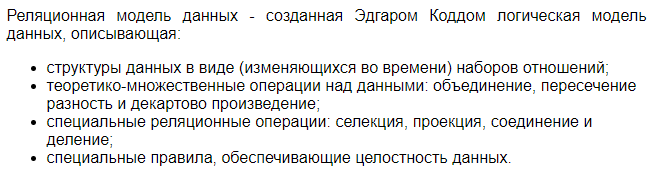
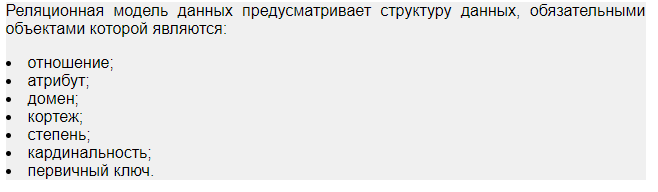
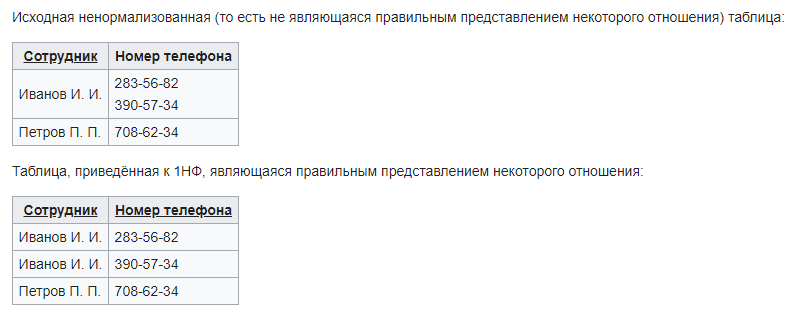
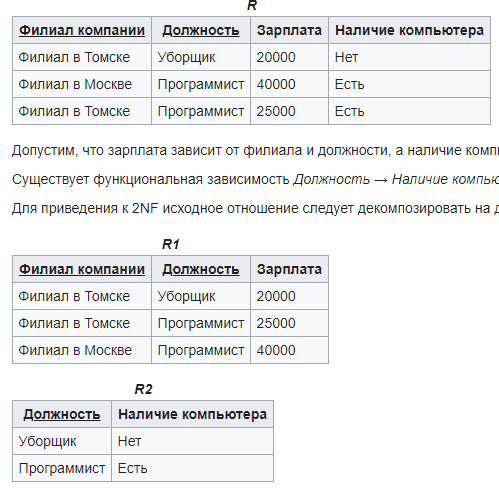
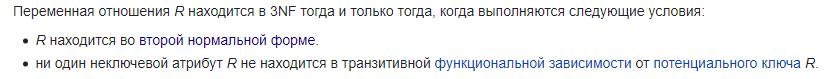
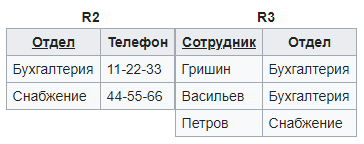
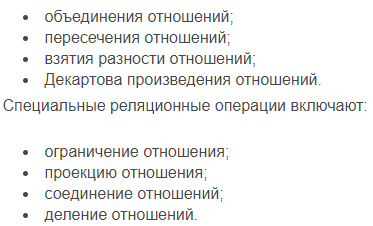
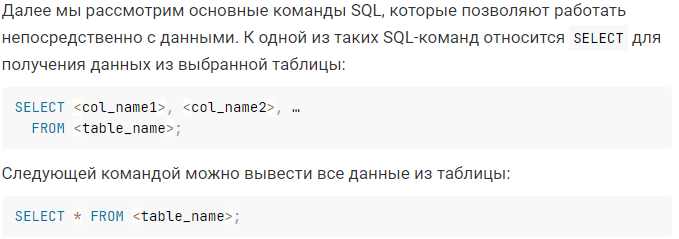
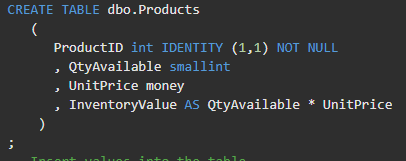
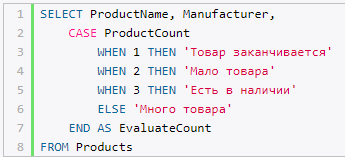
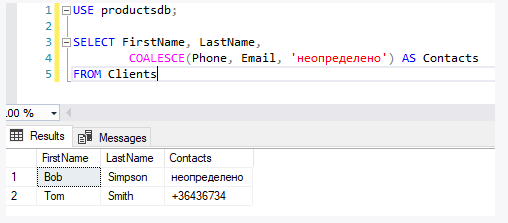
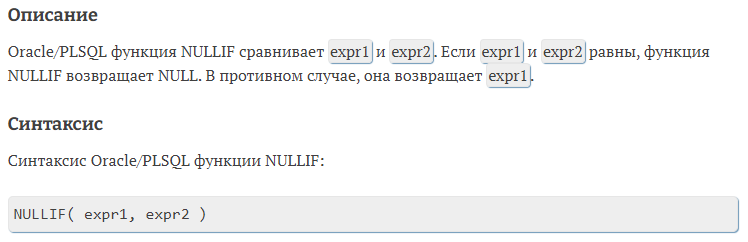
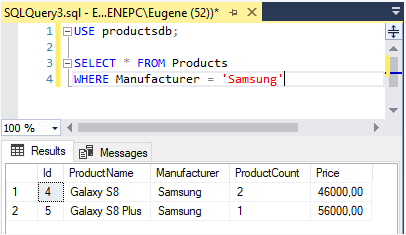
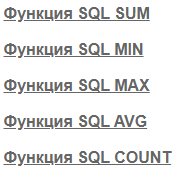
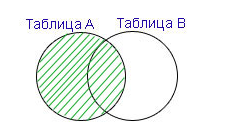
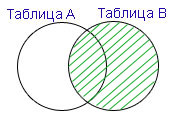
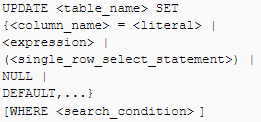
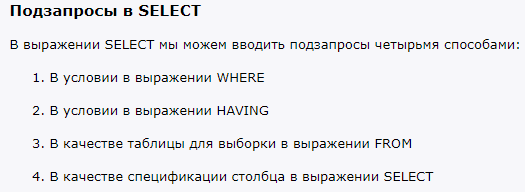
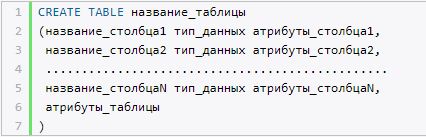
Теоретические вопросы по дисциплине «Базы данных»

1. Предмет и задачи дисциплины. Модель предметной области.
2. Бинарные отношения. Отображения. Кардинальные числа.
3. Ограничения целостности в моделях данных. Ограничения целостности для атрибутов.
4. Ограничения целостности для типа сущности. Ограничения целостности для связей между типами сущностей.
5. Инфологическая модель. Требования к инфологической модели.
6. Модель “Объект‑свойство‑отношение”.
7. Построение реляционной модели на основе инфологической модели.
8. Реляционная модель, как система отношений. Основные понятия и определения.
9. Нормальные формы (НФ) отношений в реляционных моделях данных (РМД). Нормализация отношений в РМД на примере 1НФ.
10. Понятие полной функциональной зависимости от ключа. Определение 2НФ. Нормализация до 2НФ
11. Понятие транзитивной зависимости от ключа. Определение 3НФ. Нормализация до 3НФ
12. Операции манипулирования отношениями в реляционных моделях данных.
13. История и структура SQL. Типы данных SQL.
14. Оператор выбора SELECT. Назначение и порядок выполнения предложений оператора.
15. Оператор выбора SELECT. Вычисляемые поля. Выражения CASE, COALESCE и NULLIF.
16. Оператор выбора SELECT. Предложение Where. Сортировка данных.
17. Агрегатные функции в операторе SELECT. Предложение Having.
18. Внутренние и внешние объединения в операторе выбора.
19. Множественные операции в операторе выбора.
20. Оператор добавления новых данных в SQL. Правила использования.
21. Операторы изменения и удаления данных в SQL. Правила использования.
22. Подзапросы в SQL. Основные правила использования.
23. Операторы SQL по созданию и удалению таблиц и представлений.
24. Инфологическая модель (ИЛМ) и моделирование. Требования к ИЛМ. Состав ИЛМ.
25. Построение модели “Объект‑Свойство‑Отношение”. Сложные объекты.
26. Построение РМД на основе ИЛМ.
27. Классы‑модели Qt для работы с базами данных. Назначения и правила использования.
28. Представления Qt. Их использование для работы с базами данных.
29. Создание простейшего Qt‑приложения по ведению базы данных.
30. Генерация выходных документов в Qt.
31. Целью освоения дисциплины «Базы данных» является изучение теоретических основ проектирования баз данных, компонентов банков данных, характеристик современных СУБД, современных технологий организации БД, приобретение навыков работы в среде конкретных СУБД.   
    Задачами изучения дисциплины «Базы данных» является: ­  
    сформировать системное базовое представление, первичные знания, умения и навыки студентов по основам построения систем управления базами данных как научной и прикладной дисциплины; ­  
    дать представление о роли и месте баз данных в автоматизированных системах, о назначении и основных характеристиках различных систем управления базами данных, их функциональных возможностях.  
    Понятие ***предметной области*** *базы данных* является одним из базовых понятий информатики и не имеет точного определения.  
    Совокупность реалий (объектов) внешнего мира - объектов, о которых можно задавать вопросы, - образует объектное *ядро предметной области*, которое имеет онтологический статус.
32. Бинарные отношения – отношения между двумя сущностями  
    Закон (правило) f, посредством которого каждому a∈A сопоставляется единственный b∈B, называют отображением. Обычно это записывают так: b=f(a).  
    Отображение состоит из трех объектов: множества A(откуда), множества B(куда) и правила f(как).  
    Мощность множества, кардинальное число́ множества — характеристика [множеств](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) (в том числе [бесконечных](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BA%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE)), обобщающая понятие количества (числа) [элементов](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE#%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0) конечного множества.
33. **Ограничения целостности** являются специальными средствами баз данных, которые предназначены для предупреждения возможности попадания в базу недопустимым данным.  
    Средства обеспечения доменной целостности – предназначены для недопущения ввода в поля базы данных недопустимых значений. К примеру, наименование товара должно состоять из букв, а номер телефона – из цифр. Обеспечивается такая целостность в базах данных зачастую установлением условий на значение, ключами, хранимыми процедурами, триггерами, запретом пустых значений.
34. **Сущностная целостность**, главной задачей которой является недопущение двукратного попадания данных об одной сущности в базу. Такую целостность обеспечивает установление ограничения уникальности и первичный ключ. Она не допускает, чтобы любой атрибут, который принадлежит первичному ключу, мог принимать неопределенное значение.  
    **Ссылочную целостность** обеспечивает система внешних и первичных ключей. К примеру, с помощью данных средств можно дать гарантию, что в базе не будет заказов, которые оформлены на покупателей, отсутствующих в базе данных.
35. **Инфологическое моделирование** выполняется с целью обеспечения естественных для человека способов представления и сбора информации, которая будет храниться в создаваемой БД.  
    **Инфологическая модель** данных строится в соответствии с естественным языком, который не возможно использовать в чистом виде в виду сложности обработки текстов с помощью компьютера и неоднозначности естественного языка.  
    **Инфологическая модель** – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.  
    Цель инфологического моделирования – обеспечить оптимальные способы сбора и представления информации, хранимой в базе данных.  
    Целью данной модели является обеспечение наиболее естественных для пользователя способов представления и сбора информации.
36. При отражении в информационной системе каждый объект представляется своим идентификатором, который отличает один объект класса от другого, а каждый класс объектов представляется именем этого класса. Так, для объектов класса “ИЗУЧАЕМЫЕ ПРЕДМЕТЫ” идентификатором каждого объекта будет “НАЗВАНИЕ ПРЕДМЕТА”. Идентификатор должен быть уникальным.  
    Каждый объект обладает определенным набором свойств. Для объектов одного класса набор этих свойств одинаков, а их значения, естественно, могут различаться. Например, для объектов класса “СТУДЕНТ” таким набором свойств, описывающим объекты класса, может быть “ГОД РОЖДЕНИЯ”, “ПОЛ” и др.  
    При описании предметной области надо изобразить каждый из существующих классов объектов и набор свойств, фиксируемый для объектов данного класса.  
    При описании предметной области надо отразить связи между объектом и характеризующими его свойствами. Это изображается просто в виде линии, соединяющей обозначение объекта и его свойств.  
    Связь между объектом и его свойством может быть различной. Объект может обладать только одним значением какого-то свойства. Например, каждый человек может иметь только одну дату рождения. Назовем такие свойства единичными. Для других свойств возможно существование одновременно нескольких значений у одного объекта. Пусть, например, при описании “СОТРУДНИКА” фиксируется в качестве его свойства “ИНОСТРАННЫЙ ЯЗЫК”, которым он владеет. Так как сотрудник может знать несколько иностранных языков, то такое свойство будем называть множественным. При изображении связи между объектом и его свойствами для единичных свойств будем использовать одинарную стрелку, а для множественных свойств — двойную.  
    Кроме связи между объектом и его свойствами, в инфологической модели фиксируются связи между объектами разных классов. Различают связи типа «один к одному» (1 : 1), «один ко многим» (1 : М), «многие к многим» (М: М). Иногда эти типы связей называются степенью связи.
37. Работаем уже с таблицами
38.   
      
    Атрибут — это поименованный столбец отношения.  
    Домен — это набор допустимых значений для одного или нескольких атрибутов.  
    Кортеж — это строка отношения.  
    Степень определяется количеством атрибутов, которое оно содержит  
    Кардинальность — это количество кортежей, которое содержит отношение.  
    Первичный ключ — это уникальный идентификатор для таблицы.
39. Нормальная форма — свойство [отношения](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)) в [реляционной модели данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных.  
    Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации.  
    [https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F\_%D1%84%D0%BE%D1%80%D0%BC%D0%B0#:~:text=%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D1%84%D0%BE%D1%80%D0%BC%D0%B0%20%E2%80%94%20%D1%81%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE%20%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B2,%D1%80%D0%B5%D0%B7%D1%83%D0%BB%D1%8C%D1%82%D0%B0%D1%82%D0%B0%D0%BC%20%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8%20%D0%B8%D0%BB%D0%B8%20%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85.&text=%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B9%20%D1%86%D0%B5%D0%BB%D1%8C%D1%8E%20%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%83%D0%BC%D0%B5%D0%BD%D1%8C%D1%88%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9%20%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2%D0%BE%D1%80%D0%B5%D1%87%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D0%B8%20%D1%85%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%BE%D0%B9%20%D0%B2%20%D0%B1%D0%B0%D0%B7%D0%B5%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8.](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0%23:~:text=%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D1%84%D0%BE%D1%80%D0%BC%D0%B0%20%E2%80%94%20%D1%81%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE%20%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B2,%D1%80%D0%B5%D0%B7%D1%83%D0%BB%D1%8C%D1%82%D0%B0%D1%82%D0%B0%D0%BC%20%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8%20%D0%B8%D0%BB%D0%B8%20%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85.&text=%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B9%20%D1%86%D0%B5%D0%BB%D1%8C%D1%8E%20%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%83%D0%BC%D0%B5%D0%BD%D1%8C%D1%88%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9%20%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2%D0%BE%D1%80%D0%B5%D1%87%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D0%B8%20%D1%85%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%BE%D0%B9%20%D0%B2%20%D0%B1%D0%B0%D0%B7%D0%B5%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8.)  
    Переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении этой переменной каждый [кортеж](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) отношения содержит только одно значение для каждого из атрибутов  
    
40. Полная функциональная зависимость — это состояние [нормализации базы данных,](https://gadgetshelp.com/prilozheniia/osnovy-normalizatsii-bazy-dannykh/) которое соответствует стандарту нормализации [второй нормальной формы (2NF)](https://gadgetshelp.com/prilozheniia/normalizuite-svoiu-bazu-dannykh-i-perevedite-ee-vo-vtoruiu-normalnuiu-formu-2nf/) . Вкратце, это означает, что он соответствует требованиям первой нормальной формы (1NF), и все не ключевые атрибуты полностью функционально зависят от первичного ключа.  
    Переменная отношения находится во **второй нормальной форме** тогда и только тогда, когда она находится в [первой нормальной форме](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) и каждый не ключевой атрибут неприводимо [зависит](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) от (каждого) её [потенциального ключа](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87).  
    **Неприводимость** означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную [функциональную зависимость](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Для неприводимой функциональной зависимости часто используется эквивалентное понятие «полная функциональная зависимость».  
    
41.   
    *Неключевой атрибут* отношения *R* — это атрибут, который не принадлежит ни одному из потенциальных ключей *R*.  
    *Функциональная зависимость* множества атрибутов **Z** от множества атрибутов **X** (записывается **X** → **Z**, произносится «икс определяет зет») является *транзитивной*, если существует такое множество атрибутов **Y**, что **X** → **Y** и **Y** → **Z**. При этом ни одно из множеств **X**, **Y** и **Z** не является подмножеством другого, то есть функциональные зависимости **X** → **Z**, **X** → **Y** и **Y** → **Z** не являются *тривиальными*, а также отсутствует функциональная зависимость **Y** → **X**.  
    
42. Основная идея реляционной алгебры состоит в том, что коль скоро отношения являются множествами, то средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для баз данных.  
      
    <https://www.osp.ru/dbms/1995/03/13031427>
43. <https://metanit.com/sql/sqlserver/3.3.php>
44.   
    Оператор SELECT осуществляет выборку из базы данных и имеет наиболее сложную структуру среди всех операторов языка SQL.
45. Вычисляемый столбец представляет собой виртуальный столбец, физически не хранящийся в таблице, если для него не установлен признак PERSISTED. В выражении вычисляемого столбца для вычисления значения могут использоваться данные из других столбцов.  
    Вычисляемый столбец нельзя использовать ни в качестве определения ограничения DEFAULT или FOREIGN KEY, ни вместе с определением ограничения NOT NULL. Однако если вычисляемый столбец определен детерминированным выражением и тип данных результата допускается для индексных столбцов, то вычисляемый столбец может быть использован как ключевой столбец в индексе или как часть ограничений PRIMARY KEY или UNIQUE. Например, если таблица содержит столбцы a и b со значениями целого типа, то вычисляемый столбец a + b может быть индексирован, но вычисляемый столбец a+DATEPART(dd, GETDATE()) не может быть индексирован, так как значение может меняться при каждом следующем вычислении.  
    Вычисляемый столбец не может быть целевым столбцом инструкций INSERT или UPDATE.  
      
    **Функция CASE** проверяет значение некоторого выражение, и в зависимости от результата проверки может возвращать тот или иной результат.  
      
    **Функция COALESCE** принимает список значений и возвращает первое из них, которое не равно NULL  
      
    
46. Для фильтрации в команде SELECT применяется оператор WHERE. После этого оператора ставится условие, которому должна соответствовать строка.  
    
47.   
    <https://function-x.ru/sql_sum_min_max_avg_count.html>  
    Оператор GROUP BY определяет, как строки будут группироваться.  
    Стоит учитывать, что любой столбец, который используется в выражении SELECT (не считая столбцов, которые хранят результат агрегатных функций), должны быть указаны после оператора GROUP BY.  
    Оператор HAVING определяет, какие группы будут включены в выходной результат, то есть выполняет фильтрацию групп.  
    Применение HAVING во многом аналогично применению WHERE. Только есть WHERE применяется к фильтрации строк, то HAVING используется для фильтрации групп.
48. Запрос с оператором INNER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON.  
    Запрос с оператором LEFT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из первой по порядку (левой) таблицы, даже если они не соответствуют условию. У записей левой таблицы, которые не соответствуют условию, значение столбца из правой таблицы будет NULL (неопределённым).  
      
    Запрос с оператором RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из второй по порядку (правой) таблицы, даже если они не соответствуют условию. У записей правой таблицы, которые не соответствуют условию, значение столбца из левой таблицы будет NULL (неопределённым).  
    
49. Йцу
50. Как видно из названия, оператор **INSERT** используется для вставки (добавления) строк в таблицу базы данных. Добавление можно осуществить несколькими способами.  
    Итак, чтобы добавить новую строку в таблицу, нам необходимо указать название таблицы, перечислить названия колонок и указать значение для каждой колонки с помощью конструкции **INSERT INTO***название\_таблицы (поле1, поле2 ... )***VALUES***(значение1, значение2 ...)*.  
    Однако оператор **INSERT INTO**позволяет автоматизировать этот процесс, если мы хотим вставлять данные из другой таблицы. Для этого в SQL существует такая кострукция как **INSERT INTO ... SELECT ...**. Данная конструкция позволяет одновременно выбирать данные из одной таблицы, и вставить их в другую. Предположим мы имеем еще одну таблицу **Sellers\_EU**с перечнем продавцов нашего товара в Европе и нам нужно их добавить в общую таблицу **Sellers**. Структура этих таблиц одинакова (то же количество колонок и те же их названия), однако другие данные. Для этого мы можем прописать следующий запрос:  
    **INSERT INTO Sellers (ID, Address, City, Seller\_name, Country) SELECT ID, Address, City, Seller\_name, Country FROM Sellers\_EU**
51. В операторе UPDATE нужно указать изменяемые строки и их новые значения. Новые данные могут быть константами или выражениями, или могут быть получены из других таблиц. В большинстве СУБД оператор INSERT имеет следующий синтаксис:  
      
    Ключевое слово UPDATE предшествует названию таблицы или обновляемого представления. Как и в случае с другими операторами модификации данных, в каждом операторе UPDATE можно изменять данные только одной таблицы.  
    Для удаления строк используется оператор DELETE, который позволяет удалить одну или несколько строк таблицы. Так же, как и в случае с другими операторами модификации данных, при удалении строк можно пользоваться информацией из других таблиц.  
    Оператор DELETE имеет следующий синтаксис:  
    
52. 
53. Для создания таблиц применяется команда CREATE TABLE. С этой командой можно использовать ряд операторов, которые определяют столбцы таблицы и их атрибуты. И кроме того, можно использовать ряд операторов, которые определяют свойства таблицы в целом. Одна база данных может содержать до 2 миллиардов таблиц.  
      
    Для создания представления используется команда CREATE VIEW, которая имеет следующую форму:  
    
54. Йцу
55. Йцу
56. Йцу
57. Класс **QSqlTableModel** позволяет отображать и редактировать данные в табличной форме.  
    Класс **QSqlQueryModel** наиболее подходит для отображения таблиц базы, содержимое которых не подлежит изменению.  
    Пользовательский метод UpdateDB() предназначен для обновления данных в классе **QSqlQueryModel**. Это вызвано наличием в нём буфера. Поэтому, при редактировании, например, таблицы “Pokazatel” содержимое модели Pokazatel\_SUB будет “отставать” от реальности. Сихронизировать буфер с реальным содержанием БД можно при помощи медода setQuery.  
    Выходом из данного положения является использование модели **QSqlRealtionalTableModel**. Данная модель позволяет выполнять подстановку “на лету”. Например, при работе с таблицей “Rabota” вместо поля cod\_pokaz можно отображать поле pokaz\_name, используя связь между этими полями в таблице “Pokazatel”.