### **1) Укажите основные достоинства реляционного подхода к моделированию данных**

* Простота логической структуры (все данные логически структурированы внутри отношений).
* Небольшой набор абстракций (которые позволяют просто моделировать большинство ПрО и допускают точные формальные определения, оставаясь интуитивно понятными).
* Простой, но мощный математический аппарат (опирающийся в основном на теорию множеств и математическую логику)
* Возможность спецификационного манипулирования без необходимости знания физической схемы.

### **2) Перечислите и дайте определения основных структурных понятий реляционной модели**

**Тип данных** - понятие, аналогичное понятию в языках программирования (т.е. множество значений и операций над ними)

**Домен** - множество допустимых значений данного типа.

**Атрибут** - именованный домен, представляющий семантически значимые объекты.

**Кортеж** – набор именованных значений заданных типов (это множество пар «имя атрибута, значение», которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения).

**Отношение** - множество кортежей, соответствующих одной схеме отношения.

Дополнительно:

**Схема отношения** – это именованное множество пар «имя атрибута, имя домена».

**Схема БД** – это набор именованных схем отношений.

Мощность множества - степень схемы отношения

### **3) Какие свойства характерны для отношений реляционной модели**

* Отсутствие кортежей-дубликатов
* Отсутствие упорядоченности кортежей и атрибутов
* Атомарность значений атрибутов

(Атомарный атрибут - атрибут, теряющий смысл при любом разбиении на части)

### **4) Сформулируйте простейшие правила перехода от ER схемы Чена к реляционной схеме БД**

1. Множества сущностей становятся отношениями, однозначные атрибуты множеств сущностей - атрибутами отношений, ключи сущностей - возможными ключами отношения.
2. Связи 1:1 и 1:М без атрибутов представляются дублированием первичного ключа отношения, у которого стоит единица в диаграмме, в другое.
3. Связи М:N без атрибутов становятся отношениями, куда дублируются первичные ключи отношений, участвующих в связи.
4. Множества связей с атрибутами становятся самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей. Однозначные атрибуты множества связей становятся атрибутами этого отношения.
5. Связи с атрибутами и связи степени больше двух становятся отношениями, куда дублируются первичные ключи отношений, участвующих в связи. Однозначные атрибуты множества связей становятся атрибутами отношения.
6. Многозначные атрибуты множеств сущностей или связей становятся отдельными отношениями, куда дублируется первичный ключ отношения, построенного для этого множества сущностей или связей.

### **5) Что такое представление и для чего они предназначены Какой командой SQL они создаются**

**Представление** - виртуальное отношение, кортежи (или экстенсионал) которого не хранятся на диске, а воспроизводятся на основании базовых отношений, реально существующих в БД.

**Предназначены** для

* Сокрытия некоторых частей БД от определенных пользователей
* Организации доступа пользователей к БД удобным для них образом
* Упрощения сложных операций с базовыми отношениями.

**Создаются** командой CREATE VIEW.

### **6) Какие типы ограничений целостности можно декларативно задать в командах языка SQL**

* На значения атрибутов (указывать тип данных и условия на значения атрибутов - CHECK, datatype)
* На отображения между атрибутами одного отношения (PRIMARY KEY)
* На отображения между отношениями (FOREIGN KEY)

### **7) Перечислите конструкции языка SQL связанные с ограничениями целостности**

NULL и NOT NULL (может ли атрибут иметь неопределенное значение), PRIMARY KEY (первичный ключ), UNIQUE (возможный ключ), FOREIGN KEY (внешний ключ), CHECK (условие на значение атрибутов).

### **8) Что такое неопределенное значение и логическое значение unknown Какими свойствами они обладают**

**Неопределенное значение** (NULL) указывает, что значение атрибута сейчас неизвестно или неприемлемо для кортежа.

* Не принадлежит никакому типу данных и может присутствовать среди значений любого атрибута.
* При выполнении двуместных операций с другими значениями некоторого типа данных (например, сложение) дает неопределенное значение, при сравнении значений дает unknown.

**Unknown** - третье значение логического типа, обладающее свойствами: (для удобства запоминания можно представлять unkn0wn = 0.5, true = 1, false = 0)

* NOT unkn0wn = unkn0wn
* true AND unkn0wn = unkn0wn
* true OR unkn0wn = true
* false AND unkn0wn = false
* false OR unkn0wn = unkn0wn.

### **9) Укажите два основных правила целостности реляционной модели Как они обеспечиваются**

1. **Требование целостности сущностей** (любые два кортежа одного отношения должны быть отличимы). Обеспечивается тем, что у любого отношения должен быть первичный ключ (UNIQUE и NOT NULL).
   1. Достаточно гарантировать отсутствие отношений, содержащих кортежи с одним и тем же значением первичного ключа
   2. Запрещать вхождение в значение первичного ключа неопределенных значений
2. **Требование целостности по ссылке** (для каждого значения внешнего ключа должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным). Обеспечивается следующим Побразом:
   1. Достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа;
   2. При удалении кортежа из отношения, на которое ведет ссылка, либо запрещается его удаление, пока не будут удалены все ссылки, либо автоматически значения всех ссылок становятся неопределенными, либо автоматически удаляются все кортежи, ссылающиеся на удаляемый.

### **10) Дайте определения суперключа потенциального ключа составного ключа первичного ключа альтернативного ключа суррогатного ключа? Как они соотносятся друг с другом?**

**Суперключ** - атрибут или их множество, единственным образом идентифицирующее кортеж отношения.

**Потенциальный ключ** - суперключ, никакое подмножество которого не является суперключом.

**Составной ключ** - ключ, состоящий из нескольких атрибутов.

**Первичный ключ** - потенциальный ключ, выбранный для уникальной идентификации кортежей отношения.

**Альтернативный ключ** - потенциальный ключ, не выбранный в качестве первичного.

**Суррогатный ключ** - искусственный атрибут, не имеющий связей с реальными характеристиками явлений ПрО, вводимый в схему отношения исключительно для организации связи кортежей.

### **11) Что такое суррогатный первичный ключ? Почему в последнее время проектировщики предпочитают использовать только их?**

**Суррогатный первичный ключ** - первичный ключ, уникальные значения которого генерируются СУБД.

**Преимущества по сравнению с естественными первичными ключами:**

* Единообразие всех суррогатных ключей отношений (обычно имена строятся одинаково, а ключи имеют один тип значений)
* Компактность суррогатных ключей
* Отсутствие потребности в изменениях, которые могут затронуть много отношений схемы (т.е. ошибка в значении первичного ключа не распространяется в другие отношения, где он является внешним).

### **12) Что такое внешний ключ? Должен ли он обладать свойством уникальности? Для чего и как он используется?**

**Внешний ключ** - атрибут отношения или множество атрибутов, соответствующее потенциальному ключу некоторого отношения и являющееся его подмножеством.

Сам внешний ключ **не должен** обладать свойством уникальности, им должен обладать потенциальный ключ.

**Используется** для осуществления связи отношений (если некий атрибут присутствует в нескольких отношениях, то его наличие обычно отражает определенную связь между кортежами этих отношений)

"внешний ключ должен являться подмножеством значений атрибутов на которые он ссылается"

### **13) Укажите основные компоненты команды SQL CREATE TABLE. Приведите примеры**

* schema – имя схемы, в которой создается таблица (по умолчанию используется схема пользователя).
* table – имя создаваемой таблицы.
* column – имя столбца. DEFAULT - значение столбца по умолчанию.
* datatype – тип данных столбца (CHAR, NUMBER, DATE, BLOB, LONG…)
* inline\_constraint - ограничение целостности на уровне столбца.
* out\_of\_line\_constraint - ограничение целостности на уровне таблицы.
* constraint\_name – уникальное имя ограничения целостности. (NULL, NOT NULL, UNIQUE, PRIMARY KEY, references\_clause, CHECK)
* references\_clause - ограничение ссылочной целостности для внешнего ключа таблицы.



### **14) Что такое триггер Для чего они предназначены Процедурный способ определения ограничений целостности**

**Триггер** – это программа на языке программирования сервера, которая автоматически выполняется СУБД в момент наступления определенного события.

**Предназначение:**

* Проверка правильности введенных данных и проверка выполнения ограничений целостности данных
* Осуществление косвенных модификаций данных
* Выдача предупреждений, напоминающих о необходимости что-то выполнить
* Накопление информации аудита посредством фиксации сведений о внесенных изменениях и о лицах, внесших изменения

### **15) При каких событиях в системе БД могут запускаться триггеры? Какие факторы влияют на запуск триггеров обновления данных?**

* операторы INSERT, UPDATE или DELETE, применяемые к указанной таблице
* операторы CREATE, ALTER или DROP, применяемые к любому объекту схемы;
* запуск базы данных

**Факторы, влияющие на запуск:**

* регистрация пользователя в системе или выход из нее;
* конкретное или любое сообщение об ошибке.
* триггеры могут также активизировать друг друга.

### **16) Чем отличаются триггеры для таблиц от триггеров для представлений?**

Триггеры для таблиц отличаются от триггеров для представлений использованием основных команд SQL CREATE TRIGGER:

* BEFORE, AFTER - нельзя использовать в триггерах для представлений *(Before:* в теле триггера можно изменять NEW-значения и нельзя изменять OLD-значения столбцов; *After:* в теле триггера нельзя изменять ни NEW-значения, ни OLD-значения столбцов.);
* INSTEAD OF - нельзя использовать в триггерах для таблиц; в теле триггера можно читать (в теле триггера можно читать NEW-значения и OLD-значения и нельзя изменять ни NEW-значения, ни OLD-значения столбцов.)
* NEW-значения и OLD-значения используются для модификации представлений.

### **17) Как в коде триггера можно ссылаться на значения столбцов модифицируемых строк?**

**С помощью ключевых** слов NEW и OLD

* Ключевое слово **NEW** в теле триггера используется для ссылки на новое значение столбца
* Ключевое слово **OLD** может быть использовано для ссылки на старое значение столбца.
* Синтаксис обращений {NEW|OLD}.{имя столбца}.

### **18) Какова последовательность выполнения триггеров и основного действия с данными?**

1. Исполнение табличного триггера BEFORE на уровне оператора.
2. Для каждой строки, охваченной данным оператором:
   1. исполнение любого триггера BEFORE на уровне строки;
   2. блокировка данных и выполнение над ними действия, предписанного командой SQL
   3. выполнение проверок ограничений целостности;
   4. исполнение любого триггера AFTER на уровне строки.
3. Исполнение табличного триггера AFTER на уровне оператора.

### **19) Назовите основные компоненты команды SQL CREATE TRIGGER. Приведите примеры.**

* schema – имя схемы, в которой создается триггер
* trigger – имя триггера.
* время выполнения триггера:
  + BEFORE (до выполнения основного действия),
  + AFTER (после выполнения основного действия),
  + INSTEAD OF (вместо выполнения основного действия).
* dml\_event\_clause определяет грамматику триггеров для команд языка манипулирования данными. Команды, указывающие на то, для чего будет выполняться триггер:
  + DELETE (для удаления строк таблицы),
  + INSERT (для добавления строк в таблицу),
  + UPDATE (для модификации строк таблицы),
  + FOR EACH ROW (определяет является ли триггер строковым. Если эта фраза опущена, триггер является операторным).
  + ON указывает на имя таблицы, на изменения в которой будет реагировать триггер.
* WHEN определяет дополнительное условие, при истинности которого будет выполняться триггер.
* pl/sql\_block определяет блок PL/SQL, который выполняется, когда триггер активизируется

### **20) Каковы основные особенности навигационного стиля манипулирования реляционными данными? Когда используется навигационный стиль манипулирования реляционными данными?**

**Особенность**: построчное считывание информации при помощи циклов

**Используется**: исключительно в интерфейсах с реляционными СУБД.

Все эти интерфейсы строятся по **принципу**:

1) СУБД передается запрос на выборку данных

2) В памяти программы отводится место - для размещения значений, который считываются из текущей строки

3) Затем происходит построчное сканирование результирующей выборки с чтением и обработкой следующих данных одной строки.

ПРОГРАММОЙ - интрефейсы

### **21) Что собой представляют курсоры PL/SQL?**

**Курсор** - инструмент для построчного сканирования результирующих таблиц с данными.

**Курсоры бывают:**

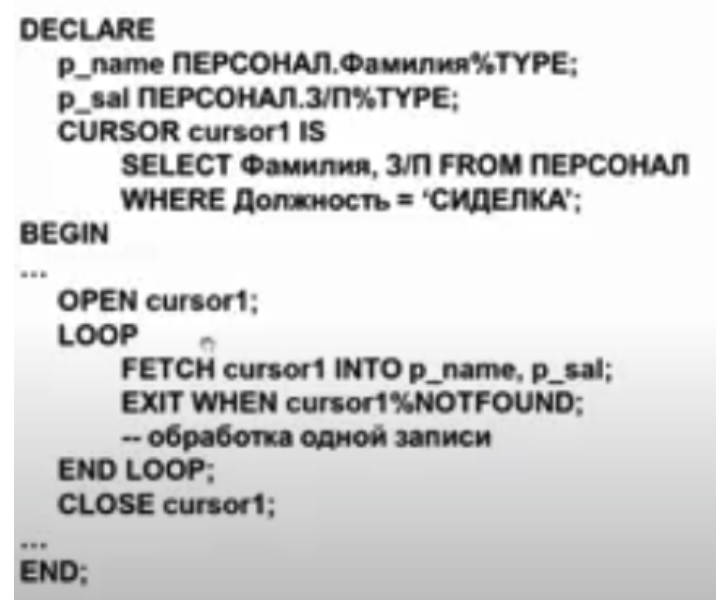
* **Неявные** - организуются системой неявно для всех команд SQL, если они селектируют и выполняют некоторые действия с одной строкой.
* **Явные** - объявляются в области объявлений, если селектируют и выполняют действия над множеством строк.

### **22) Какие команды предусмотрены в языке PL/SQL для объявления и обращения к курсорам?**

* Объявление курсора начинается с ключевого слова **CURSOR**.
* Команды **OPEN** и **CLOSE** работают как скобки, внутри которых курсор активен и может использоваться для сканирования данных.
* Чтение строк командой OPEN не производится, для этого предназначена команда **FETCH**.
* Конструкция **BULK** **COLLECT** команды FETCH позволяет за одно обращение к ней прочитать целиком все столбцы результирующей таблицы в соответствующие коллекции, типы данных которых соответствуют типам данных столбцов.
* Команды **LOOP** и **END** **LOOP** образуют безусловный цикл.

### **23) Как управлять процессом обращений к курсору с помощью атрибутов курсора?**

Каждый явный курсор имеет четыре атрибута – **ISOPEN**, **FOUND**, **NOTFOUND**, **ROWCOUNT**. Они возвращают полезную информацию о выполнении обращений к данным результирующей таблицы. Их следует использовать для адекватной обработки ситуаций, связанных с курсором.

* **ISOPEN** возвращает значение TRUE, если курсор открыт, FALSE – в противном случае.
* **FOUND** принимает значение NULL, если курсор открыт, но команды FETCH для него не выполнялись, TRUE, если последняя команда FETCH прочитала данные из текущей строки и FALSE, если очередной FETCH вышел за границу курсора.
* **NOTFOUND** отличается от FOUND тем, что он возвращает TRUE, когда FOUND дает FALSE и наоборот. В остальных случаях реакция аналогична.
* **ROWCOUNT** равен нулю сразу после открытия курсора и числу строк результирующей таблицы, прочитанных до этого командами FETCH в процессе сканирования.
* 

кодда

### **24) Укажите и охарактеризуйте классы спецификационных языков реляционной модели.**

1) Алгебраические языки, позволяющие выражать запросы средствами

специализированных операторов, применяемых к отношениям.

2) Языки исчисления предикатов, в которых запросы описывают требуемое

множество кортежей путем указания логического выражения, которому

должны удовлетворять эти кортежи. Делятся на:

* языки реляционного исчисления с переменными-кортежами (ALPHA)
* языки реляционного исчисления с переменными на доменах (QBE)

3) SQL-подобные языки. Каждая конструкция SELECT этих языков задает отображение строк исходных таблиц в строки результирующей таблицы.

### **25) Поясните деление языков на процедурные и декларативные.**

Запрос в процедурных языках - представляет алгоритм получения результата, а в декларативных языках - представляет собой некоторый образец или условие по которым необходимо найти требуемые данные.

Навигационные языки всегда процедурны.

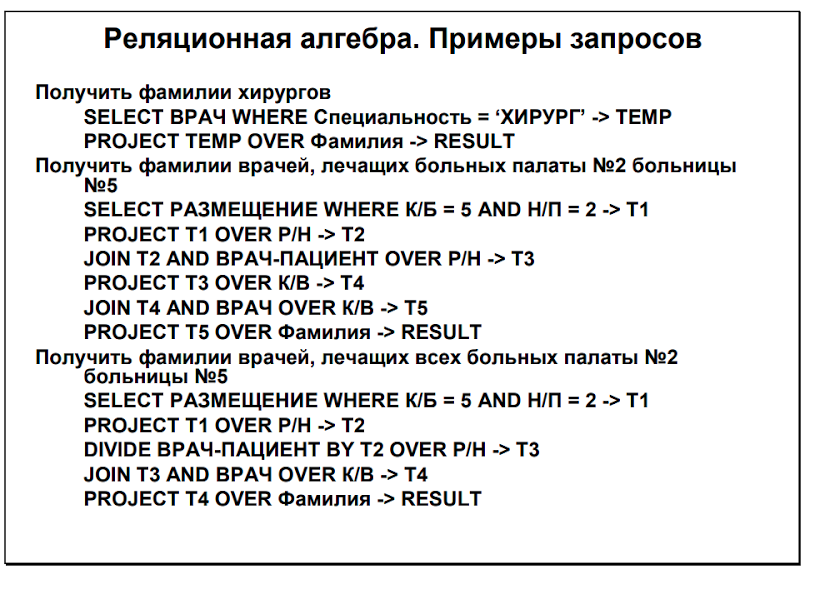
### **26) Дайте определение основных и дополнительных операций реляционной алгебры Кодда Поясните на примерах их работу.**

**Основные**:

* Объединение (UNION) - множество кортежей, которые принадлежат одному из отношений, либо им обоим.
* Разность (MINUS) - множество кортежей, принадлежащих одному отношению, но не принадлежащих второму.
* Декартово произведение отношений (TIMES) - множество всех кортежей степени k1+k2, где первые k1 компонентов которых образуют кортежи, принадлежащие первому отношению, а последние k2 – кортежи, принадлежащие второму отношению.
* Проекция (PROJECT) - берется отношение, удаляются некоторые из его атрибутов и (или) переупорядочиваются оставшиеся атрибуты.
* Селекция (SELECT) - множество кортежей, принадлежащих отношению, таких, что при подстановке i-го компонента кортежа вместо всех вхождений номера i в формулу F для всех i она станет истинной. Наряду с номерами в операциях селекции можно использовать имена атрибутов

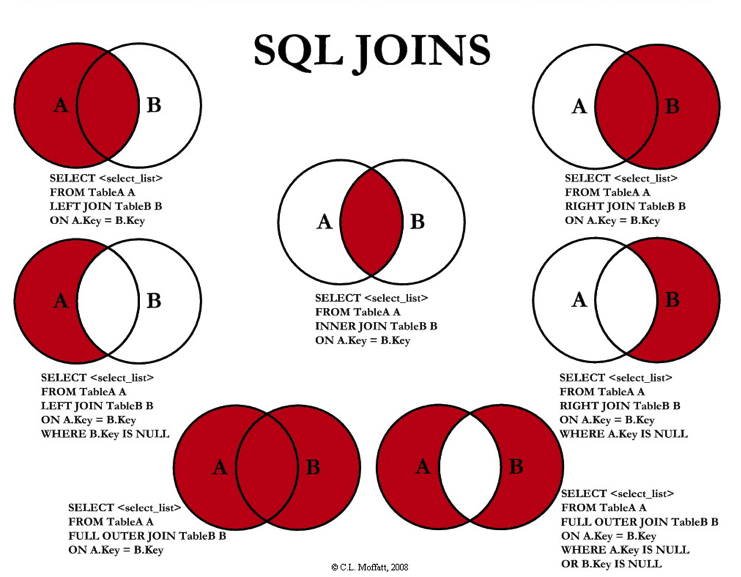
**Дополнительные:**

* Пересечение (INTERSECT) - множество кортежей, которые одновременно принадлежат обоим отношениям.ER
* Частное (DIVIDE) - множество кортежей t длины (r - s), таких, что для всех кортежей u длины s, принадлежащих S, кортеж t конкатенация u принадлежит R.
* Соединение (JOIN). Ө-соединение (тета) (Ө-JOIN) Ө-соединение отношений представляет собой множество таких кортежей их декартова произведения, что i-ый компонент первого отношения находится в отношении Ө с j-ым компонентом второго. Если Ө является оператором «равно» («=»), эта операция часто называется эквисоединением (EQUIJOIN).



### **27) Проведите на конкретном примере сравнительный анализ всех разновидностей операции соединения.**

* **Естественное соединение (INNER JOIN)** - эквисоединение, из результата которого исключены по одному экземпляру из каждой пары совпадающих атрибутов (атрибуты группы X или группы Y). То есть, в результате будет исключен один экземпляр из каждой пары совпадающих атрибутов (атрибуты группы Х или Y)
* **Композиционное соединение (COMPOSITE JOIN)** - эквисоединение, при котором атрибуты соединения (атрибуты групп X и Y) не включаются в результат. Такое соединение уместно при использовании для связей кортежей суррогатных ключей.
* **Левое внешнее соединение (LEFT OUTER JOIN)** - соединение, при котором к результату эквисоединения добавляются кортежи левого операнда, не вошедшие в эквисоединение, конкатенированные справа с заполненными NULLами кортежами.
* **Правое внешнее соединение (RIGHT OUTER JOIN)** - соединение, при котором к результату эквисоединения добавляются кортежи правого операнда, не вошедшие в эквисоединение, конкатенированные слева с заполненными NULLами кортежами.
* Результат **полного внешнего соединения (FULL OUTER JOIN)** совпадает с объединением левого и правого внешних соединений. Ничего не исключается из результата.



### **28) Как в языке реляционной алгебры выполняются действия изменяющие состояние БД?**

* Действие включения выполняется операцией UNION
* Действие удаления - операцией MINUS
* Действие модификации сводится к действию удаления и последующего включения.

Во всех этих случаях первым операндом и результатом является изменяемое базовое отношение.

А вторым операндом является либо производное отношение, полученное запросом выборки, либо отношение, кортежи которого заданы явным указанием их компонентов.

(Язык реляционной алгебры ***обычно не используют*** для изменения состояния БД, однако действия включения, модификации и удаления кортежей можно выполнить и в этом языке)

### **29) Какой вид имеют запросы в реляционном исчислении с переменными кортежами?**

{t|ψ(t)} (t такое, что “пси” от t), где t – переменная-кортеж (переменная, обозначающая кортеж некоторой фиксированной длины), а ψ (пси) – формула, построенная из атомов и совокупности операторов.

Важное дополнительное условие: указанная конструкция являлась запросом реляционного исчисления

### **30) Укажите разновидности атомов формул реляционного исчисления с переменными кортежами.**

Атомы формул ψ (пси) (реляционного исчисления) могут быть трех типов:

* R(s), где R – имя отношения, а s – переменная-кортеж. Этот атом принимает значение «истина», когда s есть кортеж отношения R.
* s[i] Ө (тета) u[j], где s и u являются переменными-кортежами, а Ө – оператор сравнения. Этот атом принимает значение «истина», когда i-ый компонент s находится в отношении Ө с j-ым компонентом u.
* s[i] Ө a - где Ө – оператор сравнения, s[i]-переменные-кортежа, a – это константа. Этот атом принимает значение «истина», когда i-ый компонент s находится в отношении Ө с константой a.

### **31) Перечислите правила построения формул реляционного исчисления с переменными кортежами.**

1. Каждый атом есть формула. Все вхождения переменных-кортежей, упомянутые в атоме, являются свободными в этой формуле.

2. Если ψ1 и ψ2 – формулы, то “пси1 или пси2”, “пси1 и пси2”, “не пси1” – тоже формулы

3. Если ψ – формула, в которой есть свободная переменная s, то “существует s от пси” и “любое s от пси” – также формулы.

4. Формулы могут заключаться в круглые скобки для изменения приоритета операторов. Приоритет: 1) оператор сравнения, 2) ∃/∀, 3) отрицание, 4) и, 5) или.

5. Ничто иное не является формулой.

### **32) Как определяется статус связана свободна переменных кортежей?**

Вхождение переменной в формулу является «связанным», если этой переменной предшествует квантор «для всех» или «существует». Иначе - переменная свободная.

### **33) Какой вид имеют запросы в реляционном исчислении с переменными на доменах?**

Запросы в реляционном исчислении с переменными на доменах имеют вид

(читается так: х1, х2 такие, что пси от х1, х2) где **x1, x2, …, xk** – переменные на доменах, т.е. переменные, обозначающие скалярные значения, взятые из определенных доменов, а **ψ** – формула, построенная из атомов и совокупности операторов

### **34) Укажите разновидности атомов формул реляционного исчисления с переменными на доменах.**

Атомы формул пси могут быть двух типов:

* R(x1 x2 … xk), где R – имя отношения степени k, а каждое xi есть константа или переменная на домене. Этот атом принимает значение «истина», когда x1 x2 … xk есть кортеж отношения R.
* x Ө (тета) y, где x и y являются константами или переменными на доменах, а Ө – оператор сравнения. Этот атом принимает значение «истина», когда x находится в отношении Ө с y.

### **35) Перечислите правила построения формул реляционного исчисления с переменными на доменах.**

1. Каждый атом есть формула.

2. Если ψ1 и ψ2 – формулы, то “пси1 или пси2”, “пси1 и пси2”, “не пси1” – тоже формулы

3. Если ψ – формула, в которой есть свободная переменная s, то “существует s от пси” и “любое s от пси” – также формулы.

4. Формулы могут заключаться в круглые скобки для изменения приоритета операторов. Приоритет: оператор сравнения, ∃/∀, отрицание, и, или.

5. Ничто иное не является формулой.

### **36) Как определяется статус связана свободна переменных на доменах?**

Вхождение переменной в формулу является «связанным», если этой переменной предшествует квантор «для всех» или «существует». Иначе - свободной.

(читается так: х1, х2 такие, что пси от х1, х2)

ИЛИ:  где x1, …, xk – свободные переменные на доменах в формуле

**Формулы, а также свободные и связанные вхождения переменных на доменах в этих формулах определяются рекурсивно следующим образом.**

* Каждый атом есть формула. Все вхождения переменных на доменах, упомянутые в атоме, являются свободными в этой формуле.
* Экземпляры переменных на доменах не меняют своего статуса «связана-свободна» в новых формулах.
* Если ψ – формула, в которой есть свободная переменная x, то ∃x (ψ) – также формула. Символ «∃» представляет собой квантор существования.
* Вхождения переменной x, свободные в формуле ψ, становятся связанными этим квантором в новой формуле. Остальные вхождения переменных на доменах, включая возможные вхождения x, связанные в ψ, своего статуса не меняют.
* Если ψ – формула, в которой есть свободная переменная x, то ∀x (ψ) – также формула. Символ «∀» представляет собой квантор всеобщности. Вхождения переменной x, свободные в формуле ψ, становятся связанными этим квантором в новой формуле. Остальные вхождения переменных на доменах, включая возможные вхождения x, связанные в ψ, своего статуса не меняют

### **37) Перечислите основные отличительные особенности языка QBE**

* Ориентация на диалоговое взаимодействие человека с системой БД.
* Двумерный синтаксис команд. (Поскольку операции задаются в табличной форме, говорят, что QBE имеет двумерный синтаксис)
* Минимум вводимых человеком символов.
* Объединение в одном и том же синтаксисе функциональности всех языков системы БД (языка определения данных, языка манипулирования данными, языка определения ограничений целостности, языка безопасности данных).
* Мощная и в то же время бескванторная «человеческая» логика.

### **38) Какие группы полей выделяются в таблице шаблоне QBE К каким элементам БД они относятся?**

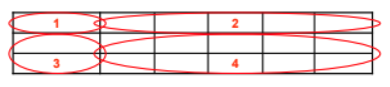
Левое верхнее поле - идентификация отношения (таблицы) и действий с ним. (**Действие с отношениями**)

Остальные поля заголовка - идентификация атрибутов и действий с ними.

(**Действие с атрибутами**)

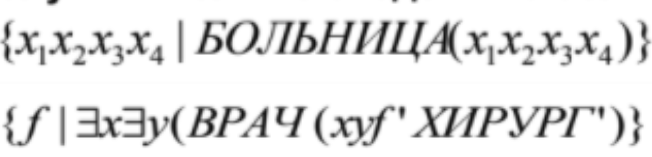
Поля первого столбца (кр. верхнего) - **действия с кортежами**.

Поля тела - **действия с отдельными компонентами кортежей**.

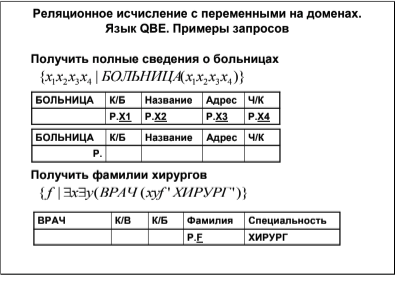


### **39) Опишите на примере последовательность совместных действий пользователя и системы по формулированию запроса QBE (Query by example).**

1. Предлагается пустой шаблон таблицы.
2. Пользователь заполняет поле в левом верхнем углу именем таблицы. Поля шапки таблицы, за исключение левого, могут заполниться автоматически.
3. После этого пользователь формулирует запрос, например



1. После того как запрос сформулирован, пользователь нажимает клавишу Enter для получения ответа.
2. Если необходимы две или более таблиц, можно сформировать дополнительные пустые шаблоны (используя специальные функциональные клавиши) и затем ввести информацию в их заголовки.
3. Условия, заданные в одной строке, связываются конъюнкцией. Для дизъюнкции условий их нужно указать в разных строках.
4. Порядок строк в запросах несущественен.



### **40) Какие различные синтаксические конструкции с ключевым словом SELECT предусмотрены в стандарте SQL Для каких ситуаций использования они предназначены? В чем особенности каждой конструкции?**

Язык допускает три конструкции, начинающиеся со слова SELECT:

* Спецификация курсора
* Оператор выборки
* Подзапрос.

**Курсор позволяет**

С помощью набора специальных операторов получить построчный доступ к результату запроса к БД.

Особенность: К табличным выражениям не предъявляются какие-либо ограничения.

**Оператор выборки позволяет**

получить результат запроса в прикладной программе без использования курсора. Особенность: Ограничения - результирующая таблица должна содержать не более одной строки.

**Подзапрос** - запрос, который может входить в предикат условия выборки оператора SQL.

Особенности: Ограничение - результирующая таблица должна содержать в точности один столбец. Вместо констант разделов WHERE и HAVING можно использовать значения столбцов текущих строк таблиц внешних запросов.

### **41) В чем заключается основная семантика табличного выражения команды SELECT Из каких разделов она состоит и для чего предназначен каждый раздел?**

Семантика табличного выражения состоит в том, что на основе последовательного применения разделов from, where, group by и having из заданных в разделе from таблиц строится некоторая новая результирующая таблица, порядок следования строк которой не определен и среди строк которой могут находиться дубликаты.

**FROM**: Результатом выполнения является расширенное декартово произведение таблиц, заданных списком таблиц раздела FROM.

**WHERE**: Результатом является таблица, состоящая из тех строк, для которых результатом вычисления условия поиска является true.

**GROUP** **BY**: Результатом является разбиение отношения на множество групп строк, в которых для каждого столбца из списка столбцов раздела GROUP BY во всех строках каждой группы, значения этого столбца совпадают.

**HAVING**: Результатом выполнения является сгруппированная таблица, содержащая только те группы строк, для которых результат вычисления условия поиска есть true.

### **42) Опишите в целом алгоритм вычисления табличного выражения команды SELECT.(селект)**

FROM -> WHERE (если есть) -> GROUP BY (если есть) -> HAVING (если есть)

* Если табличное выражение содержит только раздел FROM (это единственный обязательный раздел табличного выражения), то результат табличного выражения совпадает с результатом раздела FROM.
* Если присутствует раздел WHERE, то далее вычисляется он.
* Если присутствует раздел GROUP BY, то далее выполняется он.
* Последним выполняется раздел HAVING (если он присутствует).

### **43) Каковы особенности использования неопределенных значений атрибутов и логического значения unknown в запросах SQL?**

Значение выражения неопределено, если в его вычислении участвует хотя бы одно неопределенное значение.

**В контексте** GROUP BY, DISTINCT и ORDER BY неопределенное значение выступает как специальный вид определенного значения, т.е. возможно образование группы строк, значение указанного столбца которых является неопределенным.

**Булевские операции AND, OR и NOT** работают в трехзначной(true, false и unknown) логике следующим образом:

(для удобства запоминания можно представлять unkn0wn = 0.5, true = 1, false = 0)

* true AND unkn0wn = unkn0wn,
* false AND unkn0wn = false,
* unkn0wn AND unkn0wn = unkn0wn,
* true OR unkn0wn = true
* false OR unkn0wn = unkn0wn,
* unkn0wn OR unkn0wn = unkn0wn,
* NOT unkn0wn = unkn0wn

### **44) В каких случаях атомы предикаты принимают значения true false и unknown?**

(для удобства запоминания можно представлять unknown = 0.5, true = 1, false = 0)

*(таблицу читать надо так:   
“Предикат <имя предиката> примет значение <true, false, unknown>, если <текст на месте пересечения имени предиката и принимаемого значения>”)*

**Предикат сравнения:**

* **true:** сравнение верно (4 < 5)
* **false:** сравнение неверно (5 < 4)
* **unknown:** хотя бы один из операндов имеет значение NULL или правый операнд – подзапрос с пустым результатом

**Предикат between:**

* **true:** оба из предикатов сравнения верны
* **false:** один из предикатов сравнения неверен
* **unknown:** один из предикатов сравнения имеет неопределенное значение

**Предикат in:**

* **true:** значение левого операнда совпадает с одним из значений списка правого операнда
* **false:** список правого операнда пуст или значение левого операнда не совпадает ни с одним значением из списка правого операнда
* **unknown:** в остальных случаях (напр., значение левого операнда равно NULL).

**Предикат like:**

* **true:** значение указанного столбца удовлетворяет заданному шаблону
* **false:** значение указанного столбца НЕ удовлетворяет заданному шаблону
* **unknown:** значение шаблона или столбца неопределено.

**Предикат is null:**

* **true: указанное значение неопределено**
* **false: указанное значение определено**
* **исходова: никогда**

**Предикат с квантором ALL:**

* **true: результат вычисления подзапроса пуст, или значение предиката верно для каждого элемента подзапроса**
* **false: значение предиката неверно хотя бы для одного элемента подзапроса**
* **исходова: в остальных случаях**

**Предикат с квантором SOME:**

* **true: значение предиката верно хотя бы для одного элемента подзапроса**
* **false: результат вычисления подзапроса пуст или значение предиката неверно для каждого элемента подзапроса**
* **исходова: в остальных случаях**

**Предикат exists:**

* **true: результат вычисления подзапроса НЕ пуст**
* **false: результат вычисления подзапроса пуст**
* **исходова: никогда**

### **45) Какие виды атомов предикатов предусмотрены в стандарте SQL для логического выражения условия поиска?**

**Сравнения**, **between**, **exists**, **in**, **like**, **null** и **предикат** **с** **квантором   
(кванторы бывают all, some)**.

### **46) Перечислите и поясните все случаи при которых вычисление табличного выражения приведет к сгруппированной таблице**

**Случай: Наличие в выражении GROUP BY без HAVING:**

* **Пояснение:** Результатом этого раздела является разбиение таблицы на множество групп строк, в которых для каждого столбца из списка столбцов раздела GROUP BY во всех строках каждой группы, значения этого столбца совпадают.
* **Пример:** **SELECT COUNT(reg\_number), gender FROM Patients GROUP BY gender;** *посчитать кол-во пациентов определенных полов*

**Случай: Наличие HAVING без GROUP BY:**

* **Пояснение:** Результатом выполнения выражения будет либо пустая таблица, либо результат выполнения предыдущих разделов табличного выражения, рассматриваемых как одна группа без столбцов группировки.
* **Пример:** **SELECT birthdate, gender FROM Patients HAVING birthdate > '19990101';** *вывести ДР и пол пациентов, родившихся после 1-го января 1999-го года (тут HAVING работает как WHERE, только в конечном итоге результат будет рассматривать как одна группа)*

**Случай: Наличие GROUP BY и HAVING:**

* **Пояснение:** Результатом выполнения раздела HAVING является сгруппированная таблица, содержащая только те группы строк, для которых результат вычисления условия поиска верен.
* **Пример:** **SELECT COUNT(reg\_number), birthdate FROM Patients GROUP BY birthdate HAVING birthdate > '19990101';** *посчитать кол-во пациентов определенных дней рождения и вывести значения дат для тех, кто родился после 1 января 1999 года*

### **47) Какие дополнительные ограничения накладываются на условие поиска раздела HAVING по сравнению с условием поиска раздела WHERE?**

В выражениях предикатов, входящих в условие выборки раздела HAVING прямо можно использовать только спецификации столбцов, указанных в качестве группирования в разделе GROUP BY. Остальные столбцы можно использовать только внутри спецификаций агрегатных функций COUNT, SUM, AVG, MIN, MAX, вычисляющих в данном случае некоторое агрегатное значение для всей группы строк.

Аналогично обстоит дело с подзапросами, входящими в предикаты условия выборки раздела HAVING: если в подзапросе используется характеристика текущей группы, то она может задаваться только путем ссылки на столбцы группирования.

### **48) Укажите различные случаи применения агрегатных функций в списке выборки в зависимости от вида табличного выражения.нужны примеры**

В таблице, R - результат табличного выражения

**Случай:** R не является группированной таблицей

* **Пояснение:** Появление хотя бы одной агрегатной функции от множества строк R в списке выборки приводит к тому, что R невно расматривается как cгруппированная таблица с отсутствующими столбцами группирования. Поэтому в списке выборки не допускается прямое использование спецификации столбцов R: все они должны находиться внутри спецификации агрегатных функций. Результат запроса: таблица, состоящая не более чем из одной строки, полученной путем применения агрегатных функцй
* **Пример:** SELECT COUNT(\*) FROM Patients; Посчитать кол-во пациентов. Результат - единственная строка с кол-вом пациентов.

**Случай:** R - группированная таблица, но табличное выражение не содержит GROUP BY и имеет HAVING

* **Пояснение:** Выборка формируется только с указанием столбцов внутри спецификаций агрегатных функций. Т.к. результат табличного выражения явно объявлен группированной таблицей. Результат запроса – таблица из не более чем одной строки, полученной путем применения агрегатных функций к R
* **Пример:** SELECT COUNT(T.gender) FROM (SELECT birthdate, gender FROM Patients HAVING birthdate > '19990101') T

**Случай:** R - группированная таблица, табличное выражение имеет GROUP BY и хотя бы один столбец группирования

* **Пояснение:** Допускается прямое использование имен столбцов группирования, имена остальных столбцов R могут появляться только внутри агрег. функций. Результат - таблица, число строк в которой равно числу групп в R, каждая строка формируется на основе значений столбцов группирования и агрег. функций.
* **Пример:** SELECT COUNT(reg\_number), gender FROM Patients GROUP BY gender; Посчитать кол-во пациентов определенных полов.

### **49) Как формулируется задача проектирования реляционной базы данных? Какие цели при этом преследуются?**

**Задача**: Выбрать подходящую логическую структуру для заданного массива данных, которые требуется поместить в базу данных. Иначе говоря, нужно решить вопрос, какие необходимы базовые отношения и какой набор атрибутов они должны включать. Другой важной задачей является обеспечение целостности данных, которая решается введением в схему БД ограничений целостности.

**Цели**:

1) Возможность хранения всех необходимых данных в БД.

2) Исключение избыточного дублирования данных.

3) Сведение числа хранимых в БД отношений к минимуму.

4) Нормализация отношений для упрощения решения проблем, связанных с вставкой, обновлением и удалением данных.

### **50) Что такое универсальное отношение?**

**Универсальное** **отношение** - это отношение, в которое включаются все представляющие интерес атрибуты ПрО. И которое может таким образом содержать все данные, предполагающие к хранению в БД. Для малых БД (включающих не более 15-20 атрибутов) универсальное отношение может использоваться в качестве отправной точки при проектировании РБД.

### **51) Какие аномалии могут возникать при использовании некачественных отношений?**

Аномалии вставки, удаления, обновления.

### **52) Укажите условие первой нормальной формы отношений.**

Отношение находится в первой нормальной форме, если все его атрибуты атомарны, то есть если ни один из его атрибутов нельзя разделить на более простые атрибуты, которые соответствуют каким-то другим свойствам описываемой сущности.

### **53) Что такое декомпозиция отношения? Для чего используется декомпозиция?**

**Декомпозиция** - процесс разбиения отношения, с целью уменьшения

вероятности возникновения аномалий.

(Декомпозиция отношения R - замена R на совокупность отношений {R1, R2,… , Rn} такую, что каждое из них есть проекция R, и каждый атрибут R входит хотя бы в одну из проекций декомпозиции.)

### **54) С помощью какой операции над отношениями она осуществляется?**

С помощью операции **проекции**. Каждое из получаемых в результате декомпозиции отношений в действительности является проекцией исходного отношения. ???

### **55) Каких правил следует придерживаться при выборе Функциональной зависимости для очередной декомпозиции?**

Простым правилом выбора ФЗ для декомпозиции может служить поиск цепочек ФЗ вида A -> B -> C (A функционально определяет B, B функционально определяет C) с последующим использованием крайней правой зависимости.

Более обобщенно: всеми средствами следует избегать выбора ФЗ, зависимая часть которой сама – целиком или частично – является детерминантом другой ФЗ.

### **56) Какая операция является обратной декомпозиции?**

Операция соединения.

Если оператором декомпозиции в процедуре нормализации является операция

проекции, то обратной операцией служит операция соединения.

### **57) Как в теории реляционных БД определяется функциональная зависимость ФЗ? Какое отображение стоит за этим понятием? Что является источником информации о ФЗ?**

1) Если даны два атрибута A и B, то говорят, что B функционально зависит (ФЗ) от A, если для каждого значения A в любой момент времени существует не более одного связанного с ним значения B. A и B могут быть составными. Говорят еще, что A функционально определяет B.

2) Функциональное отображение между атрибутами A и B.

3) Единственным способом для получения информации о функциональных зависимостей для схемы отношения заключается в том, чтобы внимательно проанализировать семантику атрибутов. В этом смысле зависимости являются фактически высказываниями о закономерностях реального мира. ???

### **58) Что такое детерминант атрибута?**

Если А -> В ( A функционально определяет B) и В не зависит функционально от любого подмножества А, то говорят, что А представляет собой детерминант В.

### **59) Что такое возможный ключ отношения?**

Возможный ключ отношения - атрибут или набор атрибутов, который может быть использован для данного отношения в качестве первичного ключа. (UNIQUE)

Возможный ключ (потенциальный ключ, ключ-кандидат) – это

1) Атрибут (группа атрибутов) отношения, который функционально определяет все другие атрибуты этого отношения.

2) Атрибут (группа атрибутов) отношения, который функционально определяет отношение.

3) Атрибут (группа атрибутов) отношения, который не имеет значений-дубликатов в кортежах отношения.

### **60) Определите условие нормальной формы Бойса Кодда (НФБК). (нормальной формы Бойса Кодда)**

Отношение находится в нормальной форме Бойса-Кодда (НФБК), если каждый детерминант атрибутов отношения является возможным ключом отношения.

4.2, 164-173

### **61) Приведите первоначальный алгоритм нормализации отношений до НФБК. (нормальной формы Бойса Кодда)**

1. Разработка универсального отношения для БД и преобразование его в 1НФ (первую нормальную форму).

2. Определение всех ФЗ очередного отношения.

3.Определение того, находится ли очередное отношение в НФБК. Если да, то проектирование для него завершается, если нет – это отношение декомпозируется на два новых, которые помещаются в очередь еще не проверенных на НФБК отношений.

4. Повторение шагов 2 и 3 для каждого очередного отношения. Проектирование завершается, когда очередь еще не проверенных на НФБК отношений опустеет, а, значит, все полученные отношения находятся в НФБК.

### **62) Укажите желательные свойства декомпозиции. Дайте им определения.**

* Соединение без потерь: декомпозиция обладает свойством соединения без потерь, если она позволяет восстановить любой кортеж исходного отношения, используя соответствующие кортежи меньших отношений, полученных в результате декомпозиции.
* Сохранение зависимостей: декомпозиция называется сохраняющей зависимости, если зависимости исходного отношения сохранены в новой схеме отношений.
  + Более подробно: декомпозиция называется сохраняющей зависимости, если замыкание **соединения всех подмножеств функциональных зависимостей меньших отношений** соответствует замыканию исходного множества ФЗ.

### **63) В чем заключается метод синтеза? Приведите пример.**

В основе метода синтеза лежит утверждение о том, что необходимо все функциональные зависимости с одинаковыми детерминантами выделить в группы и для каждой группы построить свое отношение, куда включить все атрибуты всех функциональных зависимостей этой группы.

* Пример: когда атрибут зависит от двух различных детерминантов:

R(A, C, B)

A->B, C->B

При использовании метода декомпозиции для одной из зависимостей, другая неизбежно теряется:

R1(A, C) и R2(A, B) => C->B утеряна

R1(A, C) и R2(C, B) => A->B утеряна

Если воспользоваться методом синтеза и для каждого детерминанта A и B определить группу зависимых атрибутов и построить свое отношение:

R1(A, B) и R2(C, B) => оба отношения находятся в НФБК (нормальная форма Бойса Кодда), все зависимости сохранены

### **64) Что такое избыточная ФЗ (Что такое избыточная функциональная зависимость)?**

Зависимость, не заключающая в себе такой информации, которая не могла бы быть получена на основе других зависимостей из числа использованных при проектировании БД.

### **65) Перечислите правила вывода ФЗ (правила вывода функциональной зависимости).**

Рефлективность, Пополнение, Транзитивность, Объединение, Декомпозиция, Псевдотранзитивность.

* **Рефлективность** указывает, что множество атрибутов всегда функционально определяет любое из своих подмножеств.
* **Пополнение** указывает, что добавление одного и того же множества атрибутов и к левой, и к правой частям функциональной зависимости (или только к левой) приводит к получению еще одной действительной зависимости.
* **Транзитивность** указывает, что функциональные зависимости являются транзитивными.
* **Декомпозиция** определяет, что можно удалять атрибуты из правой части зависимости.
* **Объединение** указывает, что в процессе проектирования может быть выполнена обратная операция, при которой ряд зависимостей с одинаковыми левыми частями А -> В, А -> С и A -> D объединяется в одну функциональную зависимость А -> В, С, D.

### **66) Какими свойствами обладают аксиомы Армстронга?**

Надежность и полнота.

**Надежность**: если зависимость выведена из некоторого набора при помощи аксиом, то она справедлива для любого отношения, для которого справедлив исходный набор.

**Полнота**: все функциональные зависимости, производные для некоторого исходного множества, можно вывести из этого множества только при помощи этих аксиом.

Аксиомы - рефлективность, пополнение, транзитивность.

**(Аксиомы Армстронга** - это набор ссылок (или, точнее, правил вывода), используемых для вывода всех функциональных зависимостей реляционной базы данных**)**

### **67) Как определять избыточные ФЗ с использованием правил вывода ФЗ? Приведите пример.**

Если некоторая функциональная зависимость может быть выведена из других ФЗ с помощью этих правил, то она избыточна.

* Пример: пусть справедливы зависимости А -> В, А -> С и А -> В, С. Из правила объединения мы знаем, что зависимость А -> В, С следует из А -> В и А -> С, следовательно, она избыточна.

### **68) Что такое минимальное покрытие ФЗ отношения?**

Набор неизбыточных ФЗ (функциональных зависимостей), полученный путем удаления всех избыточных ФЗ из исходного набора с помощью правил вывода.

### **69) Как окончательно выглядит декомпозиционный алгоритм проектирования реляционных схем БД?**

1. Построение универсального отношения.
2. Определение всех ФЗ (функциональных зависимостей), существующих между атрибутами этого отношения.
3. Удаление всех избыточных ФЗ из исходного набора ФЗ с целью получения минимального покрытия.
4. Использование ФЗ из минимального покрытия для декомпозиции универсального отношения в набор НФБК-отношений.
   1. Определение ФЗ минимального покрытия для очередного отношения.
   2. Определение того, находится ли очередное отношение в НФБК. Если да, то проектирование для него завершается, если нет – это отношение декомпозируется на два новых, которые помещаются в очередь еще не проверенных на НФБК отношений.
   3. Повторение шагов 4.1 и 4.2 для каждого очередного отношения. Проектирование завершается, когда очередь еще не проверенных на НФБК отношений опустеет, а, значит, все полученные отношения находятся в НФБК.
5. Если может быть построено более чем одно минимальное покрытие, осуществляется сравнение результатов, полученных на основе этих минимальных покрытий, с целью определения варианта, лучше других отвечающего требованиям ПрО.

### **70) Какие проверки отношений следует провести на завершающей фазе проектирования?**

1. Составляются списки ФЗ (функциональных зависимостей) для каждого отношения. Списки проверяются по двум направлениям:
   1. Одна и та же ФЗ не должна появляться более чем в одном отношении.
   2. Набор ФЗ, полученный в результате проектирования, должен в точности совпадать с набором, присутствующем в минимальном покрытии, полученном перед началом проектирования. В противном случае, необходимо показать возможность получения итогового набора ФЗ из минимального покрытия с помощью правил вывода и наоборот.
2. Осуществляется проверка на присутствие избыточных отношений. Если устанавливается избыточность отношения, его следует исключить из проектного набора. Отношение является избыточным, если:
   1. все атрибуты этого отношения могут быть найдены в одном другом отношении проектного набора.
   2. все атрибуты этого отношения могут быть найдены в отношении, которое может быть получено из других отношений проектного набора с помощью серии операций соединения над этими отношениями.
3. Отношения рассматриваются с практической точки зрения. Изучается характер использования отношений в конструируемой БД и определяется, будут ли они обеспечивать те типы запросов и операций обновления, которые предполагается выполнять.