**14. Простые и сложные объекты. Составные, обобщенные и агрегированные объекты.***Объект* – это совокупность свойств элемента, которые могут быть исследованы и описаны.

Примеры объектов: студент или группа студентов, аудитория, время занятий, слова, числа и т.д. Обычно считается, что быть объектом – значит быть дискретным и различимым. По-видимому, примеры «необъектов» это мир, время, материя, смысл и т.п. При этом можно представить БД, в которой хранятся сведения об этих категориях.

Объекты могут быть простыми и сложными. Простые объекты представляются совокупностью характеризующих их свойств. Внутренняя структура простого объекта не раскрывается. Сложные объекты имеют определенный состав и структуру. Например: студент Петров учится в группе СЦИ-201.   
Объект может быть материальным (служащий, изделие или населенный пункт) и нематериальным (имя, понятие, абстрактная идея).

Объект имеет различные свойства (например, цвет, вес, имя), которые важны для нас в то время, когда мы обращаемся к объекту (например, выбираем среди множества других) с какой-либо целью его использования. Причем свойства объекта можно задавать как отдельными однозначно интерпретируемыми количественными показателями, так и словесными нечеткими описаниями, допускающими разную трактовку, зависящую, например, от точки зрения и наличных знаний воспринимающего субъекта.

***Составные объекты данных*** позволяют интерпретировать некоторые части информации как единое целое таким образом, чтобы затем можно было легко разделить их вновь. Возьмем, например, дату "октябрь 15, 1991". Она состоит из трех частей информации - месяц, день и год. Представим ее на рис. 1, как древовидную структypy.

**Обобщенный объект** — отражает наличие связи род-вид между объектами предметной области. Объекты СТУДЕНТ-ШКОЛЬНИК-АСПИРАНТ образуют обобщенный объект УЧАЩИЙСЯ — родовой объект из видовых объектов. Как родовой объект, так и видовые объекты, обладают определенным набором свойств, причем имеет место наследование. Видовой объект обладает всеми свойствами, которые есть у родового объекта, а также свойствами, которые присущи объектам только этого вида.

**Определение родовидовых связей —** означает классификацию объектов по определенным признакам.

Виды или подклассы объектом могут быть представлены в ИЛМ в явной форме. При графическом изображении объекта можно обозначить треугольником каждый вид; с ним связаны свойства, характерные для данного вида.

**Агрегированный объект** — обычно соответствует некоторому процессу, в который вовлечены другие объекты, присутствующие в предметной области. Принято именовать существительным, который происходит от глагола, который описывает соответствующий объект. Изображается ромбом, внутри которого записывается название объекта. Ромб соединяется отрезками прямых с условными обозначениями объектов, входящими в агрегированный объект.

**15. Инфологическое моделирование и проектирование. Компоненты ИЛМ.**

**1-2**. Описание предметной области представляется с помощью какой-либо знаковой системы, поэтому в ИЛМ отражаются не только отношения, присущие данной предметной области (ПО), но и лингвистические отношения, обусловленные особенностями предметной области в языковой среде.

Учитываются понятия: синонимы, омонимы и т.д.

3. **Алгоритмические связи показателей** — для фиксирования алгоритмических зависимостей между показателями.

В третьем компоненте отражаются алгоритмические зависимости между показателями.

Пример графа с показателями:

Для вычисления показателя P5 требуется значение P2.

В алгоритмических связях фиксируются расчетные файлы.

**4.** Четвертый компонент содержит информацию о типах запросов, поступающих от пользователей, информацию о режимах использования данных и другую информацию.

5. **Ограничение целостности** — это условия при которых имеют смысл значения, хранящиеся в базе данных или условия, при которых значения могут быть записанными в базу данных (например, частота процессора больше нуля, или объем памяти больше нуля и т.д.)

**16. Этапы проектирования баз данных ИО САПР. Взаимосвязь этапов и их последовательность.**

**I Инфологическое моделирование как результат разработки модели ПО**

**Предметная область** — это часть реального мира, которая представляет интерес.

**Инфологическая модель предметной (ИЛМ) области** — описание предметной области, выполненное без ориентации на использованные в дальнейшем СУБД и технологические средства, и отражающие информационные аспекты ПО.

**II Даталогическое проектирование**

**Даталогическая модель (ДЛМ)** строится на основе ИЛМ. ДЛМ БД является концептуальной моделью БД и отражает логические связи между информационными элементами ДЛМ. В ДЛМ фиксируются данные и связи данных между ними.

ДЛМ строится в терминах информационных единиц, допустимых в той конкретной СУБД, в среде которой проектируется БД. ДЛМ зависит от выбора СУБД для разработки БнД или информационной модели.

**Схема БД** - описание ДЛМ на языке выбранной СУБД.

**III Физическое проектирование**

Для привязки ДЛМ к среде хранения используется **модель данных физического уровня** (физическая модель, внутренняя модель). Эта модель БД определяется используемыми ЗУ, способами физической организации данных в среде хранения.

Физическая модель, также как и ДЛМ, строится с учетом особенностей выбранной СУБД.

**Физическое проектирование** — описание физической структуры БД.

**IV Этап определения подсхем**

В некоторых СУБД имеется возможность описать логическую структуру БД с точки зрения конкретной группы пользователей. Такая **модель называется внешней**, а ее описание — **подсхемой.** Если СУБД поддерживает уровень подсхем, то перед проектировщиком может встать задача определения подсхем. Тогда это будет еще одним этапом.

**Этап определения подсхем** — этап разработки внешней модели. Использование подсхем облегчает работу пользователя, т.к. он должен знать структуру не всей базы, а только той ее части, которая представляет интерес. Подсхема приспособлена под потребности конкретного пользователя.

**17. Нормальные формы отношений. Определения. Примеры.**

**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_%28%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%29) 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. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) к виду, отвечающему нормальным формам, называется **нормализацией**. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных.[[1]](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#cite_note-1) Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации

В создании и развитии теории нормализации принимали участие многие учёные. Однако первые три нормальные формы и концепцию функциональной зависимости предложил [Э. Кодд](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B4,_%D0%AD%D0%B4%D0%B3%D0%B0%D1%80).[[4]](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#cite_note-Date_Writings-4)

**Первая нормальная форма (1NF)**

Переменная отношения находится в первой нормальной форме (1НФ) [тогда и только тогда](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B3%D0%B4%D0%B0_%D0%B8_%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE_%D1%82%D0%BE%D0%B3%D0%B4%D0%B0), когда в любом допустимом значении отношения каждый его [кортеж](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия *отношение*. Что же касается различных *таблиц*, то они могут не быть правильными *представлениями отношений* и, соответственно, могут не находиться в 1НФ.

**Вторая нормальная форма (2NF)**

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 неключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа.

**Третья нормальная форма (3NF)**

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во [второй нормальной форме](https://ru.wikipedia.org/wiki/%D0%92%D1%82%D0%BE%D1%80%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 отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

**Нормальная форма Бойса — Кодда (BCNF)**

Переменная отношения находится в нормальной форме Бойса — Кодда (иначе — в усиленной третьей нормальной форме) тогда и только тогда, когда каждая ее нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый [потенциальный ключ](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).

**Четвёртая нормальная форма (4NF)**

Переменная отношения находится в четвёртой нормальной форме, если она находится в нормальной форме Бойса — Кодда и не содержит нетривиальных [многозначных зависимостей](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%B7%D0%BD%D0%B0%D1%87%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).

**Пятая нормальная форма (5NF)**

Переменная отношения находится в пятой нормальной форме (иначе — в проекционно-соединительной нормальной форме) тогда и только тогда, когда каждая нетривиальная *зависимость соединения* в ней определяется потенциальным ключом (ключами) этого отношения.

**Доменно-ключевая нормальная форма (DKNF)**

**Шестая нормальная форма (6NF)**

Введена [К. Дейтом](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D1%82,_%D0%9A%D1%80%D0%B8%D1%81%D1%82%D0%BE%D1%84%D0%B5%D1%80) в его книге,[[2]](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#cite_note-Date-2) как обобщение пятой нормальной формы для [темпоральной базы данных](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%BE%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

**18. Иерархическая модель данных. Иерархические СУБД. Организация целостности.**

**Иерархическая модель данных** — это [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), где используется представление [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в виде [древовидной](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B5%D0%B2%D0%BE%D0%B2%D0%B8%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0) (иерархической) структуры, состоящей из объектов ([данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)) различных уровней.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами (в программировании применительно к структуре данных дерево устоялось название братья).

## Известные иерархические СУБД

* Типичным представителем (наиболее известным и распространенным) является [Information Management System](https://ru.wikipedia.org/wiki/IMS_%28%D0%A1%D0%A3%D0%91%D0%94%29) (IMS) фирмы [IBM](https://ru.wikipedia.org/wiki/IBM) (1966-1968 г.).
* [Time-Shared Date Management System](https://ru.wikipedia.org/w/index.php?title=Time-Shared_Date_Management_System&action=edit&redlink=1) (TDMS) компании [System Development Corporation](https://ru.wikipedia.org/wiki/System_Development_Corporation);
* [Mark IV MultiAccess Retrieval System](https://ru.wikipedia.org/w/index.php?title=Mark_IV_MultiAccess_Retrieval_System&action=edit&redlink=1) компании [Control Data Corporation](https://ru.wikipedia.org/wiki/Control_Data_Corporation);
* [System 2000](https://ru.wikipedia.org/w/index.php?title=System_2000&action=edit&redlink=1) разработки [SAS Institute](https://ru.wikipedia.org/wiki/SAS_Institute);
* Серверы каталогов, такие, как [LDAP](https://ru.wikipedia.org/wiki/LDAP) и [Active Directory](https://ru.wikipedia.org/wiki/Active_Directory) (допускают чёткое представление в виде дерева)
* По принципу иерархической БД построены иерархические файловые системы и [Реестр Windows](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B5%D1%81%D1%82%D1%80_Windows).
* InterSystems [Caché](https://ru.wikipedia.org/wiki/Cach%C3%A9)
* [Google App Engine](https://ru.wikipedia.org/wiki/Google_App_Engine) Datastore API

19. Сетевая модель данных. Сетевые СУБД. Организация целостности, наборы записей и связей.

**Сетевая модель данных** — логическая [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), являющаяся расширением [иерархического подхода](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%87%D0%B5%D1%81%D0%BA%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), строгая математическая теория, описывающая структурный аспект, аспект целостности и аспект обработки данных в сетевых базах данных.

Разница между [иерархической моделью данных](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%87%D0%B5%D1%81%D0%BA%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) и сетевой состоит в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных у потомка может иметься любое число предков.

Сетевая БД состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора экземпляров типа записи потомка. Для данного типа связи L с типом записи предка P и типом записи потомка C должны выполняться следующие два условия:

* каждый экземпляр типа записи P является предком только в одном экземпляре типа связи L;
* каждый экземпляр типа записи C является потомком не более чем в одном экземпляре типа связи L.

## Сетевые СУБД

Сетевая СУБД — [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94), построенная на основе сетевой модели данных. Вот список самых заметных из них на 1978 год[[3]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%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#cite_note-3):

* [IDS](https://ru.wikipedia.org/wiki/Integrated_Data_Store) (Integrated Data Store) компании [General Electric](https://ru.wikipedia.org/wiki/General_Electric) — самая первая сетевая СУБД, разработаная [Чарльзом Бахманом](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D1%85%D0%BC%D0%B0%D0%BD,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7_%D0%A3%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC) в 1960 г.
* [IDS/2](https://ru.wikipedia.org/wiki/Integrated_Data_Store) или [IDS/II](https://ru.wikipedia.org/wiki/Integrated_Data_Store)) компании [Honeywell](https://ru.wikipedia.org/wiki/Honeywell), купившей [IDS](https://ru.wikipedia.org/wiki/Integrated_Data_Store) у General Electric, позднее - компании [Bull](https://ru.wikipedia.org/wiki/Bull)[[4]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%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#cite_note-Celko.E2.80.942012.E2.80.94.E2.80.946-4)[[5]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%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#cite_note-5)
* Integrated Database Management System (IDMS) компании Cullinet, развитие [IDS](https://ru.wikipedia.org/wiki/Integrated_Data_Store) на основе ее исходных кодов
* DMS-1100 (для мейнфреймов UNIVAC 1100) и DMS-90 (для миникомпьютеров, первый релиз - ноябрь 1974) компании [UNIVAC](https://ru.wikipedia.org/wiki/UNIVAC)
* DBMS-10 компании [DEC](https://ru.wikipedia.org/wiki/DEC) для Decsystem-10 и Decsystem-20
* [CDC](https://ru.wikipedia.org/wiki/Control_Data_Corporation) DMS-170
* [Burroughs](https://ru.wikipedia.org/wiki/Burroughs) Data Management System (DMS-2[[6]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%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#cite_note-6)). Продукт представлен на рынке в октябре 1974 года.

Другие подобные СУБД:[[*источник не указан 98 дней*]](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8)

* IMAGE/3000 компании [Hewlett-Packard](https://ru.wikipedia.org/wiki/Hewlett-Packard) (1974 г.)
* Norsk-Data SYBAS
* [NCR](https://ru.wikipedia.org/wiki/NCR) IDM-9000
* Cincom TOTAL
* [dbVista](https://ru.wikipedia.org/w/index.php?title=DbVista&action=edit&redlink=1)
* Universal Datenbank System (UDS) от Siemens
* СООБЗ Cerebrum[[7]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%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" \l "cite_note-7)
* [ИСУБД «CronosPRO»](https://ru.wikipedia.org/w/index.php?title=Cronospro&action=edit&redlink=1)
* [Caché](https://ru.wikipedia.org/wiki/Cach%C3%A9)
* [GT.M](https://ru.wikipedia.org/w/index.php?title=GT.M&action=edit&redlink=1)

**Набор** - именованная двухуровневая иерархическая структура, которая содержит запись владельца и запись (или записи) членов. Наборы отражают связи «один ко многим» и «один к одному» между двумя типами записей.

**Наборы** бывают нескольких **видов**:

* С одними и теми же типами записей, но разными типами наборов.
* Наборы из трех записей и более, в том числе с обратной связью.
* Сингулярный набор (только один экземпляр). У такого набора нет естественного владельца и в качестве него выступает система. В дальнейшем такие наборы могут приобрести запись - владельца.

20. Реляционная модель данных. Реляционные СУБД. Сравнительная характеристика с иерархической и сетевой моделями данных.

**Реляционная модель данных** (РМД) — [логическая](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), прикладная [теория](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F) построения [баз данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), которая является приложением к задачам обработки данных таких разделов [математики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) как [теории множеств](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2) и [логика первого порядка](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BA%D0%B0).

На реляционной модели данных строятся [реляционные базы данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Реляционная модель данных включает следующие компоненты:

* [Структурный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0) аспект (составляющая) — данные в базе данных представляют собой набор [отношений](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29).
* Аспект (составляющая) [целостности](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) — отношения (таблицы) отвечают определенным условиям [целостности](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). РМД поддерживает декларативные [ограничения целостности](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_%D1%86%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1) уровня [домена](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%BD_%28%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29) (типа данных), уровня отношения и уровня базы данных.
* Аспект (составляющая) обработки (манипулирования) — РМД поддерживает операторы манипулирования отношениями ([реляционная алгебра](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%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0), [реляционное исчисление](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5)).

Кроме того, в состав реляционной модели данных включают теорию [нормализации](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).

Термин «реляционный» означает, что теория основана на математическом понятии [отношение](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) (*relation*). В качестве неформального синонима [термину](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%BC%D0%B8%D0%BD) «отношение» часто встречается слово [таблица](https://ru.wikipedia.org/wiki/Table_%28database%29). Необходимо помнить, что «таблица» есть понятие нестрогое и неформальное и часто означает не «отношение» как [абстрактное понятие](https://ru.wikipedia.org/w/index.php?title=%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D0%B5&action=edit&redlink=1), а [визуальное представление](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) отношения на бумаге или экране. Некорректное и нестрогое использование термина «таблица» вместо термина «отношение» нередко приводит к недопониманию. Наиболее частая ошибка состоит в рассуждениях о том, что РМД имеет дело с «плоскими», или «двумерными» таблицами, тогда как таковыми могут быть только визуальные представления таблиц. Отношения же являются абстракциями, и не могут быть ни «плоскими», ни «неплоскими».

Для лучшего понимания РМД следует отметить три важных обстоятельства:

* модель является логической, то есть отношения являются логическими (абстрактными), а не физическими (хранимыми) структурами;
* для реляционных баз данных верен [информационный](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) [принцип](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF): всё информационное наполнение базы данных представлено одним и только одним способом, а именно — явным заданием значений атрибутов в [кортежах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6) отношений; в частности, нет никаких указателей (адресов), связывающих одно значение с другим;
* наличие реляционной алгебры позволяет реализовать [декларативное программирование](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%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) и декларативное описание ограничений целостности, в дополнение к навигационному (процедурному) программированию и процедурной проверке условий.

**21. Реляционная модель данных. Таблицы (отношения) и их характеристики. Взаимосвязь таблиц (отношений). Ограничение целостности.**

**Отношение** — фундаментальное понятие [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). По этой причине модель и называется *реляционной* (от [лат.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *relatio* — «отношение», «зависимость», «связь»).

Отношение обычно имеет простую графическую интерпретацию в виде таблицы, столбцы которой соответствуют атрибутам, а строки — кортежам, а в «ячейках» находятся значения атрибутов в кортежах. Тем не менее, в строгой реляционной модели *отношение* не является *таблицей*, *кортеж* — это не *строка*, а *атрибут* — это не *столбец*[[2]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29#cite_note-DinD-2)[[3]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29#cite_note-3). Термины «таблица», «строка», «столбец» могут использоваться только в неформальном контексте, при условии полного понимания, что эти более «дружественные» термины являются всего лишь *приближением* и не дают точного представления о сути обозначаемых понятий[[2]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29#cite_note-DinD-2)[[4]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29#cite_note-4).

В соответствии с определением [К. Дж. Дейта](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D1%82,_%D0%9A%D1%80%D0%B8%D1%81%D1%82%D0%BE%D1%84%D0%B5%D1%80), таблица является прямым и верным представлением некоторого отношения, если она удовлетворяет следующим пяти условиям:

1. Нет упорядочивания строк сверху-вниз (другими словами, порядок строк не несет в себе никакой информации).
2. Нет упорядочивания столбцов слева-направо (другими словами, порядок столбцов не несет в себе никакой информации).
3. Нет повторяющихся строк.
4. Каждое пересечение строки и столбца содержит ровно одно значение из соответствующего [домена](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%BD_%28%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29) (и больше ничего).
5. Все столбцы являются обычными. «Обычность» всех столбцов таблицы означает, что в таблице нет «скрытых» компонентов, которые могут быть доступны только в вызове некоторого специального оператора взамен ссылок на имена регулярных столбцов, или которые приводят к побочным эффектам для строк или таблиц при вызове стандартных операторов. Таким образом, например, строки не имеют идентификаторов кроме обычных значений потенциальных ключей (без скрытых «идентификаторов строк» или «идентификаторов объектов»). Они также не имеют скрытых временных меток[[5]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29#cite_note-DateWritings-5).

**22. Функциональные зависимости. Способы изображения.**

**Функциона́льная зави́симость** — концепция, лежащая в основе многих вопросов, связанных с [реляционными базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), включая, в частности, их проектирование. Математически представляет [бинарное отношение](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5) между множествами атрибутов данного [отношения](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29) и является, по сути, связью типа «один ко многим». Их использование обусловлено тем, что они позволяют формально и строго решить многие проблемы.

### Функциональная зависимость

Пусть дано отношение rсо схемой (заголовком) R, Aи B — некоторые подмножества множества атрибутов отношения r. МножествоB **функционально зависит** от Aтогда и только тогда, когда каждое значение множества Aсвязано в точности с одним значением множества B. Обозначается A\to B.

Другими словами, если два кортежа совпадают по атрибутам A, то они совпадают и по атрибутам B.

r\left( R \right),\ A\subseteq R,\ B\subseteq R

\left( A\to B \right)\Leftrightarrow \left( \left( \forall {{t}_{1}},{{t}_{2}}\in r:{{t}_{1}}\left( A \right)={{t}_{2}}\left( A \right) \right)\Rightarrow \left( {{t}_{1}}\left( B \right)={{t}_{2}}\left( B \right) \right) \right)

В этом случае A — детерминант, B — зависимая часть.

Функциональная зависимость называется **тривиальной**, если зависимая часть является подмножеством детерминанта.

\left( B\subseteq A \right)\Rightarrow \left( A\to B \right)

### Замыкание множества зависимостей

Одни функциональные зависимости могут подразумевать другие функциональные зависимости. Например, транзитивная функциональная зависимость,

\left( A\to B \right)\wedge \left( B\to C \right)\Rightarrow \left( A\to C \right).

Множество {{S}^{+}}всех функциональных зависимостей, которые подразумеваются данным множеством функциональных зависимостей Sназывается **замыканием** множества S.

### Замыкание множества атрибутов

Пусть Z — некоторое множество атрибутов отношения r, а S — множество функциональных зависимостей этого отношения. Замыканием {{Z}^{+}}множества атрибутов Zв пределах Sназывается такое множество всех атрибутов {{A}_{i}}отношения r, что функциональная зависимость Z\to {{A}_{i}}является членом замыкания {{S}^{+}}.

r\left( R \right),\ S,\ Z\subseteq R,\ {{A}_{i}}\subseteq R,\ i=\overline{1,n}

{{Z}^{+}}=\left\{ {{A}_{i}}:\left( Z\to {{A}_{i}} \right)\in {{S}^{+}} \right\}

### Неприводимые множества зависимостей

Пусть {{S}_{1}}и {{S}_{2}} — некоторые множества функциональных зависимостей.

* Если любая функциональная зависимость из {{S}_{1}}входит и в {{S}_{2}}, то {{S}_{2}}называют **покрытием** множества функциональных зависимостей {{S}_{1}}.
* Если {{S}_{2}} — покрытие для {{S}_{1}}, а {{S}_{1}} — для {{S}_{2}}(то есть S_{1}^{+}=S_{2}^{+}), то такие множества называются **эквивалентными**.
* Множество функциональных зависимостей Sназывается **неприводимым** тогда и только тогда, когда выполняются следующие условия:
  + В каждой функциональной зависимости зависимая часть содержит только один элемент;
  + Детерминант каждой функциональной зависимости является неприводимым (ни один атрибут не может быть удален из детерминанта без изменения замыкания {{S}^{+}});
  + Ни одну функциональную зависимость из Sнельзя исключить без изменения замыкания {{S}^{+}}.
* Для любого множества функциональных зависимостей существует по крайней мере одно эквивалентное множество, которое является неприводимым. Такое эквивалентное множество называется **неприводимым покрытием**.

**23. Понятие отношения. Свойства отношений. Предикаты.**

См. вопрос 21.

##### Свойства:

##### Отсутствие кортежей-дубликатов

То свойство, что отношения не содержат кортежей-дубликатов, следует из определения отношения как множества кортежей. В классической теории множеств по определению каждое множество состоит из различных элементов.

Из этого свойства вытекает наличие у каждого отношения так называемого первичного ключа - набора атрибутов, значения которых однозначно определяют кортеж отношения. Для каждого отношения по крайней мере полный набор его атрибутов обладает этим свойством. Однако при формальном определении первичного ключа требуется обеспечение его "минимальности", т.е. в набор атрибутов первичного ключа не должны входить такие атрибуты, которые можно отбросить без ущерба для основного свойства - однозначно определять кортеж. Понятие *первичного ключа* является исключительно важным в связи с понятием целостности баз данных.

Забегая вперед, заметим, что во многих практических реализациях РСУБД допускается нарушение свойства уникальности кортежей для промежуточных отношений, порождаемых неявно при выполнении запросов. Такие отношения являются не множествами, а мультимножествами, что в ряде случаев позволяет добиться определенных преимуществ, но иногда приводит к серьезным проблемам.

##### Отсутствие упорядоченности кортежей

Свойство отсутствия упорядоченности кортежей отношения также является следствием определения отношения-экземпляра как множества кортежей. Отсутствие требования к поддержанию порядка на множестве кортежей отношения дает дополнительную гибкость СУБД при хранении баз данных во внешней памяти и при выполнении запросов к базе данных. Это не противоречит тому, что при формулировании запроса к БД, например, на языке SQL можно потребовать сортировки результирующей таблицы в соответствии со значениями некоторых столбцов. Такой результат, вообще говоря, не отношение, а некоторый упорядоченный список кортежей.

##### Отсутствие упорядоченности атрибутов

Атрибуты отношений не упорядочены, поскольку по определению схема отношения есть множество пар {имя атрибута, имя домена}. Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута. Это свойство теоретически позволяет, например, модифицировать схемы существующих отношений не только путем добавления новых атрибутов, но и путем удаления существующих атрибутов. Однако в большинстве существующих систем такая возможность не допускается, и хотя упорядоченность набора атрибутов отношения явно не требуется, часто в качестве неявного порядка атрибутов используется их порядок в линейной форме определения схемы отношения.

##### Атомарность значений атрибутов

Значения всех атрибутов являются атомарными. Это следует из определения домена как потенциального множества значений простого типа данных, т.е. среди значений домена не могут содержаться множества значений (отношения). Принято говорить, что в реляционных базах данных допускаются только нормализованные отношения или отношения, представленные в первой нормальной форме.

## 2.3.4.2.1 Предикат сравнения

Синтаксис предиката сравнения определяется следующими правилами:

<comparison predicate> ::=

<value expression> <comp op>

{<value expression> | <subquery>}

<comp op> ::=

= | <> | < | > | <= | >=

Через "<>" обозначается операция "неравенства". Арифметические выражения левой и правой частей предиката сравнения строятся по общим правилам построения арифметических выражений и могут включать в общем случае имена столбцов таблиц из раздела FROM и константы (не обязательно литеральные; вместо литеральной константы может использоваться имя столбца таблицы, указанной в разделе FROM более внешнего подзапроса, или имя переменной программы, написанной на объемлющем языке). Типы данных арифметических выражений должны быть сравнимыми (например, если тип столбца a таблицы A является типом символьных строк, то предикат "a = 5" недопустим).

Если правый операнд операции сравнения задается подзапросом, то дополнительным ограничением является то, что мощность результата подзапроса должна быть не более единицы. Если хотя бы один из операндов операции сравнения имеет неопределенное значение или если правый операнд является подзапросом с пустым результатом, то значение предиката сравнения равняется unknown.

Заметим, что значение арифметического выражения не определено, если в его вычислении участвует хотя бы одно неопределенное значение. Еще одно важное замечание из стандарта SQL/89: в контексте GROUP BY, DISTINCT и ORDER BY неопределенное значение выступает как специальный вид определенного значения, т.е. возможно, например, образование группы строк, значение указанного столбца которых является неопределенным. Для обеспечения переносимости прикладных программ нужно внимательно анализировать специфику работы с неопределенными значениями в конкретной СУБД.

## 2.3.4.2.2 Предикат between

Предикат between имеет следующий синтаксис:

<between predicate> ::=

<value expression>

[NOT] BETWEEN <value expression> AND <value expression>

По определению результат "x BETWEEN y AND z" тот же самый, что результат логического выражения "x >= y AND x <= z". Результат "x NOT BETWEEN y AND z" тот же самый, что результат "NOT (x BETWEEN y AND z)".

## 2.3.4.2.3 Предикат in

Предикат in определяется следующими синтаксическими правилами:

<in predicate> ::=

<value expression> [NOT] IN

{<subquery> | (<in value list>)}

<in value list> ::=

<value specification>

{,<value specification>}...

Типы левого операнда и значений из списка правого операнда (напомним, что результирующая таблица подзапроса должна содержать ровно один столбец)должны быть сравнимыми.

Значение предиката равно true в том и только в том случае, когда значение левого операнда совпадает хотя бы с одним значением списка правого операнда. Если список правого операнда пуст (так может быть, если правый операнд задается подзапросом) или значение "подразумеваемого" предиката сравнения x = y (где x - значение арифметического выражения левого операнда)равно false для каждого элемента y списка правого операнда, то значение предиката in равно false. В противном случае значение предиката in равно unknown (например, так может быть, если значение левого операнда есть NULL).По определению значение предиката "x NOT IN S" равно значению предиката "NOT (x IN S)".

## 2.3.4.2.4 Предикат like

Предикат like имеет следующий синтаксис:

<like predicate> ::=

<column specification> [NOT] LIKE <pattern>

[ESCAPE <escape character>]

<pattern> ::= <value specification>

<escape character> ::= <value specification>

Типы данных столбца левого операнда и образца должны быть типами символьных строк. В разделе ESCAPE должен специфицироваться одиночный символ.

Значение предиката равно true, если pattern является подстрокой заданного столбца. При этом если раздел ESCAPE отсутствует, то при сопоставлении шаблона со строкой производится специальная интерпретация двух символов шаблона: символ подчеркивания ("\_") обозначает любой одиночный символ; символ процента ("%") обозначает последовательность произвольных символов произвольной длины (может быть, нулевой).

Если же раздел ESCAPE присутствует и специфицирует некоторый одиночный символ x, то пары символов "x\_" и "x%" представляют одиночные символы "\_" и "%", соответственно.

Значение предиката like есть unknown, если значение столбца либо шаблона не определено.

Значение предиката "x NOT LIKE y ESCAPE z" совпадает со значением "NOT x LIKE y ESCAPE z".

## 2.3.4.2.5 Предикат null

Предикат null описывается синтаксическим правилом

<null predicate> ::=

<column specification> IS [NOT] NULL

Этот предикат всегда принимает значения true или false. При этом значение "x IS NULL" равно true тогда и только тогда, когда значение х не определено. Значение предиката "x NOT IS NULL" равно значению "NOT x IS NULL".

## 2.3.4.2.6 Предикат с квантором

Предикат с квантором имеет следующий синтаксис:

<quantified predicate> ::=

<value expression>

<comp op> <quantifier> <subquery>

<quantifier> ::=

<all> | <some>

<all> ::= ALL

<some> ::= SOME | ANY

Обозначим через x результат вычисления арифметического выражения левой части предиката, а через S результат вычисления подзапроса.

Предикат "x <comp op> ALL S" имеет значение true, если S пусто или значение предиката "x <comp op> s" равно true для каждого s, входящего в S. Предикат "x <comp op> ALL S" имеет значение false, если значение предиката "x <comp op> s" равно false хотя бы для одного s, входящего в S. В остальных случаях значение предиката "x <comp op> ALL S" равно unknown.

Предикат "x <comp op> SOME S" имеет значение false, если S пусто или значение предиката "x <comp op> s" равно false для каждого s, входящего в S. Предикат "x <comp op> SOMES" имеет значение true, если значение предиката "x <comp op>s" равно true хотя бы для одного s, входящего в S. В остальных случаях значение предиката "x <comp op> SOME S" равно unknown.

## 2.3.4.2.7 Предикат exists

Предикат exists имеет следующий синтаксис:

< exists predicate> ::=

EXISTS <subquery>

Значением этого предиката всегда является true или false, и это значение равно true тогда и только тогда, когда результат вычисления подзапроса не пуст.

**24. Потенциальные ключи. Определение. Состав.**

**Потенциальный ключ** — 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) — подмножество атрибутов [отношения](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%28%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%29), удовлетворяющее требованиям уникальности и минимальности (несократимости).

*Уникальность* означает, что не существует двух [кортежей](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29#.D0.92_.D0.B1.D0.B0.D0.B7.D0.B0.D1.85_.D0.B4.D0.B0.D0.BD.D0.BD.D1.8B.D1.85) данного отношения, в которых значения этого подмножества атрибутов совпадают (равны).

*Минимальность* (несократимость) означает, что в составе потенциального ключа отсутствует *меньшее* подмножество атрибутов, удовлетворяющее условию уникальности. Иными словами, если из потенциального ключа убрать любой атрибут, он утратит свойство уникальности.

Поскольку все кортежи в отношении по определению уникальны, в нём всегда существует хотя бы один потенциальный ключ (например, включающий все атрибуты отношения).

В отношении может быть одновременно несколько потенциальных ключей. Один из них может быть выбран в качестве [*первичного ключа*](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) *отношения*, тогда другие потенциальные ключи называют *альтернативными ключами*.

Теоретически, все потенциальные ключи равно пригодны в качестве первичного ключа, на практике в качестве первичного обычно выбирается тот из потенциальных ключей, который имеет меньший размер (физического хранения) и/или включает меньшее количество атрибутов.