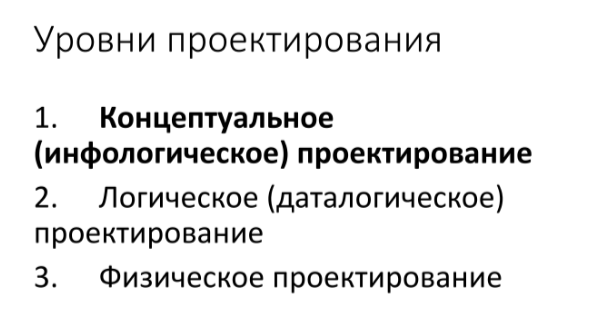
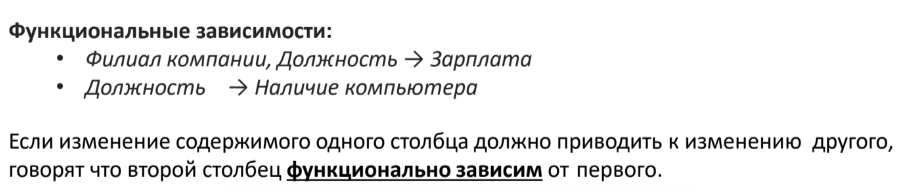
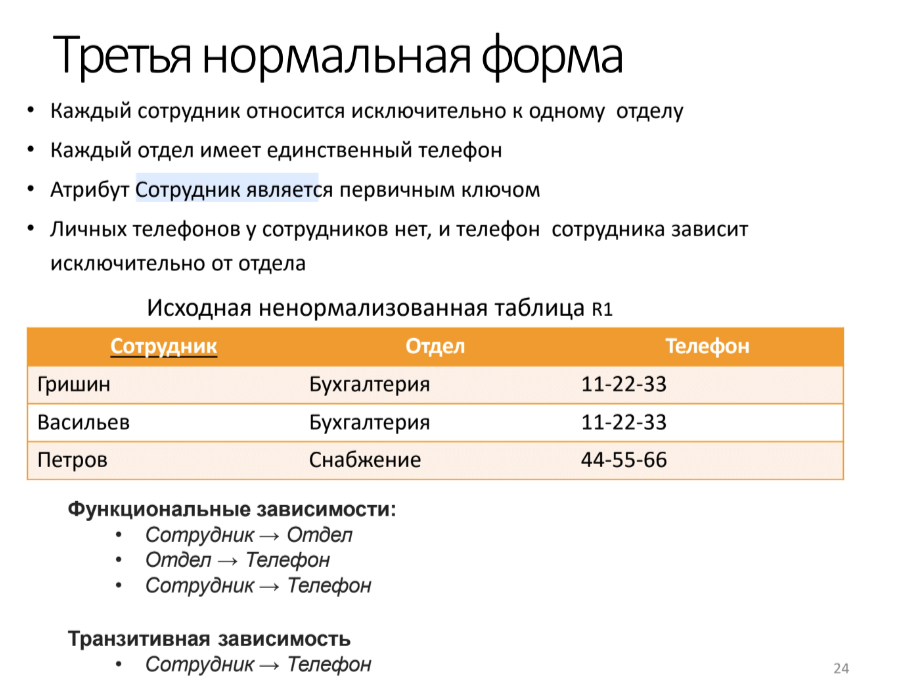
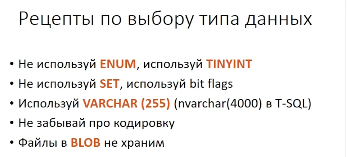
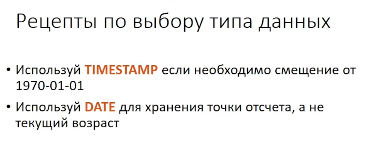
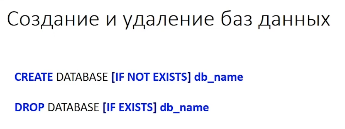
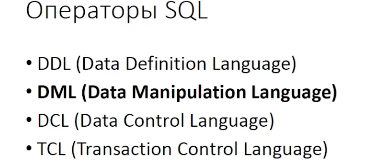
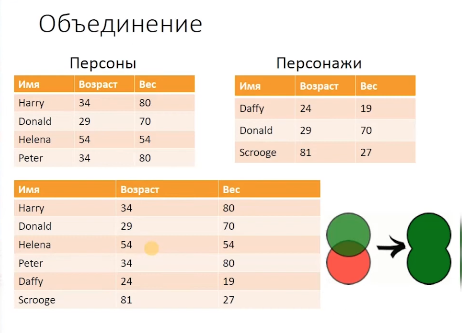
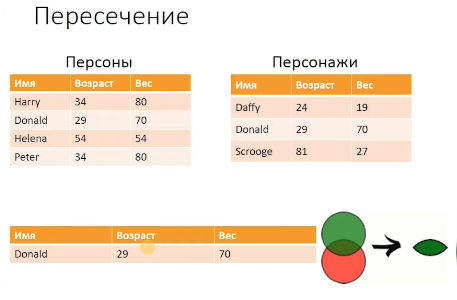
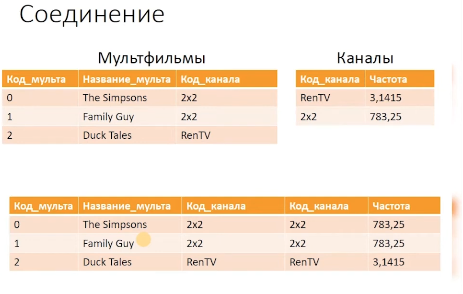
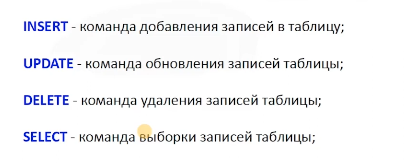
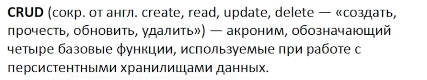
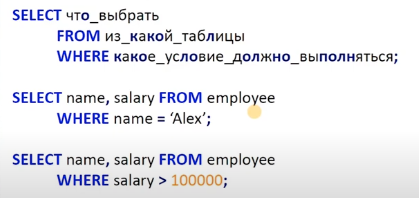
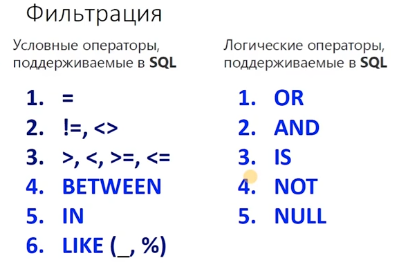
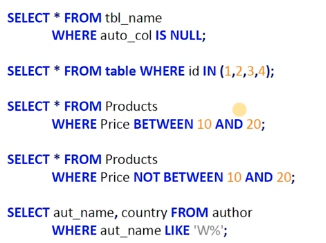
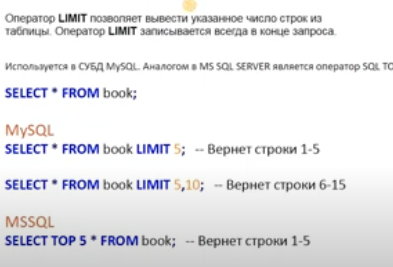
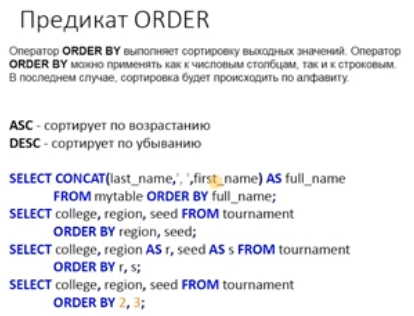
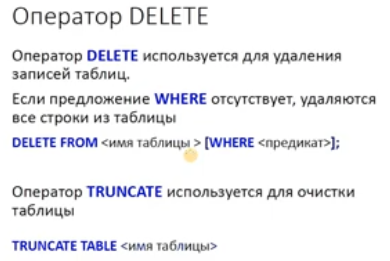
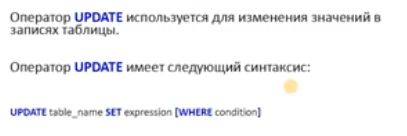
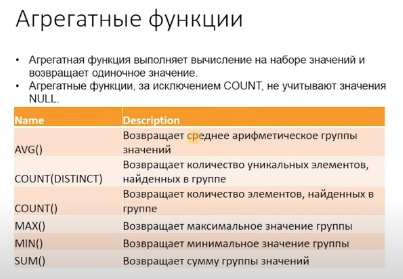
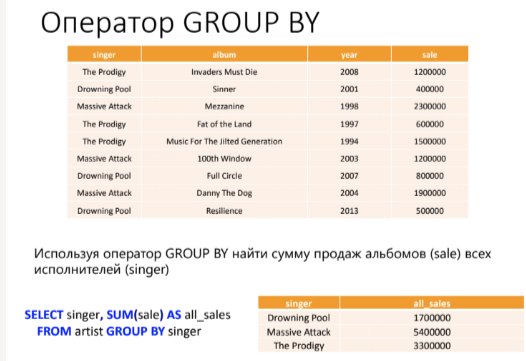
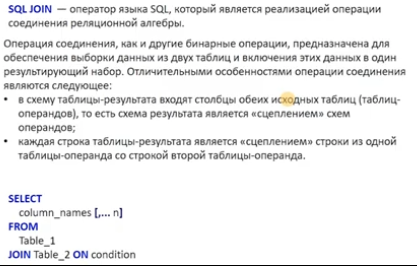
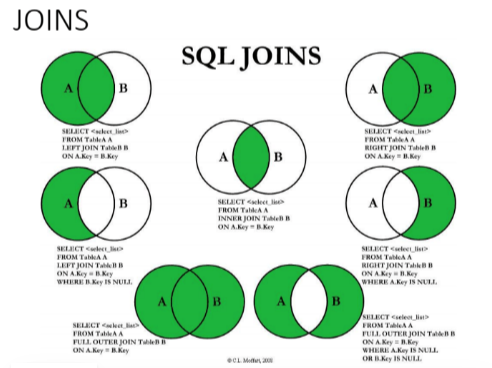
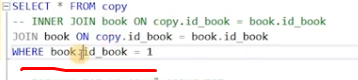
**1. [1]Чем отличает БД от СУБД? [2]Какие задачи решает СУБД? [3]Раскройте детали каждой задачи.  
  
[1]**БД – база данных – совокупность структурированных данных.СУБД – система управления базами данных (программное обеспечение для работы с данными).  
  
**[2][3]**  
СУБД умеет выполнять запросы над данными: создание, удаление, обновление.  
Какие задачи решает СУБД:  
- Надёжное хранилище больших данных (снимается с программистов вопрос как делать бэкап, как сохранять).  
- Эффективный доступ (СУБД умеет искать нужную информацию быстро).  
- Многопользовательский доступ (много людей могут изменять в одно время).  
- Разграничение прав доступа (преподаватель и ученик).  
- Удобная работа с данными (SQL стандартный язык).  
- Обеспечение доступа по сети.  
  
**2. Проектирование БД и уровни проектирования БД.**Основные задачи проектирования:  
- Сокращение избыточности и дублирования данных.  
  
**Проектирование базы данных** - создание эффективной структуры данных, обеспечивающее хранение требуемой информации.  
  
  
  
**-> Концептуальное (инфологическое) проектирование**—построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции.  
  
Процедуры концептуального проектирования:  
- Определение сущностей и их аргументов.  
- Определение связей между сущностями и их документирование.  
  
  
  
Особенности концептуального проектирования.  
- Обследование предметной области, изучение ее информационной структуры.  
- Моделирование и интеграция всех представлений.  
  
Связи: один к одному, многие ко многим, один ко многим.  
  
**-> Логическое проектирование** – Преобразование концептуальной модели на основе выбранной модели данных в логическую модель.  
Это уже представление программиста. Учитывается модель, но не специфика СУБД.  
  
Процедуры логического проектирования:  
- Выбор модели данных.  
- Определение набора таблиц.  
- Нормализация таблиц (приведение таблиц к правильному хранению инфы).  
  
Есть ключ отношения – атрибут для идентификации объекта в БД.  
Первичный ключ – ключ, используемый СУБД для идентификации объекта.  
Суррогатный ключ – ключ, генерируемый СУБД (id).  
Составной ключ – ключ, использующий несколько атрибутов.  
  
**-> Физическое проектирование (3 уровень)** – ER-диаграмма с учётом СУБД.  
  
**3. Реляционная алгебра. Реализация операций реляционной алгебры в языке SQL.**Реляционная модель данных – мыслит отношениями.  
Она разработана Э. Коддом.  
  
**4. Реляционная БД преимущества и недостатки.**  
Реляционная БД (в виде таблицы: строки и столбцы).  
  
Преимущества.  
- В основе модели – лежит мощный математический аппарат теории множеств и математической логики.  
- Контроль целостности данных.  
- Гибкость.  
  
Недостатки.  
- Большое количество таблиц в реальных БД.  
- Относительно медленный доступ к данным.   
- Некоторые предметные области плохо представляются в форме отношений.  
  
  
 **5. Нормальные формы. Для чего нужны нормальные формы?**Устранение аномалий.  
Процесс нормализации – приведение к нормальной форме.  
  
Нормальная форма – правило, которой должна соответствовать таблица, чтобы не было аномалий.  
  
Всего их 7.  
Каждая нормальная форма предполагает, что таблица приведена в предыдущей нормальной форме.  
  
Первая нормальная форма – в одной ячейке должно находится одно значение.  
Одно значение для каждого атрибута.  
  
Пример.  
  
  
  
**6. Нормальные формы. Приведение к 2 НФ. Пример.**  
Вторая нормальная форма – говорит, что таблица в уже в первой начальной форме и она должна быть функционально полная зависимость от первичного ключа.  
  
Атрибут должен функционально полно зависеть от первичного ключа.  
Первичный ключ тут – филиал компании и должность.  
  
Теперь проверяем, что атрибут зависит от первичного ключа.  
  
Нам дана не вторая нормальная форма, потому что наличие компьютера по этой таблице зависит не только от должности, но и от филиала компании.  
  
  
  
  
 **7. Нормальные формы. Приведение к 3 НФ. Пример.**  
  
Третья нормальная форма – предполагает, что таблица уже во второй нормальной форме и **при этом любой её неключевой атрибут зависит только от первичного ключа.**  
Данная таблица не в 3 нормальной форме, потому что есть транзитивная зависимость телефона от сотрудника.  
  
То есть у нас телефон не зависит напрямую от сотрудника (телефон зависит от отдела, который зависит от сотрудника).  
  
  
  
**8. Нормальные формы. Приведение к 3 НФ Бойса—Кодда. Пример.**  
  
Третья нормальная форма Бойса-Кодда.  
Часть составного первичного ключа не должна зависеть от атрибута.  
  
  
**9. Нормальные формы. Транзитивная зависимость. Функциональная зависимость. Пример.**  
  
Транзитивная зависимость – непрямая зависимость (косвенная).  
Функциональная зависимость – прямая зависимость атрибута от первичного ключа.  
  
  
  
**10.ER-диаграмма. Для чего нужна? Составные компоненты.**Entity-relationship Диаграмма – диаграмма, которая отображает отношения набора сущностей, хранящиеся в базе данных.  
  
В основе ER-диаграмм лежит принцип «рисунок нагляднее текста».  
  
Основные компоненты: сущности, атрибуты сущности, ключ сущности, связи.  
  
  
  
**11.SQL. Типы данных. Пример применения типов данных.  
  
Целые числа:**  
bigint  
int  
tinyint  
money  
smallmoney  
  
**Дробные числа:**  
float  
real.  
  
К числам можно добавлять модификатор unsigned.  
  
**Символьные типы данных:**  
CHAR и VARCHAR(лучше использовать его). Чем отличаются? CHAR всегда будет занимать данное ему место. VARCHAR может сэкономить.  
  
BLOB и TEXT, ENUM, SET – не использовать.  
  
  
  
Временные типы данных.  
Datetime.  
Date.  
Year.  
Timestamp.  
  
  
  
Есть возможность задавать дефолтные значения.  
  
NULL-значения. По умолчанию не задано. **12.SQL. DDL. Пример запроса создания таблицы. Автоинкремент vs GUID.**  
Используется для генерации уникального идентификатора для новых записей.  
  
Что использовать Автоинкремент или GUID?  
  
**Автоинкремент.**  
- Занимает меньший объём.  
- Минус в том, что быстро можно перебрать базу данных.  
  
  
Пример использования идентификатора.  
  
  
**GUID.**  
  
  
  
**DDL – data definition language.**  
Язык определения данных – позволяет создавать таблицы, удалять, изменять данные.  
  
  
  
**НО! Чаще всего мы создаём БД через графический интерфейс.**  
  
Конструкции IF NOT EXISTS, IF EXISTS – указывать обязательно.  
  
Когда мы создаем БД, мы должны указать кодировку.  
  
  
  
  
**13. SQL. DML.SELECT**SQL разбит на подмножества операторов.  
DML – data manipulation language.  
  
  
  
  
  
В результате операции над таблицей мы получаем таблицу.  
  
[1] Выборка (покажи тех у кого возраст больше 30).  
[2] Проекция (верни столбцы возраст, вес).  
[3] Объединение (2 таблицы в 1).



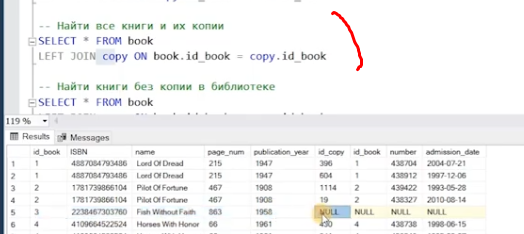
[4] Пересечение (То что есть в 2 таблицах).

  
  
[5] Разность (Все, что только в первой таблице).  
  
  
[6] Произведение (каждая запись первого отношения соотносится с записью второго отношения).  
  
  
  
[7] Соединение.  
  
  
  
Команды.  
  
  
  
  
  
- CRUD операции.   
  
  
-> Оператор SELECT.  
SELECT column\_list  
FROM table\_name  
[WHERE condition GROUP BY expression HAVING condition ORDER BY exprsn]  
  
-> Предикат WHERE.  
  
  
-> Фильтрация.  
  
  
Between – в диапазоне.  
In – Атрибут находится в массиве.  
Like – позволяет искать подстроку в строке.  
  
  
  
-> Предикат LIMIT/TOP.  
  
  
-> Предикат ORDER.  
  
  
  
  
  
  
  
  
**14. SQL. DML. INSERT, DELETE, UPDATE.**[1] Оператор INSERT.  
  
Внести записи без указания столбцов:  
INSERT INTO <название таблицы> VALUES (<выражение1>, <выражение2>..)  
  
Внести записи по некоторым столбцам:  
INSERT INTO <название таблицы> (column\_name, …) VALUES (name).  
  
Внести записи с оператором SELECT.  
INSERT INTO <название таблицы> SELECT column\_name, … FROM <названи таблицы>  
  
[2] Оператор DELETE.  
  
  
  
[3] Оператор UPDATE.  
  
  
  
  
[4 extra] Оператор GROUP BY.  
  
  
Group By как правило сам по себе не используется, но находит применение совместной с функциями агрегации.  
  
  
  
  
Оператор GROUP BY позволяет «схлопнуть» несколько записей в одну, чтобы потом применить функцию агрегации.  
  


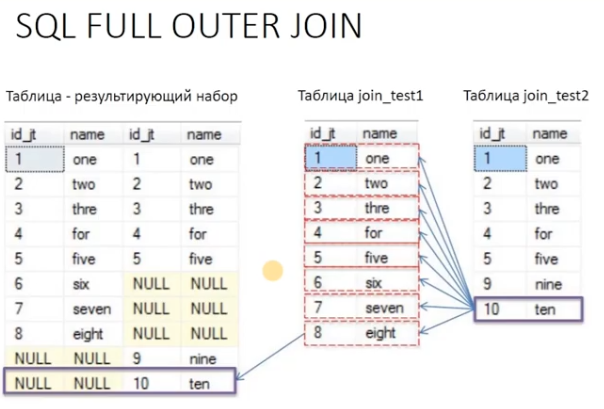
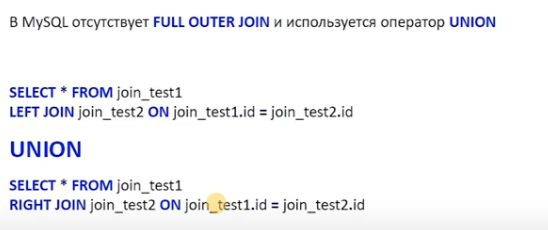
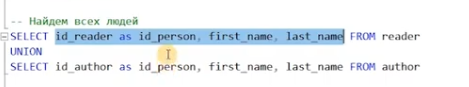
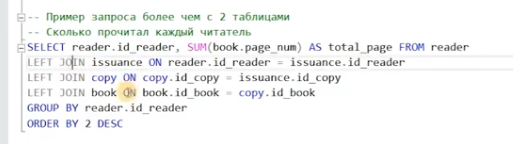
[5] Оператор Distinct. Позволяет находить уникальные записи в таблице.  
  
  
[6] Оператор JOIN.  
Зачем нам это?  
Если бы мы хотели получить информацию по таблице выдаче книги информацию о том, кто взял(имя,фамилия).  
Склейка таблиц.  
Бинарный оператор.  
Соединяет таблицы, получаем результирующий набор, к нему присоединяем следующую.  
  
В результате получаем таблицу, состоящую из полей первой и второй.  
  
  
Операторов JOIN несколько.  
  


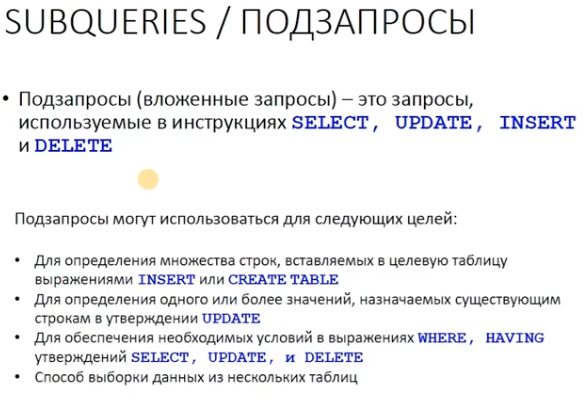
[1] Inner Join.  
  
  
То есть находим пересечение множеств, тех данных которые есть в первой таблице и в другой.  
  
JOIN – по умолчанию INNER JOIN.  
  
При WHERE по условию указывать в какой таблице.  
В примере ниже id\_book есть в 2 таблицах, нужно указать явно.  
  
  
  
INNER JOIN в обратном порядке совершенно такой же.

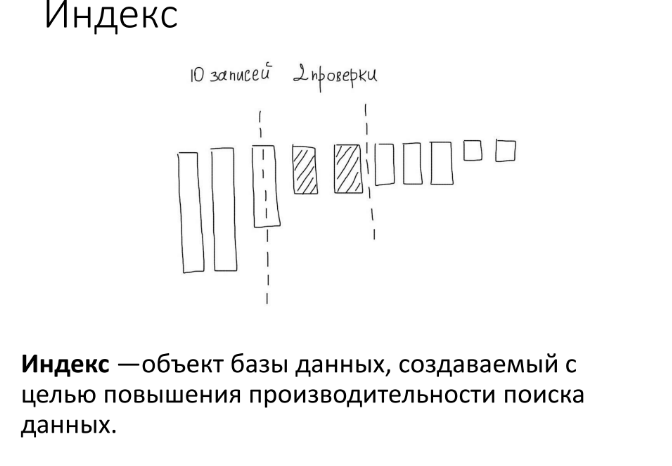
[2] Left Join.

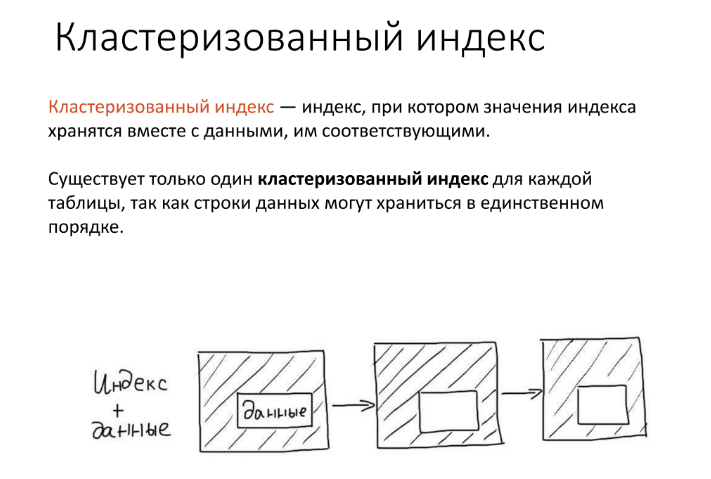
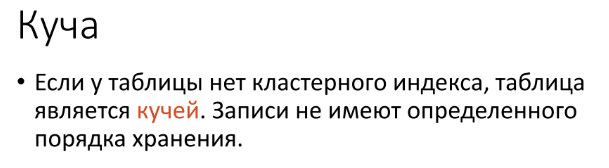
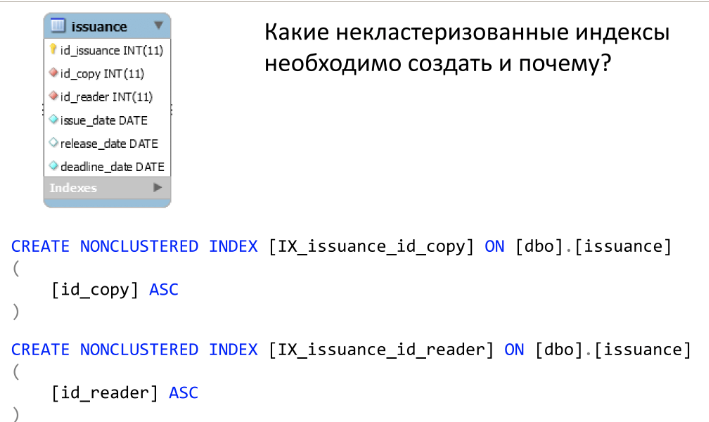
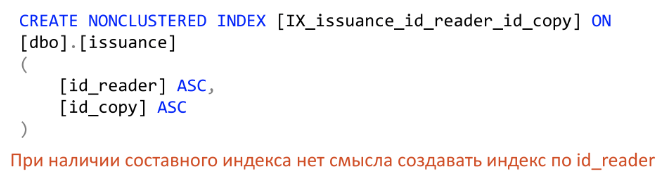
  
  


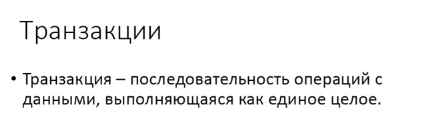
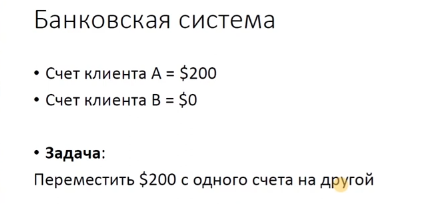
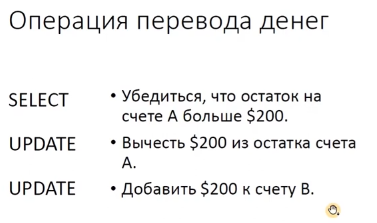
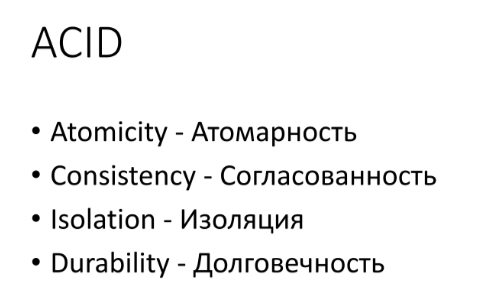
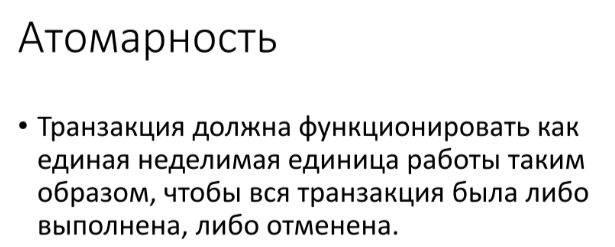
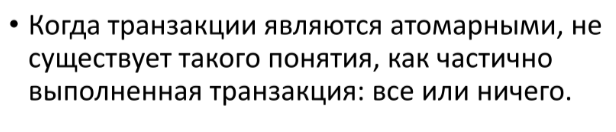
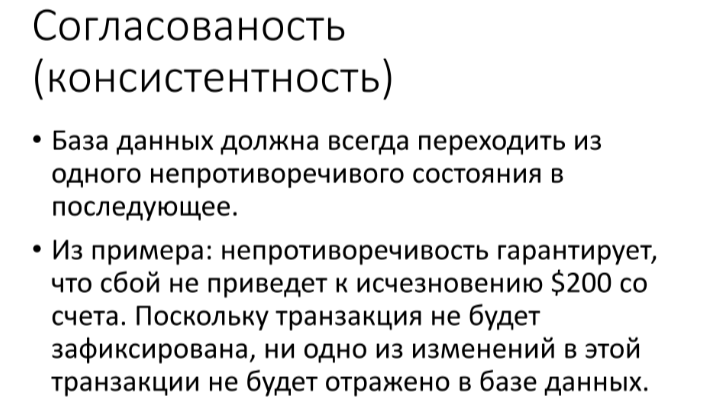
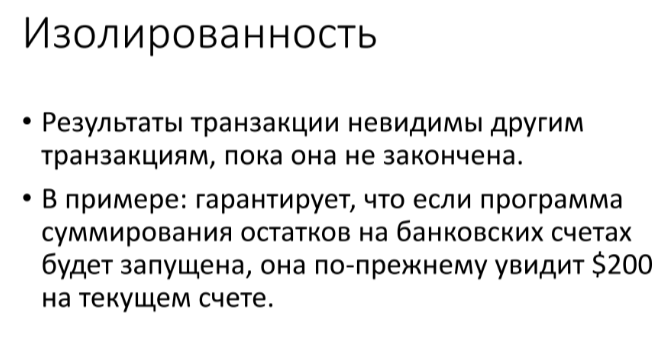
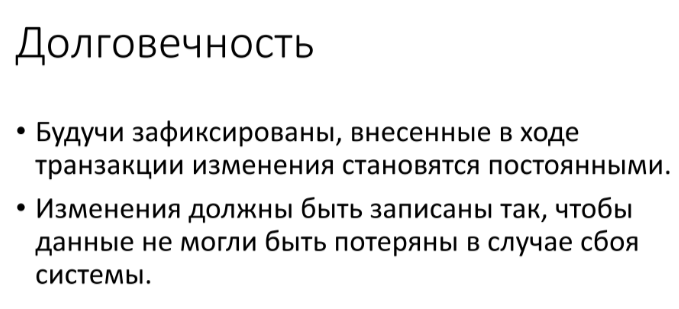
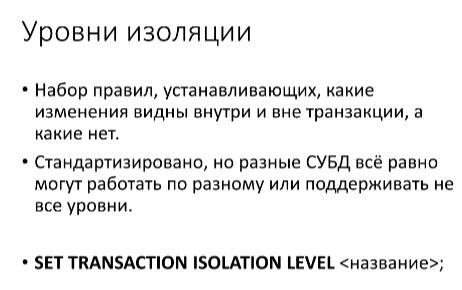
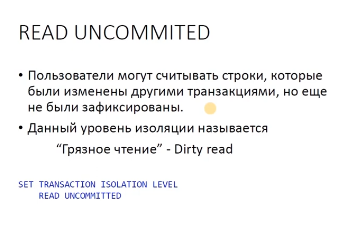
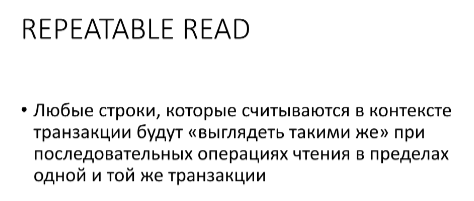
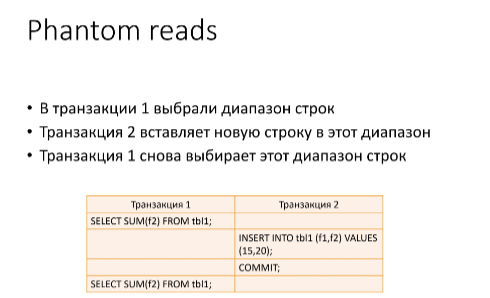
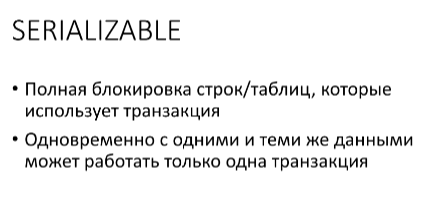
[3] Right join.  
  
  
На практике right join используется реже.  
Лучше использовать left join.  
  
[4] Full Join Outer Join.  
  

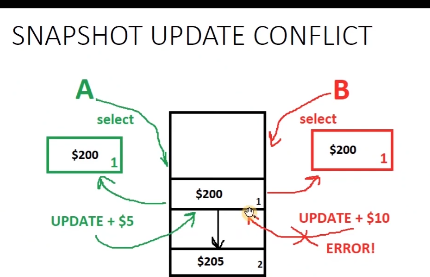

  
  
На практике используется реже.  
  
  
UNION – позволяет соединять 2 таблицы.  
  
  
  
  
Полный пример.  
  


**[7] Subqueries / подзапросы.**  
  
  
  
4 варианта подзапросов.  
  
[7.1]   


[7.2] Подзапрос использовать как таблицу.  
  
(Лаба 5)  
  
ПРИМЕРЫ:  
  
  
  
  
ы  
  
  
  
  
  
  
  
  
15. Индексы.  


  
  
  
  
  
  
  
  
  
  
  
  
  
16. Ограничения целостности

**17. Транзакции. Уровни изоляции.**  
  
  
Для чего может это потребоваться?  
Пример с банковской схемой.  
  
  
Выполнить эту операцию одним запросом не получится.  
  
  
  
Почему нам важно, чтобы эти 3 операции были как единое целое?  
Потому что при ошибке операция будет незакончена и это недопустимо.  
  
Start Transaction, Commit.  
  
Rollback – откат.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Пессимистическая работает на блокировке данных.  
Оптимистическая предполагает что мы одни работаем с данными.  
  
  
  
Проблема uncommitted – мы можем читать неподтвержденные данные.  
  
  
  
Проблема – возможно невоспроизводимое чтение.  
Один запрос – разные данные.  
  
  
  
Проблема – фантомное чтение.  
  
  


Оптимистическая стратегия.  
Располагает предположением, что мы одни.  
  
Snapshot.  
Если клиент обратился к БД, то создается копия.   
  
Read commited snapshot.  
Каждый запрос новая копия.  
  
SNAPSHOT UPDATE.  
  
  
18. Хранимые процедуры и триггеры.  
19. Масштабирование БД. Репликация.  
20. Масштабирование БД. Шардинг.  
21. Масштабирование БД. Партиционирование.  
22. Оптимизация запросов.  
23. План выполнения запроса.  
24. OLAP и OLTP. Предназначения и отличия.  
25. NoSQL. Теорема CAP.  
26. NoSQL. Виды БД и их предназначения.  
27. ORM. Назначение. Принципы работы.  
28. SQL инъекция. Причины. Вред. Как защититься.