|  |  |
| --- | --- |
| **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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | А. П. Плютто |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | М. А. Скворцова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:**

Данная лабораторная работа призвана сформировать у студента

понимание особенностей хранения данных приложения в РСУБД, а также

настройка и поддержка хранения данных.

**Задачи:**

• Получить теоретические знания по концептуальным картам.

• Ознакомится с нормализацией в БД.

• Изучение типов связей.

• Ознакомится с DDL операторами.

• Изучение типов данных.

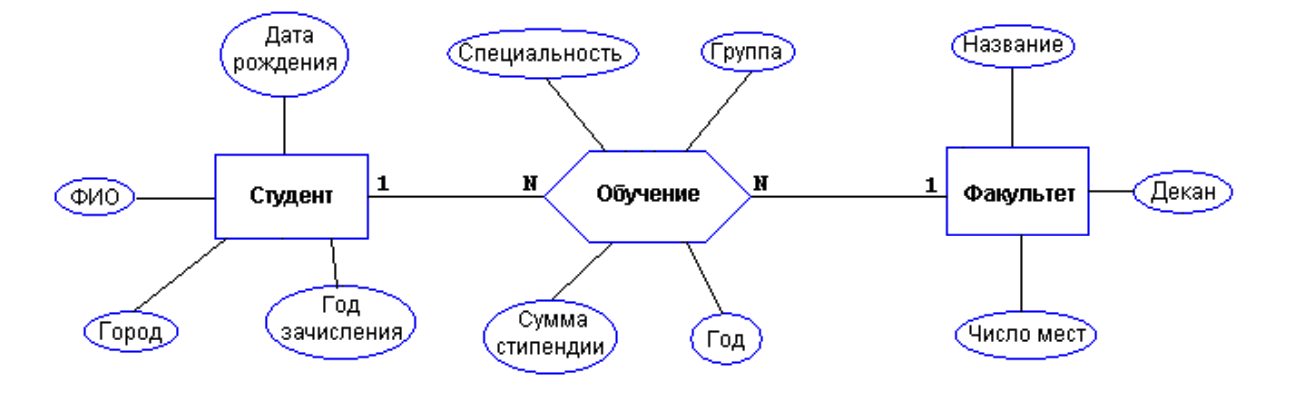
• Научится добавлять записи в таблицы.

• Научиться удалять и изменять записи в таблице.

• Научиться контролю целостности данных.

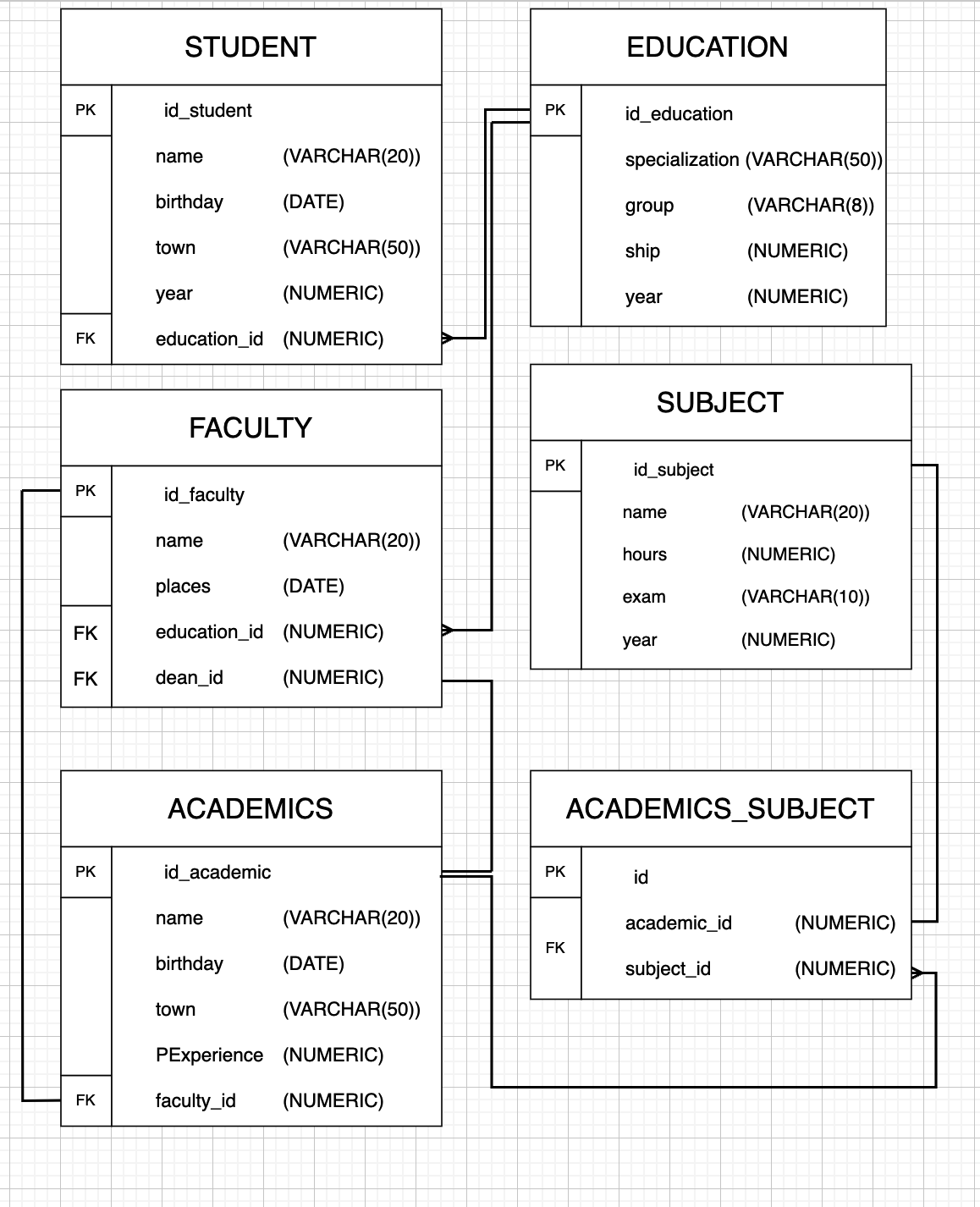
• Ознакомиться с механизмами контроля, транзакциями и триггерами.

Часть 1. Проектирование схемы базы данных



Первое практическое задание связано с проектированием схемы базы данных для работы приложения. 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, необходимых для детального решения поставленной задачи. Задачей является решить, для чего будет использоваться создаваемая база данных, и, исходя из этого, построить её концептуальную схему.

Данная база будет использоваться для электронного университета. Добавим еще таблицы: предметы и академики (преподаватели). Связь много ко много реализуем при помощи таблицы, показывающей какие предметы будут вести другие преподаватели. Связь один к одному реализуем для указания декана факультета.



**Ответы на контрольные вопросы:**

1. **Понятие первичного и вторичного ключей:**

Первичный ключ (Primary Key) представляет собой уникальный столбец или комбинацию столбцов в таблице, однозначно идентифицирующих каждую строку. Обеспечивает уникальность значений и используется для связей с другими таблицами.

Вторичный ключ (Secondary Key) — это столбец или группа столбцов, не являющихся первичным ключом, но используемых для установления связей между таблицами.

1. **Типы связей:**

Один к одному: каждая запись в одной таблице соответствует одной записи в другой.

Один ко многим: каждая запись в одной таблице соответствует нескольким записям в другой.

Многие ко многим: набор записей в одной таблице соответствует набору записей в другой через промежуточную таблицу.

1. **Нормализация:**

**Нормализация** — это процесс организации структуры базы данных для устранения избыточности данных и обеспечения целостности. Её цель — минимизировать дублирование данных и обеспечить эффективное хранение и обработку информации.

**Нормальные формы**: существует пять, но чаще всего применяются первые три.

Первая нормальная форма (1NF): устранение повторяющихся групп данных и обеспечение того, чтобы каждая ячейка содержала только одно значение.

Вторая нормальная форма (2NF): устранение частичной зависимости от первичного ключа, разделение данных на отдельные таблицы и установка связей.

Третья нормальная форма (3NF): устранение транзитивных зависимостей, гарантируя, что каждый столбец зависит только от первичного ключа.

1. **Суть первой, второй и третьей нормальных форм:**

**Первая нормальная форма (1NF):** в каждой ячейке таблицы должно содержаться только одно значение, и не должно быть повторяющихся групп данных.

**Вторая нормальная форма (2NF):** каждый столбец, не входящий в первичный ключ, должен полностью зависеть от всего первичного ключа, и не должно быть частичных зависимостей.

**Третья нормальная форма (3NF):** отсутствие транзитивных зависимостей, то есть, если A зависит от B, а B зависит от C, то A не должно зависеть от C напрямую.

Часть 2. Создание и заполнение таблиц

Создадим таблицы по схеме из части 1

1. CREATE TABLE EDUCATION(

2. id\_education SERIAL PRIMARY KEY,

3. specialization VARCHAR(50),

4. squad VARCHAR(8),

5. ship NUMERIC,

6. nyear NUMERIC

7. );

8.

1. CREATE TABLE STUDENT(

2. id\_student SERIAL PRIMARY KEY,

3. sname VARCHAR(20),

4. birsday DATE,

5. town VARCHAR(50),

6. nyear NUMERIC,

7. education\_id SERIAL,

8. FOREIGN KEY (education\_id) REFERENCES EDUCATION(id\_education)

9. );

10.

1. CREATE TABLE FACULTY(

2. id\_faculty SERIAL PRIMARY KEY,

3. fname VARCHAR(20),

4. places NUMERIC,

5. education\_id SERIAL,

6. dean\_id SERIAL,

7. FOREIGN KEY (education\_id) REFERENCES EDUCATION(id\_education),

8. UNIQUE (dean\_id)

9. );

10.

1. CREATE TABLE ACADEMIC(

2. id\_academic SERIAL PRIMARY KEY,

3. aname VARCHAR(20),

4. birsday DATE,

5. town VARCHAR(50),

6. PExperience NUMERIC,

7. faculty\_id SERIAL,

8. FOREIGN KEY (faculty\_id) REFERENCES FACULTY(id\_faculty)

9. );

10.

1. ALTER TABLE FACULTY

2. ADD FOREIGN KEY (dean\_id) REFERENCES ACADEMIC(id\_academic);

3.

1. CREATE TABLE SUBJECT(

2. id\_subject SERIAL PRIMARY KEY,

3. sname VARCHAR(20),

4. hours NUMERIC,

5. exam VARCHAR(10),

6. nyear NUMERIC

7. );

8.

1. CREATE TABLE ACADEMICS\_SUBJECT(

2. id SERIAL PRIMARY KEY,

3. academic\_id SERIAL,

4. subject\_id SERIAL,

5. FOREIGN KEY (academic\_id) REFERENCES ACADEMIC(id\_academic),

6. FOREIGN KEY (subject\_id) REFERENCES SUBJECT(id\_subject)

7. );

8.

Заполним таблицы, слишком большое количество значений сгенерируем создав запросы к бд через другой язык программирования.

1. INSERT INTO EDUCATION(specialization, squad, ship, nyear) VALUES

2. ('ivt', 'iu6-11b', 1000, 2023),

3. ('ivt', 'iu6-12b', 1000, 2023),

4. ('ivt', 'iu6-13b', 1000, 2023),

5. ('pi', 'iu6-14b', 1500, 2023),

6. ('pi', 'iu6-15b', 1500, 2023),

7. ('ivt', 'iu6-31b', 2700, 2022),

8. ('ivt', 'iu6-32b', 2700, 2022),

9. ('ivt', 'iu6-33b', 2700, 2022),

10. ('pi', 'iu6-34b', 3200, 2022),

11. ('pi', 'iu6-35b', 3200, 2022);

12.

1. INSERT INTO SUBJECT(sname, hours, exam, nyear) VALUES

2. ('Математика', 34, 'e', 2001),

3. ('Физика',72, 'e', 2002),

4. ('Программирование',72, 'a', 2004),

5. ('Теория вероятностей',34, 'a', 2010),

6. ('Электротехника',34, 'e', 2020),

7. ('Механика',72, 'e', 1999),

8. ('Инженерная графика',34, 'a', 2009),

9. ('Теория управления',34, 'a', 1998),

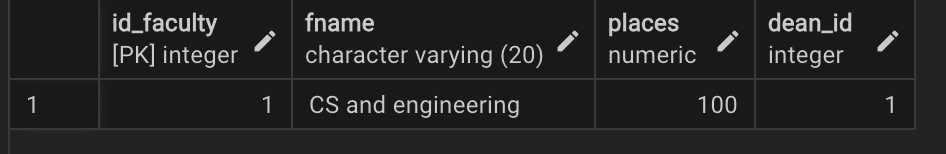
10. ('Метрология',17, 'd', 1890),

11. ('Экономика',17, 'e', 2016);

12.

Часть 3. Операторы манипулирования

Предположим, что мы должны уволить декана факультета ИУ6, а потом назначить новым деканам любого (первого найденного Андрея)



1. UPDATE FACULTY SET dean\_id = (

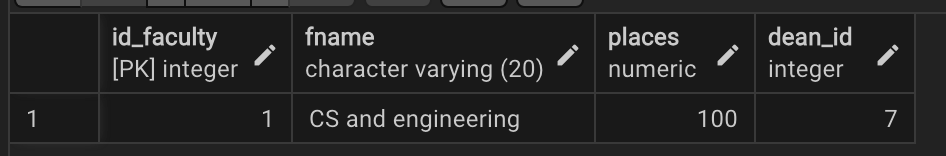
2. SELECT id\_academic FROM ACADEMIC

3. WHERE aname = 'Андрей' AND id\_academic <> 1

4. LIMIT 1

5. ) WHERE id\_faculty = 1;

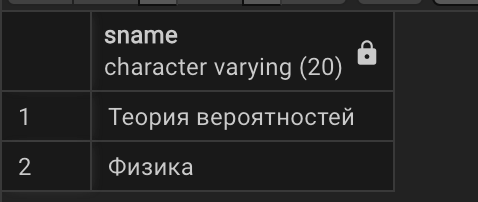
6.



Найдем все предметы которые вел этот декан и удалим записи об этом.

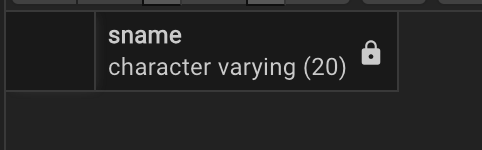
1. select sname from subject, academics\_subject where academic\_id = 1 and subject\_id = id\_subject;

2.

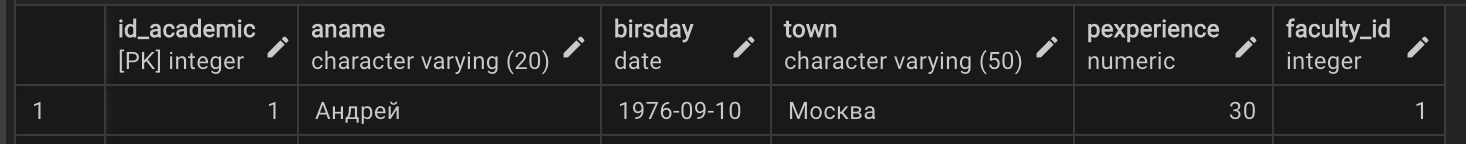


1. DELETE FROM ACADEMICS\_SUBJECT WHERE academic\_id = 1;

2.

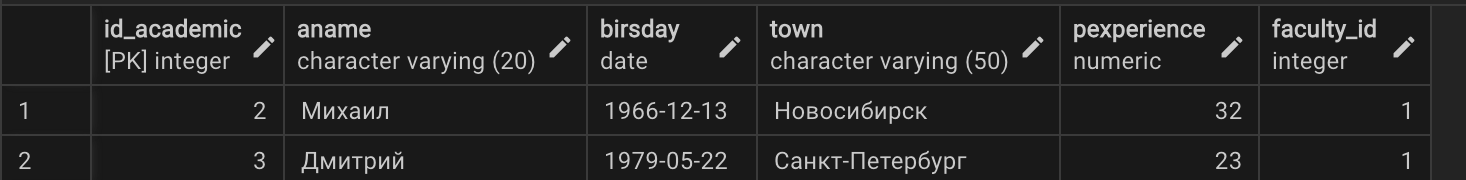


После этого удалим самого декана, вот и конец всего удаления.



1. DELETE FROM academic WHERE id\_academic = 1;

2.



**Ответы на контрольные вопросы:**

1. **Различие между типами CHAR и VARCHAR, VARCHAR и TEXT:**
   * CHAR - это фиксированная длина строки, где каждая строка занимает заданное количество символов. Если строка короче, она заполняется пробелами.
   * VARCHAR - это переменная длина строки, где каждая строка может иметь разную длину, но не заполняется пробелами.
   * VARCHAR и TEXT похожи, но VARCHAR имеет ограничение на максимальную длину, в то время как TEXT может хранить более длинные тексты без ограничений.
2. **Внешний ключ:** Внешний ключ (Foreign Key) - это столбец или набор столбцов в таблице, который ссылается на первичный ключ в другой таблице. Он используется для создания связей между таблицами и обеспечения ссылочной целостности данных.
3. **Способы поддержания ссылочной целостности:** Способы поддержания ссылочной целостности включают использование внешних ключей, каскадное обновление и каскадное удаление, ограничения CHECK, триггеры и другие механизмы, которые обеспечивают правильные связи и значения в базе данных.
4. **Уникальный ключ:** Уникальный ключ (Unique Key) - это ограничение, которое гарантирует уникальность значений в столбце или наборе столбцов. Он позволяет хранить только уникальные значения в указанных столбцах.
5. **SERIAL:** SERIAL - это псевдоним для целочисленных столбцов, которые автоматически увеличиваются при каждой вставке. Это удобно для создания автоинкрементных первичных ключей.
6. **Значения по умолчанию и неопределенные значения:** Значение по умолчанию (DEFAULT) - это значение, которое будет использовано, если не указано другое значение при вставке записи. Неопределенное значение (NULL) - представляет собой отсутствие значения. Это используется, когда значение неизвестно или не применимо.
7. **Хранение дат и времени:** Даты и времена могут храниться в столбцах с типами данных DATE (только дата), TIME (только время), TIMESTAMP (дата и время), и другими вариантами, включая интервалы и временные штампы с часовыми поясами.
8. **Числовые типы данных:** В PostgreSQL существует несколько числовых типов данных, включая INTEGER (целые числа), NUMERIC/DECIMAL (числа с фиксированной точностью), REAL (вещественные числа с одинарной точностью), DOUBLE PRECISION (вещественные числа с двойной точностью) и другие.

Часть 4. Контроль целостности данных.

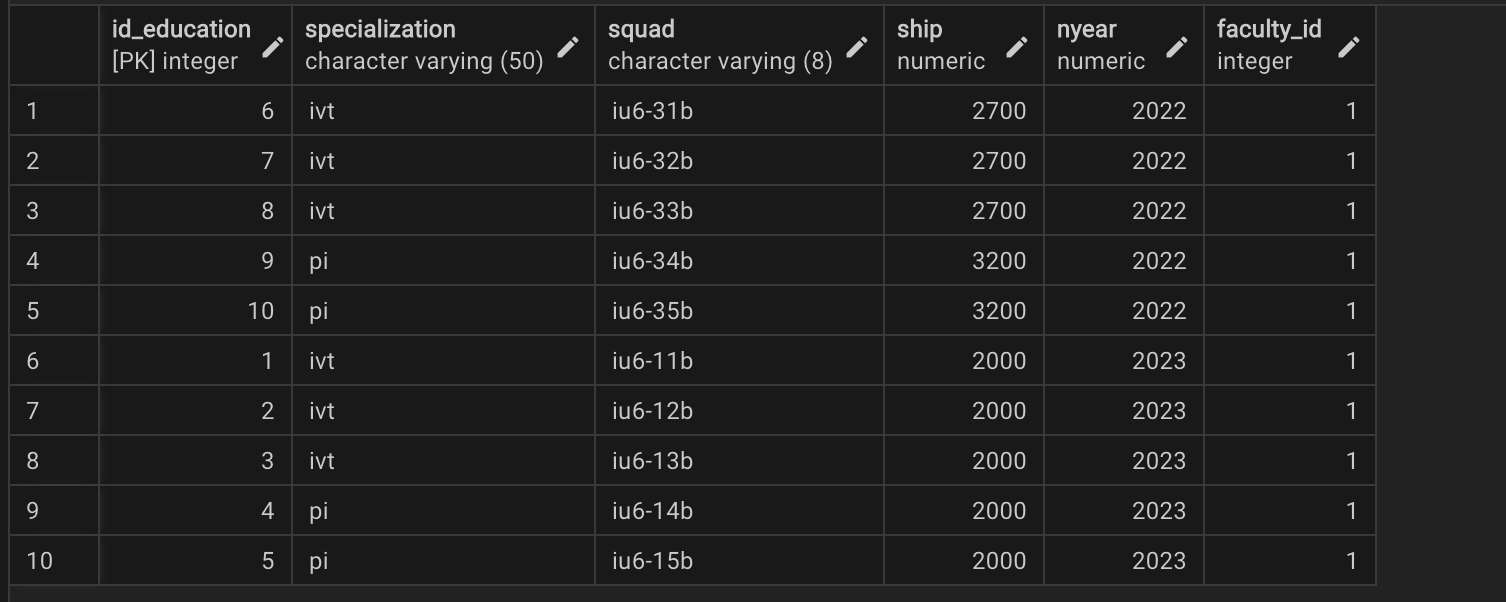
**READ COMMITTED.** Это самый низкий уровень изоляции. Согласно стандарту SQL, на этом уровне допускается чтение «грязных» (незафиксированных) данных. В PostgreSQL READ UNCOMMITTED рассматривается как READ COMMITTED.

1. BEGIN;

2. UPDATE EDUCATION SET ship = 2000 WHERE nyear = 2023;

3. SELECT \* FROM EDUCATION;

4.



1. BEGIN;

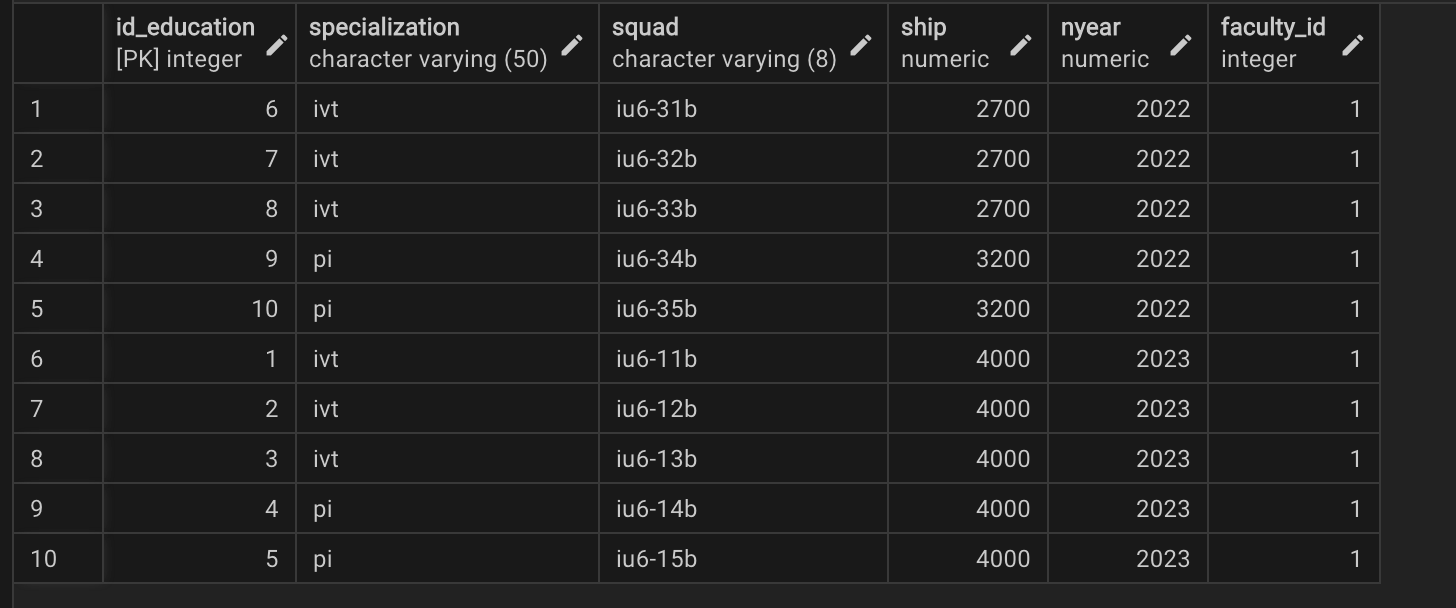
2. SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

3. SHOW transaction\_isolation;

4. UPDATE EDUCATION SET ship = ship + 2000 WHERE nyear = 2023;

5. SELECT \* FROM EDUCATION;

6.



1. ROLLBACK;

2.

**REPEATABLE READ.** Следующий уровень изоляции — REPEATABLE READ. Само его название говорит о том, что он не допускает наличия феномена неповторяющегося чтения данных. А в PostgreSQL на этом уровне не допускается и чтение фантомных строк.

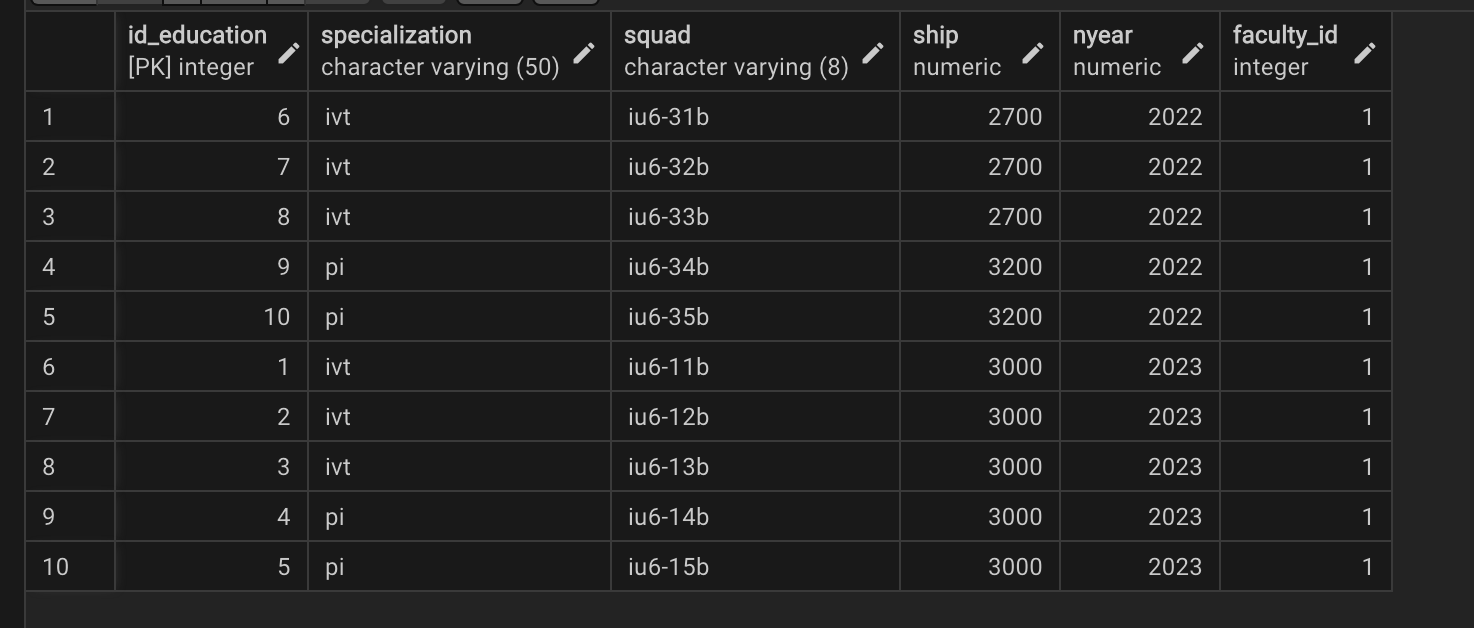
1. BEGIN;

2. SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

3. UPDATE EDUCATION SET ship = 3000 WHERE nyear = 2023;

4. SELECT \* FROM EDUCATION;

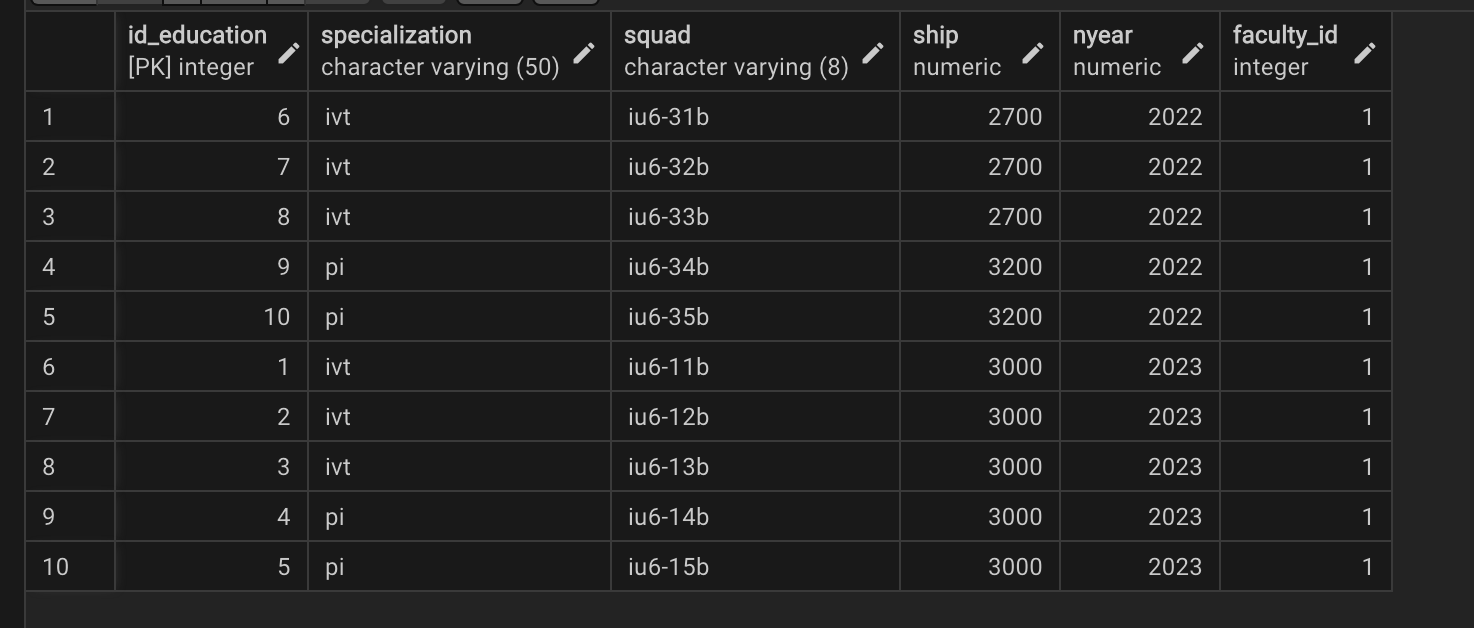
5.



1. BEGIN;

2. SELECT \* FROM EDUCATION;

3.



1. ROLLBACK;

2.

**SERIALIZABLE.** Высший уровень изоляции транзакций — SERIALIZABLE. Транзакции могут работать параллельно точно так же, как если бы они выполнялись последовательно одна за другой.

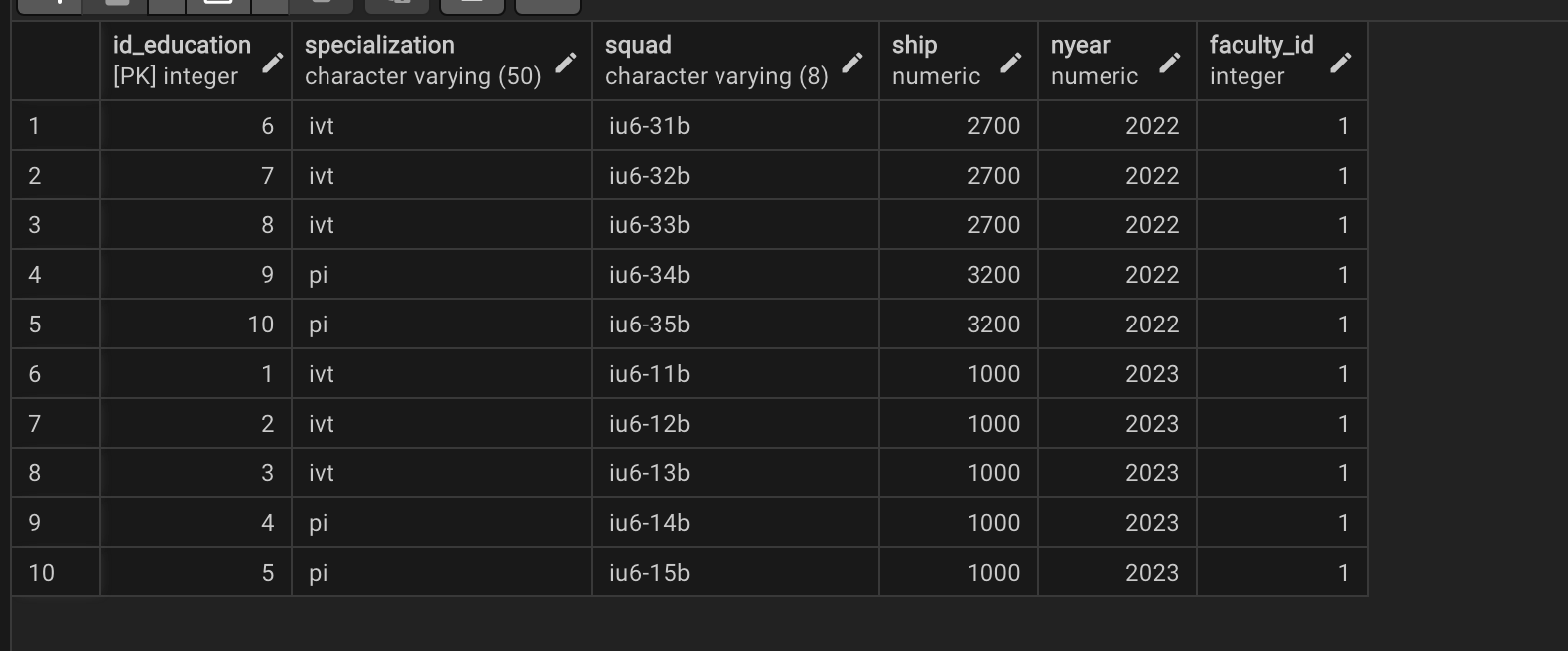
1. BEGIN;

2. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

3. UPDATE EDUCATION SET ship = 1000 WHERE nyear = 2023;

4. SELECT \* FROM EDUCATION;

5.

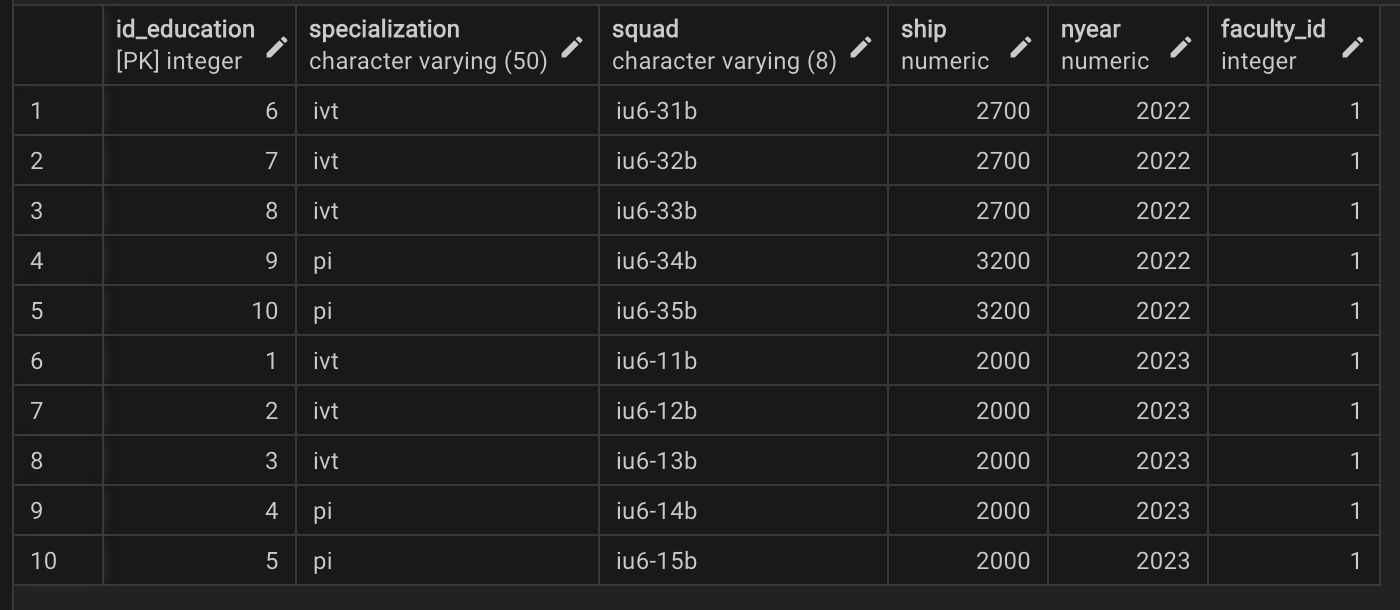


1. BEGIN;

2. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

3. SELECT \* FROM EDUCATION;

4.



1. ROLLBACK;

2.

Создадим триггер для увольнения декана из части 3 и представим ситуацию, что когда приходит новый академик он обязательно недоволен начальством, поэтому после его прихода будем обязательно менять декана на нового первого попавшегося Андрея.

1. CREATE OR REPLACE FUNCTION resign\_dean\_iu6() RETURNS TRIGGER AS $$

2. DECLARE

3. deanid numeric;

4. BEGIN

5. SELECT dean\_id INTO deanid FROM FACULTY;

6.

7. UPDATE FACULTY SET dean\_id = (

8. SELECT id\_academic FROM ACADEMIC

9. WHERE aname = 'Андрей' AND id\_academic <> deanid

10. LIMIT 1

11. ) WHERE id\_faculty = 1;

12.

13. DELETE FROM ACADEMICS\_SUBJECT WHERE academic\_id = deanid;

14. DELETE FROM academic WHERE id\_academic = deanid;

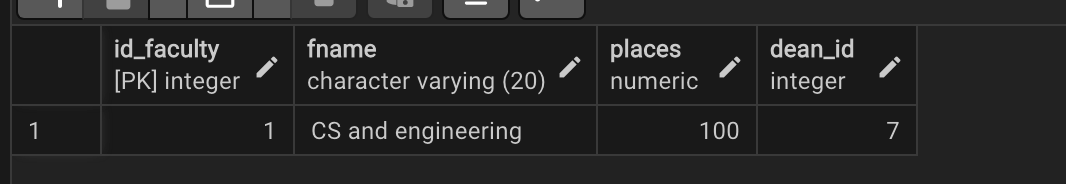
15.

16. RETURN NEW;

17. END;

18. $$ LANGUAGE plpgsql;

19.



1. CREATE TRIGGER change\_dean

2. AFTER INSERT

3. ON academic

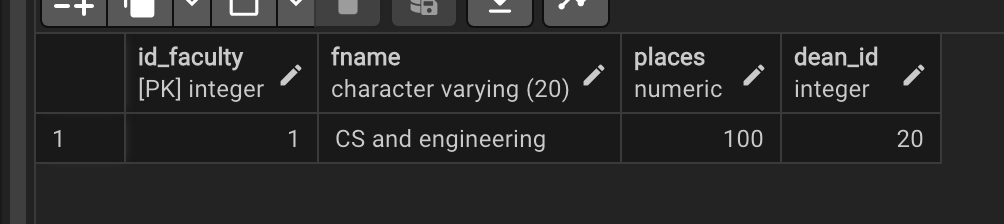
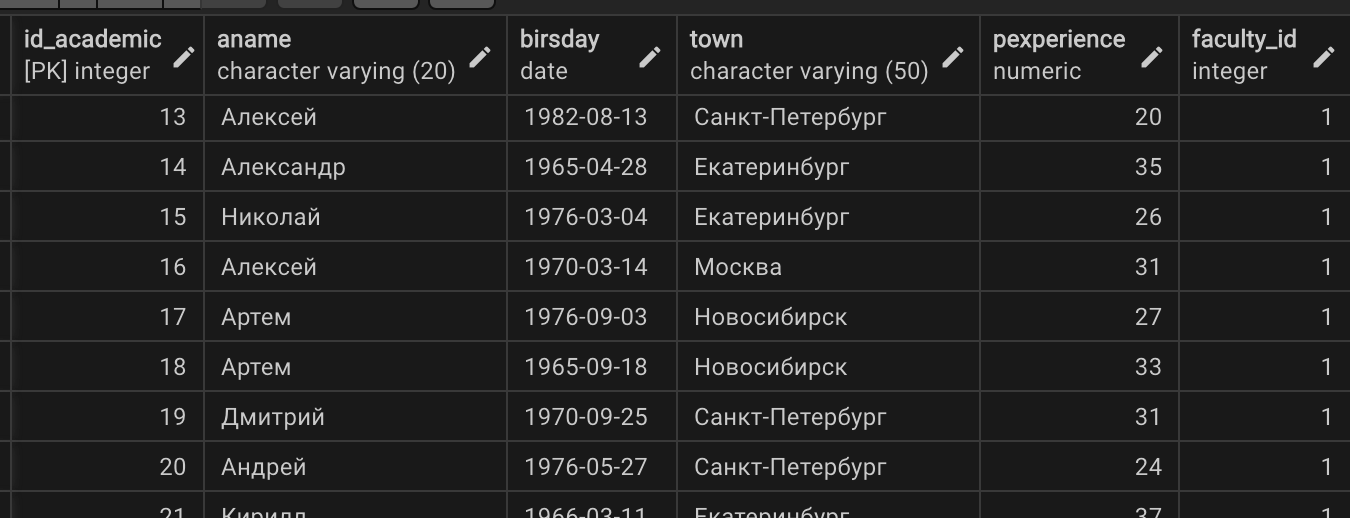
4. EXECUTE FUNCTION resign\_dean\_iu6();

5.

1. INSERT INTO ACADEMIC(aname, birsday, town, pexperience, faculty\_id)

2. VALUES ('Михаил', '10-09-1976', 'Москва', 30, 1);

3.

Как видим после прихода Михаила руководство поменялось на Андрея под номером 20.

**Ответы на контрольные вопросы:**

1. **Аномалии доступа к БД:** Аномалии доступа к базе данных представляют собой нежелательные и неконтролируемые события, возникающие при одновременном доступе к данным из разных транзакций. Эти аномалии включают три основных типа: аномалии чтения, аномалии вставки и аномалии обновления.
2. **Аномалии на разных уровнях изолированности:** Уровни изолированности включают:
   * Уровень READ UNCOMMITTED: возможны все три типа аномалий.
   * Уровень READ COMMITTED: возможны аномалии чтения и аномалии вставки.
   * Уровень REPEATABLE READ: возможны аномалии чтения.
   * Уровень SERIALIZABLE: транзакции максимально изолированы, а аномалии не возникают.
3. **Свойства транзакций:** ACID - это акроним, описывающий четыре основных свойства транзакций:
   * **Атомарность:** Транзакция либо выполняется целиком, либо не выполняется вовсе.
   * **Согласованность:** Транзакция переводит базу данных из одного согласованного состояния в другое согласованное состояние.
   * **Изолированность:** Транзакции выполняются изолированно друг от друга.
   * **Устойчивость:** Результаты выполненных транзакций остаются навсегда.
4. **Управление транзакциями:** Управление транзакциями включает команды BEGIN, COMMIT и ROLLBACK. Команда BEGIN начинает транзакцию, COMMIT фиксирует её результаты, а ROLLBACK отменяет транзакцию.
5. **Тупики и борьба с ними:** Тупик (Deadlock) представляет собой ситуацию, когда две или более транзакции блокируют друг друга, ожидая освобождения ресурсов. Борьба с тупиками включает использование таймаутов, жертво-приношений (откат изменений), и системы управления транзакциями могут автоматически обнаруживать и разрешать тупики.
6. **Уровень изолированности и тупики:** Тупики могут возникнуть на уровне изолированности SERIALIZABLE, когда множество транзакций блокируют друг друга. Однако на более низких уровнях изолированности, таких как READ COMMITTED или REPEATABLE READ, вероятность возникновения тупиков меньше.
7. **Изолированность транзакций в СУБД:** СУБД обеспечивают изолированность транзакций с использованием блокировок и уровней изоляции. Это гарантирует, что одна транзакция не увидит изменений, внесенных другой транзакцией, пока она не будет фиксирована.
8. **Борьба с проблемой фантомов:** Проблема фантомов возникает, когда транзакция видит набор данных, который ей кажется одним, но другая транзакция внесла изменения. Для борьбы с проблемой фантомов используются блокировки на чтение и механизмы изоляции данных.
9. **Журнал транзакций:** Журнал транзакций (Transaction Log) фиксирует все изменения, внесенные транзакциями в базу данных. Это позволяет восстановить базу данных после сбоев и обеспечивает устойчивость транзакций.
10. **Обеспечение постоянства хранения в СУБД:** Для обеспечения постоянства хранения СУБД использует журналы транзакций. Все изменения данных записываются в журнал перед фактической записью в базу данных, обеспечивая восстановление после сбоев.
11. **Принцип работы триггера:** Триггер - это хранимая процедура, выполняемая автоматически при возникновении определенного события в базе данных. Триггер может быть привязан к таблице и срабатывать, например, при вставке, обновлении или удалении записей в этой таблице.
12. **Типы триггеров:** Существуют два основных типа триггеров:
    * Триггеры "перед": срабатывают перед выполнением операции (например, перед вставкой новой записи).
    * Триггеры "после": срабатывают после выполнения операции (например, после вставки новой записи).
13. **Когда может срабатывать триггер:** Триггер может срабатывать при определенных событиях, таких как вставка, обновление, удаление или изменение данных в конкретной таблице. Точные условия срабатывания триггера определяются при его создании.
14. **Порядок срабатывания триггеров:** Порядок срабатывания триггеров зависит от СУБД. В некоторых СУБД триггеры выполняются в порядке их создания, в других - в порядке, который они определяются при создании или с помощью приоритетов.
15. **Изменение порядка срабатывания триггеров:** В большинстве СУБД можно управлять порядком срабатывания триггеров с помощью явных указаний при их создании или путем настройки приоритетов триггеров.
16. **Сработает ли триггер, если оператор не затрагивает таблицу:** Триггеры срабатывают только в ответ на события, связанные с определенной таблицей. Если оператор, выполненный пользователем, не затрагивает ни одну строку таблицы, триггер, связанный с этой таблицей, не сработает.

**Вывод:** В результате обучения SQL я освоил навыки создания таблиц, установки связей между ними, и использования триггеров для автоматического обновления данных. Также я овладел методами управления уровнями изоляции транзакций и обработки NULL значений в запросах. Эти навыки позволяют мне более эффективно взаимодействовать с базами данных и обеспечивают более гибкий и надежный анализ данных.