppliers  
from the table `supplier`

В случае использования `SERIAL` для `supplier_id` ...

In case if the type of `supplier_id` is `SERIAL` ...

**`supplier_id BIGINT UNSIGNED`**

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
FOREIGN KEY (supplier_id) REFERENCES  
supplier(supplier_id)
```

В `supplier_id` могут содержаться только значения  
из `supplier_id` таблицы `supplier` или значения  
`NULL`

`supplier_id` might contain only values from  
`supplier_id` of the table `supplier` and `NULL` values

`supplier_id` **BIGINT NOT NULL**

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Дополнительно можно указать правила ссылочной целостности
- In addition you can specify the referential integrity rules

FOREIGN KEY (supplier\_id) REFERENCES  
supplier(supplier\_id) **ON DELETE RESTRICT**

нельзя удалить запись о поставщике, если с ним был заключен договор

you can not remove the record about supplier if it is referenced in a contract

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Дополнительно можно указать правила ссылочной целостности
- In addition you can specify the referential integrity rules

FOREIGN KEY (supplier\_id) REFERENCES  
supplier(supplier\_id) **ON UPDATE CASCADE**

при изменении номера поставщика в таблице supplier  
соответствующие изменения вносятся в таблицу  
contract

when changing the number of supplier in the table  
supplier it affects the table contract as well

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
CREATE TABLE supplied (  
    contract_number int NOT NULL,  
    supplied_product varchar(20) NOT NULL,  
    supplied_amount decimal(4,0) NOT NULL,  
    supplied_cost decimal(8,2) NOT NULL,  
    PRIMARY KEY (contract_number, supplied_product),  
    FOREIGN KEY (contract_number) REFERENCES  
        contract(contract_number)  
) ENGINE=InnoDB;
```

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**supplied\_amount decimal(4,0) NOT NULL,  
supplied\_cost decimal(8,2) NOT NULL,**

DECIMAL

для хранения денежных сумм и других значений, для которых важно избегать ошибок округления

used to store currencies and other values when it is important to avoid rounding mistakes

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

value **DECIMAL(8, 2)**

8 – максимальное количество цифр в значении столбца / max number of digits in the column value

2 – максимальное количество цифр после десятичного разделителя / max number of digits after the comma

6 ( $6 = 8 - 2$ ) в целой части / integer part



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **! ВНИМАНИЕ !**

Обратить внимание на последовательность создания таблиц. Таблицы supplier и contract должны быть созданы раньше, чем ссылающиеся на них внешние ключи.

### **! WARNING !**

It is important to follow the sequence of tables creation. Tables supplier and contract should be created earlier than corresponding foreign keys.

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ( ... );**

**TEMPORARY**

создание таблицы, существующей только в текущем сеансе работы с БД

creates the table that exists only in the current session

**IF NOT EXISTS**

таблица будет создана, если еще нет таблицы с указанным именем

the table will be created if there is no table with the specified name

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]  
<table name> LIKE <existing table name>
```

Создание таблицы с такой же структурой, как у существующей таблицы

Creating the table with the same structure the existing table has

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Индексация столбца / Column indexing

INDEX, KEY

столбцы не обязательно содержат уникальные значения

columns are not necessary contain unique values

UNIQUE

столбцы должны содержать уникальные значения

columns should contain unique values

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Индексация столбца / Column indexing

### FULLTEXT

полнотекстовые индексы для столбцов типа TEXT, CHAR, VARCHAR

full text indexes based on columns of types TEXT, CHAR, and VARCHAR

! Только для таблиц типа **MyISAM** tables only !

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Индексы – основной способ ускорения работы БД
- Using indexes is the main way to improve the database performance
- Для столбца создается копия, постоянно поддерживаемая в отсортированном состоянии
- Indexed column is copied and maintained in sorted state constantly

### **PRIMARY KEY, KEY, UNIQUE, INDEX**

CREATE INDEX <name> ON <table>(<field>);

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

DESCRIBE <table name>;

Позволяет получить подробную информацию о структуре таблицы

Provides the detailed information about the table structure

COLUMNS (6×4)					
Field	Type	Null	Key	Default	Extra
contract_number	int(11)	NO	PRI	(NULL)	auto_increment
contract_date	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
supplier_id	int(11)	NO	MUL	(NULL)	
contract_note	varchar(100)	YES		(NULL)	

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### Типы данных для столбцов / Column data types

- Числовые / Numeric
- Строковые / String
- Календарные / Calendar
- NULL – обозначает отсутствие информации / identifies absence of information



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### Числовые типы / Numeric types

- Точные типы / Exact types

**INTEGER** и его вариации / and its variations

**DECIMAL**

- Приближенные типы / Approximate types

**FLOAT**

**DOUBLE**

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **DECIMAL (M, D)**

M – число символов для отображения всего числа / number of digits that represent total number

D – число символов для отображения дробной части / number of digits in its fractional part

price DECIMAL (5, 2)

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **UNSIGNED**

столбец содержит только положительные числа или нули

a column stores only positive numbers or zero values

### **ZEROFILL**

число будет отображаться с ведущими нулями

a number will be shown with leading zero digits

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### Текстовые типы и строки / Text and string types

- CHAR – хранение строк фиксированной длины / stores strings of the fixed length
- VARCHAR – хранение строк переменной длины / stores strings of the variable length
- TEXT, BLOB – хранение больших фрагментов текста / stores large fragments of text
- ENUM, SET – хранение значений из заданного списка / stores values from a pre-defined list

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **CHAR(M)**

независимо от длины строки, использует для ее хранения все M символов

does not depend on the string length, uses all M chars to store it

### **VARCHAR (M)**

использует количество символов, равное длине строки + 1 байт

uses the number of chars which is equal to the string length + 1 byte

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- CHAR обрабатывается эффективнее VARCHAR
- CHAR is more efficient than VARCHAR
- Нельзя смешивать в таблице столбцы CHAR и VARCHAR
- Do not mix CHAR and VARCHAR columns in a table

CHAR => VARCHAR

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### TEXT

поддерживает полнотекстовый поиск

supports full-text search

TEXT	BLOB
Учитывается кодировка Character set is taken into account	Не учитывается кодировка Character set is not taken into account
Для хранения больших объемов текста Used to store large amounts of text	Для хранения больших двоичных объектов Used to store large binary objects

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **ENUM**

содержит одно значение из указанного множества

contains a single value from the pre-defined set

### **SET**

может содержать любой или все элементы заданного множества одновременно

might contain any or all elements from the pre-defined set at the same time

status ENUM ('valid', 'invalid') **DEFAULT** 'invalid'



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

Календарные типы данных / Calendar data types

**DATE** (YYYY-MM-DD)

**TIME** (HH:MM:SS)

**DATETIME** (YYYY-MM-DD HH:MM:SS)

**TIMESTAMP**

**YEAR** (YYYY)

В качестве разделителей могут выступать любые символы, отличные от цифры

Any symbols might be used as separators, except digits

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Обработка числовых данных происходит быстрее строковых
- Numeric data is processed faster than string data
- Производительность можно увеличить за счет представления строк в виде чисел
- Performance might be improved by representing string values as numeric values

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Если есть возможность, следует выбирать типы данных, занимающие меньше места
- Choose data types that require less storage space if it is possible
- Типы фиксированной длины обрабатываются быстрее типов переменной длины
- Data types with fixed length are processed faster than data types with variable length
- Иначе необходимо использовать **OPTIMIZE TABLE** для дефрагментации таблицы
- Otherwise use **OPTIMIZE TABLE** to defragment the table

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- [CONSTRAINT <key\_name>] **PRIMARY KEY** (<columns\_list>)
- **INDEX** [<index\_name>] (<columns\_list>)
- [CONSTRAINT <index\_name>] **UNIQUE** [index\_name] (<columns\_list>)

### TEXT, BLOB

указать количество символов в начале значения, по которым будет проведено индексирование

specify the number of symbols at the beginning of a value that will be used for indexing

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**FULLTEXT** [<index\_name>] (<columns\_list>)

ускоренный поиск по значениям символьных столбцов

fast search over values of character columns

**CHAR, VARCHAR, TEXT**

можно создать только в таблицах типа MyISAM  
can be created only in MyISAM tables

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

[CONSTRAINT <fk\_name>] **FOREIGN KEY**  
[<index\_name>] (<columns\_list>) **REFERENCES**  
**KEY** [<parent\_table\_name>] (<pk\_columns\_list>)  
[<integrity\_rules>]

только для таблиц с типом InnoDB (и дочерняя и родительская должны иметь тип **InnoDB**)

only for InnoDB tables (both child and parent tables should be **InnoDB** tables)

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- для числовых столбцов должен совпадать размер и знак
- sizes and signs should match for numeric columns
- для символьных столбцов должна совпадать кодировка и правило сравнения значений
- character sets and collations should match for character columns
- столбцы с типом TEXT и BLOB не могут входить во внешний ключ
- TEXT and BLOB columns can not be included into a foreign key

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### Опциональные свойства таблицы / Optional table properties

- ENGINE <table\_type>
- AUTO\_INCREMENT <start\_value>
- CHARACTER SET <charset\_name>
- COLLATE <comparison\_rule>



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **ALTER TABLE**

используется для модификации ранее созданной таблицы

used to modify previously created table

**ALTER TABLE** <table\_name>

**ADD** <column\_name> <column\_type>  
[<column\_properties>]

[**FIRST** | **AFTER** <preceding\_column\_name>];

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### **FIRST, AFTER**

позволяют указать место добавляемого столбца  
allow to specify the place of the added column

- **FIRST** – столбец может стать первым / a column could be the first
- **AFTER** – может следовать за указанным столбцом / could follow the specified preceding column
- Становится последним, если место не указано / It would be the last column if the place is not specified

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**ALTER TABLE** <table\_name> **ADD [CONSTRAINT**  
    <key\_name>] **PRIMARY KEY** (<columns\_list>);

столбцы, которые будут входить в первичный ключ, должны уже существовать в таблице  
columns included into the PK, should already exist in a table

**ALTER TABLE** supplier **ADD PRIMARY KEY**  
    (supplier\_id)

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Добавление внешнего ключа / Add a foreign key

```
ALTER TABLE <table_name> ADD [CONSTRAINT  
  <fk_name>] FOREIGN KEY [<index_name>]  
  (<columns_list>) REFERENCES KEY  
  [<parent_table_name>] (<pk_columns_list>)  
  [<integrity_rules>]
```

- Добавление индекса / Add index

```
ALTER TABLE <table_name> ADD INDEX  
  [<index_name>] (<columns_list>);
```

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Добавление уникального индекса / Add unique index

```
ALTER TABLE <table_name> ADD [CONSTRAINT  
    <constraint_name>] UNIQUE (<columns_list>);
```

- Добавление полнотекстового индекса / Add full text index

```
ALTER TABLE <table_name> ADD FULLTEXT  
    [<index_name>] (<columns_list>);
```

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Изменение описания столбца / Change column description

**ALTER TABLE** <table\_name> **CHANGE** <old\_name>  
<new\_name> <new\_type> [<column\_properties>]  
[**FIRST** | **AFTER** <preceding\_column\_name>];

- Изменение описания столбца без переименования / Change column description without rename

**ALTER TABLE** <table\_name> **MODIFY** <column\_name>  
<new\_type> [<column\_properties>] [**FIRST** | **AFTER**  
<preceding\_column\_name>];

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

### Значения по умолчанию / Default values

- Установить значение по умолчанию / Set the default value

**ALTER TABLE <table\_name> ALTER <column\_name>  
SET DEFAULT <default\_value>;**

- Удалить значение по умолчанию / Delete the default value

**ALTER TABLE <table\_name> ALTER <column\_name>  
DROP DEFAULT;**

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

Удаление столбца / Delete column

**ALTER TABLE** <table\_name> **DROP** <column\_name>;

- столбец удаляется из всех индексов / column is deleted from all indexes
- первичный и внешние ключи нужно удалить прежде, чем удалять входящие в них столбцы / PK and FK should be removed before included columns would be removed
- пустой индекс удаляется автоматически / empty index is removed automatically



## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

**ALTER TABLE <table\_name> DROP PRIMARY KEY;**

**ALTER TABLE <table\_name> DROP FOREIGN KEY <fk\_name>;**

имя внешнего ключа, если не было указано, генерируется автоматически

if the FK name was not specified it would be generated automatically

**ALTER TABLE <table\_name> DROP INDEX <index\_name>;**

**SHOW CREATE TABLE <table\_name>;**

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Переименование таблицы / Rename table

**ALTER TABLE** <table\_name> **RENAME**  
<new\_table\_name>;

- Изменение кодировки и правила сравнения / Change the character set and collation

**ALTER TABLE** <table\_name> **CHARACTER SET**  
<charset\_name> [**COLLATE**  
<collation\_name>];

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

- Преобразовать в новую кодировку существующие столбцы / Convert existing columns to the new character set

**ALTER TABLE** <table\_name> **CONVERT TO CHARACTER SET** <charset\_name> [**COLLATE** <collation\_name>];

- Удаление таблицы / Delete a table
- DROP TABLE** <table\_name>;

## 2.3 Создание таблиц и работа со структурой БД / Creating tables and working with the DB structure

Основные операции с таблицами

Basic operations on tables

- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- SHOW TABLES
- DESCRIBE
- SHOW CREATE TABLE

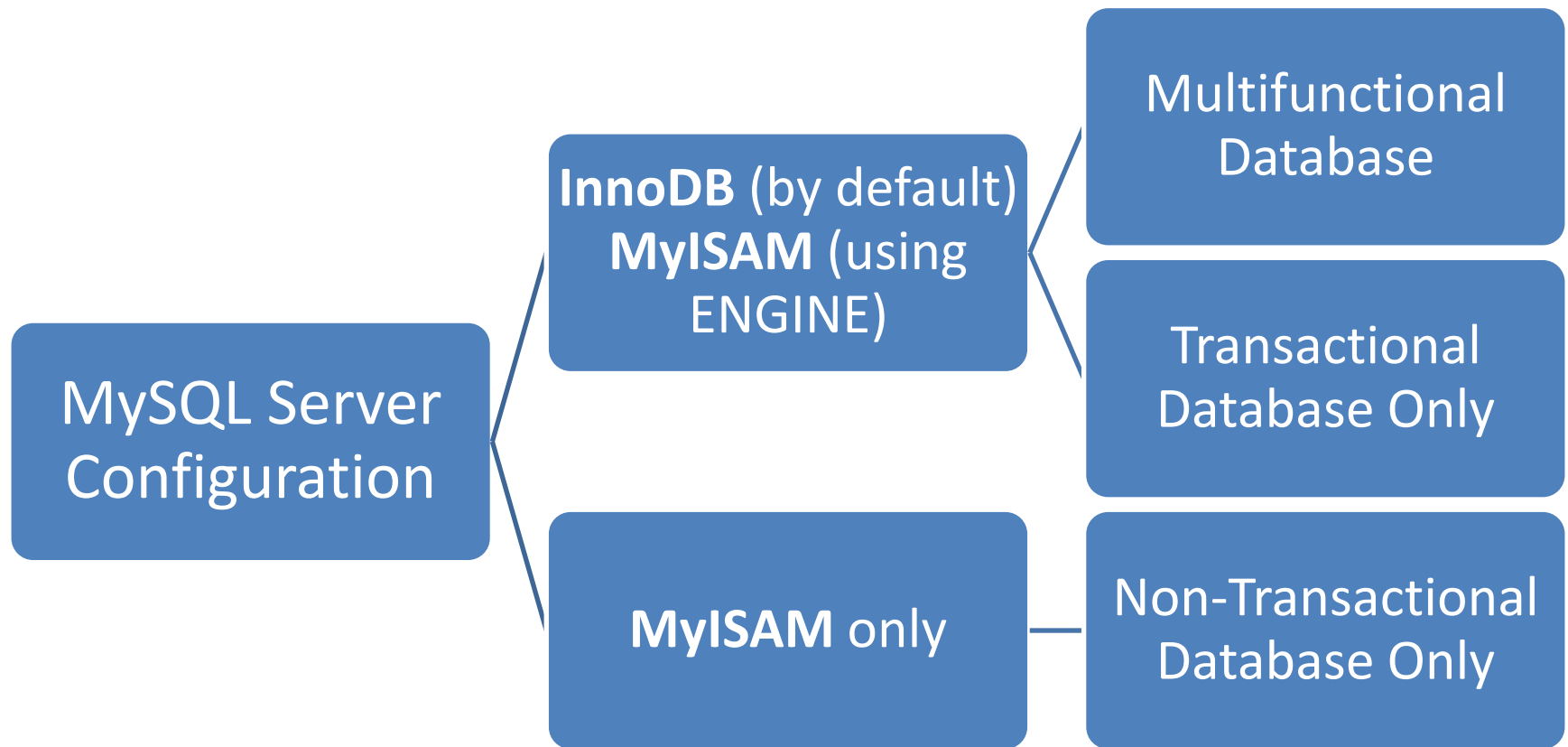
## 2.4 Типы таблиц в MySQL / Table types in MySQL

**ENGINE** <table\_type>

Основные типы таблиц / Main table types

- **InnoDB** – поддерживает транзакции, внешние ключи, каскадное удаление и блокировки на уровне строк / supports transactions, cascade delete, and row-based locks
- **MyISAM** – очень быстрая работа, поддерживает полнотекстовую индексацию / extremely high performance, supports full text indexing

## 2.4 Типы таблиц в MySQL / Table types in MySQL



## 2.4 Типы таблиц в MySQL / Table types in MySQL

- **BDB (Berkeley DB)** - нереляционный механизм, хранит пары «ключ-значения» / non-relational mechanism used to store “key-value” pairs
- **MEMORY** – таблицы целиком хранятся в оперативной памяти / tables are completely stored in RAM
- **MERGE** – объединение нескольких таблиц MyISAM с одной структурой / merge several MyISAM tables with the similar structure

## 2.4 Типы таблиц в MySQL / Table types in MySQL

- **NDB Cluster** – распределение таблиц между несколькими компьютерами / distribution of tables across multiple computers
- **ARCHIVE** – хранение большого объема данных в сжатом формате, поддержка только SELECT и INSERT / stores huge amount of data using a compressed format, supports only SELECT and INSERT
- **CSV** – текстовый файл / text file
- **FEDERATED** – данные в таблицах хранятся на другом компьютере в сети / tables data is stored in another computer in a network



# 3 Работа с данными при помощи SQL

## 3 Data manipulation using SQL

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

- Добавление в таблицу одной или нескольких строк / Add one or multiple records into a table

**INSERT INTO** <table\_name>

[(<columns\_list>)]

**VALUES**

(<values\_list\_1>),

(<values\_list\_2>),

...

(<values\_list\_N>);

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### Основные параметры команды INSERT / Basic parameters of the INSERT statement

- имя таблицы / table name
- список имен столбцов / list of columns names
  - если столбец не включен в список, при добавлении строки будет установлено значение по умолчанию
  - if column is not included into a list, a default value will be set after record insert
- значения, добавляемые в таблицу / values inserted into a table

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

Формат добавляемых значений / Inserted values  
format

- набор значений для одной строки заключается в скобки / a values set for a single row is provided in parentheses
- набор значений должен соответствовать списку столбцов / a values set should correspond to a list of columns
- если список столбцов не указан – списку всех столбцов таблицы / if a list of columns is not specified, it should correspond to a list of all table's columns (**DESCRIBE**)

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

- значения внутри набора и сами наборы отделяются запятыми / values inside a set, as well as sets are separated using commas
- символьные значения и значения даты/времени приводятся в одинарных кавычках / character values and date/time values are shown using single quotes
- для числовых значений кавычки не обязательны / quotes are not necessary for numeric values
- десятичным разделителем для чисел с дробной частью служит точка / dot symbol is used as a separator for numeric values with fractional parts

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

- время и даты вводятся в форматах / time and dates should be formatted as  
“YYYY-MM-DD”, “HH:MM:SS”
- чтобы ввести неопределенное значение, необходимо использовать ключевое слово **NULL** / the keyword **NULL** should be used to insert unidentified value
- вместо значения можно указать ключевое слово **DEFAULT** / the keyword **DEFAULT** should be used instead of a value


### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

```
INSERT INTO supplier (supplier_id,  
    supplier_address, supplier_phone) VALUES  
(1, 'Kharkiv, Nauky av., 55, apt. 108', 'phone: 32-18-  
    44'),  
(2, 'Kyiv, Peremohy av., 154, apt. 3', ''),  
(3, 'Kharkiv, Pushkinska str., 77', 'phone: 33-33-44,  
    fax: 22-12-33'),  
(4, 'Odesa, Derebasivska str., 75', ''),  
(5, 'Poltava, Soborna str., 15, apt. 43', '');
```

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

USE supply;

SELECT \* FROM supplier;

supplier (3×5)		
 supplier_id	supplier_address	supplier_phone
1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
2	Kyiv, Peremohy av., 154, apt. 3	
3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax
4	Odesa, Derebasivska str., 75	
5	Poltava, Soborna str., 15, apt. 43	



### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

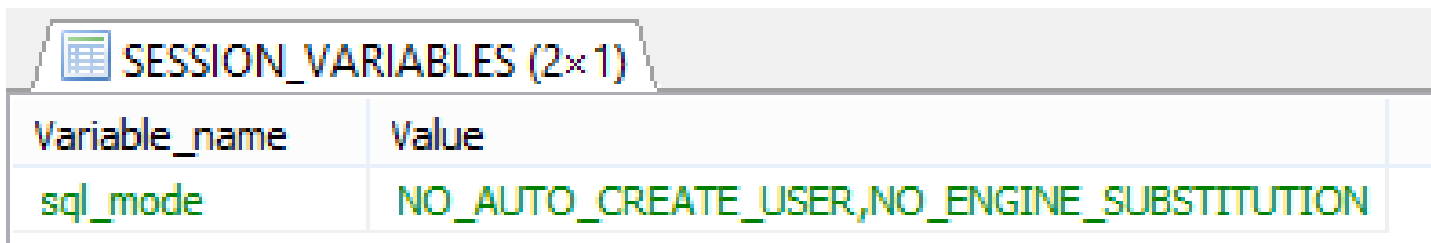
Результат команды INSERT в случае наличия некорректных значений зависит от режима взаимодействия клиентского приложения с сервером MySQL

A result of execution of INSERT statement with incorrect values depends on a mode according to which a client application is interacting with a MySQL server

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

- Проверить текущий режим взаимодействия / Check current interaction mode

SHOW VARIABLES LIKE 'sql\_mode';



The screenshot shows a window titled 'SESSION\_VARIABLES (2x1)'. It contains a table with two columns: 'Variable\_name' and 'Value'. The first row shows 'sql\_mode' with the value 'NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION'.

Variable_name	Value
sql_mode	NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION

Отсутствие в переменной sql\_mode ключевых слов  
STRICT\_TRANS\_TABLES и STRICT\_ALL\_TABLES = **нестрогий**  
режим

If the variable sql\_mode does not contain keywords  
STRICT\_TRANS\_TABLES and STRICT\_ALL\_TABLES = **non strict**  
mode

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

В **нестрогом** режиме вставляемое некорректное значение преобразуется в допустимое

In a **non strict** mode inserted incorrect value is transformed into allowed value

- некорректная дата заменяется нулевой / incorrect date is replaced by a zero-date

0000-00-00 00:00:00

- лишние символы в символьном значении отбрасываются / redundant symbols of a character value are discarded

“abc” => CHAR (2) => “ab”

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

- внесение в числовой столбец символьного значения / inserting a character value into a numeric column

“123abc” => INT => 123

“abc123” => INT => 0

- для NOT NULL столбца не указано значение и не задано значение по умолчанию / both inserted and default values are not specified for a NOT NULL table

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

Тип данных столбца Column data type	Вставляемое значение	Inserted value
Числовой Numeric	0 очередной порядковый номер	0 next ordinal value
Дата и время Date and time	нулевая дата и время текущая дата и время	zero date and time current date and time
Символьный Character	пустая строка	empty string
Перечисление ENUM	первый из элементов списка	first element from a list of values

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

При работе в **нестрогом** режиме операция добавления завершается успешно и генерируется предупреждение

When working in a **non strict** mode, insert operation is finished successfully and a warning is generated

### **SHOW WARNINGS;**

Попытка добавить повторяющееся значение в столбец первичного ключа или уникального индекса вызывает ошибку в любом из режимов

An attempt to insert duplicate value into a primary key column or unique index causes an error in all modes

То же самое касается FK / The same is also for FK (**InnoDB**)

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

sql\_mode                      STRICT\_TRANS\_TABLES  
                                 STRICT\_ALL\_TABLES

сервер работает в **строгом** режиме / server works in a **strict** mode

InnoDB – режимы эквивалентны / modes are equal

- операция INSERT полностью отменяется / INSERT operation is completely cancelled
- выдается сообщение об ошибке / error message is displayed

Числа с дробной частью округляются до целого в любом режиме без ошибок и предупреждений

Decimal values are transformed into integers without errors and warnings in both modes

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### Строгий режим / Strict mode

- необходимо задавать значения для столбцов без свойства DEFAULT / it is required to define values for columns without the DEFAULT property
- TIMESTAMP, ENUM, AUTO\_INCREMENT – в случае отсутствия значений, используются те же значения, что и в нестрогом режиме / if a value is not specified, the same values are used as in a non strict mode



### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

Изменить режим взаимодействия клиента с сервером / Change client-server interaction mode

SET SQL\_MODE = '<mode>;

- Установить нестрогий режим / Set a non strict mode

SET SQL\_MODE = '';

- Установить строгий режим / Set a strict mode

SET SQL\_MODE = 'STRICT\_TRANS\_TABLES';

SET SQL\_MODE = 'STRICT\_ALL\_TABLES;

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### **SET SQL\_MODE**

- изменяет режим взаимодействия с сервером только для текущего соединения / changes server interaction mode only for a current connection
- не влияет на взаимодействие сервера с другими клиентами / do not affect other sessions
- сохраняется до момента отключения от сервера / persists until disconnected from server

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### **SET GLOBAL SQL\_MODE**

- применяется для всех вновь подключаемых к серверу клиентов / applies to all new clients connected to a server
- ранее подключенные клиенты продолжают работать в прежнем режиме / previously connected clients continue to work in their modes
- сохраняется до перезапуска сервера MySQL / remains until a MySQL server is restarted

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### **UPDATE**

установить новые значения в одной или нескольких строках / set new values for one or several rows

**UPDATE** <table\_name>

**SET** <column\_name\_1> = <value\_1>,

...,

<column\_name\_N> = <value\_N>

[**WHERE** <condition>]

[**ORDER BY** <column\_name> [**ASC** | **DESC**]]

[**LIMIT** <rows\_number>]



### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

UPDATE supplied

SET supplied\_cost = supplied\_cost \* 0.95

WHERE contract\_number = 4 AND supplied\_product =  
'Printer';

SELECT \* FROM supplied WHERE contract\_number = 4  
AND supplied\_product = 'Printer';

supplied (4×1)			
 contract_number	 supplied_product	supplied_amount	supplied_cost
4	Printer	41	332.50

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### Основные параметры / Main parameters

- имя таблицы / table name
- список столбцов и новых значений / a list of columns and new values

**SET** <column\_name\_1> = <value\_1>, ...,  
<column\_name\_N> = <value\_N>

- можно использовать прежние значения в строке / it is possible to use previous values of a record
- условие отбора / filtering condition

**WHERE** <condition>

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

- в условиях отбора можно использовать вложенный запрос / a nested query can be used in a filtering condition
- порядок применения изменений к строкам таблицы / order of applying changes to table rows

**ORDER BY** <column\_name> [**ASC** | **DESC**]

- предельное количество изменяемых строк / set a limit number of changing records

**LIMIT** <rows\_number>

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### **REPLACE**

добавление либо замещение строк таблицы / inserts or updated table records

**REPLACE INTO** <table\_name>

[(<columns\_list>)]

### **VALUES**

(<values\_list\_1>),

(<values\_list\_2>),

...

(<values\_list\_N>);



## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

### **REPLACE == INSERT**

значение первичного ключа или уникального индекса  
не совпадает ни с одним из уже существующих  
значений

the value of the primary key or unique index does not  
match any of the existing values

в противном случае, перед добавлением новой строки  
прежняя строка удаляется

otherwise, the previous row is deleted before adding a new  
row

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

REPLACE INTO supplied  
VALUES

(4, 'Printer', 41, 332.50);

- нельзя задавать новые значения, вычисляемые с использованием прежних значений / you cannot set new values using previous values
- иначе будет подставлено значение по умолчанию / otherwise, a default value will be set

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

#### **DELETE**

Удаление строк таблицы / Remove table records

**DELETE FROM** <table>

[**WHERE** <condition>]

[**ORDER BY** <column\_name> [**ASC** | **DESC**]]

[**LIMIT** <rows\_number>];

DELETE FROM supplied WHERE contract\_number = 5;

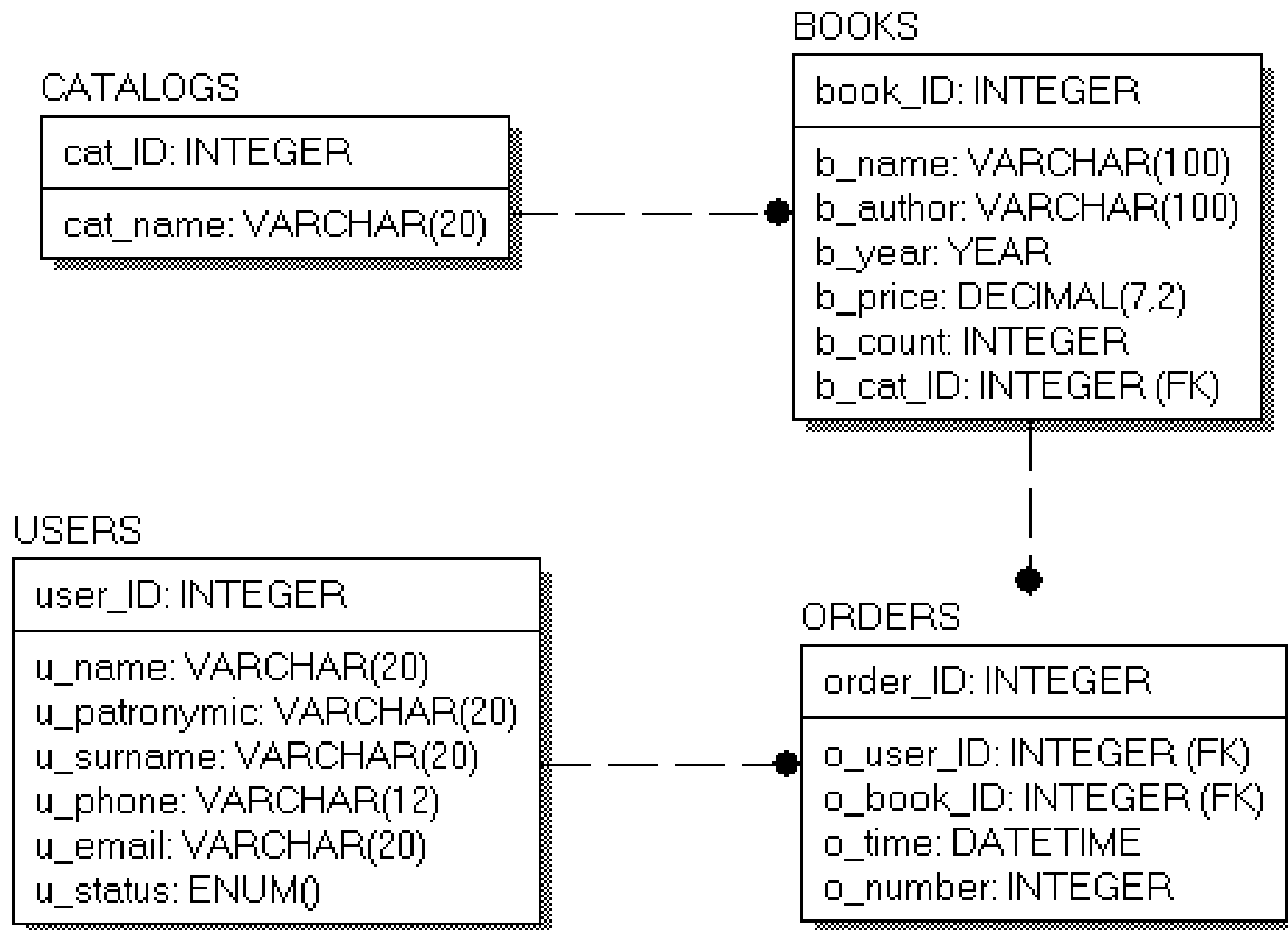
## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

Операции вставки, удаления и изменения  
строк таблицы

Operations used to insert, delete, and modify  
table rows

- INSERT
- UPDATE
- REPLACE
- DELETE

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data



## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

```
DROP DATABASE IF EXISTS book;

CREATE DATABASE book
CHARACTER SET utf8
COLLATE utf8_general_ci;

USE book;

CREATE TABLE catalogs (
    cat_ID int(6) NOT NULL
    AUTO_INCREMENT,
    cat_name varchar(20) NOT NULL,
    PRIMARY KEY (cat_ID)
) ENGINE=InnoDB;

CREATE TABLE books (
    book_ID int(6) NOT NULL
    AUTO_INCREMENT,
    b_name varchar(100) NOT NULL,
    b_author varchar(100) NOT NULL,
    b_year year NOT NULL,
    b_price decimal(7,2) NULL default
    '0.00',
    b_count int(6) NULL default '0',
    b_cat_ID int(6) NOT NULL default '0',
    PRIMARY KEY (book_ID),
    FOREIGN KEY (b_cat_ID) REFERENCES
    catalogs (cat_ID) ON DELETE
    CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;
```

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

```
CREATE TABLE users (  
    user_ID int(6) NOT NULL  
    AUTO_INCREMENT,  
    u_name varchar(20) NOT NULL,  
    u_patronymic varchar(20) NOT NULL,  
    u_surname varchar(20) NOT NULL,  
    u_phone varchar(12) NULL,  
    u_email varchar(20) NULL,  
    u_status ENUM  
    ('active','passive','lock','gold')  
    default 'passive',  
    PRIMARY KEY (user_ID)  
) ENGINE=InnoDB;
```

```
CREATE TABLE orders (  
    order_ID int(6) NOT NULL  
    AUTO_INCREMENT,  
    o_user_ID int NOT NULL,  
    o_book_ID int NOT NULL,  
    o_time datetime NOT NULL default  
    '0000-00-00 00:00:00',  
    o_number int(6) NOT NULL default '0',  
    PRIMARY KEY (order_ID),  
    FOREIGN KEY (o_book_ID)  
    REFERENCES books(book_ID) ON  
    DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (o_user_ID)  
    REFERENCES users (user_ID) ON  
    DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB;
```

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

catalogs (2×5)		catalogs (2×2)	
cat_ID	cat_name		
1	Программирование		
2	Интернет		
3	Базы данных		
4	Сети		
5	Мультимедиа		

catalogs (2×5)		catalogs (2×2)	
cat_ID	cat_name		
1	Программирование		
2	Интернет		

```
SELECT * FROM catalogs;
```

```
DELETE FROM catalogs WHERE cat_ID > 2;
```

```
SELECT * FROM catalogs;
```



### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

catalogs (2×5)		catalogs (2×0)	
cat_ID	cat_name		
1	Программирование		
2	Интернет		
3	Базы данных		
4	Сети		
5	Мультимедиа		




```
SELECT * FROM catalogs;
```

```
DELETE FROM catalogs;
```

```
SELECT * FROM catalogs;
```

catalogs (2×5)		catalogs (2×0)	
cat_ID	cat_name		




## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

orders (5×5)		orders (5×2)			
 order_ID	 o_user_ID	 o_book_ID	o_time	o_number	
1	3	8	2009-01-04 10:39:38	1	
2	6	10	2009-02-10 09:40:29	2	
3	1	20	2009-02-18 13:41:05	4	
4	4	20	2009-03-10 18:20:00	1	
5	3	20	2009-03-17 19:15:36	1	




```
SELECT * FROM orders;
```

```
DELETE FROM orders LIMIT 3;
```

```
SELECT * FROM orders;
```

orders (5×5)		orders (5×2)			
 order_ID	 o_user_ID	 o_book_ID	o_time	o_number	
4	4	20	2009-03-10 18:20:00	1	
5	3	20	2009-03-17 19:15:36	1	




## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

orders (5×5)		orders (5×2)			
 order_ID	 o_user_ID	 o_book_ID	o_time	o_number	
1	3	8	2009-01-04 10:39:38	1	
2	6	10	2009-02-10 09:40:29	2	
3	1	20	2009-02-18 13:41:05	4	
4	4	20	2009-03-10 18:20:00	1	
5	3	20	2009-03-17 19:15:36	1	

```
SELECT * FROM orders;
```

```
TRUNCATE TABLE orders;
```

```
SELECT * FROM orders;
```

orders (5×5)		orders (5×0)		
 order_ID	 o_user_ID	 o_book_ID	o_time	o_number

### 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

catalogs (2×5)	
cat_ID	cat_name
1	Программирование
2	Интернет
3	Базы данных
4	Сети
5	Мультимедиа

catalogs (2×5)	
cat_ID	cat_name
1	Программирование
2	Интернет
3	Базы данных
4	Компьютерные сети
5	Мультимедиа

```
SELECT * FROM catalogs;
```

```
UPDATE catalogs SET cat_name = 'Компьютерные сети'  
WHERE cat_name = 'Сети';
```

```
SELECT * FROM catalogs;
```

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

books (7×30)		books (7×30)				
book_ID	b_name	b_author	b_year	b_price	b_count	b_cat_ID
1	JavaScript в кармане	Рева О.Н.	2008	42.00	10	1
2	Visual FoxPro 9.0	Клепинин В.Б.	2007	660.00	2	1
3	C++ Как он есть	Тимофеев В.В.	2009	218.00	4	1
4	Создание приложений с помощью C#	Фаронов В.В.	2008	169.00	1	1
5	Delphi. Народные советы	Шкрыль А.А.	2007	243.00	6	1
6	Delphi. Полное руководство	Сухарев М.	2008	500.00	6	1
7	Профессиональное программирование на PHP	Шлосснейгл Дж.	2006	309.00	5	1
8	Совершенный код	Макконнелл С.	2007	771.00	1	1
9	Практика программирования	Керниган Б.	2004	214.00	12	1

```
SELECT * FROM books;
```

```
UPDATE books SET b_price = b_price * 0.95;
```

```
SELECT * FROM books;
```

books (7×30)		books (7×30)				
book_ID	b_name	b_author	b_year	b_price	b_count	b_cat_ID
1	JavaScript в кармане	Рева О.Н.	2008	39.90	10	1
2	Visual FoxPro 9.0	Клепинин В.Б.	2007	627.00	2	1
3	C++ Как он есть	Тимофеев В.В.	2009	207.10	4	1
4	Создание приложений с помощью C#	Фаронов В.В.	2008	160.55	1	1
5	Delphi. Народные советы	Шкрыль А.А.	2007	230.85	6	1
6	Delphi. Полное руководство	Сухарев М.	2008	475.00	6	1
7	Профессиональное программирование на PHP	Шлоссейгл Дж.	2006	293.55	5	1
8	Совершенный код	Макконнелл С.	2007	732.45	1	1
9	Практика программирования	Керниган Б.	2004	203.30	12	1

## 3.1 Вставка, удаление и обновление данных / Insert, delete, and update data

books (7×30)		books (7×30)				
book_ID	b_name	b_author	b_year	b_price	b_count	b_cat_ID
1	JavaScript в кармане	Рева О.Н.	2008	42.00	10	1
2	Visual FoxPro 9.0	Клепинин В.Б.	2007	660.00	2	1
3	C++ Как он есть	Тимофеев В.В.	2009	218.00	4	1
4	Создание приложений с помощью C#	Фаронов В.В.	2008	169.00	1	1
5	Delphi. Народные советы	Шкрыль А.А.	2007	243.00	6	1
6	Delphi. Полное руководство	Сухарев М.	2008	500.00	6	1
7	Профессиональное программирование на PHP	Шлоссейгл Дж.	2006	309.00	5	1
8	Совершенный код	Макконнелл С.	2007	771.00	1	1
9	Практика программирования	Керниган Б.	2004	214.00	12	1

```
SELECT * FROM books;
```

```
UPDATE books SET b_price = b_price * 0.95, b_count = b_count - 1;
```

```
SELECT * FROM books;
```

books (7×30)		books (7×30)				
book_ID	b_name	b_author	b_year	b_price	b_count	b_cat_ID
1	JavaScript в кармане	Рева О.Н.	2008	39.90	9	1
2	Visual FoxPro 9.0	Клепинин В.Б.	2007	627.00	1	1
3	C++ Как он есть	Тимофеев В.В.	2009	207.10	3	1
4	Создание приложений с помощью C#	Фаронов В.В.	2008	160.55	0	1
5	Delphi. Народные советы	Шкрыль А.А.	2007	230.85	5	1
6	Delphi. Полное руководство	Сухарев М.	2008	475.00	5	1
7	Профессиональное программирование на PHP	Шлосснейгл Дж.	2006	293.55	4	1
8	Совершенный код	Макконнелл С.	2007	732.45	0	1
9	Практика программирования	Керниган Б.	2004	203.30	11	1

## 3.2 Создание запросов на выборку / Creating select queries

### **SELECT**

команда для получения данных из таблиц базы  
данных

the command used to retrieve data from database  
tables

- вывести все данные таблицы / display all table  
data

**SELECT \* FROM** <table\_name>;


## 3.2 Создание запросов на выборку / Creating select queries

- можно указать список столбцов таблицы / it is possible to specify a list of table columns

**SELECT** <column\_name\_1>, ..., <column\_name\_N> **FROM**  
<table\_name>;

SELECT \* FROM supplier;

SELECT supplier\_id, supplier\_address, supplier\_phone FROM  
supplier;

supplier (3×5)		
 supplier_id	supplier_address	supplier_phone
1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
2	Kyiv, Peremohy av., 154, apt. 3	
3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax
4	Odesa, Derebasivska str., 75	
5	Poltava, Soborna str., 15, apt. 43	



## 3.2 Создание запросов на выборку / Creating select queries

- можно получать значения, вычисленные с помощью выражений / it is possible to get calculated values

SELECT supplied\_product, supplied\_amount,  
supplied\_cost / 27 FROM supplied;

supplied (3×17)		
supplied_product	supplied_amount	supplied_cost / 27
Audio Player	25	25.925926
TV	10	48.148148
Video Player	12	27.777778
Audio Player	5	16.666667
Stereo System	11	18.518519
Video Player	8	16.666667
Audio Player	11	20.370370
Monitor	85	20.370370
TV	52	33.333333

## 3.2 Создание запросов на выборку / Creating select queries

- можно вычислять значения без обращения к таблице / it is possible to calculate values without accessing any table

SELECT 2 \* 2;

- **DISTINCT** – исключить повторяющиеся строки / exclude repeating records

SELECT supplied\_product | SELECT DISTINCT supplied\_product  
FROM supplied;

supplied (1×17)	
supplied_product	
Audio Player	
TV	
Video Player	
Audio Player	
Stereo System	
Video Player	
Audio Player	
Monitor	

supplied (1×7)	
supplied_product	
Audio Player	
TV	
Video Player	
Stereo System	
Monitor	
Printer	
Phone	

## 3.2 Создание запросов на выборку / Creating select queries

- выведенные строки можно упорядочить по одному из столбцов / selected records can be sorted by one of the columns

**ORDER BY** <column\_name> [**ASC** | **DESC**]

```
SELECT supplied_product,  
       supplied_amount,  
       supplied_cost  
FROM supplied  
ORDER BY  
       supplied_amount DESC,  
       supplied_cost;
```

supplied (3×17)		
supplied_product	supplied_amount	supplied_cost
TV	14	860.00
Video Player	12	750.00
Stereo System	11	500.00
Audio Player	11	550.00
TV	10	1,300.00
TV	10	2,999.00
Video Player	8	450.00
Audio Player	5	450.00
Phone	5	5,999.00

## 3.2 Создание запросов на выборку / Creating select queries

- вместо имен столбцов можно использовать их порядковые номера / ordinal numbers of columns can be used instead of their names

SELECT supplied\_product,

supplied\_amount,

supplied\_cost

FROM supplied

ORDER BY 2 DESC, 3;

supplied (3×17)		
supplied_product	supplied_amount	supplied_cost
TV	14	860.00
Video Player	12	750.00
Stereo System	11	500.00
Audio Player	11	550.00
TV	10	1,300.00
TV	10	2,999.00
Video Player	8	450.00
Audio Player	5	450.00
Phone	5	5,999.00

## 3.2 Создание запросов на выборку / Creating select queries

- объединение таблиц / join tables

**SELECT** <columns\_list> **FROM** <tables\_list>

**WHERE** <condition>;

SELECT contract\_note, supplied\_product, supplied\_amount, supplied\_cost  
FROM supplied, contract

WHERE contract.contract\_number = supplied.contract\_number;

Result #1 (4×17)			
contract_note	 supplied_product	supplied_amount	supplied_cost
Order 34 on 30.08.2018	Audio Player	25	700.00
Order 34 on 30.08.2018	TV	10	1,300.00
Order 34 on 30.08.2018	Video Player	12	750.00
Invoice 08-78 on 28.08.2018	Audio Player	5	450.00
Invoice 08-78 on 28.08.2018	Stereo System	11	500.00
Invoice 08-78 on 28.08.2018	Video Player	8	450.00
Order 56 on 28.08.2018	Audio Player	11	550.00
Order 56 on 28.08.2018	Monitor	85	550.00
Order 56 on 28.08.2018	TV	52	900.00

## 3.2 Создание запросов на выборку / Creating select queries

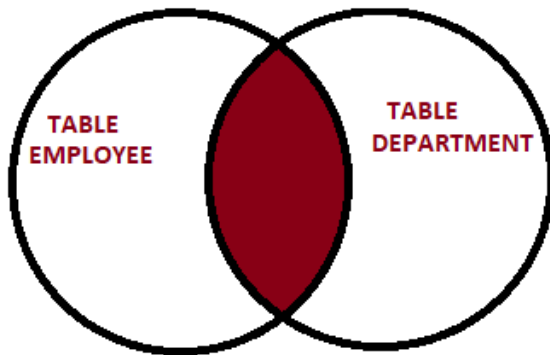
- объединение таблиц с помощью USING / join tables with USING  
**SELECT** <columns\_list> **FROM** <table1> **JOIN** <table2> **JOIN** ...  
**USING** (<table1\_id>, <table2\_id>, ...);

```
SELECT contract_note, supplied_product, supplied_amount, supplied_cost  
FROM supplied INNER JOIN contract  
USING (contract_number);
```

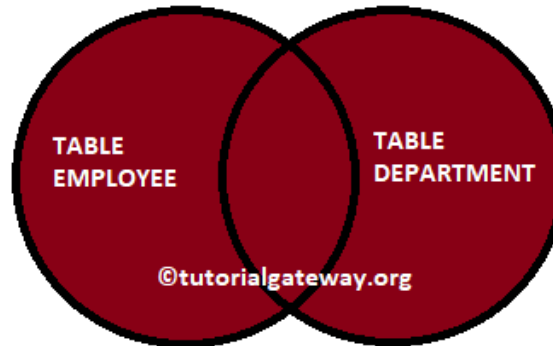
Result #1 (4×17)			
contract_note	 supplied_product	supplied_amount	supplied_cost
Order 34 on 30.08.2018	Audio Player	25	700.00
Order 34 on 30.08.2018	TV	10	1,300.00
Order 34 on 30.08.2018	Video Player	12	750.00
Invoice 08-78 on 28.08.2018	Audio Player	5	450.00
Invoice 08-78 on 28.08.2018	Stereo System	11	500.00
Invoice 08-78 on 28.08.2018	Video Player	8	450.00
Order 56 on 28.08.2018	Audio Player	11	550.00
Order 56 on 28.08.2018	Monitor	85	550.00
Order 56 on 28.08.2018	TV	52	900.00

## 3.2 Создание запросов на выборку / Creating select queries

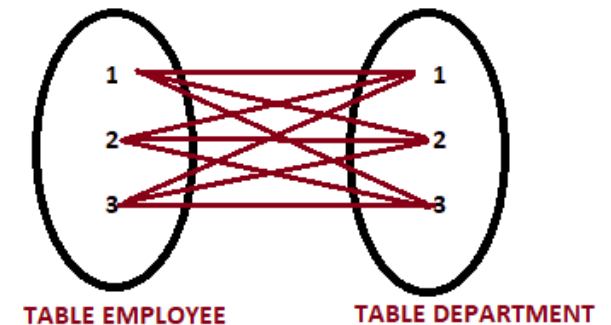
INNER JOIN EXAMPLE



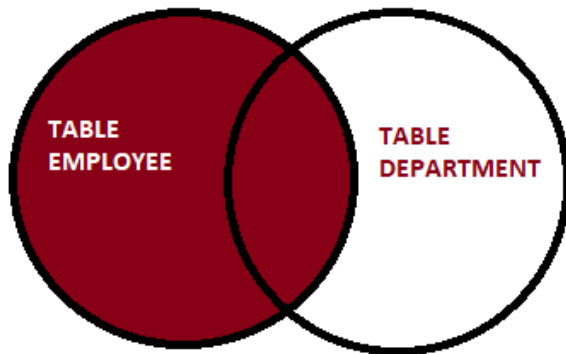
FULL JOIN EXAMPLE



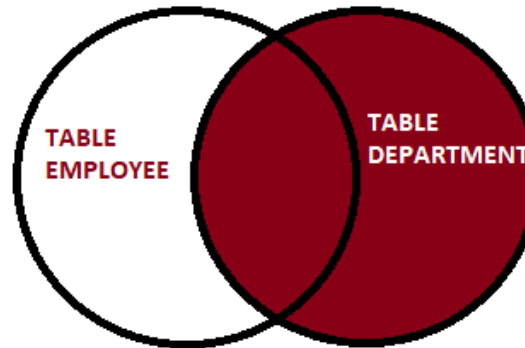
CROSS JOIN EXAMPLE



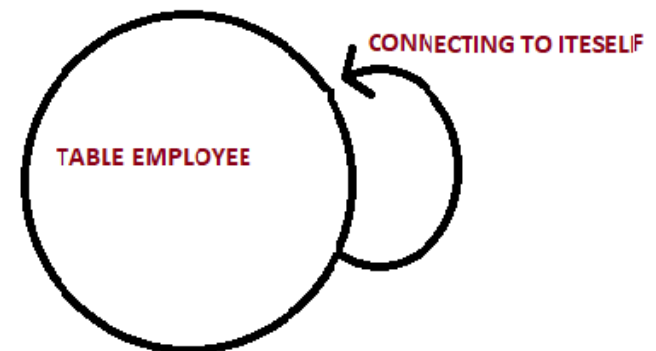
LEFT JOIN EXAMPLE



RIGHT JOIN EXAMPLE



SELF JOIN EXAMPLE



## 3.2 Создание запросов на выборку / Creating select queries

- таблицу можно объединить саму с собой / a table might be join with itself

```
SELECT L.supplied_product,  
       R.supplied_product  
FROM supplied L,  
       supplied R  
WHERE  
       L.supplied_cost =  
       R.supplied_cost
```

supplied (2×21)	
supplied_product	supplied_product
Audio Player	Audio Player
TV	TV
Video Player	Video Player
Audio Player	Audio Player
Video Player	Audio Player
Stereo System	Stereo System
Audio Player	Video Player
Video Player	Video Player
Audio Player	Audio Player



## 3.2 Создание запросов на выборку / Creating select queries

Дополнительное условие отбора, чтобы избавиться от повторений

Additional filtering condition to avoid repeating values

```
SELECT L.supplied_product,
```

```
       R.supplied_product
```

```
FROM supplied L,
```

```
     supplied R
```

```
WHERE
```

```
    L.supplied_cost = R.supplied_cost AND
```

```
    L.supplied_product <> R.supplied_product
```

supplied (2×4)	
supplied_product	supplied_product
Video Player	Audio Player
Audio Player	Video Player
Monitor	Audio Player
Audio Player	Monitor

## 3.2 Создание запросов на выборку / Creating select queries

- результат одного запроса можно использовать в другом запросе / a query result can be used in another query

SELECT \* FROM supplied

WHERE supplied\_amount = (SELECT MAX(supplied\_amount)  
FROM supplied)

supplied (4×1)		supplied (4×1)	
contract_number	supplied_product	supplied_amount	supplied_cost
3	Monitor	85	550.00

SELECT \* FROM supplied

ORDER BY supplied\_amount DESC

LIMIT 1

supplied (4×1)		supplied (4×1)	
contract_number	supplied_product	supplied_amount	supplied_cost
3	Monitor	85	550.00

## 3.2 Создание запросов на выборку / Creating select queries

- результаты нескольких запросов можно объединить / results of several queries can be combined

```
SELECT * FROM supplied
```

```
WHERE supplied_amount = (SELECT MAX(supplied_amount) FROM supplied)
```

**UNION**

```
SELECT * FROM supplied
```

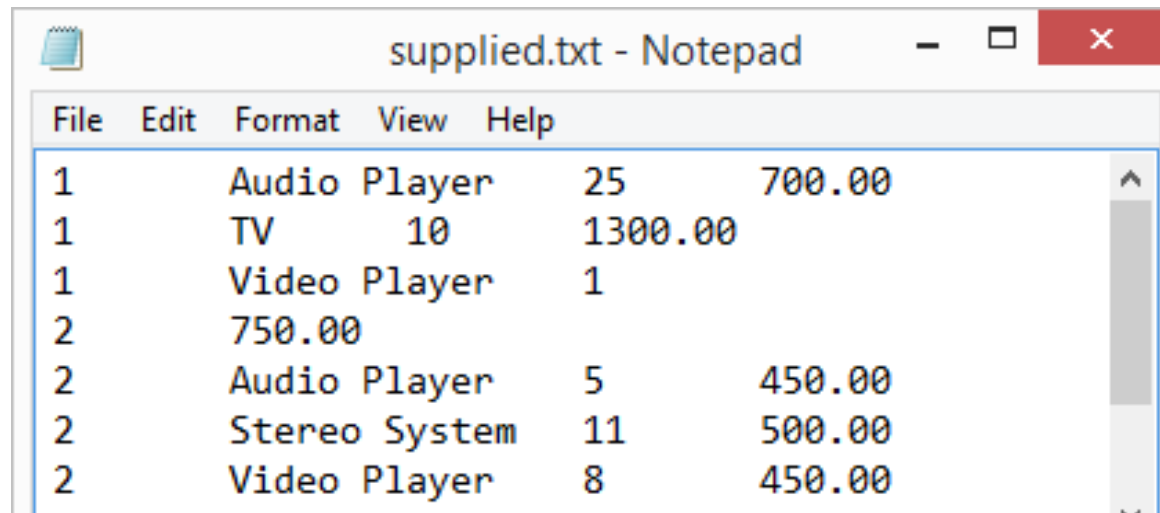
```
WHERE supplied_amount = (SELECT MIN(supplied_amount) FROM supplied)
```

Result #1 (4×3)			
contract_number	supplied_product	supplied_amount	supplied_cost
3	Monitor	85	550.00
2	Audio Player	5	450.00
7	Phone	5	5,999.00

## 3.2 Создание запросов на выборку / Creating select queries

- результат запроса можно сохранить в файл  
/ a query result can be saved to a file

**SELECT \* FROM supplied INTO OUTFILE**  
**'D:/supplied.txt'**



The screenshot shows a Notepad window with the title 'supplied.txt - Notepad'. The window contains a table with 5 columns. The first column contains integers (1, 1, 1, 2, 2, 2, 2). The second column contains product names (Audio Player, TV, Video Player, Audio Player, Stereo System, Video Player). The third column contains integers (25, 10, 1, 5, 11, 8). The fourth column contains decimal values (700.00, 1300.00, 750.00, 450.00, 500.00, 450.00). The table is displayed in a simple text format with no headers.

1	Audio Player	25	700.00
1	TV	10	1300.00
1	Video Player	1	750.00
2	Audio Player	5	450.00
2	Stereo System	11	500.00
2	Video Player	8	450.00

## 3.3 Условия выборки / Conditions

### WHERE

выбор записей, удовлетворяющих  
определенным критериям поиска  
select records that satisfy specific search  
criterias


book.users: 6 rows total (approximately)

Next

 user_ID	u_name	u_patronymic	u_surname	u_phone	u_email	u_status
1	Александр	Валерьевич	Иванов	58-98-78	ivanov@email.ru	active
2	Сергей	Иванович	Лосев	90-57-77	losev@email.ru	passive
3	Игорь	Николаевич	Симонов	95-66-61	simonov@email.ru	active
4	Максим	Петрович	Кузнецов	(NULL)	kuznetsov@email.ru	active
5	Анатолий	Юрьевич	Петров	(NULL)	(NULL)	lock
6	Александр	Александрович	Корнеев	89-78-36	korneev@email.ru	gold

## 3.3 Условия выборки / Conditions

```
SELECT user_ID, u_surname FROM users  
WHERE u_status = 'active';
```

users (2×3)	
 user_ID	u_surname
1	Иванов
3	Симонов
4	Кузнецов

```
SELECT DISTINCT u_status FROM users;
```

users (1×4)	
u_status	
active	
passive	
lock	
gold	

## 3.4 Сортировка / Sorting

```
SELECT * FROM orders ORDER BY o_user_ID;
```

orders (5×5)				
order_ID	o_user_ID	o_book_ID	o_time	o_number
3	1	20	2009-02-18 13:41:05	4
1	3	8	2009-01-04 10:39:38	1
5	3	20	2009-03-17 19:15:36	1
4	4	20	2009-03-10 18:20:00	1
2	6	10	2009-02-10 09:40:29	2

```
SELECT o_time FROM orders ORDER BY o_time DESC;
```

orders (1×5)	
o_time	
2009-03-17 19:15:36	
2009-03-10 18:20:00	
2009-02-18 13:41:05	
2009-02-10 09:40:29	
2009-01-04 10:39:38	

## 3.4 Сортировка / Sorting

- результат выборки – записи в порядке хранения в БД / select result – records in an order according to which they are stored in a DB
- можно сортировать по нескольким столбцам / it is possible to sort by multiple columns
- по умолчанию сортировка выполняется в прямом порядке / records are sorted in an ascending order by default (**ASC**)
- можно отсортировать строки в обратном порядке / it is possible to sort records in a descending order (**DESC**)




## 3.5 Ограничение выборки / Limit

### LIMIT

используется для реализации постраничной навигации  
is used to implement navigation by pages (pagination)

```
SELECT book_ID, b_count FROM books  
ORDER BY b_count DESC  
LIMIT 5;
```

books (2×5)	
 book_ID	b_count
28	20
25	20
26	15
29	12
9	12


## 3.5 Ограничение выборки / Limit

### LIMIT offset, count

**offset** – позиция, начиная с которой необходимо вернуть результат / the offset of the first row to be returned

**count** – число извлекаемых записей / the maximum number of rows to be returned

```
SELECT book_ID, b_count FROM books  
ORDER BY b_count DESC  
LIMIT 5, 5;
```

books (2×5)	
 book_ID	b_count
1	10
27	10
24	8
30	8
19	6

## 3.5 Ограничение выборки / Limit

books (3×30)		
book_ID	b_name	b_price
1	JavaScript в кармане	42.00
2	Visual FoxPro 9.0	660.00
3	C++ Как он есть	218.00
4	Создание приложений с помощью C#	169.00
5	Delphi. Народные советы	243.00
6	Delphi. Полное руководство	500.00
7	Профессиональное программирование на PHP	309.00
8	Совершенный код	771.00
9	Практика программирования	214.00
10	Принципы маршрутизации в Internet	428.00
11	Поиск в Internet	107.00
12	Web-конструирование	177.00
13	Самоучитель Интернет	121.00
14	Популярные интернет-браузеры	82.00
15	Общение в Интернете	85.00
16	Базы данных	326.00
17	Базы данных. Разработка приложений	189.00
18	Раскрытие тайн SQL	200.00
19	Практикум по Access	87.00
20	Компьютерные сети	630.00

SELECT book\_ID,  
b\_name, b\_price  
FROM books  
LIMIT 0, 5

SELECT book\_ID,  
b\_name, b\_price  
FROM books  
LIMIT 5, 5

SELECT book\_ID,  
b\_name, b\_price  
FROM books  
LIMIT 10, 5

SELECT book\_ID,  
b\_name, b\_price  
FROM books  
LIMIT 15, 5

## 3.6 Группировка записей / Group

### **GROUP BY**

позволяет группировать извлекаемые строки

allows to group selected records

- используется вместе с функциями, применяемыми к группам строк / it is used with functions applied to groups of records
- агрегатные функции вычисляют одно значение для каждой группы, создаваемой конструкцией GROUP BY / aggregate functions calculate a single value for each group created by using the GROUP BY statement

## 3.6 Группировка записей / Group

- функции позволяют узнать число строк в группе, подсчитать среднее значение, получить сумму значений столбцов / functions allow to define a number of records in a group, calculate the average value, get the sum of values for columns
- результирующее значение вычисляется для значений, не равных NULL / a result value is calculated for values do not equal to NULL
- функции можно использовать в запросах без группировки / functions might be used in queries without GROUP BY option

## 3.6 Группировка записей / Group

```
SELECT COUNT(DISTINCT b_cat_ID) FROM books;
```

Result #1 (1×1)	
COUNT(DISTINCT b_cat_ID)	
	5

```
SELECT COUNT(DISTINCT b_cat_ID) AS total FROM books;
```

Result #1 (1×1)	
total	
	5

```
SELECT * FROM catalogs WHERE cat_ID = MAX(cat_ID);
```

???

## 3.6 Группировка записей / Group

```
SELECT * FROM catalogs ORDER BY cat_ID DESC LIMIT 1;
```

catalogs (2×1)	
cat_ID	cat_name
5	Мультимедиа


```
SELECT b_cat_ID FROM books  
GROUP BY b_cat_ID ORDER BY b_cat_ID;
```

```
SELECT DISTINCT b_cat_ID FROM books;
```


books (1×5)	
b_cat_ID	
1	
2	
3	
4	
5	

## 3.6 Группировка записей / Group

```
SELECT b_cat_ID, COUNT(b_cat_ID) FROM books  
WHERE b_cat_ID > 2  
GROUP BY b_cat_ID  
ORDER BY b_cat_ID;
```

books (2×3)	
 b_cat_ID	COUNT(b_cat_ID)
3	4
4	5
5	6

```
SELECT b_cat_ID, COUNT(b_cat_ID) AS total FROM books  
GROUP BY b_cat_ID  
HAVING total > 5  
ORDER BY b_cat_ID;
```

books (2×3)	
 b_cat_ID	total
1	9
2	6
5	6



## 3.6 Группировка записей / Group

```
SELECT b_cat_ID, COUNT(b_cat_ID) FROM books  
GROUP BY b_cat_ID  
HAVING b_cat_ID > 2  
ORDER BY b_cat_ID;
```

books (2x3)	
b_cat_ID	COUNT(b_cat_ID)
3	4
4	5
5	6

- **WHERE** – выборка с применением условия и затем группировка результата / selects with a condition and then groups a result
- **HAVING** – группировка результата и затем выборка с применением условия / groups a result and then selects with a condition

## 3.7 Использование функций / Using functions

Встроенные функции MySQL используются для решения специфических задач, возникающих при выборке данных

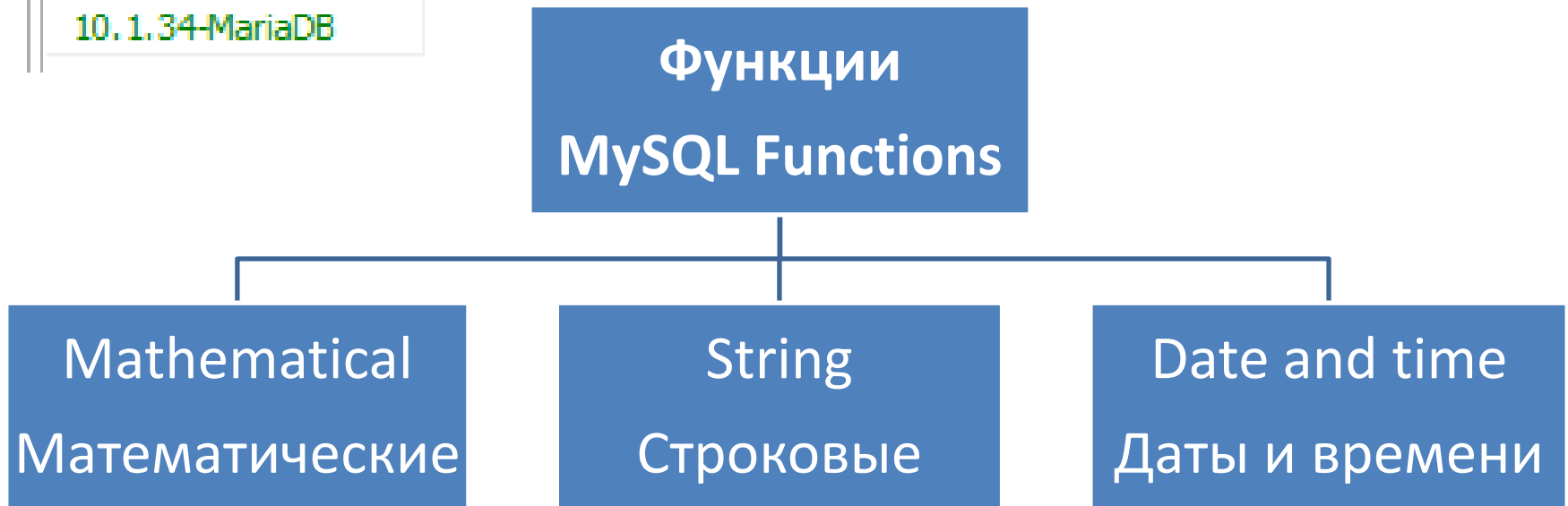
In-built MySQL functions are used to solve specific problems occurred while selecting data

- функция имеет уникальное имя / each function has a unique name
- функция может иметь несколько аргументов / each function might have multiple arguments
- круглые скобки указываются и при отсутствии аргументов / parentheses should be set even if arguments are absent

## 3.7 Использование функций / Using functions

**SELECT VERSION();**

Result #1 (1×1)	
VERSION()	
10.1.34-MariaDB	



## 3.7 Использование функций / Using functions

### Математические функции / Mathematical functions

Функция Function	Описание Description
ABS(X)	Возвращает абсолютное значение аргумента X Returns an absolute value of the argument X
ACOS(X)	Возвращает арккосинус аргумента X или NULL, если значение X не находится в диапазоне от -1 до 1 Returns an arccosine of the argument X or NULL, if the value X does not belong to the range from -1 to 1
ASIN(X)	Возвращает арксинус аргумента X или NULL, если значение X не находится в диапазоне от -1 до 1 Returns an arcsine of the argument X or NULL, if the value X does not belong to the range from -1 to 1

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
ATAN(X)	Возвращает арктангенс аргумента X Returns an arctangent of the argument X
CEIL(X)	Принимает дробный аргумент X и возвращает первое целое число, находящееся справа от аргумента Returns a first left integer number on the right of the fractional argument X
COS(X)	Вычисляет косинус угла X, заданного в радианах Calculates a cosine of the angle X, set in radians
COT(X)	Вычисляет котангенс угла X, заданного в радианах Calculates a cotangent of the angle X, set in radians
DEGREES(X)	Преобразует значение угла X из радиан в градусы Transforms the angle X value from radians to degrees

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
EXP(X)	Вычисляет значение $\exp(x)$ Calculates the value $\exp(x)$
FLOOR(X)	Принимает дробный аргумент X и возвращает первое целое число, находящееся слева от аргумента Returns a first left integer number on the left of the fractional argument X
LN(X)	Вычисляет натуральный логарифм числа X Calculates the logarithm of X
LOG2(X)	Вычисляет логарифм числа X по основанию 2 Calculates the logarithm of X for base 2
LOG10(X)	Вычисляет десятичный логарифм числа X Calculates the decimal logarithm of X

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
MOD(M, N)	Возвращает остаток от деления целого числа M на целое число N Returns the remainder of dividing an integer M by an integer N
PI()	Используется без аргументов. Возвращает число $\pi$ Used without arguments. Returns the number $\pi$
POW(X, Y)	Возвращает значение числа X, возведенного в степень Y Returns the value of the number X raised to the power Y
RADIANS(X)	Возвращает значение угла X, преобразованное из градусов в радианы Returns the angle X converted from degrees to radians
RAND(X)	Возвращает случайное значение с плавающей точкой в диапазоне от 0.0 до 1.0 Returns a random floating point value from 0.0 to 1.0.

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
ROUND(X)	Возвращает округленное до ближайшего целого значение числа X Returns the rounded to the nearest integer value of the number X
SIGN(X)	Возвращает -1, 0 или 1, если X отрицательно, равно нулю или положительно Returns -1, 0 or 1, if X is negative, is zero or positive
SIN(X)	Вычисляет синус угла X, заданного в радианах Calculates the sine of the angle X, given in radians.
SQRT(X)	Вычисляет квадратный корень числа X Calculates the square root of X
TAN(X)	Вычисляет тангенс угла X, заданного в радианах Calculates the tangent of angle X, given in radians
TRUNCATE(X, D)	Возвращает число X с дробной частью, D знаков после запятой Returns the number X with a fractional part having D decimal points



# 3.7 Использование функций / Using functions

## Строковые функции / String functions

Функция Function	Описание Description
ASCII(str)	Возвращает значение ASCII-кода первого символа строки str Returns the ASCII code value of the first character of the string str
BIN(N)	Принимает десятичное число N и возвращает его двоичное представление Takes a decimal number N and returns its binary representation
BIT_LENGTH(str)	Принимает строку str и возвращает ее длину в битах Takes the string str and returns its length in bits
CHAR(N1, N2, ...)	Принимает последовательность из ASCII-кодов и возвращает строку, построенную путем объединения соответствующих им символов Takes a sequence of ASCII codes and returns a string constructed by combining the corresponding characters

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
CHAR_LENGTH(str)	Принимает строку str и возвращает число символов в строке Takes the string str and returns the number of characters in the string
CHARSET(str)	Возвращает имя кодировки, в которой представлена строка Returns the name of the encoding in which the string is represented
COLLATION(str)	Возвращает порядок сортировки, установленный для кодировки аргумента str Returns the collation rule set for the encoding of the argument str
CONCAT(str1, str2, ...)	Возвращает строку, созданную путем объединения всех аргументов, количество которых не ограничено Returns a string created by combining all of the arguments, the number of which is unlimited
CONCAT_WS(separator, str1, str2, ...)	Также объединяет аргументы str1, str2 и т.д., помещая между ними разделитель separator Also combines the arguments str1, str2, etc., placing between them separator

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
CONV(N, from_base, to_base)	Преобразует число N из одной системы счисления from_base в другую to_base Converts the number N from one number system from_base to another to_base
ELT(N, str1, str2, ...)	Возвращает N-ю строку из списка аргументов str1, str2, ... Returns the N-th string from the list of arguments str1, str2, ...
FIELD(str, str1, str2, ...)	Находит строку str в списке str1, str2, ...и возвращает номер строки в этом списке Finds the string str in the list str1, str2, ... and returns the line number in this list
FIND_IN_SET(str, str_list)	Ищет вхождение строки str в список str_list и возвращает номер строки в этом списке Searches for str in the str_list list and returns the line number in the list

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
HEX(N_or_S)	Возвращает значение аргумента в виде шестнадцатеричного числа. Returns the value of the argument as a hexadecimal number.
INSERT(str, pos, len, new_str)	Возвращает строку str, в которой подстрока, начинающаяся с позиции pos и имеющая длину len символов, заменена подстрокой new_str Returns the string str, in which the substring beginning at position pos and having the length len characters is replaced by the substring new_str
INSERT(str, pos, len, new_str)	Возвращает строку str, в которой подстрока, начинающаяся с позиции pos и имеющая длину len символов, заменена подстрокой new_str Returns the string str, in which the substring beginning at position pos and having the length len characters is replaced by the substring new_str
INSTR(str, substr)	Возвращает позицию первого вхождения подстроки substr в строку str Returns the position of the first occurrence of the substring substr in the string str

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
LEFT(str, len)	Возвращает len крайних левых символов строки str Returns len of the leftmost characters of the string str
LENGTH(str)	Возвращает длину строки str Returns the length of the string str
LOCATE(substr, str [,pos])	Возвращает позицию первого вхождения подстроки substr в строку str Returns the position of the first occurrence of the substring substr in the string str
LOWER(str)	Возвращает строку str, записанную строчными символами Returns the string str, written in lower case characters
LPAD(str, len, padstr)	Возвращает строку str, дополненную слева строкой padstr до длины len символов Returns the string str, padstr padded to the left to the length of len characters

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
LTRIM(str)	Возвращает строку str, в которой удалены все начальные пробелы Returns the string str, in which all leading spaces are removed
MID(str, pos [,len])	Возвращает подстроку строки str, которая начинается с позиции pos и имеет длину len символов Returns a substring of the string str, which starts at position pos and has length len characters
OCT(N)	Принимает десятичное число N и возвращает его в восьмеричной системе счисления Takes a decimal number N and returns it in octal number system
ORD(str)	Возвращает значение ASCII-кода первого символа строки str Returns the ASCII code value of the first character of the string str
REPEAT(str, count)	Возвращает строку, полученную из count повторений строки str Returns a string str repeated count times

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
REPLACE(str, from_str, to_str)	Возвращает строку str, в которой все подстроки from_str заменены to_str Returns the string str, in which all the substrings from_str are replaced by to_str
REVERSE(str)	Возвращает строку str, записанную в обратном порядке Returns the string str, written in reverse order
RIGHT(str, len)	Возвращает len крайних правых символов строки str Returns len of the rightmost characters of the str string
RPAD(str, len, padstr)	Возвращает строку str, дополненную справа строкой padstr до длины len символов Returns the string str, padded to the right with the string padstr to len characters length
RTRIM(str)	Возвращает строку str, в которой удалены все конечные пробелы Returns the string str, in which all trailing spaces are removed

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
SPACE(N)	Возвращает строку, состоящую из N пробелов, или пустую строку, если N имеет отрицательное значение Returns a string consisting of N spaces, or an empty string if N is negative
SUBSTRING_ INDEX(str, delim, N)	Возвращает подстроку строки str Returns a substring of the string str Если $N > 0$ ( $N < 0$ ), то функция возвращает всю часть строки, расположенную слева (справа) от N-го вхождения подстроки delim If $N > 0$ ( $N < 0$ ), then the function returns the whole part of the string located to the left (right) of the N-th occurrence of the substring delim
TRIM([[BOTH   LEADING   TRAILING] [remstr] FROM] str)	Удаляет из строки str расположенные в начале (в конце) символы, указанные в строке remstr Removes from the str line the characters at the beginning (at the end) specified in the remstr line



## 3.7 Использование функций / Using functions

Функция Function	Описание Description
UNHEX(str)	Является обратной функции HEX( ) и интерпретирует каждую пару символов строки str как шестнадцатеричный код, который необходимо преобразовать в символ It is the inverse of the HEX() function and it interprets every pair of str characters as hex code that needs to be converted to a character.
UPPER(str)	Переводит все символы строки str в верхний регистр Converts all str characters to uppercase

## 3.7 Использование функций / Using functions

### Функции даты и времени / Date and time functions

Функция Function	Описание Description
ADDDATE(date, INTERVAL expr type)	Возвращает дату date, к которой прибавлен временной интервал, определяемый вторым параметром Returns the <i>date</i> to which the time interval is added, defined by the second parameter ADDDATE('2019-03-14', INTERVAL 7 DAY)
ADDTIME(expr1, expr2)	Возвращает результат сложения двух временных значений Returns the result of adding two time values
CURDATE()	Возвращает текущую дату в формате 'YYYY-MM-DD' Returns the current date in the format 'YYYY-MM-DD'
CURTIME()	Возвращает текущее время суток в формате 'hh:mm:ss' Returns the current time of day in the format 'hh:mm:ss'

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
DATE(datetime)	Извлекает из значения datetime дату, отсекая часы, минуты и секунды Extract date from datetime, cutting off hours, minutes and seconds
DATEDIFF(begin, end)	Вычисляет разницу в днях между датами begin и end Calculates the difference in days between the begin and end dates
DATE_FORMAT(date, format)	Форматирует время date в соответствии со строкой format Formats date time according to the format string
DAY(date)	Возвращает порядковый номер дня в месяце (от 1 до 31) Returns the ordinal number of a day in a month (from 1 to 31)
DAYNAME(date)	Возвращает день недели в виде полного английского названия Returns the day of the week as a full English name
DAYOFWEEK(date)	Возвращает порядковый номер дня недели Returns the ordinal number of day of the week

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
DAYOFYEAR(date)	Возвращает порядковый номер дня в году (от 1 до 366) Returns the ordinal number of a day in a year (from 1 to 366)
EXTRACT(type FROM datetime)	Принимает дату и время суток и возвращает часть, определяемую параметром type Takes the date and time and returns the part defined by the type param.
FROM_DAYS(N)	Принимает число дней N, прошедших с нулевого года, и возвращает дату в формате 'YYYY-MM-DD' Takes the number of days N that have passed since the zero year, and returns the date in the format 'YYYY-MM-DD'
hour(datetime)	Извлекает из значения datetime часы (от 0 до 23) Extracts an hour value from a datetime value (0 to 23)
LAST_DAY(datetime)	Возвращает дату – последний день текущего месяца Returns the date - the last day of the current month

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
MAKEDATE(year, dayofyear)	Принимает год <i>year</i> и номер дня в году <i>dayofyear</i> и возвращает дату в формате 'YYYY-MM-DD' Takes the <i>year</i> and the day number in the year <i>dayofyear</i> and returns the date in the format 'YYYY-MM-DD'
MAKETIME(hour, minute, second)	Принимает час <i>hour</i> , минуты <i>minute</i> и секунды <i>second</i> и возвращает время суток в формате 'hh:mm:ss' Accepts hour <i>hour</i> , minutes <i>minute</i> and second <i>second</i> and returns the time of day in the format 'hh: mm: ss'
MINUTE(datetime)	Извлекает из значения <i>datetime</i> минуты (от 0 до 59) Retrieves minutes from <i>datetime</i> (from 0 to 59)
MONTH(datetime)	Возвращает числовое значение месяца года (от 1 до 12) Returns the numeric value of the month of the year (from 1 to 12)
MONTHNAME(datetime)	Возвращает название месяца в виде полного английского названия Returns the name of the month in the form of a full English name

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
NOW()	Возвращает текущую дату и время в формате 'YYYY-MM-DD hh:mm:ss' Returns the current date and time in the format 'YYYY-MM-DD hh:mm:ss'
PERIOD_ADD( period, N)	Добавляет N месяцев к значению даты <i>period</i> Adds N months to the date <i>period</i>
PERIOD_DIFF( period1, period2)	Вычисляет разницу в месяцах между двумя датами, представленными в числовом формате YYYYMMDD или YYYYMM Calculates the difference in months between two dates, represented in the numeric format YYYYMMDD or YYYYMM
QUARTER(date time)	Возвращает значение квартала года (от 1 до 4) Returns the quarter value of a year (from 1 to 4)
SECOND(dateti me)	Извлекает из значения datetime секунды (от 0 до 59) Extracts from datetime seconds (from 0 to 59)

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
SUBDATE(date, INTERVAL expr type)	Возвращает дату date, из которой вычитается временной интервал Returns the date from which the time interval is subtracted
SUBTIME(date time, time)	Вычитает из величины datetime время time Subtracts the time from the value of datetime
TIME(datetime )	Извлекает из значения datetime время суток Retrieves time of a day from the daytime value
TIMEDIFF(expr 1, expr2)	Возвращает разницу между временными значениями expr1 и expr2 Returns the difference between the values of expr1 and expr2
TIMESTAMP(date, time)	Принимает в качестве аргумента дату date и время time и возвращает полный вариант в формате 'YYYY-MM-DD hh:mm:ss' Takes as an argument the date <i>date</i> and time <i>time</i> , and returns the full version in the format 'YYYY-MM-DD hh: mm: ss'

## 3.7 Использование функций / Using functions

Функция Function	Описание Description
TIMESTAMPADD( interval, int_expr, datetime_expr)	Прибавляет к дате и времени суток datetime_expr временной интервал int_expr, единицы измерения которого задаются параметром interval Adds to the date and time of day datetime_expr the time interval int_expr, the units of measurement of which are specified by the interval parameter
TIMESTAMPDIFF( interval, datetime_expr1, datetime_expr2)	Возвращает разницу между двумя датами datetime_expr1 и datetime_expr2. Единицы измерения интервала задаются параметром interval Returns the difference between two dates, datetime_expr1 and datetime_expr2. Interval units are specified by the interval parameter
TO_DAYS(date)	Принимает дату date и возвращает число дней N, прошедших с нулевого года Accepts the date <i>date</i> and returns the number of days N since the year zero






## 3.7 Использование функций / Using functions

Функция Function	Описание Description
WEEK(date)	Возвращает номер недели в году (от 0 до 53) для даты <i>date</i> . Предполагается, что неделя начинается с воскресенья Returns the week number of the year (0 to 53) for the <i>date</i> . The week is supposed to start on Sunday
WEEKDAY(date)	Возвращает номер дня недели (0 – для понедельника, 1 – для вторника, 6 – для воскресенья) для даты <i>date</i> Returns the day of the week number (0 for Monday, 1 for Tuesday, 6 for Sunday) for the <i>date</i>
YEAR(datetime)	Возвращает год из значения <i>datetime</i> Returns the year from the <i>datetime</i> value
YEARWEEK(date)	Возвращает число в формате YYYYWW, представляющее год и номер недели (от 0 до 53) в году и соответствующее <i>date</i> Returns a number in YYYYWW format representing the year and week number (from 0 to 53) in the year which corresponds to the <i>date</i>

## Query 3

```
SELECT contract.contract_number, contract.contract_date,  
supplied.supplied_product, supplied.supplied_cost, supplier.*  
FROM (supplier INNER JOIN contract ON supplier.supplier_id =  
contract.supplier_id) INNER JOIN supplied ON contract.contract_number =  
supplied.contract_number  
WHERE MONTH(contract.contract_date) = 9 AND  
YEAR(contract.contract_date) = 2018;
```

 contract_number	contract_date	 supplied_product	supplied_cost	 supplier_id	supplier_address	supplier_phone
1	2018-09-01 00:00:00	Audio Player	700.00	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
1	2018-09-01 00:00:00	New Product	100.00	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
1	2018-09-01 00:00:00	TV	1,300.00	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
1	2018-09-01 00:00:00	Video Player	750.00	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
4	2018-09-24 00:00:00	Audio Player	320.00	2	Kyiv, Peremohy av., 154, apt. 3	
4	2018-09-24 00:00:00	Printer	332.50	2	Kyiv, Peremohy av., 154, apt. 3	
4	2018-09-24 00:00:00	TV	990.00	2	Kyiv, Peremohy av., 154, apt. 3	

## Query 4



```
SELECT contract.contract_number, contract.contract_date,  
contract.supplier_id, SUM(supplied.supplied_amount *  
supplied.supplied_cost) AS `Sum`  
FROM contract INNER JOIN supplied ON contract.contract_number =  
supplied.contract_number  
GROUP BY contract.contract_number, contract.contract_date,  
contract.supplier_id  
ORDER BY contract.contract_number;
```

contract_number	contract_date	supplier_id	Sum
1	2018-09-01 00:00:00	1	41,000.00
2	2019-03-21 15:18:46	1	11,350.00
3	2018-09-23 00:00:00	3	99,600.00
4	2018-09-24 00:00:00	2	76,112.50
5	2018-10-02 00:00:00	2	45,630.00
7	2018-12-27 13:30:04	1	59,985.00

## Query 6


```
SELECT contract.contract_number, contract.contract_date,  
contract.contract_note, supplier.*, supplied.supplied_amount  
FROM contract, supplied, supplier  
WHERE contract.contract_number = supplied.contract_number AND  
contract.supplier_id = supplier.supplier_id AND  
supplied.supplied_amount = (SELECT  
MAX(supplied.supplied_amount) FROM supplied);
```

Result #1 (7×1)

 contract_number	contract_date	contract_note	 supplier_id	supplier_address	supplier_phone	supplied_amount
3	2018-09-23 00:00:00	Order 56 on 28.08.2018	3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax	85

## Query 11

```
SELECT supplier.supplier_id, supplier.supplier_address,  
       IFNULL(supplier_org.supplier_org_name,  
       CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',  
               SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',  
               SUBSTRING(supplier_person.supplier_middle_name, 1, 1),  
               '. ')) AS `Supplier`  
FROM (supplier LEFT JOIN supplier_person ON supplier.supplier_id =  
supplier_person.supplier_id) LEFT JOIN supplier_org ON supplier.supplier_id =  
supplier_org.supplier_id;
```

 supplier_id	supplier_address	Supplier
1	Kharkiv, Nauky av., 55, apt. 108	Petrov P. P.
2	Kyiv, Peremohy av., 154, apt. 3	Interfruit Ltd.
3	Kharkiv, Pushkinska str., 77	Ivanov I. I.
4	Odesa, Derebasivska str., 75	Transservice LLC
5	Poltava, Soborna str., 15, apt. 43	Sydorov S. S.

## Query 16

```
SELECT supplied_product, SUM(IF(MONTH(contract_date) = 1, supplied_amount, 0)) AS `Jan`,  
      SUM(IF(MONTH(contract_date) = 2, supplied_amount, 0)) AS `Feb`,  
      SUM(IF(MONTH(contract_date) = 3, supplied_amount, 0)) AS `Mar`,  
      SUM(IF(MONTH(contract_date) = 4, supplied_amount, 0)) AS `Apr`,  
      ...  
      SUM(IF(MONTH(contract_date) = 10, supplied_amount, 0)) AS `Oct`,  
      SUM(IF(MONTH(contract_date) = 11, supplied_amount, 0)) AS `Nov`,  
      SUM(IF(MONTH(contract_date) = 12, supplied_amount, 0)) AS `Dec`  
FROM contract, supplied  
WHERE contract.contract_number = supplied.contract_number AND YEAR(contract_date) = 2018  
GROUP BY supplied_product ORDER BY supplied_product;
```

supplied_product	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Audio Player	0	0	0	0	0	0	0	0	58	33	0	0
Monitor	0	0	0	0	0	0	0	0	85	0	0	0
New Product	0	0	0	0	0	0	0	0	15	0	0	0
Phone	0	0	0	0	0	0	0	0	0	0	0	5
Printer	0	0	0	0	0	0	0	0	41	0	0	0
TV	0	0	0	0	0	0	0	0	118	14	0	10
Video Player	0	0	0	0	0	0	0	0	12	17	0	0

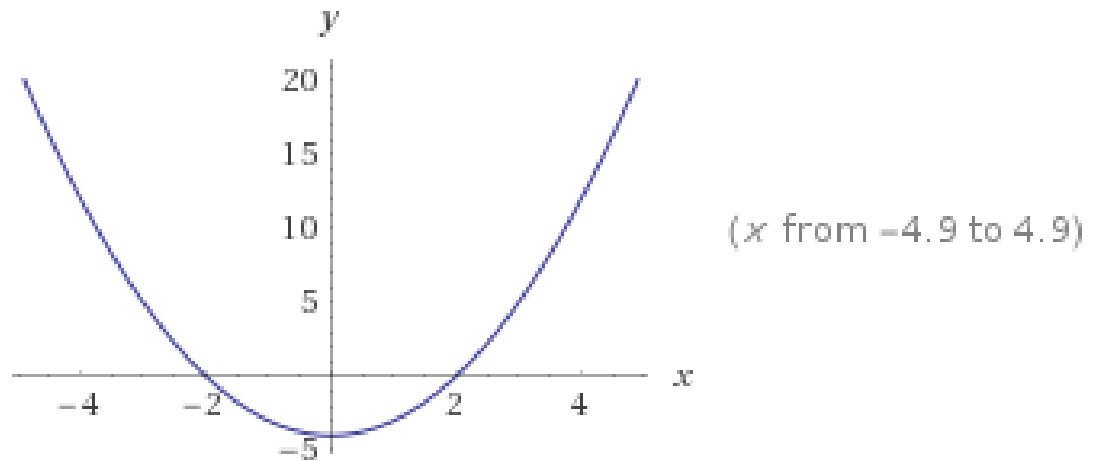
## Query 17

```
SELECT supplied.contract_number, supplied.supplied_product,  
supplied.supplied_amount, supplied.supplied_cost,  
contract.contract_date,  
MONTHNAME(contract.contract_date) AS `Month`,  
YEAR(contract.contract_date) AS `Year`  
FROM supplied, contract  
WHERE contract.contract_number = supplied.contract_number;
```

🔑 contract_number	🔑 supplied_product	supplied_amount	supplied_cost	contract_date	Month	Year
1	Audio Player	25	700.00	2018-09-01 00:00:00	September	2,018
1	New Product	15	100.00	2018-09-01 00:00:00	September	2,018
1	TV	10	1,300.00	2018-09-01 00:00:00	September	2,018
1	Video Player	12	750.00	2018-09-01 00:00:00	September	2,018
2	Audio Player	5	450.00	2019-03-21 15:18:46	March	2,019
2	Stereo System	11	500.00	2019-03-21 15:18:46	March	2,019
2	Video Player	8	450.00	2019-03-21 15:18:46	March	2,019
3	Audio Player	11	550.00	2018-09-23 00:00:00	September	2,018

## 3.7 Использование функций / Using functions

$$y = f(x) = x^2 - 4$$



```
CREATE FUNCTION <name> (<args>) RETURNS <type>  
    [NOT] DETERMINISTIC  
BEGIN  
<body>  
END
```



## 3.7 Использование функций / Using functions

### **RETURNS**

- определяет тип данных, который возвращает функция
- RETURNS defines a data type of returning value

### **[NOT] DETERMINISTIC**

- определяет, будет ли функция детерминированной или нет
- defines whether the function is deterministic or not

## 3.7 Использование функций / Using functions

```
DELIMITER $$
```

```
CREATE FUNCTION my_parabola(x DOUBLE) RETURNS  
    DOUBLE DETERMINISTIC
```

```
BEGIN
```

```
    RETURN x * x - 4;
```

```
END $$
```

```
DELIMITER ;
```

## 3.7 Использование функций / Using functions

### **DELIMITER**

используется для изменения разделителя операций  
(окончание операции) в MySQL

стандартный разделитель – ;

is used to change operations separator (operations end  
character) in MySQL

Standard operations separator is ;

```
DELIMITER //
```

```
...
```

```
DELIMITER ;
```

## 3.7 Использование функций / Using functions

```
SELECT b_name, b_author, b_year, b_price,  
       my_parabola(b_price)  
FROM books;
```

books (5×30)				
b_name	b_author	b_year	b_price	my_parabola(b_price)
JavaScript в кармане	Рева О.Н.	2008	42.00	1,760
Visual FoxPro 9.0	Клепинин В.Б.	2007	660.00	435,596
C++ Как он есть	Тимофеев В.В.	2009	218.00	47,520
Создание приложений с помощью C#	Фаронов В.В.	2008	169.00	28,557
Delphi. Народные советы	Шкрыль А.А.	2007	243.00	59,045
Delphi. Полное руководство	Сухарев М.	2008	500.00	249,996
Профессиональное программирование на PHP	Шлоссейгл Дж.	2006	309.00	95,477
Совершенный код	Макконнелл С.	2007	771.00	594,437
Практика программирования	Керниган Б.	2004	214.00	45,792

## 3.7 Использование функций / Using functions

DELIMITER \$\$

**CREATE FUNCTION** LAST\_ORDERED\_BOOK() **RETURNS** INT **NOT DETERMINISTIC**

**BEGIN**

**DECLARE** last\_order INT **DEFAULT** -1;

**SELECT** o\_book\_ID **INTO** last\_order **FROM** orders **ORDER BY** order\_ID **DESC** **LIMIT** 1;



**RETURN** last\_order;

**END** \$\$

DELIMITER ;

**SELECT** \* **FROM** books

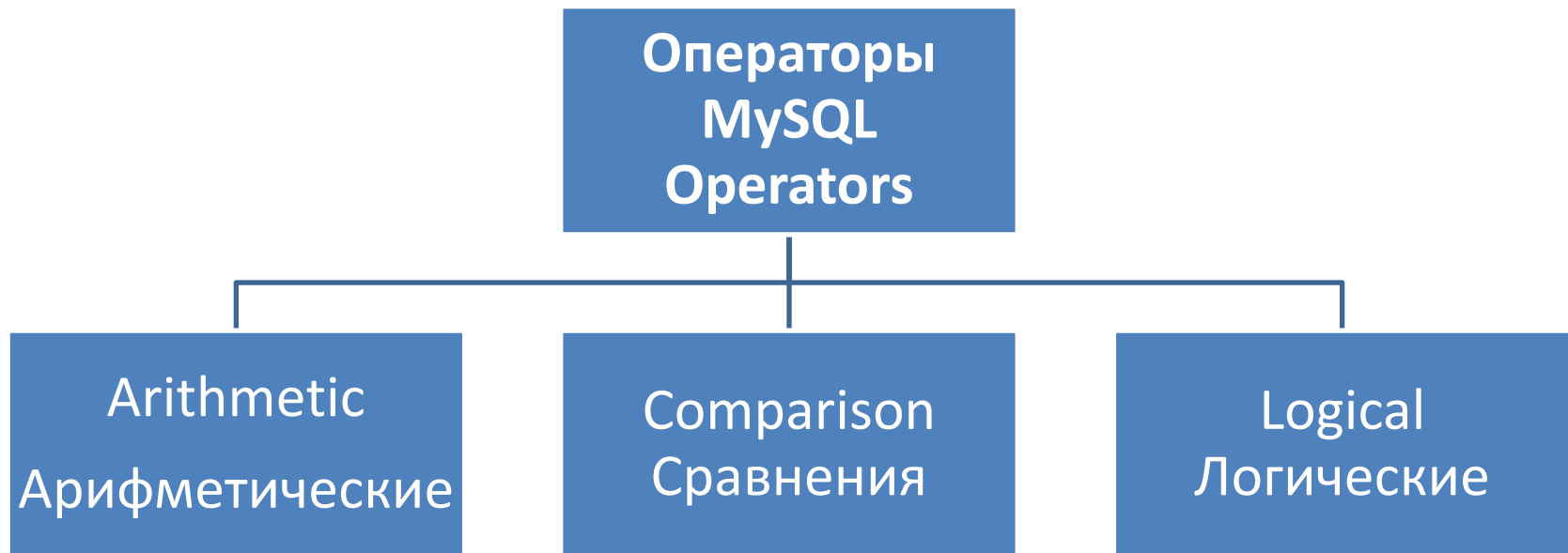
**WHERE** book\_ID = ***LAST\_ORDERED\_BOOK()***;

books (7×1)						
 book_ID	b_name	b_author	b_year	b_price	b_count	 b_cat_ID
20	Компьютерные сети	Танненбаум Э.	2007	630.00	6	4

## 3.8 Операторы / Operators

Оператор – конструкция языка, которая производит преобразование данных (операндов).

Operator is a language construction used to transform data (operands).



## 3.8 Операторы / Operators

Арифметические операторы / Arithmetic operators:

- (+) сложение / addition
- (-) вычитание / subtraction
- (\*) умножение / multiplication
- (/) деление / division
- **DIV** целочисленное деление / integer division

Деление на 0 дает безопасный результат NULL

Division by zero produces a safe result NULL

## 3.8 Операторы / Operators

SELECT NULL = NULL;

Result #1 (1×1)	
NULL = NULL	
(NULL)	

SELECT NULL <> NULL;

Result #1 (1×1)	
NULL <> NULL	
(NULL)	

SELECT NULL IS NULL;

Result #1 (1×1)	
NULL IS NULL	
1	



## 3.8 Операторы / Operators

### Операторы сравнения / Comparison operators:

Оператор Operator	Описание Description
=	Возвращает 1, если операнды равны, и 0, если не равны Returns 1, if operands are equal, and 0, if they are not equal
<=>	То же самое, но не возвращает NULL It is similar to the previous operator, but it does not return NULL
<>	Возвращает 1, если операнды не равны, и 0, если равны Returns 1, if operands are not equal, and 0, if they are equal
<	Возвращает 1, если левый операнд меньше правого Returns 1, if the left operand is left than right operand
<=	Возвращает 1, если левый операнд меньше правого или они равны Returns 1, if the left operand is left than right operand or they are equal

## 3.8 Операторы / Operators

Оператор Operator	Описание Description
>	Возвращает 1, если левый операнд больше правого Returns 1, if the left operand is greater than right operand
>=	Возвращает 1, если левый операнд больше правого или они равны Returns 1, if the left operand is greater than right operand or they are equal
n BETWEEN min AND max	Возвращает 1, если проверяемое значение n находится между min и max Returns 1, if the value n is between min and max
IS NULL IS NOT NULL	Проверяет, является ли значение значением NULL или нет Checks, whether a value is NULL or not
n IN (set)	Возвращает 1, если проверяемое значение n входит в множество Returns 1, if the value n is included into a set

## 3.8 Операторы / Operators

Логические операторы / Logical operators:

Оператор Operator	Описание Description
n AND m	true AND true = true, false AND any = false
n OR m	true OR any = true, false OR false = false
NOT n	NOT true = false, NOT false = true
n XOR m	true XOR true = false, true XOR false = true, false XOR true = true, false XOR false = false

- ненулевое значение, отличное от NULL, интерпретируется как «ИСТИНА»
- non-zero and non-NULL value is interpreted as “true”

## 3.9 Переменные / Variables

- Часто результаты запроса необходимо использовать в последующих запросах
- Often the query results are required in subsequent queries
- Для этого полученные данных необходимо сохранить во временных структурах
- Therefore, retrieved data should be stored in temporary structures
- Эта задача решается при помощи переменных SQL и временных таблиц
- This problem is solved by using SQL variables and temporary tables

## 3.9 Переменные / Variables

- Объявление переменной начинается с символа @, за которым следует имя переменной
- Variable definition starts with the @ symbol followed by the name of a variable
- Значения переменным присваиваются при помощи оператора SELECT с использованием оператора присваивания :=
- Values are assigned to the variables by using the SELECT operator with the assignment operator :=
- Например / For example  
`SELECT @total := COUNT(*) FROM books;`

## 3.9 Переменные / Variables

```
SELECT @total := COUNT(*) FROM books;
```

```
SELECT @total;
```

- Объявляется переменная @total, которой присваивается число записей в таблице books
- The variable @total is declared, the number of records in the table books is assigned to this variable
- Затем данная переменная используется в последующем запросе в рамках текущего сеанса
- Then this variable is used in the sequential query within the same session
- Переменные действуют только в рамках сеанса
- Variables are available only within a session

Result #1 (1×1)	Result #2 (1×1)
@total := COUNT(*)	
	30

Result #1 (1×1)	Result #2 (1×1)
@total	
	30

## 3.9 Переменные / Variables

```
SET @last = CURDATE() - INTERVAL 7 DAY;
```

```
SELECT CURDATE(), @last;
```

- Переменные также могут объявляться при помощи оператора SET
- Variables also might be declared using the SET operator
- При этом может быть использован обычный знак равенства = в качестве оператора присваивания
- The usual equals sign = can be used as the assignment operator
- Оператор SET удобен тем, что он не возвращает результат присваивания значения переменной
- The SET operator is convenient because it doesn't return the result of variable assignment

Result #1 (2×1)	
CURDATE()	@last
2019-03-19	2019-03-12

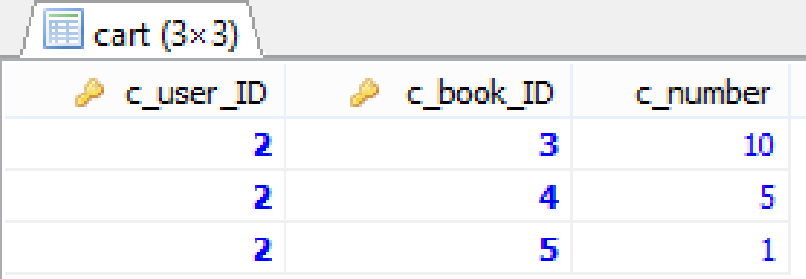
## 3.10      Временные таблицы / Temporary tables



- Переменная SQL позволяет сохранить только одно промежуточное значение
- The SQL variable allows to store only a single intermediate value
- Когда необходимо сохранить результирующую таблицу, прибегают к временным таблицам
- Temporary tables are used to store the result table
- CREATE TEMPORARY TABLE



## 3.10 Временные таблицы / Temporary tables

```
CREATE TEMPORARY TABLE cart (  
    c_user_ID INT NOT NULL,  
    c_book_ID INT NOT NULL,  
    c_number INT NOT NULL,  
    PRIMARY KEY (c_user_ID, c_book_ID)  
) ENGINE=InnoDB;
```



 c_user_ID	 c_book_ID	c_number
2	3	10
2	4	5
2	5	1

```
INSERT INTO cart (c_user_ID, c_book_ID, c_number) VALUES  
(2, 3, 10), (2, 4, 5), (2, 5, 1);
```

```
SELECT * FROM cart;
```

Временная таблица автоматически удаляется по завершении сеанса и может быть использована только в течение данного соединения

Temporary tables are automatically removed when the session is over, they might be used only within a current session

- Вывод списка покупателей и числа осуществленных ими покупок, покупателей отсортировать по убыванию числа заказов
- Print the list of customers and numbers of their purchases, sort customers by a number of orders in a descending order

```
SELECT users.u_surname, users.u_name, users.u_patronymic,  
COUNT(orders.order_ID) AS total  
FROM users INNER JOIN orders ON users.user_ID = orders.o_user_ID  
GROUP BY users.user_ID  
ORDER BY total DESC;
```

users (4x4)			
u_surname	u_name	u_patronymic	total
Симонов	Игорь	Николаевич	2
Корнеев	Александр	Александрович	1
Кузнецов	Максим	Петрович	1
Иванов	Александр	Валерьевич	1

- Вывести полный список покупателей, включая тех, которые не сделали ни одной покупки
- Print the complete list of customers, including those who did not make any orders

```
SELECT users.u_surname, users.u_name, users.u_patronymic,  
COUNT(orders.order_ID) AS total  
FROM users LEFT JOIN orders ON users.user_ID = orders.o_user_ID  
GROUP BY users.user_ID  
ORDER BY total DESC;
```

users (4×6)			
u_surname	u_name	u_patronymic	total
Симонов	Игорь	Николаевич	2
Кузнецов	Максим	Петрович	1
Корнеев	Александр	Александрович	1
Иванов	Александр	Валерьевич	1
Петров	Анатолий	Юрьевич	0
Лосев	Сергей	Иванович	0

- Выбрать строки из таблицы catalogs, у которых первичный ключ совпадает с одним из значений, возвращаемых вложенным запросом
- Select records from the table catalog, which primary key matches one of the values returned by the nested query

```
SELECT cat_name FROM catalogs
WHERE cat_ID IN (SELECT b_cat_ID
FROM books GROUP BY b_cat_ID);
```

cat_name
Программирование
Интернет
Базы данных
Сети
Мультимедиа

- Вывести имена и фамилии покупателей, совершивших хотя бы одну покупку
- Print names and surnames of customers that made at least one order

```
SELECT u_name, u_surname FROM users
WHERE user_ID = ANY(SELECT o_user_ID
FROM orders);
```

u_name	u_surname
Александр	Иванов
Игорь	Симонов
Максим	Кузнецов
Александр	Корнеев

- Вывести все товарные позиции, цена которых превышает среднюю цену каждого из каталогов
- Print all products which price is greater than average price of each category

```
SELECT b_name, b_price FROM books
WHERE b_price > ALL(SELECT AVG(b_price)
FROM books GROUP BY b_cat_ID);
```

b_name	b_price
Visual FoxPro 9.0	660.00
Delphi. Полное руководство	500.00
Совершенный код	771.00
Принципы маршрутизации в Internet	428.00
Компьютерные сети	630.00
Сети. Поиск неисправностей	434.00
Безопасность сетей	462.00

- Вывести имена и фамилии покупателей, совершивших хотя бы одну покупку
- Print names and surnames of customers that made at least one order

```
SELECT u_name, u_surname FROM users
WHERE EXISTS (SELECT * FROM orders
WHERE orders.o_user_ID = users.user_ID);
```

u_name	u_surname
Александр	Иванов
Игорь	Симонов
Максим	Кузнецов
Александр	Корнеев

# 4 Реализация бизнес-логики

## 4 Business logic implementation

## 4.1 Создание и использование представлений / Creating and using views

**Представление** – виртуальная/логическая таблица, которая базируется на SQL запросе SELECT.

A database **view** is a virtual table or logical table which is defined as a SQL SELECT query.

Поскольку представление, также как и таблица базы данных, содержит строки и столбцы, к нему могут быть выполнены **запросы**.

Because a database view is similar to a database table, which consists of rows and columns, so you can **query** data against it.

## 4.1 Создание и использование представлений / Creating and using views

Большинство СУБД, включая MySQL, позволяют **обновлять** данные в базовых таблицах через представление с некоторыми ограничениями.

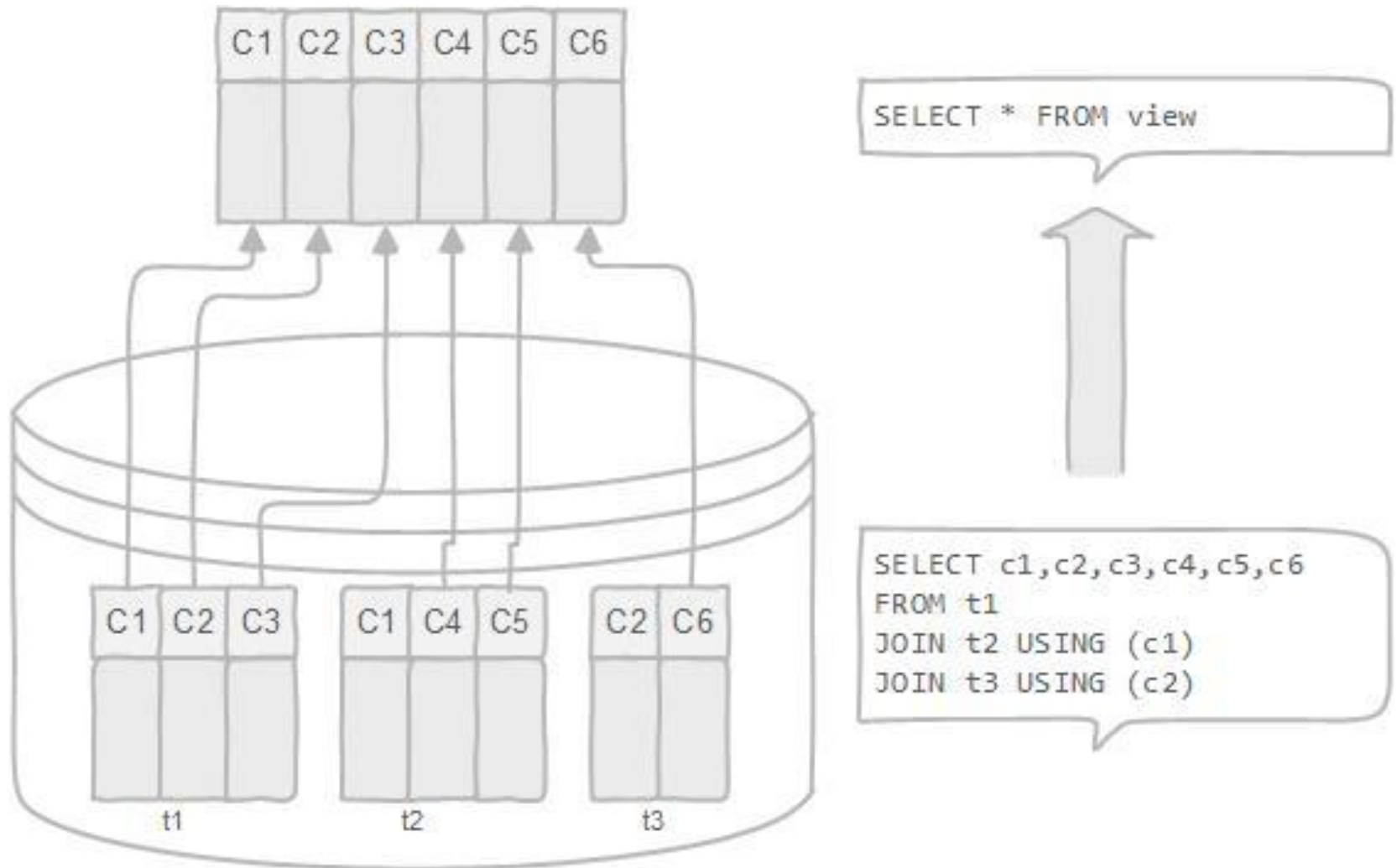
Most database management systems, including MySQL, allow you to **update** data in the underlying tables through the database view with some prerequisites.

Представления являются **динамическими**, поскольку они не связаны с физической схемой данных. СУБД хранит представления как SQL SELECT запросы. Когда данные таблиц изменяются, представления также **отражают** эти изменения.

A database view is **dynamic** because it is not related to the physical schema. The database system stores views as a SQL SELECT statement with joins. When the data of the tables changes, the view **reflects** that changes as well.



## 4.1 Создание и использование представлений / Creating and using views



## 4.1 Создание и использование представлений / Creating and using views

### Преимущества / Advantages

- Представление позволяет **упростить сложные запросы**: представление определяется оператором SQL, который связывается с несколькими базовыми таблицами.
- A database view allows you to **simplify complex queries**: a database view is defined by an SQL statement that associates with many underlying tables.

## 4.1 Создание и использование представлений / Creating and using views

Представления можно использовать, чтобы скрыть сложность базовых таблиц для конечных пользователей и внешних приложений. При помощи представлений достаточно использовать только простые операторы SQL вместо сложных с множеством объединений.

You can use database view to hide the complexity of underlying tables to the end-users and external applications. Through a database view, you only have to use simple SQL statements instead of complex ones with many joins.

```


SELECT supplier.supplier_id, supplier.supplier_address,
IFNULL(supplier_org.supplier_org_name,
        CONCAT(supplier_person.supplier_last_name, ' ',
        supplier_person.supplier_first_name, ' ',
        supplier_person.supplier_middle_name)) AS `Info`
FROM supplier LEFT OUTER JOIN supplier_org ON supplier.supplier_id =
supplier_org.supplier_id LEFT OUTER JOIN supplier_person ON
supplier.supplier_id = supplier_person.supplier_id;

```

```

SELECT * FROM supplier_info;

```

Result #1 (3×5)		
 supplier_id	supplier_address	Info
<b>1</b>	Kharkiv, Nauky av., 55, apt. 108	Petrov Pavlo Petrovych
<b>2</b>	Kyiv, Peremohy av., 154, apt. 3	Interfruit Ltd.
<b>3</b>	Kharkiv, Pushkinska str., 77	Ivanov Ilia Illych
<b>4</b>	Odesa, Derebasivska str., 75	Transservice LLC
<b>5</b>	Poltava, Soborna str., 15, apt. 43	Sydorov Serhii Stepanovych

## 4.1 Создание и использование представлений / Creating and using views

- Представление базы данных помогает **ограничить доступ к данным** для определенных пользователей. Возможно, вы не захотите, чтобы подмножество конфиденциальных данных могло запрашиваться всеми пользователями. Вы можете использовать представление базы данных, чтобы предоставлять только общедоступные данные определенной группе пользователей.
- A database view helps **limit data access** to specific users. You may not want a subset of sensitive data can be queryable by all users. You can use a database view to expose only non-sensitive data to a specific group of users.

user_ID	u_name	u_patronymic	u_surname	u_phone	u_email	u_status
1	Александр	Валерьевич	Иванов	58-98-78	ivanov@email.ru	active
2	Сергей	Иванович	Лосев	90-57-77	losev@email.ru	passive
3	Игорь	Николаевич	Симонов	95-66-61	simonov@email.ru	active
4	Максим	Петрович	Кузнецов	(NULL)	kuznetsov@email.ru	active
5	Анатолий	Юрьевич	Петров	(NULL)	(NULL)	lock
6	Александр	Александрович	Корнеев	89-78-36	korneev@email.ru	gold

```
SELECT CONCAT_WS(' ', u_name, u_patronymic, u_surname), u_status
FROM users;
```

CONCAT_WS(' ', u_name, u_patronymic, u_surname)	u_status
Александр Валерьевич Иванов	active
Сергей Иванович Лосев	passive
Игорь Николаевич Симонов	active
Максим Петрович Кузнецов	active
Анатолий Юрьевич Петров	lock
Александр Александрович Корнеев	gold

## 4.1 Создание и использование представлений / Creating and using views

- Представления обеспечивают **дополнительный уровень безопасности**. Безопасность является важной частью любой СУБД. Представления обеспечивают дополнительную защиту для СУБД. Можно создавать не обновляемые представления, чтобы предоставлять данные только для чтения конкретным пользователям. Пользователи могут только получать данные в режиме read-only, но не могут их обновлять.
- A database view provides **extra security layer**. Security is a vital part of any relational database management system. The database view offers additional protection for a database management system. The database view allows you to create the read-only view to expose read-only data to specific users. Users can only retrieve data in read-only view but cannot update it.

## 4.1 Создание и использование представлений / Creating and using views

- Представления, в отличие от таблиц базы данных, могут содержать **вычисляемые столбцы**.
- A database view enables computed columns. A database table should not have **calculated columns** however a database view should.

SELECT \*, supplied\_amount \* supplied\_cost FROM supplied;

supplied (5x17)				
contract_number	supplied_product	supplied_amount	supplied_cost	supplied_amount * supplied_cost
1	Audio Player	25	700.00	17,500.00
1	TV	10	1,300.00	13,000.00
1	Video Player	12	750.00	9,000.00
2	Audio Player	5	450.00	2,250.00
2	Stereo System	11	500.00	5,500.00
2	Video Player	8	450.00	3,600.00
3	Audio Player	11	550.00	6,050.00
3	Monitor	85	550.00	46,750.00
3	TV	52	900.00	46,800.00



## 4.1 Создание и использование представлений / Creating and using views

- Представление базы данных обеспечивает **обратную совместимость**. Предположим, есть база данных, которую используют приложения. Возникает необходимость изменения структуры базы данных, чтобы выполнить новые бизнес-требования. Некоторые таблицы удаляются и создаются новые таблицы, при этом нежелательно, чтобы такие изменения влияли на другие приложения. В этом случае можно создать представления с той же структурой, что и устаревшие таблицы, которые были удалены.
- A database view enables **backward compatibility**. Suppose you have a central database, which many applications are using it. One day, you decide to redesign the database to adapt to the new business requirements. You remove some tables and create new tables, and you don't want the changes to affect other applications. In this scenario, you can create database views with the same schema as the legacy tables that you will remove.

## 4.1 Создание и использование представлений / Creating and using views

### Недостатки / Disadvantages

- **Производительность:** запрос данных из представления может быть медленным, особенно если представление создается на основе других представлений.
- **Performance:** querying data from a database view can be slow especially if the view is created based on other views.

## 4.1 Создание и использование представлений / Creating and using views

- **Зависимость таблиц:** представление создается на основе базовых таблиц базы данных. Всякий раз, когда структуру этих таблиц, с которыми связано представление, изменяется, также требуется изменить представление.
- **Tables dependency:** you create a view based on underlying tables of the database. Whenever you change the structure of these tables that view associated with, you have to change the view as well.

## 4.1 Создание и использование представлений / Creating and using views

### **CREATE VIEW**

Используется для создания нового представления в базе данных MySQL

Is used to create a new view in MySQL database

**CREATE [OR REPLACE] [ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}]**

**VIEW** view\_name [(column\_list)]

**AS** select-statement

**[WITH [CASCADED | LOCAL] CHECK OPTION];**

# 4.1 Создание и использование представлений / Creating and using views

## OR REPLACE

в случае существования представления с таким именем старое будет удалено, а новое создано

is used to replace the existing view with the same name, the old view will be removed and the new one will be created

Внутри базы данных **представления и таблицы совместно используют одно и то же пространство имен**, поэтому представление и таблица не могут иметь одинаковые имена. Кроме того, имя представления должно соответствовать правилам именования таблицы.

Within a database, **views and tables share the same namespace**, therefore, a view and a table cannot have the same name. In addition, the name of a view must follow the table's naming rules.

# 4.1 Создание и использование представлений / Creating and using views

## **ALGORITHM**

определяет алгоритм, используемый при обращении к представлению

defines the algorithm used when a view is addressed

- **MERGE**

MySQL добавляет в использующийся оператор соответствующие части из определения представления и выполняет получившийся оператор

MySQL adds corresponding parts from the view definition into the used operator and then executes the result operator

## 4.1 Создание и использование представлений / Creating and using views

- **TEMPTABLE**

MySQL заносит содержимое представления во временную таблицу, над которой затем выполняется оператор обращенный к представлению

MySQL inserts a content of the view into the temporary table and then executes the operator applied to the view

Представление, созданное при помощи оператора TEMPTABLE, не может быть обновляемым

A view created with the TEMPTABLE operator cannot be updatable

## 4.1 Создание и использование представлений / Creating and using views

- **UNDEFINED**

MySQL сам выбирает какой алгоритм использовать при обращении к представлению

MySQL chooses itself which algorithm should be used when a view is addressed

Используется по умолчанию, если конструкция

**[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]**

отсутствует

Is used by default if the **ALGORITHM** clause is not specified when view created



## 4.1 Создание и использование представлений / Creating and using views

### **MERGE**

строки представления должны полностью соответствовать строкам базовой таблицы

view rows must fully match the rows of the base table

**CREATE VIEW** v\_total **AS**

**SELECT** supplied\_product,

supplied\_amount \* supplied\_cost **AS** total

**FROM** supplied

SELECT \* FROM v\_total;

🔑 supplied_product	total		🔑 contract_number	🔑 supplied_product	supplied_amount	supplied_cost
Audio Player	17,500.00	→	1	Audio Player	25	700.00
TV	13,000.00	→	1	TV	10	1,300.00
Video Player	9,000.00	→	1	Video Player	12	750.00
Audio Player	2,250.00	→	2	Audio Player	5	450.00
Stereo System	5,500.00	→	2	Stereo System	11	500.00
Video Player	3,600.00	→	2	Video Player	8	450.00
Audio Player	6,050.00	→	3	Audio Player	11	550.00
Monitor	46,750.00	→	3	Monitor	85	550.00

Каждая строка соответствует строке из таблицы supplied, используется алгоритм MERGE

Each row corresponds to each row from the supplied table, the MERGE algorithm is used

- имя представления заменяется на имя таблицы
- the name of view is replaced with the name of table
- список полей заменяется на определения полей из представления
- the list of fields is replaced with the fields definition from the view

## 4.1 Создание и использование представлений / Creating and using views

MERGE не может быть использован / cannot be used:

- агрегатные функции / aggregate functions

AVG, COUNT, MAX, MIN, SUM

- подзапросы в SELECT или WHERE / nested queries in SELECT or WHERE clauses
- DISTINCT
- GROUP BY
- HAVING
- UNION, UNION ALL

```
CREATE VIEW v_max AS  
SELECT supplied_product, COUNT(*) AS num  
FROM supplied  
GROUP BY supplied_product;
```

supplied_product	num
Audio Player	5
Monitor	1
Phone	1
Printer	1
Stereo System	1
TV	5
Video Player	3

```
SELECT supplied_product, MAX(num) FROM v_max;
```



supplied_product	MAX(num)
Audio Player	5

```
SELECT supplied_product, MAX(COUNT(*))  
FROM supplied  
GROUP BY supplied_product;
```

**SQL Error (1111): Invalid use of group function**

```
SELECT supplied_product, MAX(num) FROM v_max;
```



```
DROP TABLE IF EXISTS tmp_table;
```

```
CREATE TEMPORARY TABLE tmp_table  
SELECT supplied_product, COUNT(*) AS num  
FROM supplied  
GROUP BY supplied_product;
```

```
SELECT supplied_product, MAX(num) FROM v_max;
```

```
DROP TABLE tmp_table;
```

supplied (2×1)	
supplied_product	MAX(num)
Audio Player	5

## 4.1 Создание и использование представлений / Creating and using views

### **WITH [CASCADED | LOCAL]CHECK OPTION**

все добавляемые или изменяемые строки проверяются  
на соответствие определению представления

all inserted or updated records are checked to be  
compatible with the view definition

проверяется соответствие новой строки условию WHERE  
в определении представления

compatibility of a new record to the WHERE clause of the  
view definition is checked

## 4.1 Создание и использование представлений / Creating and using views

### **CASCADED, LOCAL**

определяют глубину проверки для представлений, основанных на других представлениях

define the depth of validation for views based on another views

**LOCAL** – проверка условия WHERE только в определении представления / checks the WHERE clause of the view definition

**CASCADED (default)** – проверка для всех представлений, на которых основано данное представление / checks all views on which the current view is based

```
CREATE VIEW v_upd_supplied AS
SELECT * FROM supplied
WHERE supplied.supplied_amount > 10
WITH CHECK OPTION;
```

```
INSERT INTO v_upd_supplied VALUES(1, 'New Product', 5, 100);
```

**SQL Error (1369): CHECK OPTION failed 'supply.v\_upd\_supplied'**

```
INSERT INTO v_upd_supplied VALUES(1, 'New Product', 15, 100);
```

v_upd_supplied (4×13)			
contract_number	supplied_product	supplied_amount	supplied_cost
1	Audio Player	25	700.00
1	New Product	15	100.00
1	Video Player	12	750.00
2	Stereo System	11	500.00

```
SELECT *
FROM supplied;
```

contract_number	supplied_product	supplied_amount	supplied_cost
1	Audio Player	25	700.00
1	New Product	15	100.00



## 4.1 Создание и использование представлений / Creating and using views

Обновление представлений, основанных на нескольких таблицах

Update of views based on multiple tables

- **INSERT** работает только при добавлении данных в одну реальную таблицу / works if data is inserted into a single physical table
- **UPDATE** аналогично / the same
- **DELETE** не поддерживается / is not supported
- **INNER JOIN** используется для объединения таблиц / is used to join the tables

## 4.1 Создание и использование представлений / Creating and using views

**Вертикальные** представления – для ограничения доступа пользователей к столбцам таблицы

**Горизонтальные** представления – для ограничения доступа пользователей к строкам таблицы

**Vertical** views are used to restrict users' access to the table's columns

**Horizontal** views are used to restrict users' access to the table's rows

## 4.1 Создание и использование представлений / Creating and using views

```
mysql> CREATE VIEW manager1
-> AS SELECT * FROM books
-> WHERE b_catID IN (SELECT catID
-> FROM catalogs
-> WHERE cat_name = 'Интернет' OR cat_name = 'Сети')
-> ORDER BY b_name;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT b_name, b_price, b_count FROM manager1;
```

b_name	b_price	b_count
Web-конструирование	177.00	6
Анализ и диагностика компьютерных сетей	344.00	3
Безопасность сетей	462.00	5
Компьютерные сети	630.00	6
Общение в Интернете	85.00	5
Принципы маршрутизации в Internet	428.00	4
Поиск в Internet	107.00	2
Популярные интернет-браузеры	82.00	6
Локальные вычислительные сети	82.00	8
Сети. Поиск неисправностей	434.00	4
Самоучитель Интернет	121.00	4

```
11 rows in set (0.05 sec)
```

## 4.1 Создание и использование представлений / Creating and using views

**DROP VIEW** [IF EXISTS] view\_name [, view\_name] ... ;

позволяет удалить одно или несколько представлений  
used to remove one or multiple views

```
mysql> DROP VIEW cat, list_user, price;  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_book |  
+-----+  
| books          |  
| catalogs       |  
| orders         |  
| users          |  
+-----+  
4 rows in set (0.00 sec)
```

## 4.1 Создание и использование представлений / Creating and using views

### list\_user

позволяет отображать фамилию и инициалы покупателей, скрывая другие поля

allows to display the last name and initials of customers, whereas other fields are hidden

```
mysql> CREATE OR REPLACE VIEW list_user
-> AS SELECT CONCAT(u_surname, " ",
-> SUBSTRING(u_name,1,1), ".",
-> SUBSTRING(u_patronymic,1,1), ".") AS name
-> FROM users ORDER BY name;
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> SELECT * FROM list_user;
```

name
Кузнецов М.П.
Корнеев А.А.
Петров А.Ю.
Иванов А.В.
Лосев С.И.
Симонов И.Н.

```
6 rows in set (0.02 sec)
```

## 4.1 Создание и использование представлений / Creating and using views

### price

позволяет получить общую стоимость книг в каждом каталоге  
allows to retrieve the total cost of books in each catalog

```
mysql> CREATE VIEW price  
-> AS SELECT b_catID, SUM(b_price*b_count) AS price  
-> FROM books  
-> GROUP BY b_catID  
-> ORDER BY price;
```

Query OK, 0 rows affected (0.08 sec)

```
mysql> SELECT * FROM price;
```

b_catID	price
3	2908.00
2	4389.00
4	9514.00
1	12123.00
5	15177.00

5 rows in set (0.01 sec)

## 4.1 Создание и использование представлений / Creating and using views

Представление можно использовать в одном запросе SELECT наряду с таблицей

A view might be used in the same SELECT query with a table

```
mysql> SELECT catalogs.cat_name, price.price  
-> FROM price, catalogs  
-> WHERE price.b_catID = catalogs.catID  
-> ORDER BY catalogs.catID;
```

cat_name	price
Программирование	12123.00
Интернет	4389.00
Базы данных	2908.00
Сети	9514.00
Мультимедиа	15177.00

```
5 rows in set (0.02 sec)
```

## 4.1 Создание и использование представлений / Creating and using views

Запросы к представлениям могут содержать функции группировки, также как и запросы к обычным таблицам

Views might be queried using aggregation functions just like the ordinary database tables

```
mysql> SELECT MIN(price), MAX(price), SUM(price) FROM price;
+-----+-----+-----+
| MIN(price) | MAX(price) | SUM(price) |
+-----+-----+-----+
|      2908.00 |      15177.00 |      44111.00 |
+-----+-----+-----+
1 row in set (0.02 sec)
```



## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

На практике часто требуется повторять последовательность одинаковых запросов

On practice it is often required to repeat a sequence of similar queries

Хранимые процедуры позволяют объединить последовательности таких запросов и сохранить их на сервере

Stored procedures allow to merge sequences of such queries and to store them on a server

После этого клиентам достаточно послать один запрос на выполнение хранимой процедуры

Then clients only have to execute a single query in order to run a stored procedure

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Преимущества хранимых процедур / Advantages of stored procedures:

### 1 Повторное использование кода / Code reuse

после создания хранимой процедуры ее можно вызывать из любых приложений и SQL запросов

after a stored procedure is created it can be executed by any applications and SQL queries

### 2 Сокращение сетевого трафика / Reduce a network traffic

вместо нескольких запросов на сервер можно послать запрос на выполнение хранимой процедуры и сразу получить ответ

instead of running several queries a single query can be sent to a server in order to execute a stored procedure and take the complete answer

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

### 3 Безопасность / Security

для выполнения хранимой процедуры пользователь должен обладать привилегией

a user should have a privilege in order to execute a stored procedure

### 4 Простота доступа / Ease of access

хранимые процедуры позволяют инкапсулировать сложный код и оформить его в виде простого вызова

stored procedures allow to encapsulate a complex code and run it by using a simple call

### 5 Выполнение бизнес-логики / Execution of business logic

бизнес-логика в виде хранимых процедур не зависит от языка разработки приложения

business logic provided by stored procedures does not depend on application programming language

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

```
CREATE PROCEDURE procedure_name ( [ parameter [, ...] ] )  
[ characteristic ... ] procedure_body
```

В скобках передается необязательный список параметров, перечисленных через запятую

The unnecessary list of parameters is passed in parentheses

Каждый параметр позволяет передать в процедуру (из процедуры) входные данные (результат работы)

Each parameter allows to pass input data (or the result of execution) into procedure (or retrieve from a procedure)

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

[ IN | OUT | INOUT ] parameter\_name type

### **IN**

данные передаются внутрь хранимой процедуры

data passed into a stored procedure

при выходе из процедуры новое значение для такого параметра не сохраняется

a new value for such parameter will not be stored after the procedure is completed

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

### **OUT**

данные передаются из хранимой процедуры

data retrieved from a stored procedure

начальное значение такого параметра не принимается во внимание  
внутри хранимой процедуры

initial value of such parameter will not be used in a stored procedure

### **INOUT**

принимается во внимание внутри процедуры, сохраняет значение

such parameter is used within a stored procedure and its value will be stored  
after the procedure is completed, as well

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Список аргументов, заключенных в круглые скобки, необходимо указывать всегда

It is always required to provide a list of arguments within parentheses

Если аргументы отсутствуют, следует использовать пустой список

You should use the empty list if there are no arguments required

Если ни один из модификаторов не указан, считается, что параметр объявлен с ключевым словом IN

If there is no modifier (parameter's type) provided, the parameter will be treated as if this parameter is provided with the modifier IN

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Телом процедуры является составной оператор BEGIN ... END, внутри которого могут располагаться другие операторы

The composite operator BEGIN ... END is considered as the procedure's body in which another operators can be placed

[ label: ] BEGIN  
statements  
END [ label ]

```
CREATE PROCEDURE myProc()  
  outer_block: BEGIN  
    DECLARE l_status int;  
    SET l_status=1;  
    inner_block: BEGIN  
      IF (l_status=1) THEN  
        LEAVE inner_block;  
      END IF;  
      SELECT 'This statement will never be executed';  
    END inner_block;  
    SELECT 'End of program';  
  END outer_block$$
```

Оператор, начинающийся с необязательной метки *label* (любое уникальное имя) может заканчиваться выражением *END label*.

Operator that starts with the optional label *label* (any unique name) can be ended with the statement *END label*.



## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

При работе с хранимыми процедурами символ точки с запятой в конце запроса воспринимается консольным клиентом как сигнал к отправке запроса на сервер

When working with stored procedures, the semicolon at the end of the query is considered by the console client as a signal to send a query to the server

Поэтому следует переопределить разделитель запросов – например, вместо точки с запятой использовать последовательность //

Therefore, you should override the query separator – for example, instead of a semicolon, use the sequence //

```
mysql> DELIMITER //
```

<pre>mysql&gt; SELECT VERSION() +-----+   VERSION()   +-----+   5.0.51b-community-nt   +-----+ 1 row in set (0.00 sec)</pre>	<pre>mysql&gt; CREATE PROCEDURE my_version() -&gt; BEGIN -&gt; SELECT VERSION(); -&gt; END // Query OK, 0 rows affected (0.00 sec)</pre>
--	--

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Чтобы вызвать хранимую процедуру, необходимо применить оператор CALL, после которого помещается имя процедуры и ее параметры в круглых скобках

To call a stored procedure, use the CALL statement, followed by the name of the procedure and its parameters in parentheses

```
mysql> CALL my_version();//
```

```
+-----+  
! VERSION() !  
+-----+  
! 5.0.51b-community-nt !  
+-----+  
1 row in set (0.00 sec)
```

```
Query OK, 0 rows affected (0.00 sec)
```

Рекомендуется избегать использования названий хранимых процедур, совпадающих с именами встроенных функций MySQL

It is recommended to avoid using names of stored procedures that match the names of MySQL built-in functions

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

В теле хранимой процедуры можно использовать многострочный комментарий, который начинается с последовательности `/*` и заканчивается последовательностью `*/`

In the body of the stored procedure, you can use a multi-line comment that starts with the sequence `/*` and ends with the sequence `*/`

Процедура присваивает пользовательской переменной `@x` новое значение

The procedure assigns the user variable `@x` a new value

```
mysql> CREATE PROCEDURE set_x(IN value INT)
-> BEGIN
-> SET @x = value;
-> END//
```

Query OK, 0 rows affected (0.00 sec)

```
mysql>
mysql> CALL set_x(123456)//
Query OK, 0 rows affected (0.00 sec)
```

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Через параметр value процедуре передается числовое значение 123456, которое она присваивает пользовательской переменной @x

Through the value parameter, the procedure takes a numeric value 123456, which it assigns to the user variable @x

```
mysql> SELECT @x//
+-----+
| @x     |
+-----+
| 123456 |
+-----+
1 row in set (0.00 sec)
```

Пользовательская переменная @x является глобальной, она доступна как внутри хранимой процедуры set\_x(), так и вне ее

The @x user variable is global, it is available both inside the set\_x() stored procedure and outside of it

Параметры хранимой процедуры являются локальными

Parameters of the stored procedure are local

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Хранимая процедура `numcatalogs()` имеет один целочисленный параметр `total`, в котором сохраняется число записей в таблице `catalogs`

The *numcatalogs()* stored procedure has one integer parameter *total*, which stores the number of entries in the *catalogs* table

```
mysql> CREATE PROCEDURE numcatalogs(OUT total INT)
-> BEGIN
-> SELECT COUNT(*) INTO total FROM catalogs;
-> END //
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> CALL numcatalogs(@a)//
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT @a//
```

```
+-----+
| @a    |
+-----+
| 5     |
+-----+
```

```
1 row in set (0.00 sec)
```

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Хранимая процедура `catalogname()` возвращает по первичному ключу `catID` название каталога `cat_name`

The stored procedure *catalogname()* returns the catalog name *cat\_name* by the primary key *catID*

```
mysql> CREATE PROCEDURE catalogname(IN id INT, OUT catalog TINYTEXT)
-> BEGIN
-> SELECT cat_name INTO catalog FROM catalogs
-> WHERE catID = id;
-> END //
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> SET @id = 5//
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> CALL catalogname(@id, @name)//
```

Query OK, 0 rows affected (0.02 sec)

```
mysql> SELECT @id, @name//
```

+	-----+	-----+
@id	@name	
+	-----+	-----+
5	Мультимедиа	
+	-----+	-----+

1 row in set (0.00 sec)

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

Хранимые процедуры позволяют реализовать сложную логику с помощью операторов ветвления и циклов

Stored procedures allow to implement complex logic using branching statements and loops

**IF** – оператор ветвления / branching statement

**CASE** – множественный выбор / multiple choice

**WHILE** – оператор цикла / loop statement

**LEAVE** – досрочный выход из цикла / early exit from the cycle (= break)

**ITERATE** – досрочное прекращение итерации / early exit from the iteration (= continue)

**REPEAT** – оператор цикла / loop statement

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

**IF** condition **THEN** statement

[ **ELSEIF** condition **THEN** statement ] ...

[ **ELSE** statement ]

**END IF ;**

Логические выражения можно комбинировать с помощью операторов **&&** (И), а также **||** (ИЛИ)

Logical expressions can be combined with the help of the operators **&&** (AND), as well as **||** (OR)

Если в блоках **IF**, **ELSEIF** и **ELSE** – два или более операторов, необходимо использовать составной оператор **BEGIN ... END**

If there are two or more statements in the **IF**, **ELSEIF** and **ELSE** blocks, you must use the composite **BEGIN ... END** statement



## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

**CASE** expression

**WHEN** value **THEN** statement

[ **WHEN** value **THEN** statement ] ...

[ **ELSE** statement ]

**END CASE ;**

Выражение сравнивается со значениями.

Как только найдено соответствие, выполняется соответствующий оператор или ELSE, если соответствия не найдены

The expression is compared with the values

Once a match is found, the corresponding statement or ELSE (if no match is found) is executed

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

```
[ label: ] WHILE condition DO  
    statements  
END WHILE [ label ] ;
```

Операторы выполняются в цикле, пока истинно условие

Operators are executed in a loop while the condition is true

Если в цикле выполняется более одного оператора, не обязательно заключать их в блок BEGIN ... END, т. к. эту функцию выполняет сам оператор WHILE

If more than one statement is executed in a loop, it is not necessary to enclose them in a BEGIN ... END block, since this role is played by the WHILE statement itself

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

[ label: ] **REPEAT**

statements

**UNTIL** condition **END REPEAT** [ label ] ;

Условие проверяется не в начале, а в конце оператора цикла

The condition is not checked at the beginning, but at the end of the cycle operator

Следует отметить, что цикл выполняется, пока условие ложно

It should be noted that the loop is executed while the condition is false

[ label : ] **LOOP**

statements

**END LOOP** [ label ];

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

```
DELIMITER //
```

```
CREATE PROCEDURE sp_contract_ops(IN op CHAR(1), IN c_num INT, IN c_date TIMESTAMP,  
                                IN s_id INT, IN c_note VARCHAR(100))  
BEGIN  
    IF op = 'i' THEN  
        INSERT INTO contract(contract_date, supplier_id, contract_note)  
            VALUES(CURRENT_TIMESTAMP(), s_id, c_note);  
    ELSEIF op = 'u' THEN  
        UPDATE contract SET contract_date = c_date,  
                            supplier_id = s_id,  
                            contract_note = c_note  
        WHERE contract_number = c_num;  
    ELSE  
        DELETE FROM contract WHERE contract_number = c_num;  
    END IF;  
END //
```

```
CALL sp_contract_ops('i', 0, '2018-12-16', 2, 'contract inserted');  
CALL sp_contract_ops('u', 6, '2018-12-31', 2, 'contract updated');  
CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
```

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

```
drop table if exists m2_products;
```

```
create table m2_products (  
    product_id int not null,  
    product_name varchar(50) not null,  
    product_price decimal(8,2) not null,  
    primary key (product_id)  
);
```

```
insert into m2_products (product_id, product_name, product_price) values  
(1, 'iPhone X', 999),  
(2, 'Samsung S10', 1099),  
(3, 'Honor 8X', 299),  
(4, 'Huawei P Smart', 199),  
(5, 'Xiaomi Mi8', 399);
```

test.m2\_products: 5 rows total (approximately)

 product_id	product_name	product_price
1	iPhone X	999.00
2	Samsung S10	1,099.00
3	Honor 8X	299.00
4	Huawei P Smart	199.00
5	Xiaomi Mi8	399.00

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

```
delimiter $$
create or replace procedure m2_cart_ops(in op_id char(1), in p_id int, in p_amount int)
begin
    create temporary table if not exists m2_cart (
        product_id int not null,
        product_amount int not null,
        primary key (product_id)
    );

    if op_id = 'a' then
        begin
            declare p_count int;
            select count(*) into p_count from m2_cart where product_id = p_id;

            if p_count < 1 then
                insert into m2_cart (product_id, product_amount) values (p_id, p_amount);
            else
                update m2_cart set product_amount = product_amount + p_amount where product_id = p_id;
            end if;

            select concat('Product [' , p_id, ' ] x [' , p_amount, ' ] added to the cart!');
        end;
    end if;

    if op_id = 'c' then
        begin
            select 'Check out';
            select m2_cart.product_id, product_name, product_amount, product_amount * product_price as total
            from m2_cart, m2_products
            where m2_cart.product_id = m2_products.product_id;
        end;
    end if;
end $$
```

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures


```
drop table if exists m2_cart;
```

```
call m2_cart_ops('a', 1, 2);  
call m2_cart_ops('a', 3, 1);  
call m2_cart_ops('a', 4, 4);  
call m2_cart_ops('c', 0, 0);
```

Result #1 (1×1)	Result #2 (1×1)	Result #3 (1×1)	Result #4 (1×1)
concat('Product [, p_id, ] x [, p_amount, ] added to the... Product [1] x [2] added to the cart!')			

Result #1 (1×1)	Result #2 (1×1)	Result #3 (1×1)	Result #4 (1×1)
concat('Product [, p_id, ] x [, p_amount, ] added to the... Product [3] x [1] added to the cart!')			

Result #1 (1×1)	Result #2 (1×1)	Result #3 (1×1)	Result #4 (1×1)
concat('Product [, p_id, ] x [, p_amount, ] added to the... Product [4] x [4] added to the cart!')			

Result #1 (1×1)	Result #2 (1×1)	Result #3 (1×1)	Result #4 (1×1)	Result #5 (4×3)
 product_id	product_name	product_amount	total	
1	iPhone X	2	1,998.00	
3	Honor 8X	1	299.00	
4	Huawei P Smart	4	796.00	

## 4.2 Создание и использование хранимых процедур / Creating and using stored procedures

**DROP PROCEDURE [ IF EXISTS ] procedure\_name;**

используется для удаления хранимых процедур

is used to remove stored procedures

Если удаляемой процедуры с таким именем не существует, оператор возвращает ошибку, которую можно подавить, если использовать необязательное ключевое слово IF EXISTS

If a deleted procedure with that name does not exist, the statement returns an error that can be suppressed by using the optional keyword IF EXISTS



## 4.3 Создание и использование триггеров / Creating and using triggers

Триггер – хранимая процедура, привязанная к событию изменения содержимого конкретной таблицы

Trigger is a stored procedure associated with the event of the contents change of a specific table

Триггер можно привязать к трем событиям, связанным с изменением содержимого таблицы

The trigger can be tied to three events associated with changing the contents of the table

**INSERT**

**DELETE**

**UPDATE**

## 4.3 Создание и использование триггеров / Creating and using triggers

Например, при оформлении нового заказа, т. е. при добавлении новой записи в таблицу *orders*, можно создать триггер, автоматически вычитающий число заказанных товарных позиций в таблице *books*

For example, when placing a new order, that is, when adding a new entry to the *orders* table, you can create a trigger that automatically subtracts the number of ordered items in the *books* table

```
CREATE TRIGGER trigger_name trigger_time trigger_event  
ON table_name FOR EACH ROW  
BEGIN  
    statements  
END ;
```

## 4.3 Создание и использование триггеров / Creating and using triggers

Оператор создает триггер с именем `trigger_name`, привязанный к таблице `table_name`

The operator creates a trigger named *trigger\_name* associated with the table *table\_name*

Не допускается привязка триггера к временной таблице или представлению

Binding a trigger to a temporary table or view is not allowed

Конструкция `trigger_time` указывает момент выполнения триггера

The *trigger\_time* construction specifies the time at which the trigger is executed

## 4.3 Создание и использование триггеров / Creating and using triggers

*trigger\_time*

может принимать два значения / can take two values:

### **BEFORE**

действия триггера производятся до выполнения операции изменения  
таблицы

trigger actions are performed before performing a table change operation

### **AFTER**

действия триггера производятся после выполнения операции изменения  
таблицы

trigger actions are performed after the change table operation

## 4.3 Создание и использование триггеров / Creating and using triggers

Конструкция `trigger_event` показывает, на какое событие должен реагировать триггер, и может принимать три значения

The *trigger\_event* construct indicates which event the trigger should respond to, and can take three values

**INSERT, UPDATE, DELETE**

Для таблицы `table_name` может быть создан только один триггер для каждого из событий `trigger_event` и момента `trigger_time`

For table *table\_name*, only one trigger can be created for each of the *trigger\_event* event and *trigger\_time* time

Таким образом, для каждой из таблиц может быть создано всего шесть триггеров

Thus, for each of the tables, only six triggers can be created

## 4.3 Создание и использование триггеров / Creating and using triggers

**BEGIN**

statements

**END ;**

Тело триггера – оператор, который необходимо выполнить при возникновении события *trigger\_event* в таблице *table\_name*

A trigger body is an operator that must be executed when a *trigger\_event* event occurs in a *table\_name* table

Если требуется выполнить несколько операторов, то необходимо использовать составной оператор BEGIN ... END

If several statements are required, then the composite statement BEGIN ... END must be used

## 4.3 Создание и использование триггеров / Creating and using triggers

Внутри составного оператора BEGIN ... END допускаются все специфичные для хранимых процедур операторы и конструкции:

Inside a BEGIN ... END composite statement, all operators and structures specific to stored procedures are allowed:

- другие составные операторы BEGIN ... END
- another composite operators BEGIN ... END
- операторы управления потоком (IF, CASE, WHILE, LOOP, REPEAT, LEAVE, ITERATE)
- control flow statements (IF, CASE, WHILE, LOOP, REPEAT, LEAVE, ITERATE)
- объявления локальных переменных при помощи оператора DECLARE и назначение им значений при помощи оператора SET
- local variable declarations using the DECLARE operator and assigning values to them using the SET operator

## 4.3 Создание и использование триггеров / Creating and using triggers

Триггеры сложно использовать, не имея доступа к новым записям, которые вставляются в таблицу, или старым записям, которые обновляются или удаляются

Triggers are difficult to use without access to new records that are inserted into a table, or old records that are updated or deleted

Для доступа к новым и старым записям используются префиксы NEW и OLD соответственно

To access new and old records, the prefixes NEW and OLD are used, respectively

Если в таблице обновляется поле *total*, то получить доступ к старому значению можно по имени *OLD.total*, а к новому – *NEW.total*

If the *total* field is updated in the table, then the old value can be accessed by the name *OLD.total*, and the new value – *NEW.total*



## 4.3 Создание и использование триггеров / Creating and using triggers

Рассмотрим триггер, который будет включаться до вставки новых записей в таблицу *orders* и ограничивает число заказываемых товаров до 1

Let's consider a trigger that will be called before inserting new entries into the *orders* table and limits the number of items to be ordered to 1

```
mysql> CREATE TRIGGER restrict_count BEFORE INSERT ON orders
-> FOR EACH ROW
-> BEGIN
-> SET NEW.o_number=1;
-> END//
```

Query OK, 0 rows affected (0.05 sec)

```
mysql> INSERT INTO orders VALUES (NULL,1,2,NOW(),10)//
```

Query OK, 1 row affected (0.01 sec)

```
mysql> SELECT * FROM orders WHERE orderID = LAST_INSERT_ID()//
```

orderID	o_userID	o_bookID	o_time	o_number
16	1	2	2009-10-23 20:26:19	1

1 row in set (0.00 sec)

## 4.3 Создание и использование триггеров / Creating and using triggers

Создадим триггер, который при оформлении нового заказа (при добавлении новой записи в таблицу *orders*) будет увеличивать на 1 значение пользовательской переменной *@tot*

Create a trigger that, when placing a new order (when adding a new entry to the *orders* table), will increase by 1 the value of the user variable *@tot*

```
mysql> delimiter //
mysql> CREATE TRIGGER sub_count AFTER INSERT ON orders
    -> FOR EACH ROW
    -> BEGIN
    -> SET @tot =@tot+1;
    -> END//
```

Query OK, 0 rows affected (0.02 sec)

```
mysql> SELECT @tot //
+-----+
| @tot |
+-----+
| NULL |
+-----+
1 row in set (0.00 sec)
```

## 4.3 Создание и использование триггеров / Creating and using triggers

Для корректной работы триггера необходимо, чтобы пользовательская переменная `@tot` имела значение, отличное от `NULL`, т. к. операция сложения с `NULL` также приводит к `NULL`

For the trigger to work correctly, the `@tot` user variable must have a value other than `NULL`, since the addition operation with `NULL` also results in `NULL`

```
mysql> SET @tot=5//
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO orders VALUES (NULL,1,5,NOW(),10)//
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @tot//
+-----+
| @tot |
+-----+
| 6    |
+-----+
1 row in set (0.00 sec)
```

## 4.3 Создание и использование триггеров / Creating and using triggers

Создадим триггер, который при добавлении новых покупателей преобразует имена и отчества покупателей в инициалы

Create a trigger that when adding new customers converts the names and patronymic of customers into initials

```
mysql> CREATE TRIGGER restrict_user BEFORE INSERT ON users
-> FOR EACH ROW
-> BEGIN
-> SET NEW.u_name = LEFT(NEW.u_name,1);
-> SET NEW.u_patronymic = LEFT(NEW.u_patronymic,1);
-> END//
```

Query OK, 0 rows affected (0.06 sec)

```
mysql> INSERT INTO users VALUES (NULL, 'Светлана', 'Петровна', 'Титова',
-> '83-89-00', NULL, 'active')//
```

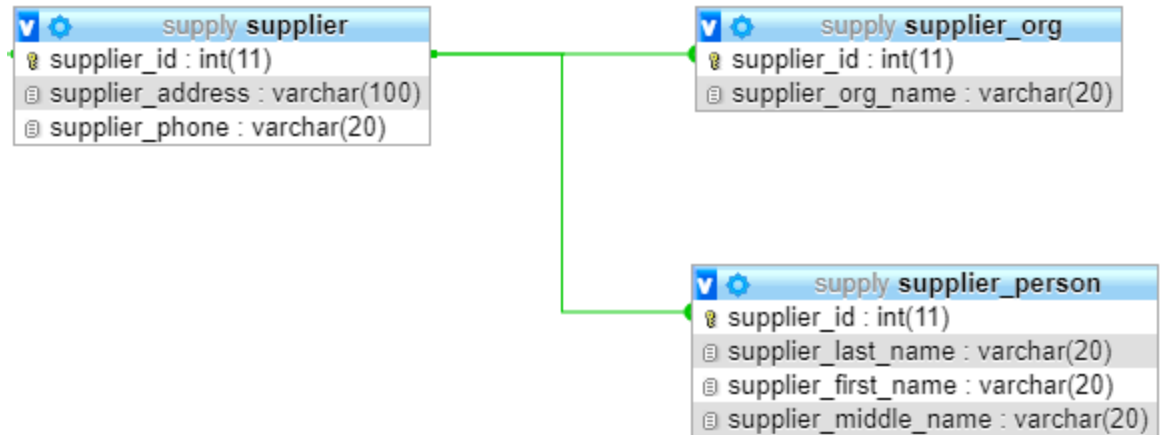
Query OK, 1 row affected (0.03 sec)

```
mysql> SELECT u_surname, u_name, u_patronymic FROM users
-> WHERE userID = LAST_INSERT_ID()//
```

u_surname	u_name	u_patronymic
Титова	С	П

1 row in set (0.00 sec)

## 4.3 Создание и использование триггеров / Creating and using triggers



```
DELIMITER //
```

```
CREATE TRIGGER check_supplier_org BEFORE INSERT ON supplier_person
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF NEW.supplier_id IN (SELECT supplier_id FROM supplier_org) THEN
```

```
        SET @message = CONCAT('The person with id ', NEW.supplier_id,
```

```
                                ' is already stored as the organization!');
```

```
        SIGNAL SQLSTATE '45001'
```

```
        SET MESSAGE_TEXT = @message;
```

```
    END IF;
```

```
END //
```

```
INSERT INTO supplier_person VALUES (2, 'Makarov', 'Oleg', 'Petrovych');
```

## 4.3 Создание и использование триггеров / Creating and using triggers

test delivered
delivery_id : int(11)
supply_id : int(11)
product_id : int(11)
# amount : int(11)
# price : decimal(8,2)

test supplied
supply_id : int(11)
product_id : int(11)
# amount : int(11)
# price : decimal(8,2)

```
create table delivered (  
    delivery_id int not null,  
    supply_id int not null,  
    product_id int not null,  
    amount int not null,  
    price decimal(8,2) not null,  
    primary key (delivery_id, supply_id, product_id)  
) engine=innodb;
```

```
create table supplied (  
    supply_id int not null,  
    product_id int not null,  
    amount int not null,  
    price decimal(8,2) not null,  
    primary key (supply_id, product_id)  
) engine=innodb;
```

```
alter table delivered  
add constraint foreign key (supply_id, product_id) references supplied(supply_id, product_id);
```

## 4.3 Создание и использование триггеров / Creating and using triggers

```
delimiter $$
create trigger tr_dlv_r_amount before insert on delivered
for each row
begin
    DECLARE available int;
    SELECT amount INTO available FROM supplied
        WHERE supplied.supply_id = NEW.supply_id AND
            supplied.product_id = NEW.product_id;
    IF available < NEW.amount THEN
        SET @message = CONCAT('Product ', NEW.product_id, ' is out of stock! Only ',
            available, ' items available. ');
        SIGNAL SQLSTATE '45001'
        SET MESSAGE_TEXT = @message;
    END IF;
end $$

insert into supplied (supply_id, product_id, amount, price)
values (1, 1, 15, 5), (1, 2, 50, 10), (1, 3, 25, 15);

insert into delivered (delivery_id, supply_id, product_id, amount, price)
values (1, 1, 1, 25, 5);
```

**SQL Error (1644): Product 1 is out of stock! Only 15 items available.**

## 4.3 Создание и использование триггеров / Creating and using triggers

```
delimiter $$
create or replace trigger tr_m2_emp_dates before insert on employee
for each row
begin
    insert into t_emp values (new.employee_id, new.first_name, new.last_name,
        new.birth_date, new.onboarding_date);

    if new.onboarding_date <= new.birth_date then
        set @inv_emp_id = new.employee_id;
    end if;
end $$
delimiter ;

drop table if exists t_emp;
create temporary table if not exists t_emp like employee;

insert into employee values (2, 'Adam', 'Lee', '1990-01-01', '1989-01-01');

delete from employee where employee_id = @inv_emp_id;

select * from employee;

select * from t_emp;
```

employee (5x1)				
employee_id	first_name	last_name	birth_date	onboarding_date
1	John	Smith	1993-04-11 00:00:00	2016-01-12 00:00:00

employee (5x1)				
employee_id	first_name	last_name	birth_date	onboarding_date
2	Adam	Lee	1990-01-01 00:00:00	1989-01-01 00:00:00



5 Целостность данных, транзакции, права пользователей

## 5 Data integrity, transactions, user privileges

## 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

```
FOREIGN KEY [name_key] (col1, ... ) REFERENCES tbl (tbl_col, ... )  
[ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT |  
SET DEFAULT}]  
[ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT |  
SET DEFAULT}]
```

Конструкция позволяет задать внешний ключ с необязательным именем `name_key` на столбцах, которые задаются в круглых скобках (один или несколько)

The design allows you to specify a foreign key with the optional name *name\_key* on the columns, which are specified in parentheses (one or more)

## 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

Ключевое слово REFERENCES указывает таблицу *tbl*, на которую ссылается внешний ключ, в круглых скобках указываются имена столбцов

The keyword REFERENCES indicates the *tbl* table referenced by the foreign key, column names are indicated in parentheses

Необязательные конструкции ON DELETE и ON UPDATE позволяют задать поведение СУБД при удалении и обновлении строк из таблицы-предка

Optional constructions ON DELETE and ON UPDATE allow you to specify the behavior of the DBMS when deleting and updating rows from the parent table

## 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

Параметры, следующие за этими ключевыми словами, имеют следующие значения

The parameters following these keywords have the following meanings

### **CASCADE**

при удалении или обновлении записи в таблице-предке, содержащей первичный ключ, записи со ссылками на это значение в таблице-потомке удаляются или обновляются автоматически

when deleting or updating an entry in the parent table containing the primary key, entries with references to this value in the child table are deleted or updated automatically

# 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

## **SET NULL**

при удалении или обновлении записи в таблице-предке, содержащей первичный ключ, в таблице-потомке значения внешнего ключа, ссылающегося на таблицу-предка, устанавливаются в NULL

when deleting or updating an entry in the parent table containing the primary key, in the child table, the foreign key values referring to the ancestor table are set to NULL

## **NO ACTION**

при удалении или обновлении записей, содержащих первичный ключ, с таблицей-потомком никаких действий не производится

when deleting or updating records containing the primary key, no action is taken with the child table

## 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

### **RESTRICT**

если в таблице-потомке имеются записи, ссылающиеся на первичный ключ таблицы-предка, при удалении или обновлении записей с таким первичным ключом возвращается ошибка

if there are records in the child table that refer to the primary key of the parent table, an error is returned when the records with this primary key are deleted or updated

### **SET DEFAULT**

согласно стандарту SQL, при удалении или обновлении первичного ключа в таблице-потомке для ссылающихся на него записей в поле внешнего ключа должно устанавливаться значение по умолчанию (в MySQL это ключевое слово зарезервировано, но не обрабатывается)

according to the SQL standard, when deleting or updating the primary key in the child table, the default key value must be set for the records referring to it in the foreign key field (in MySQL this keyword is reserved but not processed)

## 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

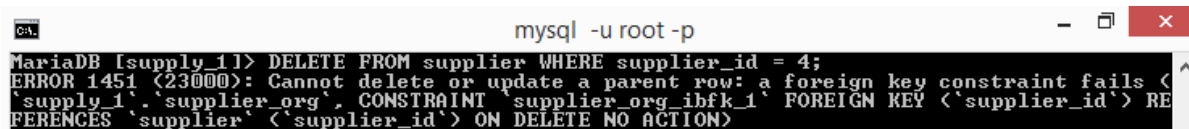
ALTER TABLE supplier_org
DROP FOREIGN KEY supplier_org_ibfk_1;

ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE supplier_person
DROP FOREIGN KEY supplier_person_ibfk_1;

ALTER TABLE supplier_person
ADD CONSTRAINT supplier_person_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

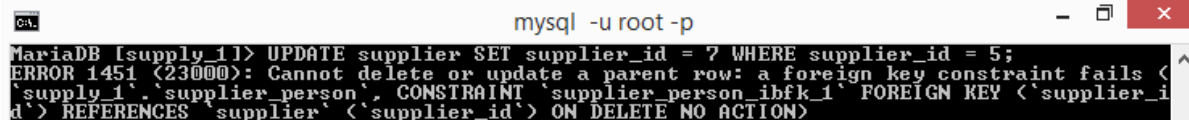
DELETE FROM supplier WHERE supplier_id = 4;
```



A terminal window titled 'mysql -u root -p' shows the following error message:

```
MariaDB [supply_1]> DELETE FROM supplier WHERE supplier_id = 4;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<
'supply_1','supplier_org', CONSTRAINT 'supplier_org_ibfk_1' FOREIGN KEY (<'supplier_id') RE
FERENCES 'supplier' (<'supplier_id') ON DELETE NO ACTION)
```

```
UPDATE supplier SET supplier_id = 7 WHERE supplier_id = 5;
```



A terminal window titled 'mysql -u root -p' shows the following error message:

```
MariaDB [supply_1]> UPDATE supplier SET supplier_id = 7 WHERE supplier_id = 5;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<
'supply_1','supplier_person', CONSTRAINT 'supplier_person_ibfk_1' FOREIGN KEY (<'supplier_i
d') REFERENCES 'supplier' (<'supplier_id') ON DELETE NO ACTION)
```

# 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

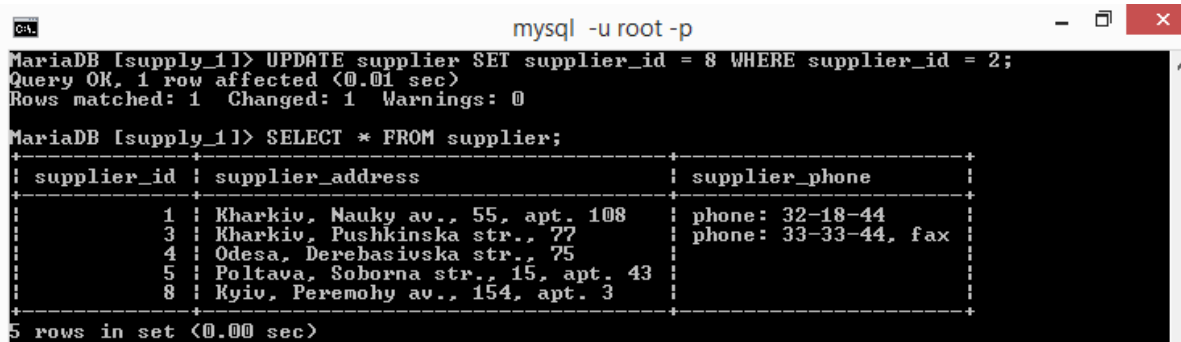
ALTER TABLE supplier_org
DROP FOREIGN KEY supplier_org_ibfk_1;

ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE supplier_person
DROP FOREIGN KEY supplier_person_ibfk_1;

ALTER TABLE supplier_person
ADD CONSTRAINT supplier_person_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

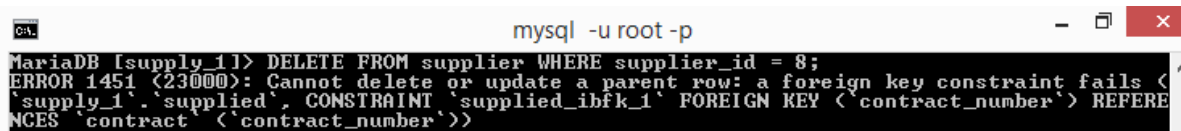
```
UPDATE supplier SET supplier_id = 8 WHERE supplier_id = 2;
```



Terminal window titled 'mysql -u root -p' showing the execution of an UPDATE statement and a SELECT query. The UPDATE statement successfully updates the supplier\_id of the row where supplier\_id = 2 to 8. The SELECT query returns a table with 5 rows.

supplier_id	supplier_address	supplier_phone
1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax
4	Odesa, Derebasivska str., 75	
5	Poltava, Soborna str., 15, apt. 43	
8	Kyiv, Peremohy av., 154, apt. 3	

```
DELETE FROM supplier WHERE supplier_id = 8;
```



Terminal window titled 'mysql -u root -p' showing an attempt to delete a row from the supplier table where supplier\_id = 8. The operation fails with an error message indicating a foreign key constraint failure.

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<supply_1>.`supplier`, CONSTRAINT `supplier_ibfk_1` FOREIGN KEY (<contract_number>) REFERENCES <contract> (<contract_number>))
```



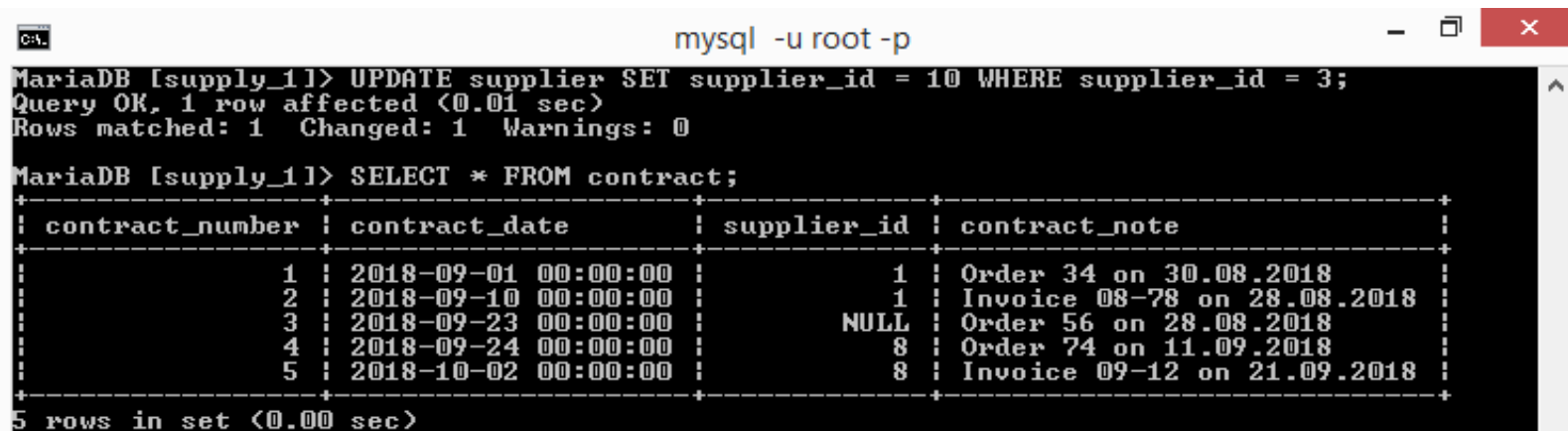
## 5.1 Механизмы контроля целостности данных / Data integrity control mechanisms

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;
```

```
ALTER TABLE contract
MODIFY supplier_id INT NULL;
```

```
ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE SET NULL ON UPDATE SET NULL;
```

```
UPDATE supplier SET supplier_id = 10 WHERE supplier_id = 3;
```



The screenshot shows a MySQL terminal window with the title bar "mysql -u root -p". The terminal output is as follows:

```
MariaDB [supply_11] > UPDATE supplier SET supplier_id = 10 WHERE supplier_id = 3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [supply_11] > SELECT * FROM contract;
```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	NULL	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	8	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	8	Invoice 09-12 on 21.09.2018

```
5 rows in set (0.00 sec)
```

## 5.2 Механизм транзакций / Transactional mechanism

Изменения БД часто требуют выполнения нескольких запросов, например при покупке в электронном магазине требуется добавить запись в таблицу заказов и уменьшить число товарных позиций на складе

Changes in the database often require several requests, for example, when buying from an electronic store, you need to add an entry to the order table and reduce the number of items in the warehouse

В промышленных БД одно событие может затрагивать большее число таблиц и требовать многочисленных запросов

In enterprise databases, one event may affect a lot of tables and require multiple queries

## 5.2 Механизм транзакций / Transactional mechanism

Если на этапе выполнения одного из запросов происходит сбой, это может нарушить целостность БД (товар может быть продан, а число товарных позиций на складе не обновлено)

If at the stage of execution of one of the requests a failure occurs, it can break the integrity of the database (the goods can be sold and the number of items in the warehouse is not updated)

Чтобы сохранить целостность БД, все изменения должны выполняться как единое целое

To preserve the integrity of the database, all changes must be made as a whole

## 5.2 Механизм транзакций / Transactional mechanism

Либо все изменения успешно выполняются, либо, в случае сбоя,  
БД принимает состояние, которое было до начала изменений  
Either all changes are successfully executed, or, in the case of a failure,  
the database returns to a state that was before the start of changes

Это обеспечивается средствами обработки транзакций  
This is provided by transaction processing mechanism

Транзакция – последовательность операторов SQL,  
выполняющихся как единая операция, которая не прерывается  
другими клиентами

Transaction is a sequence of SQL statements executed as a single  
operation that is not interrupted by other clients

## 5.2 Механизм транзакций / Transactional mechanism

Пока происходит работа с записями таблицы (обновление или удаление), никто другой не может получить доступ к этим записям, т. к. MySQL автоматически блокирует доступ к ним

While working with table entries (update or delete), no one else can access these records, since MySQL automatically blocks access to them

Таблицы ISAM, MyISAM и HEAP не поддерживают транзакции, в настоящий момент их поддержка осуществляется только в таблицах BDB и InnoDB

ISAM, MyISAM and HEAP tables do not support transactions, currently they are only supported in BDB and InnoDB tables

## 5.2 Механизм транзакций / Transactional mechanism

Транзакции позволяют объединять операторы в группу и гарантировать, что все операторы группы будут выполнены успешно

Transactions allow you to combine statements into a group and ensure that all statements of the group are executed successfully

Если часть транзакции выполняется со сбоем, результаты выполнения всех операторов транзакции до места сбоя отменяются, приводя БД к виду, в котором она была до выполнения транзакции

If a part of the transaction fails, the results of the execution of all transaction statements before the point of failure are canceled, leading the database to the form in which it was before the execution of the transaction

## 5.2 Механизм транзакций / Transactional mechanism

По умолчанию MySQL работает в режиме автоматического завершения транзакций, т. е. как только выполняется оператор обновления данных, который модифицирует таблицу, изменения тут же сохраняются на диске

By default, MySQL operates in the mode of automatic completion of transactions, i.e., as soon as the data update statement that modifies the table is executed, the changes are immediately saved on disk

Чтобы объединить операторы в транзакцию, следует отключить этот режим: `SET AUTOCOMMIT=0;`

To combine operators into a transaction, you should disable this mode: `SET AUTOCOMMIT = 0;`

## 5.2 Механизм транзакций / Transactional mechanism

После отключения режима для завершения транзакции необходимо ввести оператор COMMIT, для отката – ROLLBACK  
After disabling the mode, you must enter the COMMIT statement to complete the transaction, and ROLLBACK for a rollback

Включить режим автоматического завершения транзакций для отдельной последовательности операторов можно при помощи оператора START TRANSACTION

You can enable the automatic completion of transactions for a separate sequence of statements using the START TRANSACTION operator



## 5.2 Механизм транзакций / Transactional mechanism

Для таблиц InnoDB есть операторы SAVEPOINT и ROLLBACK TO SAVEPOINT, которые позволяют работать с именованными точками начала транзакции

For InnoDB tables, there are SAVEPOINT and ROLLBACK TO SAVEPOINT statements that allow you to work with named transaction start points

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Периферия');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT point1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Разное');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM catalogs;
+-----+-----+
| cat_ID | cat_name |
+-----+-----+
|      1 | Программирование |
|      2 | Интернет |
|      3 | Базы данных |
|      4 | Сети |
|      5 | Мультимедиа |
|     12 | Периферия |
+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM catalogs;
+-----+-----+
| cat_ID | cat_name |
+-----+-----+
|      1 | Программирование |
|      2 | Интернет |
|      3 | Базы данных |
|      4 | Сети |
|      5 | Мультимедиа |
|     12 | Периферия |
|     13 | Разное |
+-----+-----+
7 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT point1;
Query OK, 0 rows affected (0.02 sec)
```

## 5.2 Механизм транзакций / Transactional mechanism

### **Атомарность / Atomicity**

транзакция является атомарной единицей обработки данных,  
следовательно она либо выполняется полностью, либо не  
выполняется совсем

a transaction is an atomic unit of processing, that is, either it is performed in  
its entirety or not performed at all

### **Согласованность / Consistency**

транзакция должна перевести базу данных из одного согласованного  
состояния в другое согласованное состояние

a transaction should take the database from one consistent state to another  
consistent state

## 5.2 Механизм транзакций / Transactional mechanism

### **Изоляция / Isolation**

транзакция должна быть выполнена так, как если бы она была единственной в системе, не должно быть никаких помех от других одновременных транзакций, которые одновременно выполняются

a transaction should be executed as if it is the only one in the system, there should not be any interference from the other concurrent transactions that are simultaneously running

### **Долговечность / Durability**

если зафиксированная транзакция приводит к изменению, это изменение должно быть долговременным в базе данных и не теряться в случае любого сбоя

If a committed transaction brings about a change, that change should be durable in the database and not lost in case of any failure

## 5.2 Механизм транзакций / Transactional mechanism

При параллельном выполнении транзакций возможны следующие проблемы

The following problems are possible when executing transactions in parallel

### **потерянное обновление / lost update**

при одновременном изменении одного блока данных разными транзакциями теряются все изменения, кроме последнего

when simultaneously changing one data block with different transactions, all changes are lost except the last

Transaction 1	Transaction 2
UPDATE table SET a = a + 20 WHERE a = 1	UPDATE table SET a = a + 25 WHERE a = 1

## 5.2 Механизм транзакций / Transactional mechanism

### грязное чтение / dirty read

чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится)

reading data added or modified by a transaction that is not subsequently confirmed (rolled back)

Transaction 1	Transaction 2
UPDATE table SET a = a + 1 WHERE a = 1	
	SELECT a FROM table WHERE a = 1
ROLLBACK	

## 5.2 Механизм транзакций / Transactional mechanism

### неповторяющееся чтение / non-repeatable read

при повторном чтении в рамках одной транзакции ранее прочитанные  
данные оказываются изменёнными

when re-reading in the same transaction, the previously read data is changed

Transaction 1	Transaction 2
	SELECT a FROM table WHERE a = 1
UPDATE table SET a = a + 1 WHERE a = 1	
COMMIT	
	SELECT a FROM table WHERE a = 1

## 5.2 Механизм транзакций / Transactional mechanism

### фантомное чтение / phantom read

Ситуация, когда при повторном чтении в рамках одной транзакции одна и та же выборка дает разные множества строк

The situation when, when re-reading in the same transaction, the same sample gives different sets of rows

Transaction 1	Transaction 2
	SELECT SUM(b) FROM table
INSERT INTO table (a, b) VALUES (15,20)	
COMMIT	
	SELECT SUM(b) FROM table

## 5.2 Механизм транзакций / Transactional mechanism

### **Уровень изоляции транзакций / Transaction isolation level**

степень обеспечиваемой внутренними механизмами СУБД защиты от всех или некоторых видов вышеперечисленных несогласованности данных, возникающих при параллельном выполнении транзакций

the degree of protection provided by the internal mechanisms of the DBMS against all or some of the above listed inconsistencies of data arising during the parallel execution of transactions

### **READ UNCOMMITTED**

Если несколько параллельных транзакций пытаются изменять одну и ту же строку таблицы, то в окончательном варианте строка будет иметь значение, определенное всем набором успешно выполненных транзакций

If several parallel transactions attempt to change the same row of the table, then in the final version the row will have the value defined by the entire set of successfully completed transactions



## 5.2 Механизм транзакций / Transactional mechanism

### **READ COMMITED**

На этом уровне обеспечивается защита от «грязного» чтения, тем не менее, в процессе работы одной транзакции другая может быть успешно завершена и сделанные ею изменения зафиксированы

At this level, protection against a “dirty” reading is provided, however, during the execution of one transaction, the other one can be successfully completed and the changes made by it are fixed

### **REPEATABLE READ**

Уровень, при котором читающая транзакция «не видит» изменения данных, которые были ею ранее прочитаны. При этом никакая другая транзакция не может изменять данные, читаемые текущей транзакцией, пока та не окончена

The level at which the reading transaction "does not see" the changes in the data that it had previously read. However, no other transaction can change the data read by the current transaction until it is completed

## 5.2 Механизм транзакций / Transactional mechanism

### **SERIALIZABLE**

Самый высокий уровень изолированности; транзакции полностью изолируются друг от друга, каждая выполняется последовательно, как будто параллельных транзакций не существует. Только на этом уровне параллельные транзакции не подвержены эффекту «фантомного чтения»

The highest level of isolation; transactions are completely isolated from each other, each performed sequentially, as if parallel transactions do not exist. Only at this level parallel transactions are not affected by "phantom reading"

## 5.2 Механизм транзакций / Transactional mechanism

Isolation level	Phantom reads	Non-repeatable read	Dirty read	Lost update
SERIALIZABLE	+	+	+	+
REPEATABLE READ	-	+	+	+
READ COMMITTED	-	-	+	+
READ UNCOMMITTED	-	-	-	+
NULL	-	-	-	-

## 5.2 Механизм транзакций / Transactional mechanism

**SET [ GLOBAL | SESSION ] TRANSACTION ISOLATION LEVEL**

**{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |  
SERIALIZABLE }**

По умолчанию уровень изоляции устанавливается для последующей (не начальной) транзакции

By default, the isolation level is set for a subsequent (non-initial) transaction

При использовании ключевого слова GLOBAL данная команда устанавливает уровень изоляции по умолчанию глобально для всех новых соединений, созданных от этого момента

When using the GLOBAL keyword, this command sets the default isolation level globally for all new connections created from this moment

При использовании ключевого слова SESSION устанавливается уровень изоляции по умолчанию для всех будущих транзакций, выполняемых в текущем соединении

Using the SESSION keyword sets the default isolation level for all future transactions performed on the current connection

## 5.2 Механизм транзакций / Transactional mechanism

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,  
       supplier.supplier_address, contract.contract_date  
FROM supplied, contract, supplier  
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id  
AND contract.contract_number = 1;
```

```
SET AUTOCOMMIT = 0;  
START TRANSACTION;  
INSERT INTO supplied VALUES (1, 'Vacuum cleaner', 22, 390);
```

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,  
       supplier.supplier_address, contract.contract_date  
FROM supplied, contract, supplier  
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id  
AND contract.contract_number = 1;
```

```
ROLLBACK;
```

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,  
       supplier.supplier_address, contract.contract_date  
FROM supplied, contract, supplier  
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id  
AND contract.contract_number = 1;
```

## 5.2 Механизм транзакций / Transactional mechanism

SQL

mysql -u root -p

```
MariaDB [supply_11] SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

3 rows in set (0.00 sec)

SQL

mysql -u root -p

```
MariaDB [supply_11] SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Vacuum cleaner	390.00	22	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

4 rows in set (0.00 sec)

SQL

mysql -u root -p

```
MariaDB [supply_11] ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

MariaDB [supply_11]
MariaDB [supply_11] SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

3 rows in set (0.00 sec)

## 5.2 Механизм транзакций / Transactional mechanism

```
create table m2_order (  
    order_id int not null,  
    product_id int not null,  
    product_amount int not null,  
    primary key (order_id, product_id),  
    foreign key (product_id) references m2_products(product_id)  
);
```

```
set AUTOCOMMIT = 0;
```

```
start transaction;
```



```
insert into m2_order (order_id, product_id, product_amount) values (1, 1, 1);  
insert into m2_order (order_id, product_id, product_amount) values (1, 3, 3);  
insert into m2_order (order_id, product_id, product_amount) values (1, 4, 2);
```

```
select * from m2_order;
```

```
rollback;
```

```
select * from m2_order;
```

```
set AUTOCOMMIT = 1;
```

m2_order (3×3)			m2_order (3×0)		
 order_id	 product_id	product_amount			
1	1	1			
1	3	3			
1	4	2			

## 5.3 Управление правами пользователей / Manage user privileges

СУБД MySQL является многопользовательской средой, поэтому для доступа к таблицам БД могут быть созданы различные учетные записи с разным уровнем привилегий

MySQL is a multi-user environment, so different accounts with different levels of privileges can be created to access the database tables

Учетной записи пользователя можно предоставить привилегии на просмотр таблицы, добавление новых записей и обновление уже существующих

The user's account can be granted privileges to view the table, add new entries, and update existing ones

Администратору БД можно предоставить более широкие полномочия (возможность создания таблиц, редактирования и удаления уже существующих)

The DBA can be given greater authority (the ability to create tables, edit and delete existing ones)



## 5.3 Управление правами пользователей / Manage user privileges

Для гостя достаточно лишь просмотра таблиц

For a guest, just viewing the tables is enough

Рассмотрим следующие вопросы / Let's consider the following questions:

- создание, редактирование и удаление учетных записей пользователей
- create, edit and delete user accounts
- назначение и отмена привилегий
- assignment and cancellation of privileges

## 5.3 Управление правами пользователей / Manage user privileges

Учетная запись является составной и принимает форму '*username*' @ '*host*', где *username* – имя пользователя, а *host* – наименование хоста, с которого пользователь может обращаться к серверу

The account has composite structure and takes the form of '*username*' @ '*host*', where *username* is the name of the user, and *host* is the name of the host from which the user can access the server

Например, записи '*root*' @ '*127.0.0.1*' и '*wet*' @ '*62.78.56.34*' означают, что пользователь с именем *root* может обращаться с хоста, на котором расположен сервер, а *wet* – только с хоста с IP-адресом 62.78.56.34

For example, the entries '*root*' @ '*127.0.0.1*' and '*wet*' @ '*62.78.56.34*' mean that the user with the name *root* can access from the host where the server is located, and *wet* – only from the host with IP address 62.78.56.34

## 5.3 Управление правами пользователей / Manage user privileges

IP-адрес 127.0.0.1 всегда относится к локальному хосту

IP address 127.0.0.1 always refers to the local host

Если сервер и клиент установлены на одном хосте, то сервер слушает соединения по этому адресу, а клиент отправляет на него SQL-запросы

If the server and client are installed on the same host, the server listens for connections to this address, and the client sends SQL queries to it

IP-адрес 127.0.0.1 имеет псевдоним *localhost*, поэтому учетные записи вида 'root' @ '127.0.0.1' можно записывать в виде 'root' @ 'localhost'

The IP address 127.0.0.1 has an alias of *localhost*, so accounts like 'root' @ '127.0.0.1' can be written as 'root' @ 'localhost'

## 5.3 Управление правами пользователей / Manage user privileges

Число адресов, с которых необходимо обеспечить доступ пользователю, может быть значительным

The number of addresses from which user access should be provided can be significant

Для задания диапазона в имени хоста используется специальный символ "%"

The special character "%" is used to set the range in the host name

Так, учетная запись 'wet' @ '%' позволяет пользователю *wet* обращаться к серверу MySQL с любых компьютеров сети

So, the 'wet' @ '%' account allows the *wet* user to access the MySQL server from any network computers

## 5.3 Управление правами пользователей / Manage user privileges

Все учетные записи хранятся в таблице `user` системной базы данных с именем `mysql`

All accounts are stored in the *user* table of the system database named *mysql*

```
mysql> SELECT Host,User,Password FROM mysql.user;
```

Host	User	Password
localhost	root	
production.mysql.com	root	
127.0.0.1	root	
localhost		
production.mysql.com		

```
5 rows in set (0.27 sec)
```

## 5.3 Управление правами пользователей / Manage user privileges

```
CREATE USER 'username' @ 'host'  
[IDENTIFIED BY [PASSWORD] 'password'];
```

Оператор создает новую учетную запись с необязательным паролем  
The operator creates a new account with an optional password

Если пароль не указан, в его качестве выступает пустая строка  
If the password is not specified, an empty string is used as the password

Разумно хранить пароль в виде хэш-кода, полученного в результате  
необратимого шифрования  
It is reasonable to store the password in the form of a hash code obtained  
from irreversible encryption

## 5.3 Управление правами пользователей / Manage user privileges

Чтобы воспользоваться этим механизмом шифрования, необходимо поместить между ключевым словом IDENTIFIED BY и паролем ключевое слово PASSWORD

To use this encryption mechanism, you should place the keyword PASSWORD between the IDENTIFIED BY keyword and the password

**DROP USER** 'username' @ 'host';

Данный оператор позволяет удалить учетную запись

This operator allows you to delete an account

Изменение имени пользователя в учетной записи осуществляется с помощью оператора

Use the operator to change the username of the account

**RENAME USER** old\_name **TO** new\_name;

## 5.3 Управление правами пользователей / Manage user privileges

Рассмотренные выше операторы позволяют создавать, удалять и редактировать учетные записи, но они не позволяют изменять привилегии пользователя – сообщить MySQL, какой пользователь имеет право только на чтение информации, какой на чтение и редактирование, а кому предоставлены права изменять структуру БД и создавать учетные записи

The above operators allow you to create, delete and edit accounts, but they do not allow changing user privileges – tell MySQL which user has the right to read information only, which one to read and edit, and who has the right to change the database structure and create accounts



## 5.3 Управление правами пользователей / Manage user privileges

Для решения этих задач предназначены операторы **GRANT** (назначает привилегии) и **REVOKE** (удаляет привилегии)

The **GRANT** (assigns privileges) and **REVOKE** (deletes privileges) statements are intended for solving these tasks

Если учетной записи, которая показана в операторе **GRANT**, не существует, то она автоматически создается

If the account shown in the **GRANT** statement does not exist, it is automatically created

Удаление всех привилегий с помощью оператора **REVOKE** не приводит к автоматическому уничтожению учетной записи

Removing all privileges using the **REVOKE** statement does not automatically destroy the account

## 5.3 Управление правами пользователей / Manage user privileges

В простейшем случае оператор GRANT выглядит следующим образом

In the simplest case, the GRANT statement looks like this

```
mysql> GRANT ALL ON *.* TO 'wet'@'localhost' IDENTIFIED BY 'pass';  
Query OK, 0 rows affected (0.17 sec)
```

Данный запрос создает пользователя с именем *wet* и паролем *pass*, который может обращаться к серверу с локального хоста (*localhost*) и имеет все права (*ALL*) для всех баз данных (*\*.\**)

This query creates a user with the name *wet* and a password *pass*, which can access the server from the local host (*localhost*) and has all rights (*ALL*) for all databases (*\*.\**)

Если такой пользователь существует, то его привилегии будут изменены на *ALL*

If such a user exists, his privileges will be changed to *ALL*

## 5.3 Управление правами пользователей / Manage user privileges

Ключевое слово ON в операторе GRANT задает уровень привилегий, которые могут быть заданы на одном из четырех уровней

The ON keyword in the GRANT statement sets the level of privileges that can be set at one of four levels

Для таблиц можно установить только следующие типы привилегий:

**SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX**  
и **ALTER**

Only the following types of privileges can be set for tables: **SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX** and **ALTER**

Это следует учитывать при использовании конструкции GRANT ALL, которая назначает привилегии на текущем уровне

This should be considered when using the GRANT ALL construct, which assigns privileges at the current level

## 5.3 Управление правами пользователей / Manage user privileges

Так, запрос уровня базы данных GRANT ALL ON db.\* не предоставляет никаких глобальных привилегий

So the database level query GRANT ALL ON db. \* does not provide any global privileges

Для отмены привилегий используется оператор **REVOKE**

To cancel privileges, use the **REVOKE** operator

```
mysql> REVOKE DELETE, UPDATE ON *.* FROM 'wet'@'localhost';  
Query OK, 0 rows affected (0.02 sec)
```

Оператор REVOKE отменяет привилегии, но не удаляет учетные записи

Operator REVOKE revokes privileges, but does not delete accounts

## 5.3 Управление правами пользователей / Manage user privileges

Privilege	Description
ALL [PRIVILEGES]	Комбинация всех привилегий, за исключением привилегии GRANT OPTION, которая задается отдельно Combination of all privileges, except GRANT OPTION privilege, which is specified separately
ALTER	Позволяет редактировать таблицу с помощью оператора ALTER TABLE Allows you to edit a table using the ALTER TABLE statement
ALTER ROUTINE	Позволяет редактировать или удалять хранимую процедуру Allows you to edit or delete a stored procedure
CREATE	Позволяет создавать таблицу при помощи оператора CREATE TABLE Allows you to create a table using the operator CREATE TABLE
CREATE ROUTINE	Позволяет создавать хранимую процедуру Allows you to create a stored procedure

## 5.3 Управление правами пользователей / Manage user privileges

Privilege	Description
CREATE TEMPORARY TABLES	Позволяет создавать временные таблицы Allows you to create temporary tables
CREATE USER	Позволяет работать с учетными записями с помощью CREATE USER, DROP USER, RENAME USER и REVOKE ALL PRIVILEGES Allows you to work with accounts using CREATE USER, DROP USER, RENAME USER and REVOKE ALL PRIVILEGES
CREATE VIEW	Позволяет создавать представление с помощью оператора CREATE VIEW Allows you to create a view using the CREATE VIEW statement
DELETE	Позволяет удалять записи при помощи оператора DELETE Allows you to delete records using the operator DELETE
DROP	Позволяет удалять таблицы при помощи оператора DROP TABLE Allows you to delete tables using the DROP TABLE statement

## 5.3 Управление правами пользователей / Manage user privileges

Privilege	Description
EXECUTE	Позволяет выполнять хранимые процедуры Allows you to execute stored procedures
INDEX	Позволяет работать с индексами, в частности, использовать операторы CREATE INDEX и DROP INDEX Allows you to work with indexes, in particular, to use the operators CREATE INDEX and DROP INDEX
INSERT	Позволяет добавлять в таблицу новые записи оператором INSERT Allows you to add new entries to the table using the INSERT statement
LOCK TABLES	Позволяет осуществлять блокировки таблиц при помощи операторов LOCK TABLES и UNLOCK TABLES Allows locking tables using the LOCK TABLES and UNLOCK TABLES statements

## 5.3 Управление правами пользователей / Manage user privileges

Privilege	Description
SELECT	Позволяет осуществлять выборки таблиц оператором SELECT Allows table selection with a SELECT statement
SHOW DATABASES	Позволяет просматривать список всех таблиц на сервере при помощи оператора SHOW DATABASES Allows you to view a list of all tables on the server using the operator SHOW DATABASES
SHOW VIEW	Позволяет использовать оператор SHOW CREATE VIEW Allows the use of the SHOW CREATE VIEW statement
UPDATE	Позволяет обновлять содержимое таблиц оператором UPDATE Allows updating table contents with UPDATE statement
USAGE	Синоним для статуса «отсутствуют привилегии» Synonym for “missing privileges” status
GRANT OPTION	Позволяет управлять привилегиями других пользователей, без данной привилегии невозможно выполнить операторы GRANT и REVOKE Allows you to manage the privileges of other users, without this privilege it is impossible to execute GRANT and REVOKE statements



## 5.3 Управление правами пользователей / Manage user privileges

ON Keyword	Level
ON *.*	Глобальный уровень – пользователь с полномочиями на глобальном уровне может обращаться ко всем БД и таблицам, входящим в их состав Global level – a user with authority at the global level can access all databases and tables included in them
ON db.*	Уровень базы данных – привилегии распространяются на таблицы базы данных db Database Level - Privileges apply to <i>db</i> database tables
ON db.tbl	Уровень таблицы – привилегии распространяются на таблицу <i>tbl</i> базы данных db Table Level - Privileges apply to the <i>tbl</i> table of the <i>db</i> database
ON db.tbl	Уровень столбца – привилегии касаются отдельных столбцов в таблице <i>tbl</i> базы данных db. Список столбцов указывается в скобках через запятую после ключевых слов SELECT, INSERT, UPDATE Column Level — Privileges relate to individual columns in the <i>tbl</i> table of the <i>db</i> database. The list of columns is indicated in parentheses, separated by commas after the keywords SELECT, INSERT, UPDATE

# 6 Построение ПО для работы с БД

## 6 Database application design

## 6.1 Технология JDBC / JDBC technology

### **JDBC (Java DataBase Connectivity)**

стандартный прикладной интерфейс языка Java для  
организации взаимодействия между приложением и СУБД  
standard API (Application Programming Interface) used to  
organize interaction between the application and DBMS

Взаимодействие осуществляется с помощью драйверов JDBC,  
обеспечивающих реализацию общих интерфейсов для  
конкретных СУБД и конкретных протоколов

Interaction is implemented using JDBC drivers that provide  
common interfaces for certain DBMS and protocols

# 6.1 Технология JDBC / JDBC technology

## JDBC drivers

- 1 Используемый другой интерфейс взаимодействия с СУБД, в частности ODBC (**JDBC-ODBC bridge**)  
Driver uses another API to interact with ODBC (**JDBC-ODBC bridge**)  
JDK: sun.jdbc.odbc.JdbcOdbcDriver
- 2 Работающий через внешние (**native**) библиотеки (клиента СУБД)  
Driver uses external (**native**) libraries (through the DBMS client)
- 3 Работающий по **сетевому** и независимому от СУБД протоколу с промежуточным Java-сервером  
Driver uses **network** and DBMS-independent protocol, interacts with intermediate Java-server
- 4 Сетевой драйвер, работающий **напрямую** с СУБД  
Network driver that works **directly** with the DBMS

## 6.1 Технология JDBC / JDBC technology

JDBC предоставляет интерфейс для разработчиков, использующих различные СУБД

JDBC provides the interface for developers that use various DBMS

С помощью JDBC отсылаются SQL-запросы только к реляционным базам данных, для которых существуют драйверы, знающие способ общения с сервером баз данных

JDBC is used to send SQL queries only to relational databases, for which drivers available to interact with the database server exist

JDBC не относится напрямую к J2EE, но так как взаимодействие с СУБД является неотъемлемой частью корпоративных приложений, часто рассматривается в данном контексте

JDBC does not belong to J2EE directly, but it is often considered as its part, since interaction with DBMS is integral part of enterprise applications

## 6.1 Технология JDBC / JDBC technology

Последовательность действий / Sequence of actions

- 1 Загрузка класса драйвера базы данных

Load the class of the database driver

```
String driverName = "org.gjt.mm.mysql.Driver";
```

```
Class.forName(driverName);
```

- 2 Установка соединения с базой данных

Create connection with the database

```
Connection cn =
```

```
DriverManager.getConnection("jdbc:mysql://localhost/my_db", "root",  
"pass");
```

- 3 Создание объекта для передачи запросов

Create object to send queries

```
Statement st = cn.createStatement();
```

## 6.1 Технология JDBC / JDBC technology

Объект класса **Statement** используется для выполнения SQL-запроса без его предварительной подготовки

The object of the **Statement** class is used to execute a SQL query without prior preparation

Могут применяться также объекты классов **PreparedStatement** и **CallableStatement** для выполнения подготовленных запросов и хранимых процедур

Objects of the **PreparedStatement** and **CallableStatement** classes can also be used to execute prepared queries and stored procedures.

4 Выполнение запроса / Query execution

```
ResultSet rs = st.executeQuery("SELECT * FROM my_table");
```

## 6.1 Технология JDBC / JDBC technology

Для добавления, удаления или изменения информации в таблице вместо метода **executeQuery()** запрос помещается в метод **executeUpdate()**

To add, remove or modify information in the table instead of the **executeQuery()** method, the query is placed in the **executeUpdate()** method

- 5 Обработка результатов выполнения запроса производится методами интерфейса **ResultSet**

Processing the results of the query is performed by the methods of the interface **ResultSet**

При первом вызове метода **next()** указатель перемещается на таблицу результатов выборки в позицию первой строки таблицы ответа

The first time the **next()** method is called, the pointer is moved to the table of sample results at the position of the first line of the response table



## 6.1 Технология JDBC / JDBC technology

### 6 Заккрытие соединения / Close the connection

**cn.close();**

Дополнительно требуется подключить библиотеку, содержащую драйвер MySQL

Additionally, you need to connect the library containing the MySQL driver

**mysql-connector-java-3.1.12.jar**

Пользовательская база данных имеет имя db2 и одну таблицу users

The user database is named *db2* and has one *users* table

Field Name	Data Type
name	String
phone	Numeric

## 6.1 Технология JDBC / JDBC technology

```
1  try {
2      try {
3          Class.forName("org.gjt.mm.mysql.Driver");
4          Connection cn = null;
5
6          try {
7              cn = DriverManager.getConnection("jdbc:mysql://localhost/db2", "root", "pass");
8              Statement st = null;
9
10             try {
11                 st = cn.createStatement();
12                 ResultSet rs = null;
13
14                 try {
15                     rs = st.executeQuery("SELECT * FROM users");
16
17                     while (rs.next()) {
18                         System.out.println("Name:-> " + rs.getString(1) +
19                             " Phone:-> " + rs.getInt(2));
20                     }
21                 } finally {
22                     if (rs != null) {
23                         rs.close();
24                     } else {
25                         System.out.println("Error while reading from DB");
26                     }
27                 }
28             }
29         }
30     }
31 }
```

## 6.1 Технология JDBC / JDBC technology

```
28         } finally {
29             if (st != null) {
30                 st.close();
31             } else {
32                 System.out.println("Statement is not created");
33             }
34         }
35     } finally {
36         if (cn != null) {
37             cn.close();
38         } else {
39             System.out.println("Connection is not created");
40         }
41     }
42 } catch (ClassNotFoundException e) {
43     System.out.println("Error while loading DB driver");
44 }
45 } catch (SQLException e) {
46 } catch (IOException e) {
47 }
48
```

## 6.1 Технология JDBC / JDBC technology

Еще один способ соединения с базой данных возможен с использованием файла ресурсов **database.properties**, в котором хранятся, как правило, путь к БД, логин и пароль доступа

Another way to connect to the database is possible using the resource file **database.properties**, which usually stores the database path, login and access password

```
url=jdbc:mysql://localhost/my_db?useUnicode=true&  
characterEncoding=Cp1251
```

```
driver=org.gjt.mm.mysql.Driver
```

```
user=root
```

```
password=pass
```

## 6.1 Технология JDBC / JDBC technology

```
1 public Connection getConnection() throws SQLException {
2     ResourceBundle resource = ResourceBundle.getBundle("database");
3
4     String url = resource.getString("url");
5     String driver = resource.getString("driver");
6     String user = resource.getString("user");
7     String pass = resource.getString("password");
8
9     try {
10         Class.forName(driver).newInstance();
11     } catch (ClassNotFoundException e) {
12         throw new SQLException("Driver is not loaded!");
13     } catch (InstantiationException e) {
14         e.printStackTrace();
15     } catch (IllegalAccessException e) {
16         e.printStackTrace();
17     }
18
19     return DriverManager.getConnection(url, user, pass);
20 }
```

## 6.1 Технология JDBC / JDBC technology

**ResultSetMetaData rsMetaData = rs.getMetaData();** (interface)

**int getColumnCount()**

**String getColumnName(int column)**

**int getColumnType(int column)**

**DatabaseMetaData dbMetaData = cn.getMetaData();** (interface)

**String getDatabaseProductName()**

**String getDatabaseProductVersion()**

**String getDriverName()**

**String getUsername()**

**String getURL()**

## 6.1 Технология JDBC / JDBC technology

Для представления запросов существуют еще два типа объектов **PreparedStatement** и **CallableStatement**. Объекты первого типа используются при выполнении часто повторяющихся запросов SQL. Такой оператор предварительно готовится и хранится в объекте, что ускоряет обмен информацией с базой данных. Второй интерфейс используется для выполнения хранимых процедур, созданных средствами самой СУБД.

There are two other types of objects **PreparedStatement** and **CallableStatement** for representing queries. Objects of the first type are used when performing frequently repeated SQL queries. Such an operator is pre-prepared and stored in the object, which speeds up the exchange of information with the database. The second interface is used to execute stored procedures created by using the DBMS itself.

## 6.1 Технология JDBC / JDBC technology

Для подготовки SQL-запроса, в котором отсутствуют конкретные параметры, используется метод **prepareStatement(String sql)** интерфейса **Connection**, возвращающий объект **PreparedStatement**

To prepare an SQL query that does not contain specific parameters, use the **prepareStatement (String sql)** method of the **Connection** interface, which returns a **PreparedStatement** object

Установка входных значений конкретных параметров этого объекта производится с помощью методов **setString()**, **setInt()** и подобных им, после чего и осуществляется непосредственное выполнение запроса методами **executeUpdate()**, **executeQuery()**

The input values of specific parameters of this object are set using the methods **setString()**, **setInt()** and similar ones, after which the query is executed directly by the methods **executeUpdate()**, **executeQuery()**



## 6.1 Технология JDBC / JDBC technology

```
try {
    Class.forName("org.gjt.mm.mysql.Driver");
    Connection cn = null;

    try {
        cn = DriverManager.getConnection("jdbc:mysql://localhost/db3", "root", "");

        PreparedStatement ps = null;

        String sql = "INSERT INTO emp (id, name, surname, salary) VALUES (?, ?, ?, ?)";
        ps = cn.prepareStatement(sql);

        Rec.insert(ps, 2505, "Mike", "Call", 620);
    } finally {
        if (cn != null) {
            cn.close();
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

## 6.1 Технология JDBC / JDBC technology

```
public static void insert(PreparedStatement ps, int id, String name, String surname, int salary)
    throws SQLException {
    ps.setInt(1, id);
    ps.setString(2, name);
    ps.setString(3, surname);
    ps.setInt(4, salary);

    ps.executeUpdate();
}
```

Интерфейс **CallableStatement** расширяет возможности интерфейса **PreparedStatement** и обеспечивает выполнение хранимых процедур

The **CallableStatement** interface extends the capabilities of the **PreparedStatement** interface and ensures that stored procedures are executed

Пусть в БД существует хранимая процедура **getempname**, которая по уникальному для каждой записи в таблице **employee** числу SSN будет возвращать соответствующее ему имя

Suppose there is a **getempname** stored procedure in the database, which, by the unique number of SSN for each record in the **employee** table, will return the corresponding name

## 6.1 Технология JDBC / JDBC technology

Получение имени работника через вызов хранимой процедуры

Getting the name of employee through a stored procedure call

```
String SQL = "{call getempname (?,?)}";
```

```
CallableStatement cs = conn.prepareCall(SQL);
```

```
cs.setInt(1,822301);
```

```
// регистрация выходного параметра / output parameter registration
```

```
cs.registerOutParameter(2, java.sql.Types.VARCHAR);
```

```
cs.execute();
```

```
String empName = cs.getString(2);
```

```
System.out.println("Employee with SSN:" + ssn + " is " + empName);
```

## 6.1 Технология JDBC / JDBC technology

Для фиксации результатов работы SQL-операторов, логически выполняемых в рамках некоторой транзакции, используется SQL-оператор **COMMIT**. В API JDBC эта операция выполняется по умолчанию после каждого вызова методов **executeQuery()** и **executeUpdate()**.

To fix the results of the work of SQL statements that are logically executed within a certain transaction, use the SQL **COMMIT** statement. In the JDBC API, this operation is performed by default after each call to the **executeQuery()** and **executeUpdate ()** methods.

Если же необходимо сгруппировать запросы и только после этого выполнить операцию **COMMIT**, сначала вызывается метод **setAutoCommit(boolean param)** интерфейса **Connection** с параметром **false**, в результате выполнения которого текущее соединение с БД переходит в режим неавтоматического подтверждения операций.

If it is necessary to group the queries and only after that perform the **COMMIT** operation, first the **Connection** interface's **setAutoCommit(boolean param)** method is called with the parameter **false**, as a result of which the current connection to the database goes into the non-automatic confirmation of operations.

## 6.1 Технология JDBC / JDBC technology

Подтверждает выполнение SQL-запросов метод **commit()** интерфейса **Connection**, в результате действия которого все изменения таблицы производятся как одно логическое действие

The **commit()** method of the **Connection** interface confirms the execution of SQL queries, as a result of which all changes to the table are made as one logical action

Если же транзакция не выполнена, то методом **rollback()** отменяются действия всех запросов SQL, начиная от последнего вызова **commit()**

If the transaction is not completed, then the **rollback()** method cancels the actions of all SQL queries, starting from the last **commit()** call

## 6.1 Технология JDBC / JDBC technology

```
1  Connection cn = null;
2
3  try {
4      cn = getConnection();
5      cn.setAutoCommit(false);
6
7      Statement st = cn.createStatement();
8
9      try {
10         // execute updates
11         // ...
12
13         cn.commit();
14     } catch (SQLException e) {
15         cn.rollback();
16
17         // print errors
18         // ...
19     } finally {
20         if (cn != null) {
21             cn.close();
22         }
23     }
24 } catch (SQLException e) {
25     // print errors
26     // ...
27 }
```

## 6.1 Технология JDBC / JDBC technology

Уровни изоляции транзакций определены в виде констант интерфейса **Connection** (по возрастанию уровня ограничения):

Transaction isolation levels are defined as the constants of the **Connection** interface (by the level of isolation):

**TRANSACTION\_NONE**

**TRANSACTION\_READ\_UNCOMMITTED**

**TRANSACTION\_READ\_COMMITTED**

**TRANSACTION\_REPEATABLE\_READ**

**TRANSACTION\_SERIALIZABLE**

Метод **boolean supportsTransactionIsolationLevel(int level)** интерфейса **DatabaseMetaData** определяет, поддерживается ли заданный уровень изоляции транзакций

Method **boolean supportsTransactionIsolationLevel(int level)** of the interface **DatabaseMetaData** tells whether the transaction isolation level supported

## 6.1 Технология JDBC / JDBC technology

Методы интерфейса **Connection** определяют доступ к уровню изоляции  
Methods of the **Connection** interface provide access to the transaction  
isolation level

**int getTransactionIsolation()**

возвращает текущий уровень изоляции  
returns the current isolation level

**void setTransactionIsolation(int level)**

устанавливает нужный уровень  
sets the required isolation level



## 6.1 Технология JDBC / JDBC technology

При большом количестве клиентов, работающих с приложением, к его базе данных выполняется большое количество запросов. Установление соединения с БД является дорогостоящей (по требуемым ресурсам) операцией. Эффективным способом решения данной проблемы является организация **пула** (pool) используемых соединений, которые не закрываются физически, а хранятся в очереди и предоставляются повторно для других запросов.

With a large number of clients working with the application, a large number of queries are made to its database. Establishing a database connection is an expensive (by the required resources) operation. An effective way to solve this problem is to organize a **pool** of used connections that are not physically closed, but are stored in a queue and re-provided for other requests.

## 6.1 Технология JDBC / JDBC technology

Пул соединений – это одна из стратегий предоставления соединений приложению

Connection pooling is one of the strategies for providing connections to an application

Пул соединений можно организовать с помощью класса **PoolProperties** контейнера Apache Tomcat

A pool of connections can be organized using the **PoolProperties** class of the Apache Tomcat container

Для облегчения создания пула соединений в ApacheTomcat определен собственный класс **DataSource** на основе интерфейса **javax.sql.DataSource**

To facilitate the creation of a connection pool, ApacheTomcat defines its own **DataSource** class based on the **javax.sql.DataSource** interface.

## 6.1 Технология JDBC / JDBC technology

The screenshot shows the Eclipse IDE interface. At the top, the title bar says 'Welcome' with a close button. Below it, a dark blue banner reads 'Eclipse Java EE IDE for Web Developers' with the Eclipse logo on the left. The main area displays three options, each with an orange icon:

- [Review IDE configuration settings](#)  
Review the IDE's most fiercely contested preferences
- Create a new Java EE Web Project**  
Create a new Eclipse project for Java EE Web development
- Create a new Java project**  
Create a new Eclipse project for Java development

The 'Create a new Java EE Web Project' option is circled in red. Below it, the 'Create a new Java EE Web Project' dialog is open. It has the following fields and buttons:

- Project name:** booksdb
- Project location:**
  - ☒ Use default location
  - Location:** D:\GitHub\db-design\booksdb
  - Browse...** button
- Target runtime:**
  - <None>**
  - New Runtime...** button (circled in red)

# 6.1 Технология JDBC / JDBC technology

The screenshot displays the Eclipse IDE interface during the configuration of a web module. The main window is titled "Web Module" and contains the following elements:

- Select the type of runtime environment:** A tree view under "Apache" lists various Tomcat versions. "Apache Tomcat v7.0" is selected and highlighted with a blue border.
- Name:** A text field containing "Apache Tomcat v7.0".
- Tomcat installation directory:** A text field with "apache-tomcat-7.0.47" and a "Browse..." button. The "Download and Install..." button is circled in red.
- JRE:** A dropdown menu set to "Workbench default JRE" with an "Installed JREs..." button.
- License Agreement:** Two radio buttons: "I accept the terms of the license agreement" (selected) and "I do not accept the terms of the license agreement".
- Buttons:** "Finish" and "Cancel" buttons at the bottom right.

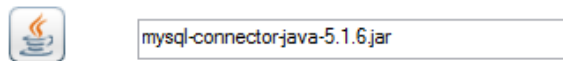
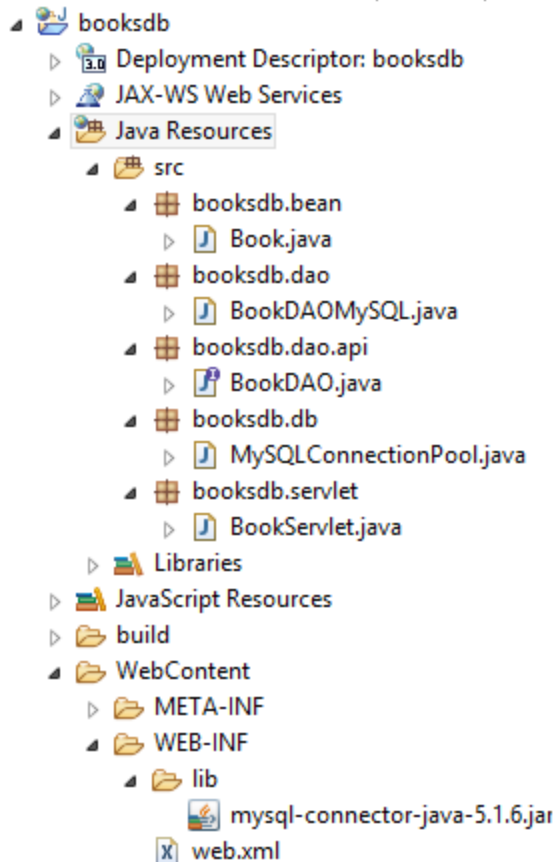
Below the main window, the "Project Explorer" is visible, showing the project structure for "bookssdb":

- bookssdb
  - Deployment Descriptor: bookssdb
  - JAX-WS Web Services
  - Java Resources
    - src
  - Libraries
  - JavaScript Resources
  - build
  - WebContent
    - META-INF
    - WEB-INF
      - lib
      - web.xml

At the bottom left, the "Web Module" configuration panel shows:

- Context root:** bookssdb
- Content directory:** WebContent
- ☒ Generate web.xml deployment descriptor

# 6.1 Технология JDBC / JDBC technology



Type of file: Executable Jar File (.jar)

Opens with: Java(TM) Platform SE b Change...

Location: D:\tomcat\lib

## booksdb.db.MySQLConnectionPool

```
1 package booksdb.db;
2
3 import java.sql.Connection;
4 import java.sql.SQLException;
5
6 import org.apache.tomcat.jdbc.pool.DataSource;
7 import org.apache.tomcat.jdbc.pool.PoolProperties;
8
9 public class MySQLConnectionPool {
10
11     public static Connection getConnection() throws SQLException {
12         PoolProperties p = new PoolProperties();
13
14         p.setUrl("jdbc:mysql://localhost:3306/book");
15         p.setDriverClassName("com.mysql.jdbc.Driver");
16         p.setUsername("root");
17         p.setPassword("");
18
19         DataSource ds = new DataSource();
20         ds.setPoolProperties(p);
21
22         return ds.getConnection();
23     }
24 }
```

# 6.1 Технология JDBC / JDBC technology

## booksdb.bean.Book

```
1 package booksdb.bean;
2
3 public class Book {
4     private int id;
5     private String name;
6     private String author;
7     private int year;
8     private long price;
9     private int count;
10 }
```

## booksdb.dao.BookDAOMySQL.getAll()

```
public List<Book> getAll() {
    List<Book> books = new ArrayList<Book>();
    Connection cn = null;
    try {
        cn = MySQLConnectionPool.getConnection();
        Statement st = cn.createStatement();
        try {
            ResultSet rs = st.executeQuery("SELECT * FROM books");
            while (rs.next()) {
                Book book = new Book();
                book.setId(rs.getInt("book_ID"));
                book.setName(rs.getString("b_name"));
                book.setAuthor(rs.getString("b_author"));
                book.setYear(rs.getInt("b_year"));
                book.setPrice(rs.getLong("b_price"));
                book.setCount(rs.getInt("b_count"));
                books.add(book);
            }
        } finally {
            if (cn != null) {
                cn.close();
            }
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return books;
}
```

## booksdb.dao.api.BookDAO

```
1 package booksdb.dao.api;
2
3 import java.util.List;
4
5 import booksdb.bean.Book;
6
7 public interface BookDAO {
8
9     List<Book> getAll();
10 }
```

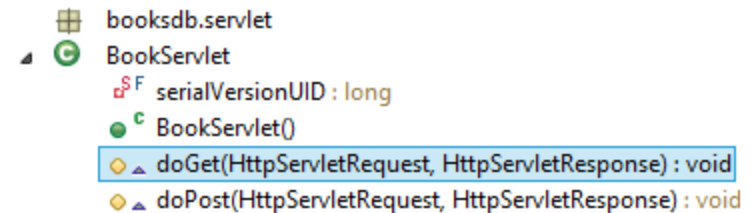
Book

- id: int
- name: String
- author: String
- year: int
- price: long
- count: int
- hashCode(): int
- equals(Object): boolean
- toString(): String
- getId(): int
- setId(int): void
- getName(): String
- setName(String): void
- getAuthor(): String
- setAuthor(String): void
- getYear(): int
- setYear(int): void
- getPrice(): long
- setPrice(long): void
- getCount(): int
- setCount(int): void

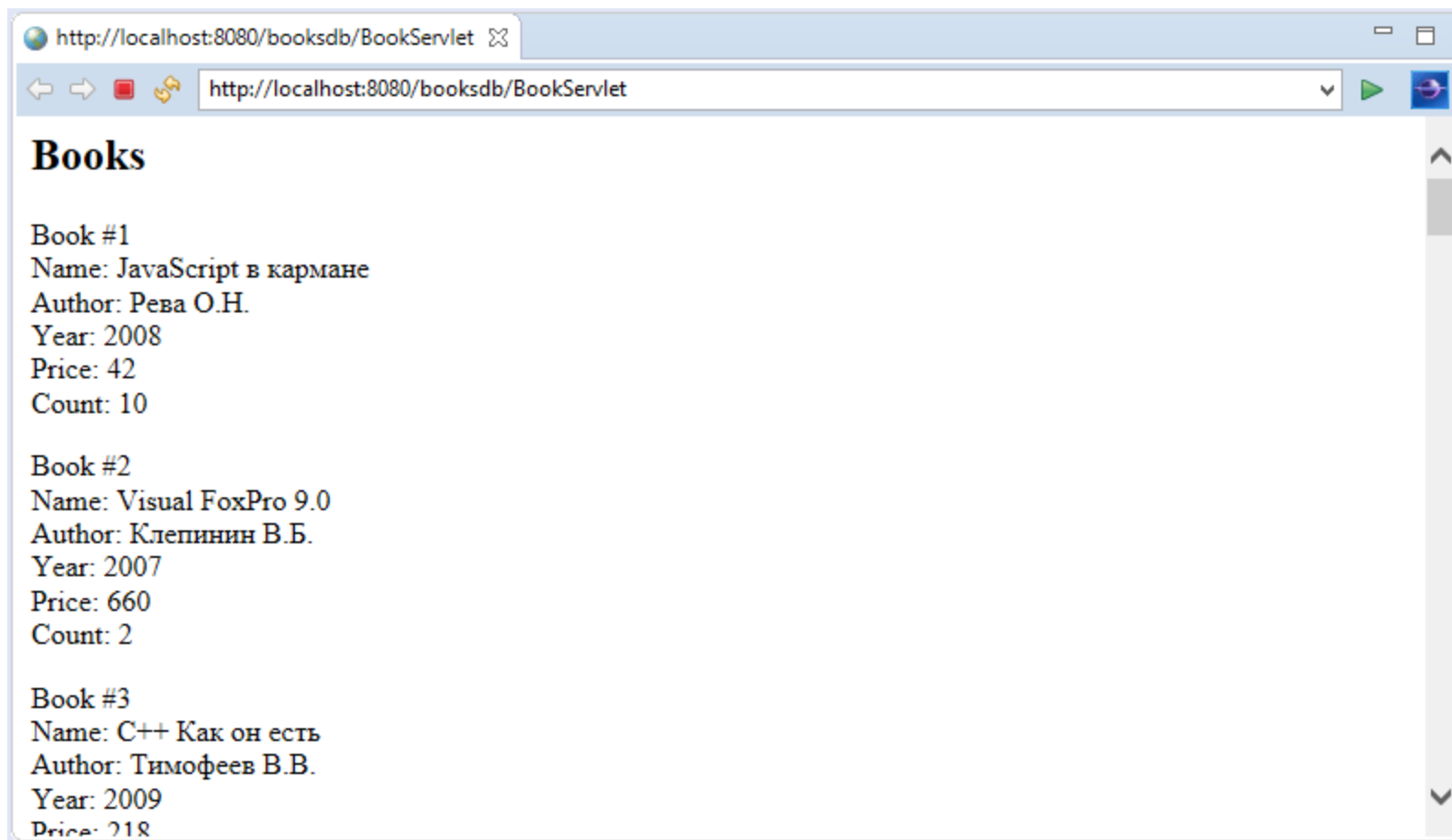
# 6.1 Технология JDBC / JDBC technology

## booksdb.servlet.BookServlet

```
16 @WebServlet("/BookServlet")
17 public class BookServlet extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19
20     public BookServlet() {
21         super();
22     }
23
24     protected void doGet(HttpServletRequest request,
25         HttpServletResponse response) throws ServletException, IOException {
26         response.setContentType("text/html; charset=UTF-8");
27
28         BookDAO bookDAO = new BookDAOMySQL();
29
30         PrintWriter out = response.getWriter();
31         out.print("<h2>Books</h2>");
32
33         for (Book book : bookDAO.getAll()) {
34             out.print("<p>Book #" + book.getId() + "</br>Name: " + book.getName()
35                 "</br>Author: " + book.getAuthor() + "</br>Year: " +
36                 book.getYear() + "</br>Price: " + book.getPrice() +
37                 "</br>Count: " + book.getCount() + "</p>");
38         }
39
40         out.close();
41     }
```



## 6.1 Технология JDBC / JDBC technology





## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

PHP 5 и следующие версии могут работать с СУБД MySQL при помощи  
PHP 5 and later can work with a MySQL database using

- **MySQLi**
- **PDO** (PHP Data Objects)

В более ранних версиях PHP используется расширение **MySQL**,  
признанное устаревшим с 2012 года

Earlier versions of PHP used the MySQL extension. However, this extension  
was deprecated in 2012

Что следует использовать – MySQLi или PDO?

Should I use MySQLi or PDO?

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

PDO будет работать в 12 различных системах баз данных, тогда как MySQLi будет работать только с базами данных MySQL

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases

Таким образом, если вам нужно переключить свой проект на использование другой базы данных, PDO упростит этот процесс. Вам нужно только изменить строку подключения и несколько запросов. С MySQLi вам нужно будет переписать весь код – включая запросы.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Оба являются объектно-ориентированными, но MySQLi также поддерживает процедурный API

Both are object-oriented, but MySQLi also offers a procedural API

Оба поддерживают подготовленные выражения (запросы)

Both support prepared statements

Подготовленные операторы защищают от SQL-инъекций и очень важны для безопасности веб-приложений.

Prepared Statements protect from SQL injection, and are very important for web application security.

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Соединение с MySQL / Open a connection to MySQL

### MySQLi Object-oriented

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Соединение с MySQL / Open a connection to MySQL

### MySQLi Procedural

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Соединение с MySQL / Open a connection to MySQL

### PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}

?>
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Заккрытие соединения / Close the connection

### **MySQLi Object-Oriented**

```
$conn->close();
```

### **MySQLi Procedural**

```
mysqli_close($conn);
```

### **PDO**

```
$conn = null;
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Создание базы данных / Create a database

### MySQLi Object-Oriented

```
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}
```



## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Создание базы данных / Create a database

### MySQLi Procedural

```
// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Создание базы данных / Create a database

### PDO

```
try {  
    $conn = new PDO("mysql:host=$servername", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
    $sql = "CREATE DATABASE myDBPDO";  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    echo "Database created successfully<br>";  
}  
catch(PDOException $e)  
{  
    echo $sql . "<br>" . $e->getMessage();  
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Создание таблицы / Create a table

### MySQLi Object-Oriented

```
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Создание таблицы / Create a table

### MySQLi Procedural

```
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Создание таблицы / Create a table

### PDO

```
$conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
// set the PDO error mode to exception  
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
// sql to create table  
$sql = "CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg_date TIMESTAMP  
)";  
  
// use exec() because no results are returned  
$conn->exec($sql);  
echo "Table MyGuests created successfully";
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

### INSERT

#### MySQLi Object-Oriented

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

---

#### MySQLi Procedural

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

---

#### PDO

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
// use exec() because no results are returned
$conn->exec($sql);
echo "New record created successfully";
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

### DELETE

#### MySQLi Object-Oriented

```
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}
```

---

#### MySQLi Procedural

```
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}
```

---

#### PDO

```
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

// use exec() because no results are returned
$conn->exec($sql);
echo "Record deleted successfully";
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

### UPDATE

#### MySQLi Object-Oriented

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
```

#### MySQLi Procedural

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}
```

#### PDO

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

// Prepare statement
$stmt = $conn->prepare($sql);

// execute the query
$stmt->execute();

// echo a message to say the UPDATE succeeded
echo $stmt->rowCount() . " records UPDATED successfully";
```



## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

### SELECT

### MySQLi Object-Oriented

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]." ".$row["lastname"]."</td>
</tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

### SELECT

#### MySQLi Procedural

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "
<br>";
    }
} else {
    echo "0 results";
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

### SELECT

### PDO

```
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}
```

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

```
$stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
$stmt->execute();

// set the resulting array to associative
$result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
    echo $v;
}
```

Id	Firstname	Lastname
1	John	Doe
2	Mary	Moe
3	Julie	Dooley

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Подготовленные запросы / Prepared statements

### MySQLi

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
```

Заменяемые параметры (?) / Substituted parameters (?)

- i – Integer
- d – Double
- s – String
- b – BLOB

## 6.2 Технологии MySQLi и PDO / MySQLi and PDO technologies

Подготовленные запросы / Prepared statements

### PDO

```
// prepare sql and bind parameters
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (:firstname, :lastname, :email)");
$stmt->bindParam(':firstname', $firstname);
$stmt->bindParam(':lastname', $lastname);
$stmt->bindParam(':email', $email);

// insert a row
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
```

The End