

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №9

Специальность ПО5

Выполнил:
А.А. Игнатюк,
студент группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2022 г.

Брест 2022

Цель работы: Приобрести практические навыки разработки баз данных и начальной интеграции БД с кодом Java с помощью JDBC.

Вариант 5.

Задание.

Реализовать базу данных из не менее 5 таблиц на заданную тематику. При реализации продумать типизацию полей и внешние ключи в таблицах.

Визуализировать разработанную БД с помощью схемы, на которой отображены все таблицы и связи между ними.

На языке Java с использованием JDBC реализовать подключение к БД и выполнить основные типы запросов, продемонстрировать результаты преподавателю и включить тексты составленных запросов в отчет.

Основные типы запросов:

1. На выборку/на выборку с упорядочиванием (SELECT);
2. На добавление (INSERT INTO);
3. На удаление (DELETE FROM);
4. На модификацию (UPDATE).

Базу данные можно реализовать в любой СУБД (MySQL, PostgreSQL, SQLite и др.).

5) База данных “Сборка компьютера”.

Спецификация ввода: -

Спецификация вывода: содержимое базы данных в процессе работы программы.

Схема таблиц базы данных:

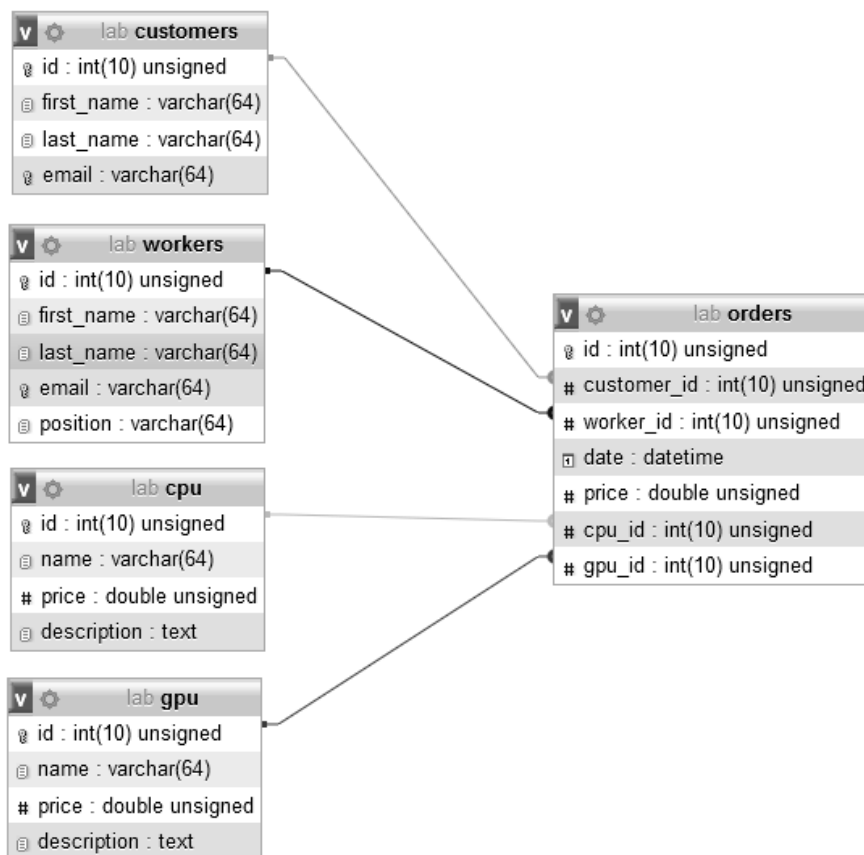


Рисунок 1 - Схема таблиц базы данных.

Код программы и результаты тестирования:

Main.java X

src > lab > Main.java > ...

```
1 package lab;
2
3 import java.sql.*;
4 import java.util.ArrayList;
5
6 public final class Main {
7     private final static String DATABASE_NAME = "lab";
8     private final static String CUSTOMERS_TABLE = "customers";
9     private final static String WORKERS_TABLE = "workers";
10    private final static String CPU_TABLE = "cpu";
11    private final static String GPU_TABLE = "gpu";
12    private final static String ORDERS_TABLE = "orders";
13
14    private final static String HOST = "localhost";
15    private final static String PORT = "3306";
16    private final static String USERNAME = "root";
17    private final static String PASSWORD = "secret";
18    private final static String URL = "jdbc:mysql://" + HOST + ':' + PORT;
19
20    private final static void create() {
21        try {
22            String[] create = {
23                new String("CREATE DATABASE IF NOT EXISTS `" + DATABASE_NAME + "`;"),
24                new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`." +
25                    + CUSTOMERS_TABLE
26                    + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `first_name` VARCHAR(64) NOT NULL , "
27                    + "`last_name` VARCHAR(64) NOT NULL , `email` VARCHAR(64) NOT NULL , PRIMARY KEY (`id`) , "
28                    + "UNIQUE (`email`) ) ENGINE = InnoDB;"),
29                new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`." +
30                    + WORKERS_TABLE
31                    + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `first_name` VARCHAR(64) NOT NULL , "
32                    + "`last_name` VARCHAR(64) NOT NULL , `email` VARCHAR(64) NOT NULL , `position` VARCHAR(64) NOT NULL , "
33                    + "PRIMARY KEY (`id`) , UNIQUE (`email`) ) ENGINE = InnoDB;"),
34                new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`." + CPU_TABLE
35                    + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `name` VARCHAR(64) NOT NULL , "
36                    + "`price` DOUBLE UNSIGNED NOT NULL , `description` TEXT NULL , PRIMARY KEY (`id`) ) ENGINE = InnoDB;"),
37                new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`." + GPU_TABLE
38                    + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `name` VARCHAR(64) NOT NULL , "
39                    + "`price` DOUBLE UNSIGNED NOT NULL , `description` TEXT NULL , PRIMARY KEY (`id`) ) ENGINE = InnoDB;"),
40                new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`." + ORDERS_TABLE
41                    + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `customer_id` INT UNSIGNED NOT NULL , "
42                    + "`worker_id` INT UNSIGNED NOT NULL , `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP , "
43                    + "`price` DOUBLE UNSIGNED NOT NULL , `cpu_id` INT UNSIGNED NOT NULL , `gpu_id` INT UNSIGNED NOT NULL , "
44                    + "PRIMARY KEY (`id`) , INDEX `customer_id_index` (`customer_id`) , INDEX `worker_id_index` (`worker_id`) , "
45                    + "INDEX `cpu_id_index` (`cpu_id`) , INDEX `gpu_id_index` (`gpu_id`) ) ENGINE = InnoDB;"),
46                new String("ALTER TABLE `" + DATABASE_NAME + "`." + ORDERS_TABLE
47                    + "` ADD FOREIGN KEY ( `customer_id` ) REFERENCES `"
48                    + CUSTOMERS_TABLE
49                    + "` ( `id` ) ON DELETE CASCADE ON UPDATE RESTRICT ;"),
50                new String("ALTER TABLE `" + DATABASE_NAME + "`." + ORDERS_TABLE
51                    + "` ADD FOREIGN KEY ( `worker_id` ) REFERENCES `"
52                    + WORKERS_TABLE
53                    + "` ( `id` ) ON DELETE CASCADE ON UPDATE RESTRICT ;"),
54                new String("ALTER TABLE `" + DATABASE_NAME + "`." + ORDERS_TABLE
55                    + "` ADD FOREIGN KEY ( `cpu_id` ) REFERENCES `" + CPU_TABLE
56                    + "` ( `id` ) ON DELETE CASCADE ON UPDATE RESTRICT ;"),
57                new String("ALTER TABLE `" + DATABASE_NAME + "`." + ORDERS_TABLE
58                    + "` ADD FOREIGN KEY ( `gpu_id` ) REFERENCES `" + GPU_TABLE
59                    + "` ( `id` ) ON DELETE CASCADE ON UPDATE RESTRICT ;")
60            };
61
62            Connection connection = DriverManager
63                .getConnection(URL + "?user=" + USERNAME + "&password=" + PASSWORD);
64            Statement statement = connection.createStatement();
65
66            for (final String sql : create) {
67                statement.executeUpdate(sql);
68            }
69
70            statement.close();
71            connection.close();
72            System.out.println("Database " + DATABASE_NAME + " has been created successfully!");
73        } catch (final Exception exception) {
74            exception.printStackTrace();
75        }
76    }
77 }
```

Рисунок 2 - Исходный код программы.

Продолжение рисунка 2.

```
78 private final static void fill() {
79     fillCustomers();
80     fillWorkers();
81     fillCPU();
82     fillGPU();
83     fillOrders();
84 }
85
86 private final static void fillCustomers() {
87     try {
88         String[] customers = {
89             new String("INSERT INTO `" + CUSTOMERS_TABLE
90                 + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Gale', 'Bean', 'Gale_Bean@gmail.com');"),
91             new String("INSERT INTO `" + CUSTOMERS_TABLE
92                 + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Abel', 'Wood', 'Abel_Wood@gmail.com');"),
93             new String("INSERT INTO `" + CUSTOMERS_TABLE
94                 + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Roosevelt', 'Glover', 'Roosevelt_Glover@gmail.com');"),
95             new String("INSERT INTO `" + CUSTOMERS_TABLE
96                 + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Avery', 'Schaefer', 'Avery_Schaefer@gmail.com');"),
97             new String("INSERT INTO `" + CUSTOMERS_TABLE
98                 + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Terry', 'Duffy', 'Terry_Duffy@gmail.com');")
99         };
100
101         Class.forName("com.mysql.cj.jdbc.Driver");
102         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
103             PASSWORD);
104         Statement statement = connection.createStatement();
105
106         for (final String sql : customers) {
107             statement.executeUpdate(sql);
108         }
109
110         statement.close();
111         connection.close();
112         System.out.println(
113             "Database " + DATABASE_NAME + " has been filled with customers successfully!");
114     } catch (final Exception exception) {
115         exception.printStackTrace();
116     }
117 }
118
119 private final static void fillWorkers() {
120     try {
121         String[] workers = {
122             new String("INSERT INTO `" + WORKERS_TABLE
123                 + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Lilac', 'Rick', 'Lilac_Rick@gmail.com', 'junior');"),
124             new String("INSERT INTO `" + WORKERS_TABLE
125                 + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Abilene', 'Debra', 'Abilene-Debra@gmail.com', 'senior');"),
126             new String("INSERT INTO `" + WORKERS_TABLE
127                 + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Caroline', 'Jean', 'Caroline_Jean@gmail.com', 'junior');"),
128             new String("INSERT INTO `" + WORKERS_TABLE
129                 + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Kiki', 'Gene', 'Kiki_Gene@gmail.com', 'middle');"),
130             new String("INSERT INTO `" + WORKERS_TABLE
131                 + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Joanie', 'Julianne', 'Joanie_Julianne@gmail.com', 'middle');")
132         };
133
134         Class.forName("com.mysql.cj.jdbc.Driver");
135         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
136             PASSWORD);
137         Statement statement = connection.createStatement();
138
139         for (final String sql : workers) {
140             statement.executeUpdate(sql);
141         }
142
143         statement.close();
144         connection.close();
145         System.out.println("Database " + DATABASE_NAME + " has been filled with workers successfully!");
146     } catch (final Exception exception) {
147         exception.printStackTrace();
148     }
149 }
150 }
```

Продолжение рисунка 2.

```
151 private final static void fillCPU() {
152     try {
153         String[] CPU = {
154             new String("INSERT INTO `" + CPU_TABLE
155                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Ryzen 3 4100', 99.00, "
156                 + "'The AMD Ryzen 3 4100 is a desktop processor with 4 cores, launched in April 2022. "
157                 + "It is part of the Ryzen 3 lineup, using the Zen 2 (Renoir) architecture with Socket AM4.');" ),
158             new String("INSERT INTO `" + CPU_TABLE
159                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Ryzen 7 5800X3D', 449.00, "
160                 + "'The AMD Ryzen 7 5800X3D is a desktop processor with 8 cores, launched in April 2022. "
161                 + "It is part of the Ryzen 7 lineup, using the Zen 3 (Vermeer) architecture with Socket AM4.');" ),
162             new String("INSERT INTO `" + CPU_TABLE
163                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD FX-6300', 132.00, 'The AMD FX-6300 was "
164                 + "a desktop processor with 6 cores, launched in October 2012. It is part of the FX lineup, using the "
165                 + "Vishera architecture with Socket AM3+. FX-6300 has 8MB of L3 cache and operates at 3.5 GHz by default, "
166                 + "but can boost up to 4.1 GHz, depending on the workload.');" ),
167             new String("INSERT INTO `" + CPU_TABLE
168                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'Intel Core i5-10400F', 220.00, "
169                 + "'The Intel Core i5-10400F is a desktop processor with 6 cores, launched in April 2020. It is part "
170                 + "of the Core i5 lineup, using the Comet Lake architecture with Socket 1200. Thanks to Intel "
171                 + "Hyper-Threading the core-count is effectively doubled, to 12 threads. Core i5-10400F has 12MB of L3 cache "
172                 + "and operates at 2.9 GHz by default, but can boost up to 4.3 GHz.');" ),
173             new String("INSERT INTO `" + CPU_TABLE
174                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'Intel Core i9-12900K', 599.00, "
175                 + "'The Intel Core i9-12900K is a desktop processor with 16 cores, launched in November 2021. "
176                 + "It is part of the Core i9 lineup, using the Alder Lake-S architecture with Socket 1700. Thanks "
177                 + "to Intel Hyper-Threading the core-count is effectively doubled, to 24 threads. Core i9-12900K has "
178                 + "30MB of L3 cache and operates at 3.2 GHz by default, but can boost up to 5.2 GHz.');" )
179         };
180
181         Class.forName("com.mysql.cj.jdbc.Driver");
182         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
183             PASSWORD);
184         Statement statement = connection.createStatement();
185
186         for (final String sql : CPU) {
187             statement.executeUpdate(sql);
188         }
189
190         statement.close();
191         connection.close();
192         System.out.println("Database " + DATABASE_NAME + " has been filled with cpu successfully!");
193     } catch (final Exception exception) {
194         exception.printStackTrace();
195     }
196 }
197
198 private final static void fillGPU() {
199     try {
200         String[] GPU = {
201             new String("INSERT INTO `" + GPU_TABLE
202                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'NVIDIA GeForce RTX 3060', 329.00, "
203                 + "'The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. "
204                 + "Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, "
205                 + "the card supports DirectX 12 Ultimate.');" ),
206             new String("INSERT INTO `" + GPU_TABLE
207                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Radeon RX 6600 XT', 379.00, "
208                 + "'The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. "
209                 + "Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, "
210                 + "the card supports DirectX 12 Ultimate.');" ),
211             new String("INSERT INTO `" + GPU_TABLE
212                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'NVIDIA GeForce GTX 1060 6 GB', 299.00, "
213                 + "'The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on July 19th, 2016. "
214                 + "Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1 variant, "
215                 + "the card supports DirectX 12.');" ),
216             new String("INSERT INTO `" + GPU_TABLE
217                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Radeon RX 6700 XT', 479.00, "
218                 + "'The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. "
219                 + "Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, "
220                 + "the card supports DirectX 12 Ultimate.');" ),
221             new String("INSERT INTO `" + GPU_TABLE
222                 + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'NVIDIA GeForce GTX 1080', 599.00, "
223                 + "'The GeForce GTX 1080 was a high-end graphics card by NVIDIA, launched on May 27th, 2016. "
224                 + "Built on the 16 nm process, and based on the GP104 graphics processor, in its GP104-400-A1 variant, "
225                 + "the card supports DirectX 12.');" )
226         };
227
```

Продолжение рисунка 2.

```
228     Class.forName(className: "com.mysql.cj.jdbc.Driver");
229     Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
230     |     |     PASSWORD);
231     Statement statement = connection.createStatement();
232
233     for (final String sql : GPU) {
234     |     statement.executeUpdate(sql);
235     }
236
237     statement.close();
238     connection.close();
239     System.out.println("Database " + DATABASE_NAME + " has been filled with gpu successfully!");
240 } catch (final Exception exception) {
241 |     exception.printStackTrace();
242 }
243 }
244
245 private final static void fillOrders() {
246     try {
247         String[] orders = {
248             new String("INSERT INTO `" + ORDERS_TABLE
249             |     + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
250             |     + "VALUES (NULL, '3', '2', current_timestamp(), 549.00, '4', '1');"),
251             new String("INSERT INTO `" + ORDERS_TABLE
252             |     + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
253             |     + "VALUES (NULL, '5', '1', current_timestamp(), 748.00, '2', '3');"),
254             new String("INSERT INTO `" + ORDERS_TABLE
255             |     + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
256             |     + "VALUES (NULL, '4', '5', current_timestamp(), 478.00, '1', '2');"),
257             new String("INSERT INTO `" + ORDERS_TABLE
258             |     + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
259             |     + "VALUES (NULL, '2', '3', current_timestamp(), 1078.00, '5', '4');"),
260             new String("INSERT INTO `" + ORDERS_TABLE
261             |     + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
262             |     + "VALUES (NULL, '1', '4', current_timestamp(), 731.00, '3', '5');")
263         };
264
265         Class.forName(className: "com.mysql.cj.jdbc.Driver");
266         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
267         |     |     PASSWORD);
268         Statement statement = connection.createStatement();
269
270         for (final String sql : orders) {
271         |     statement.executeUpdate(sql);
272         }
273
274         statement.close();
275         connection.close();
276         System.out.println("Database " + DATABASE_NAME + " has been filled with orders successfully!");
277     } catch (final Exception exception) {
278     |     exception.printStackTrace();
279     }
280 }
281
282 private final static void print() {
283     printCustomers();
284     printWorkers();
285     printCPU();
286     printGPU();
287     printOrders();
288 }
289
```

Продолжение рисунка 2.

```
290 private final static void printCustomers() {
291     try {
292         System.out.println(x: "Printing customers...\n");
293
294         Class.forName(className: "com.mysql.cj.jdbc.Driver");
295         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
296             PASSWORD);
297         Statement statement = connection.createStatement();
298         ResultSet result = statement
299             .executeQuery(new String(
300                 "SELECT * FROM `" + DATABASE_NAME + "`." + CUSTOMERS_TABLE + "` ORDER BY `first_name`"););
301
302         while (result.next()) {
303             System.out.println("ID : " + result.getInt(columnLabel: "id")
304                 + "\nFirst Name : " + result.getString(columnLabel: "first_name")
305                 + "\nLast Name : " + result.getString(columnLabel: "last_name")
306                 + "\nEmail : " + result.getString(columnLabel: "email") + '\n');
307         }
308
309         result.close();
310         statement.close();
311         connection.close();
312     } catch (final Exception exception) {
313         exception.printStackTrace();
314     }
315 }
316
317 private final static void printWorkers() {
318     try {
319         System.out.println(x: "Printing workers...\n");
320
321         Class.forName(className: "com.mysql.cj.jdbc.Driver");
322         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
323             PASSWORD);
324         Statement statement = connection.createStatement();
325         ResultSet result = statement
326             .executeQuery(new String(
327                 "SELECT * FROM `" + DATABASE_NAME + "`." + WORKERS_TABLE + "` ORDER BY `first_name`"););
328
329         while (result.next()) {
330             System.out.println("ID : " + result.getInt(columnLabel: "id")
331                 + "\nFirst Name : " + result.getString(columnLabel: "first_name")
332                 + "\nLast Name : " + result.getString(columnLabel: "last_name")
333                 + "\nPosition : " + result.getString(columnLabel: "position")
334                 + "\nEmail : " + result.getString(columnLabel: "email") + '\n');
335         }
336
337         result.close();
338         statement.close();
339         connection.close();
340     } catch (final Exception exception) {
341         exception.printStackTrace();
342     }
343 }
344
345 private final static void printCPU() {
346     try {
347         System.out.println(x: "Printing CPU...\n");
348
349         Class.forName(className: "com.mysql.cj.jdbc.Driver");
350         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
351             PASSWORD);
352         Statement statement = connection.createStatement();
353         ResultSet result = statement
354             .executeQuery(
355                 new String("SELECT * FROM `" + DATABASE_NAME + "`." + CPU_TABLE + "` ORDER BY `price`"););
356
357         while (result.next()) {
358             System.out.println("ID : " + result.getInt(columnLabel: "id")
359                 + "\nName : " + result.getString(columnLabel: "name")
360                 + "\nPrice : " + result.getDouble(columnLabel: "price")
361                 + "\nDescription : " + result.getString(columnLabel: "description") + '\n');
362         }
363
364         result.close();
365         statement.close();
366         connection.close();
367     } catch (final Exception exception) {
368         exception.printStackTrace();
369     }
370 }
371 }
```

Продолжение рисунка 2.

```
372 private final static void printGPU() {
373     try {
374         System.out.println(x: "Printing GPU...\n");
375
376         Class.forName(className: "com.mysql.cj.jdbc.Driver");
377         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
378             PASSWORD);
379         Statement statement = connection.createStatement();
380         ResultSet result = statement
381             .executeQuery(
382                 new String("SELECT * FROM `" + DATABASE_NAME + "`." + GPU_TABLE + "` ORDER BY `price`");
383
384         while (result.next()) {
385             System.out.println("ID : " + result.getInt(columnLabel: "id")
386                 + "\nName : " + result.getString(columnLabel: "name")
387                 + "\nPrice : " + result.getDouble(columnLabel: "price")
388                 + "\nDescription : " + result.getString(columnLabel: "description") + '\n');
389         }
390
391         result.close();
392         statement.close();
393         connection.close();
394     } catch (final Exception exception) {
395         exception.printStackTrace();
396     }
397 }
398
399 private final static void printOrders() {
400     try {
401         System.out.println(x: "Printing orders...\n");
402
403         Class.forName(className: "com.mysql.cj.jdbc.Driver");
404         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
405             PASSWORD);
406         Statement statement = connection.createStatement();
407         ResultSet result = statement
408             .executeQuery(new String(
409                 "SELECT * FROM `" + DATABASE_NAME + "`." + ORDERS_TABLE + "` ORDER BY `date`");
410
411         while (result.next()) {
412             System.out.println("ID : " + result.getInt(columnLabel: "id")
413                 + "\nCustomer ID : " + result.getInt(columnLabel: "customer_id")
414                 + "\nWorker ID : " + result.getInt(columnLabel: "worker_id")
415                 + "\nDate : " + result.getDate(columnLabel: "date")
416                 + "\nPrice : " + result.getDouble(columnLabel: "price")
417                 + "\nCPU ID : " + result.getInt(columnLabel: "cpu_id")
418                 + "\nGPU ID : " + result.getInt(columnLabel: "gpu_id") + '\n');
419         }
420
421         result.close();
422         statement.close();
423         connection.close();
424     } catch (final Exception exception) {
425         exception.printStackTrace();
426     }
427 }
428
429 private final static void popBack() {
430     System.out.println(x: "Deleting last order...\n");
431     popBack(ORDERS_TABLE);
432     System.out.println(x: "Deleting last customer...\n");
433     popBack(CUSTOMERS_TABLE);
434     System.out.println(x: "Deleting last worker...\n");
435     popBack(WORKERS_TABLE);
436     System.out.println(x: "Deleting last CPU...\n");
437     popBack(CPU_TABLE);
438     System.out.println(x: "Deleting last GPU...\n");
439     popBack(GPU_TABLE);
440 }
441
```


Продолжение рисунка 2.

```
442 private final static void popBack(final String table) {
443     try {
444         Class.forName("com.mysql.cj.jdbc.Driver");
445         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
446             PASSWORD);
447         Statement statement = connection.createStatement();
448         ResultSet result = statement
449             .executeQuery(new String(
450                 "SELECT * FROM `" + DATABASE_NAME + "`." + table + "` ORDER BY `id`");
451
452         int lastID = -1;
453
454         while (result.next()) {
455             lastID = result.getInt(columnLabel: "id");
456         }
457
458         result.close();
459
460         if (lastID != -1) {
461             statement.executeUpdate(
462                 "DELETE FROM `" + DATABASE_NAME + "`." + table + "` WHERE `id` = '" + lastID + '\''");
463         }
464
465         statement.close();
466         connection.close();
467     } catch (final Exception exception) {
468         exception.printStackTrace();
469     }
470 }
471
472 private final static void miningBoom() {
473     try {
474         System.out.println(x: "Increasing GPU prices...\n");
475
476         Class.forName("com.mysql.cj.jdbc.Driver");
477         Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
478             PASSWORD);
479         Statement statement = connection.createStatement();
480         ResultSet result = statement
481             .executeQuery(
482                 new String("SELECT * FROM `" + DATABASE_NAME + "`." + GPU_TABLE + "` ORDER BY `price`");
483
484         ArrayList<Integer> ID = new ArrayList<Integer>();
485         ArrayList<Double> prices = new ArrayList<Double>();
486
487         while (result.next()) {
488             ID.add(result.getInt(columnLabel: "id"));
489             prices.add(result.getDouble(columnLabel: "price"));
490         }
491
492         result.close();
493
494         for (int i = 0, size = ID.size(); i < size; ++i) {
495             statement.executeUpdate(
496                 new String("UPDATE `" + DATABASE_NAME + "`." + GPU_TABLE + "` SET `price` = '"
497                     + Double.toString(prices.get(i) * 2.5) + "' WHERE `id` = '" + ID.get(i) + '\''");
498         }
499
500         statement.close();
501         connection.close();
502     } catch (final Exception exception) {
503         exception.printStackTrace();
504     }
505 }
506
507 Run | Debug
508 public final static void main(final String[] args) {
509     create();
510     fill();
511     print();
512     popBack();
513     print();
514     miningBoom();
515     printGPU();
516 }
517 }
```

Результат работы программы:

PS D:\Documents\Visual Studio Code\Java\lab> & 'C:\Program Files\Eclipse Foundation\jdk-11.0.12-hotspot\bin\java.exe' '@C:\Users\User\AppData\Local\Temp\cp_bmjpnsdon1bnc1o8uul0nco3p.argfile' 'lab.Main'
Database lab has been created successfully!
Database lab has been filled with customers successfully!
Database lab has been filled with workers successfully!
Database lab has been filled with cpu successfully!
Database lab has been filled with gpu successfully!
Database lab has been filled with orders successfully!
Printing customers...

ID: 2

First Name: Abel

Last Name: Wood

Email: Abel_Wood@gmail.com

ID: 4

First Name: Avery

Last Name: Schaefer

Email: Avery_Schaefer@gmail.com

ID: 1

First Name: Gale

Last Name: Bean

Email: Gale_Bean@gmail.com

ID: 3

First Name: Roosevelt

Last Name: Glover

Email: Roosevelt_Glover@gmail.com

ID: 5

First Name: Terry

Last Name: Duffy

Email: Terry_Duffy@gmail.com

Printing workers...

ID: 2

First Name: Abilene

Last Name: Debra

Position: senior

Email: Abilene_Debra@gmail.com

ID: 3

First Name: Caroline

Last Name: Jean

Position: junior

Email: Caroline_Jean@gmail.com

ID: 5

First Name : Joanie

Last Name : Julianne

Position : middle

Email : Joanie_Julianne@gmail.com

ID: 4

First Name : Kiki

Last Name : Gene

Position : middle

Email : Kiki_Gene@gmail.com

ID: 1

First Name : Lilac

Last Name : Rick

Position : junior

Email : Lilac_Rick@gmail.com

Printing CPU...

ID: 1

Name : AMD Ryzen 3 4100

Price : 99.0

Description : The AMD Ryzen 3 4100 is a desktop processor with 4 cores, launched in April 2022. It is part of the Ryzen 3 lineup, using the Zen 2 (Renoir) architecture with Socket AM4.

ID: 3

Name : AMD FX-6300

Price : 132.0

Description : The AMD FX-6300 was a desktop processor with 6 cores, launched in October 2012. It is part of the FX lineup, using the Vishera architecture with Socket AM3+. FX-6300 has 8MB of L3 cache and operates at 3.5 GHz by default, but can boost up to 4.1 GHz, depending on the workload.

ID: 4

Name : Intel Core i5-10400F

Price : 220.0

Description : The Intel Core i5-10400F is a desktop processor with 6 cores, launched in April 2020. It is part of the Core i5 lineup, using the Comet Lake architecture with Socket 1200. Thanks to Intel Hyper-Threading the core-count is effectively doubled, to 12 threads. Core i5-10400F has 12MB of L3 cache and operates at 2.9 GHz by default, but can boost up to 4.3 GHz.

ID: 2

Name : AMD Ryzen 7 5800X3D

Price : 449.0

Description : The AMD Ryzen 7 5800X3D is a desktop processor with 8 cores, launched in April 2022. It is part of the Ryzen 7 lineup, using the Zen 3 (Vermeer) architecture with Socket AM4.

ID: 5

Name : Intel Core i9-12900K

Price : 599.0

Description : The Intel Core i9-12900K is a desktop processor with 16 cores, launched in November 2021. It is part of the Core i9 lineup, using the Alder Lake-S architecture with Socket 1700. Thanks to Intel Hyper-Threading the core-count is effectively doubled, to 24 threads. Core i9-12900K has 30MB of L3 cache and operates at 3.2 GHz by default, but can boost up to 5.2 GHz.

Printing GPU...

ID: 3

Name : NVIDIA GeForce GTX 1060 6 GB

Price : 299.0

Description : The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on July 19th, 2016. Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1 variant, the card supports DirectX 12.

ID: 1

Name : NVIDIA GeForce RTX 3060

Price : 329.0

Description : The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, the card supports DirectX 12 Ultimate.

ID: 2

Name : AMD Radeon RX 6600 XT

Price : 379.0

Description : The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, the card supports DirectX 12 Ultimate.

ID: 4

Name : AMD Radeon RX 6700 XT

Price : 479.0

Description : The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, the card supports DirectX 12 Ultimate.

ID: 5

Name : NVIDIA GeForce GTX 1080

Price : 599.0

Description : The GeForce GTX 1080 was a high-end graphics card by NVIDIA, launched on May 27th, 2016. Built on the 16 nm process, and based on the GP104 graphics processor, in its GP104-400-A1 variant, the card supports DirectX 12.

Printing orders...

ID: 1

Customer ID: 3

Worker ID: 2

Date : 2022-04-23

Price : 549.0

CPU ID: 4

GPU ID: 1

ID: 2

Customer ID: 5

Worker ID: 1

Date : 2022-04-23

Price : 748.0

CPU ID: 2

GPU ID: 3

ID : 3
Customer ID : 4
Worker ID : 5
Date : 2022-04-23
Price : 478.0
CPU ID : 1
GPU ID : 2

ID : 4
Customer ID : 2
Worker ID : 3
Date : 2022-04-23
Price : 1078.0
CPU ID : 5
GPU ID : 4

ID : 5
Customer ID : 1
Worker ID : 4
Date : 2022-04-23
Price : 731.0
CPU ID : 3
GPU ID : 5

Deleting last order...

Deleting last customer...

Deleting last worker...

Deleting last CPU...

Deleting last GPU...

Printing customers...

ID : 2
First Name : Abel
Last Name : Wood
Email : Abel_Wood@gmail.com

ID : 4
First Name : Avery
Last Name : Schaefer
Email : Avery_Schaefer@gmail.com

ID : 1
First Name : Gale
Last Name : Bean
Email : Gale_Bean@gmail.com

ID : 3
First Name : Roosevelt
Last Name : Glover
Email : Roosevelt_Glover@gmail.com

Printing workers...

ID: 2

First Name : Abilene

Last Name : Debra

Position : senior

Email : Abilene_Debra@gmail.com

ID: 3

First Name : Caroline

Last Name : Jean

Position : junior

Email : Caroline_Jean@gmail.com

ID: 4

First Name : Kiki

Last Name : Gene

Position : middle

Email : Kiki_Gene@gmail.com

ID: 1

First Name : Lilac

Last Name : Rick

Position : junior

Email : Lilac_Rick@gmail.com

Printing CPU...

ID: 1

Name : AMD Ryzen 3 4100

Price : 99.0

Description : The AMD Ryzen 3 4100 is a desktop processor with 4 cores, launched in April 2022. It is part of the Ryzen 3 lineup, using the Zen 2 (Renoir) architecture with Socket AM4.

ID: 3

Name : AMD FX-6300

Price : 132.0

Description : The AMD FX-6300 was a desktop processor with 6 cores, launched in October 2012. It is part of the FX lineup, using the Vishera architecture with Socket AM3+. FX-6300 has 8MB of L3 cache and operates at 3.5 GHz by default, but can boost up to 4.1 GHz, depending on the workload.

ID: 4

Name : Intel Core i5-10400F

Price : 220.0

Description : The Intel Core i5-10400F is a desktop processor with 6 cores, launched in April 2020. It is part of the Core i5 lineup, using the Comet Lake architecture with Socket 1200. Thanks to Intel Hyper-Threading the core-count is effectively doubled, to 12 threads. Core i5-10400F has 12MB of L3 cache and operates at 2.9 GHz by default, but can boost up to 4.3 GHz.

ID: 2

Name : AMD Ryzen 7 5800X3D

Price : 449.0

Description : The AMD Ryzen 7 5800X3D is a desktop processor with 8 cores, launched in April 2022. It is part of the Ryzen 7 lineup, using the Zen 3 (Vermeer) architecture with Socket AM4.

Printing GPU...

ID: 3

Name : NVIDIA GeForce GTX 1060 6 GB

Price : 299.0

Description : The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on July 19th, 2016. Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1 variant, the card supports DirectX 12.

ID: 1

Name : NVIDIA GeForce RTX 3060

Price : 329.0

Description : The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, the card supports DirectX 12 Ultimate.

ID: 2

Name : AMD Radeon RX 6600 XT

Price : 379.0

Description : The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, the card supports DirectX 12 Ultimate.

ID: 4

Name : AMD Radeon RX 6700 XT

Price : 479.0

Description : The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, the card supports DirectX 12 Ultimate.

Printing orders...

ID: 1

Customer ID: 3

Worker ID: 2

Date : 2022-04-23

Price : 549.0

CPU ID: 4

GPU ID: 1

Increasing GPU prices...

Printing GPU...

ID: 3

Name : NVIDIA GeForce GTX 1060 6 GB

Price : 747.5

Description : The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on July 19th, 2016. Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1 variant, the card supports DirectX 12.

ID : 1

Name : NVIDIA GeForce RTX 3060

Price : 822.5

Description : The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, the card supports DirectX 12 Ultimate.

ID : 2

Name : AMD Radeon RX 6600 XT

Price : 947.5

Description : The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, the card supports DirectX 12 Ultimate.

ID : 4

Name : AMD Radeon RX 6700 XT

Price : 1197.5

Description : The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, the card supports DirectX 12 Ultimate.

PS D:\Documents\Visual Studio Code\Java\lab>

Вывод: Приобрел практические навыки разработки баз данных и начальной интеграции БД с кодом Java с помощью JDBC.