|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 2 |

**Название:**

Создание БД для приложения

**Дисциплина:** Базы данных

**Вариант:** 23

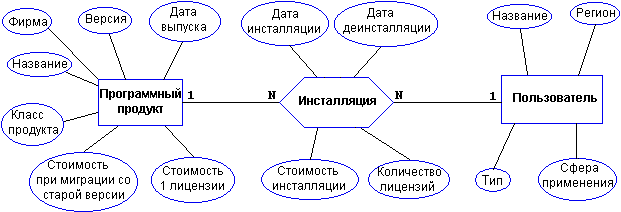
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б |  |  | Д. И. Мироненко |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | М. А. Скворцова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Часть 1. Проектирование схемы базы данных**

**Задание**

Каждый индивидуальный вариант содержит предметную область, из которой должна быть проектируемая база данных. К данной предметной области необходимо добавить не менее 2-х дополнительных таблиц (сущностей), необходимых для детального решения поставленной задачи. Задачей студента является решить, для чего будет использоваться создаваемая база данных, и, исходя из этого, построить её концептуальную схему. Результатом данной части лабораторной работы является схема базы данных (в виде ER-диаграммы, содержащей таблицы и связи между ними, с уточнением типов столбцов, с описание внешних и первичных ключей). При сдаче задания студент должен обосновать соответствие созданной схемы поставленной задаче.



**Рисунок 1** – предметная схема

A screenshot of a computer

Description automatically generated

**Рисунок 2** – ER-диаграмма

**Часть 2. Создание и заполнения таблиц**

**Задание**

Необходимо определить первичные и внешние ключи, а также декларативные ограничения целостности (возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д.). Таблицы следует создавать в отдельной базе данных. Кроме того, нужно подготовить данные для заполнения созданных таблиц. Кроме того, нужно подготовить данные для заполнения созданных таблиц. Объем подготовленных данных должен составлять не менее 10 экземпляров для каждой из стержневых сущностей и 1000 экземпляров для целевой сущности. На основе этих данных необходимо создать SQL-скрипт для вставки соответствующих строк в таблицы БД.

**SQL-скрипт для разметки таблицы**

|  |
| --- |
| DROP SCHEMA IF EXISTS "public" CASCADE;  CREATE SCHEMA IF NOT EXISTS "public";  CREATE TABLE "products" (  "id" integer PRIMARY KEY,  "name" varchar UNIQUE NOT NULL,  "organization" varchar NOT NULL,  "release\_date" date NOT NULL,  "category" integer NOT NULL,  "migration\_price" integer NOT NULL,  "license\_price" integer NOT NULL  );  CREATE TABLE "users" (  "id" integer PRIMARY KEY,  "username" varchar UNIQUE NOT NULL,  "email" varchar UNIQUE NOT NULL,  "password" varchar NOT NULL,  "first\_name" varchar,  "last\_name" varchar,  "country" integer  );  CREATE TABLE "installations" (  "id" integer PRIMARY KEY,  "installation\_date" date NOT NULL,  "uninstallation\_date" date NOT NULL,  "user\_id" integer NOT NULL,  "software\_id" integer NOT NULL  );  CREATE TABLE "product\_categories" (  "id" integer PRIMARY KEY,  "name" varchar NOT NULL  );  CREATE TABLE "countries" (  "id" integer PRIMARY KEY,  "name" varchar NOT NULL  );  ALTER TABLE "products" ADD FOREIGN KEY ("category") REFERENCES "product\_categories" ("id") ON DELETE CASCADE ON UPDATE CASCADE;  ALTER TABLE "users" ADD FOREIGN KEY ("country") REFERENCES "countries" ("id") ON DELETE CASCADE ON UPDATE CASCADE;  ALTER TABLE "installations" ADD FOREIGN KEY ("user\_id") REFERENCES "users" ("id") ON DELETE CASCADE ON UPDATE CASCADE;  ALTER TABLE "installations" ADD FOREIGN KEY ("software\_id") REFERENCES "products" ("id") ON DELETE CASCADE ON UPDATE CASCADE; |

**SQL-скрипт для вставки значений в таблицу**

|  |
| --- |
| INSERT INTO "public"."countries" ("id", "name") VALUES ('1', 'Russia');  INSERT INTO "public"."countries" ("id", "name") VALUES ('2', 'Argentina');  INSERT INTO "public"."countries" ("id", "name") VALUES ('3', 'Brazil');  INSERT INTO "public"."countries" ("id", "name") VALUES ('4', 'Canada');  INSERT INTO "public"."countries" ("id", "name") VALUES ('5', 'Denmark');  INSERT INTO "public"."countries" ("id", "name") VALUES ('6', 'Estonia');  INSERT INTO "public"."countries" ("id", "name") VALUES ('7', 'France');  INSERT INTO "public"."countries" ("id", "name") VALUES ('8', 'Germany');  INSERT INTO "public"."countries" ("id", "name") VALUES ('9', 'Honduras');  INSERT INTO "public"."countries" ("id", "name") VALUES ('10', 'India');  INSERT INTO "public"."countries" ("id", "name") VALUES ('11', 'Japan');  INSERT INTO "public"."countries" ("id", "name") VALUES ('12', 'Kenya');  INSERT INTO "public"."countries" ("id", "name") VALUES ('13', 'Lithuania');  INSERT INTO "public"."countries" ("id", "name") VALUES ('14', 'Malaysia');  INSERT INTO "public"."countries" ("id", "name") VALUES ('15', 'Netherlands');  INSERT INTO "public"."countries" ("id", "name") VALUES ('16', 'Oman');  INSERT INTO "public"."countries" ("id", "name") VALUES ('17', 'Poland');  INSERT INTO "public"."countries" ("id", "name") VALUES ('18', 'Qatar');  INSERT INTO "public"."countries" ("id", "name") VALUES ('19', 'South Korea');  INSERT INTO "public"."countries" ("id", "name") VALUES ('20', 'Tanzania');  INSERT INTO "public"."countries" ("id", "name") VALUES ('21', 'Venezuela');  INSERT INTO "public"."countries" ("id", "name") VALUES ('22', 'Wales');  INSERT INTO "public"."countries" ("id", "name") VALUES ('23', 'Xinjiang');  INSERT INTO "public"."countries" ("id", "name") VALUES ('24', 'Yemen');  INSERT INTO "public"."countries" ("id", "name") VALUES ('25', 'Zimbabwe');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('1', 'OS');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('2', 'VR');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('3', 'Business');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('4', 'Education');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('5', 'Finance');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('6', 'Graphics & Design');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('7', 'Kids');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('8', 'Magazines & Newspapers');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('9', 'Music');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('10', 'News');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('11', 'Productivity');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('12', 'Browser Extension');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('13', 'Social Networking');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('14', 'Travel');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('15', 'Books');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('16', 'Developer Tools');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('17', 'Entertainment');  INSERT INTO "public"."product\_categories" ("id", "name") VALUES ('18', 'Food & Drink');  ... |

**Часть 3. Создание и заполнения таблиц**

**Задание**

- Подготовить 3-4 выборки, которые имеют осмысленное значение для предметной области, и также составить для них SQL-скрипты.

- Сформулировать 3-4 запроса на изменение и удаление из базы данных. Запросы должны быть сформулированы в терминах предметной области. Среди запросов обязательно должны быть такие, которые будут вызывать срабатывание ограничений целостности. Составить SQL-скрипты для выполнения этих запросов.

*Получение списка установленных продуктов для конкретного пользователя, включая информацию о дате установки и удаления*

|  |
| --- |
| SELECT products.name, installations.installation\_date, installations.uninstallation\_date  FROM installations  INNER JOIN products ON installations.software\_id = products.id  WHERE installations.user\_id = 34; |

A screenshot of a computer

Description automatically generated

**Рисунок 3** – результат SELECT запроса

*Получение общего количества установок каждого продукта*

|  |
| --- |
| SELECT products.name, COUNT(installations.id) AS installation\_count  FROM products  LEFT JOIN installations ON products.id = installations.software\_id  GROUP BY products.name; |

A screenshot of a computer

Description automatically generated

**Рисунок 4** – результат SELECT запроса

*Получение списка всех установок продуктов, которые были установлены после определенной даты, включая информацию о пользователе и продукте*

|  |
| --- |
| SELECT users.username, products.name, installations.installation\_date  FROM installations  INNER JOIN users ON installations.user\_id = users.id  INNER JOIN products ON installations.software\_id = products.id  WHERE installations.installation\_date > '2023-11-1'; |

A screenshot of a computer

Description automatically generated

**Рисунок 5** – результат SELECT запроса

*Обновление цены миграции и лицензии для продукта с определенным идентификатором*

|  |
| --- |
| UPDATE products SET migration\_price = 30, license\_price = 100 WHERE id = 6; |

A screenshot of a computer

Description automatically generated

**Рисунок 6**– результат UPDATE запроса

*Обновление имени, фамилии и страны пользователя с определенным идентификатором*

|  |
| --- |
| UPDATE users SET  first\_name = 'Ivan',  last\_name = 'Ivanov',  country = (SELECT countries.id FROM countries WHERE name = 'Russia')  WHERE id = 98; |

A screenshot of a computer

Description automatically generated

**Рисунок 7** – результат UPDATE запроса

*Удаление всех установок, которые были удалены до 2020 года*

|  |
| --- |
| DELETE FROM installations WHERE uninstallation\_date < '2020-01-01'; |

A screenshot of a computer

Description automatically generated

**Рисунок 8** – результат DELETE запроса

**Часть 4. Контроль целостности данных**

**Задание**

Необходимо подготовить SQL-скрипты для проверки наличия аномалий (потерянных изменений, грязных чтений, неповторяющихся чтений, фантомов) при параллельном исполнении транзакций на различных уровнях изолированности SQL/92 (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE). Подготовленные скрипты должны работать с одной из таблиц, созданных в практическом задании №2.1. Для проверки наличия аномалий потребуются два параллельных сеанса, операторы в которых выполняются пошагово:

* Установить в обоих сеансах уровень изоляции READ UNCOMMITTED. Выполнить сценарии проверки наличия аномалий потерянных изменений и грязных чтений.
* Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.
* Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.
* Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарий проверки наличия фантомов.

Необходимо составить скрипт для создания триггера, а также подготовить несколько запросов для проверки и демонстрации его полезных свойств:

* Изменение данных для сохранения целостности.
* Проверка транзакций и их откат в случае нарушения целостности.

**SQL транзакция с уровнем READ UNCOMMITED**

A screenshot of a computer

Description automatically generated

**Рисунок 9** – результат транзакции c READ UNCOMMITED

**SQL транзакция с уровнем READ COMMITED**

A screenshot of a computer

Description automatically generated

**Рисунок 10** – результат транзакции c READ COMMITED

**SQL транзакция с уровнем REPEATABLE READ**

A screenshot of a computer

Description automatically generated

**Рисунок 11** – результат транзакции c REPEATABLE READ

**SQL транзакция с уровнем SERIALIZABLE**

A screenshot of a computer

Description automatically generated

**Рисунок 12** – результат транзакции c SERIALIZABLE

**SQL триггер**

A screenshot of a computer

Description automatically generated

**Рисунок 13** – результат работы триггера

**Вывод**

Создана и настроена реляционная база данных; реализован скрипт на python для генерации случайных данных; написаны различные SQL запросы; получены и применены на практике знания о транзакциях и триггерах.