#  Введение

В современном обществе появление персональных компьютеров (ПК) и компьютерных технологий предвещало информационный переворот в деятельности человека. В результате чего была создана система баз данных (БД), основной целью которой, была сортировка и систематизация таким способом, чтобы можно было найти и обработать данные с помощью электронной вычислительной машины.

БД формируются и работают благодаря специальным программным обеспечениям (ПО), которые называются – системы управления базами данных (СУБД).

Возникновение СУБД это результат решения задач изменившиеся враз с вычислительными машинами.

Увеличение потока информации, объема и структурной сложности хранящихся данных, расширение круга пользователей информационных систем послужило к распространению более удобных и простых для усвоения реляционных БД.

Внутри отношений реляционной модели БД все данные логически структурированы. Они представляют собой именованные атрибуты (столбцы) данных, их отношения имеют имена, все строки включают в себя хотя бы одно значение каждого атрибута БД. Данная простота логической структуры позволяет превосходить данную модель над другими.

Однако информатизация пока не привела к желаемым результатам в образовательных системах, где информационные системы (ИС) призваны решать, прежде всего, задачи моделирования в интересах поддержки организационного управления, в том числе и управления учебным процессом.

Также осуществление каких-то процессов при работе на компьютере реально только при наличие математического описания данного процесса, что имеет отношение именно к процессам происходящих в базах данных.

В работе рассматривается возможность частичной программной и полной методической поддержки курса «Математические основы реляционных баз данных», присутствующей в учебных планах направления 09.03.03 «Прикладная информатика (уровень бакалавриата)», 44.03.01 «Педагогическое образование (уровень магистратуры)». Также данный курс может частично или полностью использоваться в курсах «Базы данных», «Проектирование баз данных», «Информационные системы» и т.д. Курс является практико-ориентированным, то есть все учебное аудиторное время посвящено практическим занятиям.

Объектом проводимого исследования является реляционная структура данных.

Предметом проводимого исследования является реляционная база данных.

Целью данного исследования является анализ математических основ языков манипулирования данными в реляционных информационных системах, а именно – с реляционной алгеброй и реляционными исчислением. Сформировать навыки решения задач при работе с реляционными базами данных.

Для реализации поставленной цели предполагается решение следующих задач:

- описать реляционную структуру данных, реляционные базы данных и способы манипулирования ими;

- рассмотреть основные операции реляционной алгебры и реляционного исчисления;

- привести примеры реляционной алгебры и реляционного исчисления;

- рассмотреть программное обеспечение RELOP для реляционной алгебры.

# Глава 1. Теоретические основы реляционных баз данных.

#  1.1. Реляционные базы данных.

Реляционная модель данных берет свое начало в работах сотрудника IBM компании, британского ученого Эдгара Франка Кодда. Еще в 60-х – 70-х годах он рассматривал возможности применения, хранения и способы представления различных табличных моделей данных. Первой работой в представленной теме считается изданная в июне 1970 года под названием «A Relational Model of Data for Large Shared Data Banks».

В этой работе обсуждены несоответствия общих сетевых моделей данных. Введена модель, которая основывается на n-арных отношениях, понятие универсального подъязыка данных и нормальная форма для этих отношений БД. Некоторые операции над отношениями (кроме логического вывода), обсуждаются и используются в проблемах избыточности и согласованности пользовательской модели.

В марте 1972 года Кодд публикует статью «Relational Completeness of Data Base Sublanguages». В ней он пытается привести теоретическую основу; определяет реляционную алгебру и реляционное исчисление; дает понятие реляционной полноты; представляет алгоритм для уменьшения произвольного отношения определяющего выражение (на основе исчисления) в семантически эквивалентное выражение реляционной алгебры; излагает некоторые мнения, касающиеся относительных достоинств исчисление-ориентированной против алгебра-ориентированных подъязыков данных с точки зрения поиска оптимального и весьма разборчивых схем авторизации.

Преимуществами реляционного представления является:

* улучшенный в некоторых отношениях в графовой или сетевой модели;
* для задач машинного представления база данных подает средства представления данных его естественной структуры, и без наложения каких-нибудь специальных структур;
* выше сказанное с одной стороны предоставляет основу необходимую языку данных высокого уровня, позволяющему получить максимальную независимость между программами и с другой стороны, организации данных и машинного представления;
* организовывает прочную основу для лечения выводимости, избыточности, и последовательность отношений.

Дальше приведем описание самой базы данных. Более распространенное принадлежит одному из крупнейших специалистов в области реляционной модели данных К. Дейту.

Прежде всего, существует собственно база данных – это совокупность данных, хранящихся во вторичной памяти (на дисках, барабанах или каком-либо другом носителе). Во-вторых, имеется набор обычных прикладных программ пакетной обработки, которые работают с этими данными, обрабатывая их обычными способами (выборка, обновление, включение, удаление). Дополнительно можно взаимодействовать с базой данных с удаленных терминалов, которые выполняют похожие функции. В-третьих, база данных является «интегрированной».[8] Это позволяет ей быть доступной многим пользователям. Это же в свою очередь предполагает, что каждый отдельный пользователь, который будет работать в режиме пакетной или оперативной обработки, связан только с небольшой частью этих данных.

Реляционная база данных (РБД) – это логическая модель данных, коллекция таблиц, которая организуется истинно реляционной модели. Ячейки таких таблиц имеют формальное описание.

Реляционная база данных – это набор отношений.

Реляционная модель данных – логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.[17]

Согласно работам Дейта, реляционная модель состоит из трех частей:

1. Структурной части.

Она описывает объекты, которые проверяются реляционной моделью. Принято считать, что единственной структурой данных, оперируемой в реляционной модели, являются нормализованные n-арные отношения.

1. Целостной части.

Она описывает ограничения особенного вида, работающего для каждых отношений в любых реляционных базах. Это общность сущностей и общность внешних ключей.

1. Манипуляционной части.

Которая описывает два равнозначных способа манипулирования реляционными данными – реляционную алгебру и реляционное исчисление.

Основным свойством реляционных структур данных является то, что связи между кортежами (строками) представлены исключительно значениями данных в столбцах, полученных из общего домена.[5]

# 1.2. Дюжина правил Э. Кодда.

Доктор Эдгар Кодд, после его обширного исследования в области реляционной модели систем баз данных, придумал двенадцать собственных правил, которые, по его словам, должна соблюсти база данных, чтобы рассматриваться как истинная реляционная база данных.

Эти правила могут быть применены в любой системе базы данных, которая управляет хранившими данными, используя только его реляционные возможности. Это - правило фундамент, которое действует как основа для всех других правил.

Изначально Эдгар Кодд предложил 12 (дюжину) правил, которым должна удовлетворять РБД.

1. *Правило информации*.

Вся информация в реляционной базе данных включая имена таблиц, имена столбцов представлены значениями в таблицах. Это простое представление проекта скоростей передачи данных и изучение. Пользовательская производительность улучшена, так как знание только одного языка необходимо, чтобы получить доступ ко всем данным, таким как описание таблицы и определения атрибута, ограничения целостности. Меры могут быть приняты, когда ограничения нарушены. Доступ к данным может быть ограничен. Все они информация также сохранены в таблицах.

1. *Правило гарантированного доступа*.

Для каждой части данных в реляционной базе данных, может быть получен доступ при помощи комбинации имени таблицы, значение первичного ключа, которое идентифицирует строку и имя столбца, которое идентифицировало ячейку. Пользовательская производительность улучшена, так как нет никакой потребности обращаться к использованию физических адресов указателей. Обеспечивает независимость данных.

1. *Система поддержки Null-значений*.

Реляционная система управления базами данных (РСУБД) обрабатывает записи, у которых есть неизвестные или неподходящие значения предопределенным способом. Кроме того, РСУБД различает нули, пробелы и провалов в записи вручную обрабатывает такие значения в согласованном порядке, который производит правильные ответы, сравнений и расчетов. Через ряд правил для обработки обнуляет, пользователи могут отличить результаты запросов, которые включают, NULLS, нули и пробелы. Даже при том, что правило не определяет то, что должно быть сделано в случае провалов, он определяет, что должна быть последовательная политика в обработке нулей.

1. *Динамический оперативный каталог на основе реляционной модели*.

Описание базы данных и в ее содержании является таблицами базы данных и поэтому может быть запрошено онлайн через язык манипулирования данными. Производительность администратора базы данных улучшена, так как изменения и дополнения к каталогу могут быть сделаны с теми же командами, которые используются, чтобы получить доступ к любой другой таблице. Все запросы и отчеты могут также быть сделаны как любая другая таблица.

1. *Правило исчерпывающегося подъязыка данных*.

РСУБД может поддерживать несколько языков. Но по крайней мере один из них должен позволить пользователю делать все следующее: определите таблицы и представления, запросите и обновите данные, установите ограничения целостности, установите полномочия и определите транзакции. Пользовательская производительность улучшена, так как есть всего один подход, который может использоваться для всех операций базы данных. В многопользовательской среде пользователь не должен волновать по поводу целостности данных такой вещи, которые будут заботиться системой. Кроме того, только пользователи с надлежащей авторизацией будут в состоянии получить доступ к данным.

1. *Правило обновления представления (вида, View)*.

Любое представление, которое теоретически обновляемо, может быть обновлено, используя РСУБД. Последовательность данных обеспечена, так как изменения, внесенные в представлении, переданы к базисной таблице и наоборот.

1. *Ввод, обновление и удаление данных на высоком уровне*.

РСУБД поддерживает вставки, обновление и удаление на уровне таблицы. Производительность улучшается, так как команды действуют на наборе данных, а не одну запись одновременно.

1. *Физическая независимость данных*.

Выполнение оперативных запросов и прикладных программ не затронуто изменениями в физическом доступе к данным и методах хранения. Администраторы базы данных могут внести изменения в физический доступ и метод хранения, которые улучшают производительность и не требуют изменений в прикладных программах или запросах. Здесь пользователь определил то, что он хочет потребность не, волнуются о том, как данные получены.

1. *Логическая независимость данных*.

Логические изменения в таблицах и представлениях таких столбцов, добавления/удаления или изменение полевых длин не должны требовать модификаций в программах или в формате оперативных запросов. База данных может измениться и вырасти, чтобы отразить изменения в действительности, не требуя вмешательства пользователя или изменений в приложениях. Например, добавление атрибута или столбца к базовой таблице не должно разрушать программы или интерактивную команду, которым не нравится новый атрибут.

1. *Независимость целостности*.

Как таблица/определение представления, ограничения целостности сохранены в интернет-каталоге и могут поэтому быть изменены, не требуя изменений в прикладных программах. Ограничения целостности, определенные для определенного РБД, должны быть определимыми в реляционном подъязыке данных и сохраняемых в каталоге. По крайней мере, целостность сущности и ссылочная целостность должны поддерживаться.

1. *Независимость распределени*я.

Прикладные программы и оперативные запросы не затронуты изменением в распределении физических данных. Совершенствование систем надежности начиная с прикладных программ будет работать, даже если программы и данные будут перемещены в различные сайты.

1. *Правило соблюдения правил*.

Если у РСУБД есть язык, который получает доступ к информации отчета за один раз, этот язык не должен использоваться, чтобы обойти ограничения целостности. Это необходимо для целостности данных.

По словам доктора Эдгара. Ф. Кодд, реляционная система управления базами данных должна быть в состоянии управлять базой данных полностью через ее реляционные возможности.

# 1.3. Основные понятия реляционных баз данных.

Дальше следует рассмотреть трактовку ключевых понятий реляционных баз данных, таких как: отношение, кортеж, атрибут, домен, тип данных, первичный ключ.

На представленном рис. 1.3.1 можно увидеть основную суть этих понятий рассмотрев пример отношения СЛУЖАЩИЕ, которые охватывают информацию о служащих какого-нибудь ПРЕДПРИЯТИЯ.



Рис. 1.3.1. Соотношение основных понятий реляционного подхода.

Заголовок отношения, входящий в базу данных имеет название схема реляционной базы данных.

Термин тип данных равнозначен как в языках программирования, так и в реляционной модели данных. В РБД поддерживаемые данные являются типизированными (то есть в них уже определен тип каждого поддерживаемого значения). Для установления типа данных существует три традиционных компонента:

* категоризация множества возможных значений данного типа;
* категоризация аргументов операций над значениями типа;
* определение способа внешнего представления значений типа данных.

В РБД дозволено хранить:

* символьные;
* числовые данные;
* битовые строки;
* специализированные числовые данные;
* специальные «темпоральные» данные (дата, время).

Для реляционных моделей требуется использовать простые данные, то есть, в реляционных операциях не обязательно учитывать внутреннюю структуру данных. Но при этом следует описать действия, которые возможно реализовать с данными как с единым целым. Простые типы данных также имеют название атомарные, и при этом они не имеют внутренней структурой. Подобные данные имеют название скаляры. Но следует описать действия, проводимые с данными как с единым целым. В данном случае не принципиален тип данных, который используется.

Таким образом, в немногих пост-реляционных СУБД, которые создаются пользователями, возможно работать со многими сложными типами данных. Тип данных стоит относить к множеству всех вероятных значений имеющегося типа, при этом домен больше похож на подмножество этих множеств.

Понятием, которому теория отношений придает особое значение, но которое не имеет, как нам кажется, укоренившегося термина в области обработки данных, является понятие *домена*. Оно достаточно сильно связано с понятием тип данных и при этом домен следует считать уточнением данного типа данных.

Домен – это семантическое понятие. Это пул значений, из которого извлекаются фактические значения, появляющиеся в столбце.[5]

Характерные свойства домена:

* в рамках базы данных обладает неповторимым именем;
* домен может быть установлен на другом домене или на некоем простом типе данных;
* он может обладать неким логическим условием, которое дает возможность описать подмножество данных, благоприятных для имеющегося домена;
* передает конкретную смысловую нагрузку.

Домены передают некоторую смысловую нагрузку. Они обуславливают сравнение и это их ведущее значение. Отличительной чертой домена от подмножества заключается в том, что он выражает семантику. Домены выражают разный смысл, даже если они совпадают как подмножества. Также они способствуют правильному моделированию предметной области.

Ограничение возможных значений домена не каждому домену позволяет иметь логические условия. И поэтому множество вероятных значений домена схож с множеством вероятных значений типа данных.

Таблицы, представленные на рис. 1.3.2 являются фактически особым случаем конструкции, известной в математике как *отношение* – термин, который имеет намного более точное определение, чем более привычный термин обработки данных «файл» или «таблица».[5] Понятие отношения относится к фундаментальным понятиям реляционных моделей данных, так как n-арное отношение является единственной родовой структурой данных, хранящихся в реляционной базе данных.[18]

Обычно отношение имеет вид и записывается – R.

 



Рис. 1.3.2. Пример данных в реляционной форме.

При всем этом, понятие отношение не является совсем точным. Это происходит по причине того, что для сохранения данных необходимо иметь ввиду достаточно много факторов, таких как: тип данных, значение этого типа и переменные, в которых хранятся значения, и не обойтись без вспомогательного понятия кортежа.

Характеристики отношений:

* в рамках базы данных обладает неповторимым именем;
* в ячейках хранятся исключительно неделимые (атомарные) отношения;
* атрибуты берут свои значения из доменов;
* очередность соблюдения атрибутов не принципиален;
* кортежи не обладают дубликатами;
* очередность соблюдения кортежей не принципиален.

*Заголовком (или схемой) отношения* является конечное множество упорядоченных пар вида <X,Y>, где X – это имя атрибута, а Y – имя некоторого базового типа или домена. И конечно же стоит учитывать, что имена атрибутов в заголовке отношения не должны совпадать.

Мощность множества кортежей отношения называют мощностью отношения.[18]

*Заголовок отношения* (рис.1.3.1) описывает декартово произведение доменов, на котором задано отношение.[18] При этом он постоянен, что позволяет не изменяться в процессе работы с БД. И мы получим новое отношение, если внесем изменение или удалим, или добавим какие-либо атрибуты, даже если они будут иметь прежние имена.

*Тело отношения* (рис.1.3.1) имеет вид множества кортежей, то есть подмножество декартового произведения доменов.[18] Таким образом именно оно и является отношением в математическом смысле слова. И тело отношения в отличии от заголовка уже способно изменяться в процессе работы с базами данных (кортежи можно изменять, добавлять, удалять).

*Значением отношения* (рис.1.3.1) называется пара множеств.

Строки подобных таблиц представленных на рис. 1.3.2 обычно называют *кортежами*, так как этот термин определен точнее, чем «строка» или «запись».[5]

В отношение не имеется идентичных кортежей. Это связано с тем, что тело отношения имеет вид множества кортежей, а множество не включает в себя неотличимые элементы.

Кортежи являются не упорядоченными (сверху вниз). Все потому, что как множество не включает в себя неразличимые элементы, так и множество не упорядочено, а отношение – это множество кортежей.

Эти причины не разрешают отождествлять отношения и таблицы, так в таблицах могут содержаться идентичные строки и они упорядочены.

Столбцы подобных таблиц представленных на рис. 1.3.2 называют *атрибутами*. Их имена обязаны быть единичны в границах отношения, но при этом они (имена атрибутов отношения) довольно нередко соответствуют именам подходящих доменов, что очень удобно, если атрибуты заголовка отношения располагаются на различных доменах. Соответственно стоит помнить, что это не ликвидирует отличия между понятиями домена и атрибута.

В связи с тем, что у атрибутов уникальные имена в пределах отношения они не являются упорядоченными (слева направо), он просто не имеет смысла. И это является еще одной причиной, почему не стоит отождествлять такие понятия как отношения и таблицы, по причине того, что в таблицах столбцы упорядочены.

Количество атрибутов в отношении называют *степенью (или –арностью) отношения.*[18]

Все значения атрибутов атомарны. Это следует из того, что лежащие в их основе атрибуты имеют атомарные значения. Это еще одно отличие отношений от таблиц – в ячейки таблиц можно поместить что угодно – массивы, структуры, и даже другие таблицы[6].

*Первичный ключ* – это поле или набор полей, однозначно идентифицирующих запись.

Важно иметь ввиду, что первичный ключ должен обладать уникальностью, т.е. не может существовать более одной записи с одинаковым его значением и при этом не должен содержать пустых значений. Следует выбирать первичный ключ постоянный атрибут, т.е. его значение не будет изменяться в течение времени существования экземпляра.

Первичный ключ следует использовать для того, чтобы установить связь между таблицами в РБД. Поэтому ему (первичному ключу) родительской таблицы соответствует внешний ключ (он содержит значения связанного с ним поля, которое является первичным ключом) дочерней таблицы. Первичный и внешний ключ обязаны быть одинакового типа, и значение во внешнем ключе возможно не уникальны, но не стоит их оставлять пустыми.

Говоря о базах данных, стоит рассмотреть и вопрос о связях между таблицами. Известны три типа связей:

1. Один к одному – это когда всякая запись родительской таблицы связана только с одной записью дочерней. Эту связь мы можем наблюдать в практике не часто, в отличии от связи один ко многим и реализуется путем определения уникального внешнего ключа. Базы данных с подобной связью не считаются полностью нормализованными и используют ее для того, чтобы таблица не была громоздкой из-за большого количества полей.
2. Один ко многим – это когда всякая запись родительской таблицы связана с одной или несколькими записями дочерней. Данная связь является самой распространенной в реляционных базах данных. Например, один покупатель может купить несколько книг в магазине, однако несколько покупателей не могут купить одну и туже книгу.
3. Многие ко многим – это когда несколько записей одной таблицы связаны с несколькими связями другой. При этой связи возникают сложности в том, чтобы определить какая запись одной таблицы соответствует выбранной записи другой таблицы, из-за чего становится неосуществима физическая реализация такой связи между соответствующими таблицами. Поэтому следует переопределять данную связь перед переходом к физической модели. Например, один автор может написать несколько книг и несколько авторов могут написать одну книгу.

При всем этом становится понятно, что реляционная структура данных достаточно доступна для понимания.

Также если посмотреть со стороны пользователя немаловажным считается набор операторов для работы с данными в реляционной форме.

Известно, что в реляционных базах данных (РБД) как централизованных, так и распределенных [12] для получения новой информации из отношений баз данных необходим язык манипулирования данными (ЯМД), который будет способен выполнять соответствующие операции над отношениями. Наиболее важной частью ЯМД является его раздел для формулировки запросов. Так как запросы в общем случае представляют собой произвольные функции над отношениями, необходимо решить вопрос о требуемой выразительности языка запросов. Для исследования этого вопроса были разработаны три типа теоретических языков: реляционная алгебра, реляционное исчисление с переменными-кортежами и реляционное исчисление с переменными-доменами. Языки этих трех типов были предложены Е.Ф. Коддом и служат теоретической основой для описания действий, выполняемых над отношениями.

Языки запросов первого типа – алгебраические языки – основаны на классической теории множеств (с некоторыми уточнениями) и позволяют выражать запросы средствами специализированных операторов, применяемых к отношениям.[15] Неформально реляционную алгебру описывают как (высокоуровневый) процедурный язык, благодаря которому сообщается системе управления базами данных, как нужно построить новое отношение из одного или нескольких существующих в базе данных отношений.

Языки запросов второго и третьего типов – языки исчисления – основаны на классическом логическом аппарате исчисления предикатов первого порядка и позволяют выражать свойства желаемого результата, фактически не указывая, как его получить.[13] Таким образом, неформально реляционное исчисление, представляет собой непроцедурный язык. Его используют для определения каким будет некоторое отношение, которое создано на основе одного или нескольких других отношений базы данных.

# 1.4. Достоинства и недостатки реляционных баз данных.

Следует начать с одного не мало важного недостатка, который относится в принципе к реляционным базам данных и к проектированию этих данных в частности – это недостаток литературы, которая помогла бы научить студентов решать задачи, то есть имея текст, который написан на естественном языке, записать запрос к базе данных на языке, который будет воспринимать сама СУБД.

В работе рассматриваются примеры (Глава 2) решения задач для имеющейся базы данных, которые частично можно проверить с помощью имеющегося программного обеспечения RELOP, работа которого описана в 3 главе.

Также стоит выделить некоторые недостатки реляционной модели:

* *строгость структур страдает негибкостью*;

Реляционная модель вынуждена задавать строгую однотипность объектов в таблице, хотя в реальности они носят разные характеры. Возникает часто вопрос, вынести объекты другого типа за рамки или же стоит расширить рамки? Хотя и пропущенные значения способствуют примириться с обязательным набором атрибутов.

* *сильная зависимость структур от данных;*

Работа структурных данных достаточно сильно зависит от состава (статистики) данных. Можно расположить в пределах реляционной структуры большое количество известных моделей данных, и при этом пользоваться различными способами реализации для разных массивов данных. Проще сказать, что выбор структуры зависит от набора данных (при изменении данных стоит изменять и структуры).

* *независимость кортежей оборачивается неэффективностью;*

Этот недостаток применим для простых массивов. БД и табличная форма не имеет смысла в том случае, если в предметной области имеются тесные связи и упорядоченность объектов.

* *самый медленный доступ к данным;*
* *Трудоемкость разработки.*

Хотя выше мы уже перечислили не мало недостатков реляционных моделей, но так же они обладают немалыми достоинствами, такими как:

* эта модель данных отображает информацию в наиболее простой для пользователя форме;
* основана на развитом математическом аппарате, который позволяет достаточно лаконично описать основные операции над данными;
* позволяет создавать языки манипулирования данными не процедурного типа;
* манипулирование данными на уровне выходной базы данных и возможность изменения.

Стоит отметить, что при немалой однородности данных табличное реляционное воплощение становится интересным и из подобного хранилища возможно выбирать данные по слоям, по районам, по признаку соседства и работать с ними в прикладных программах, которые нужны пользователю.

Слабо формализованные, неоднородные, нерегулярные данные являются непригодными для РБД.

# 1.5. Обзор реляционной алгебры

Механизм алгебры замкнут релятивно понятия отношения. Имеется ввиду то, что отношениями будут считаться итоговые высчитывания и выражения этой алгебры формируются над отношениями РБД. В итоге можно толковать как отношение всякое выражение.

Основная задача реляционной алгебры является снабдить совокупность операций по связям всех степеней (не обязательно бинарные) подходящих для выбора данных из реляционной базы данных.

Ключевые понятия, которые лежат в основе РБД - «отношение» и «связь».

В сущности, реляционной алгеброй является множество кортежей, как и отношение. В связи с этим, способы манипулирования отношениями следует использовать теоретико-множественные операции, работающих с помощью дополненных нестандартными для баз данных.[2] В итоге чего они применяют отношения как операндов, а результатом этого будут отношения.

В свое время Кодд определил восемь алгебраических операций, которые определяются естественным образом и падают на два класса: традиционный набор операций (или теоретико-множественные) и менее традиционных наборов операций по связям (или реляционные).

В теоретико-множественные операции над отношениями включены:

* Объединение (union);
* Пересечение (intersect);
* Разность (minus);
* Декартово произведение (times).

В реляционные операции над отношениями включены:

* Выборка или ограничение (where);
* Проекция (project);
* Соединение (join);
* Деление (divideby).

Для начало стоит подробнее рассмотреть теоретико-множественные операции, так как именно на теории множеств основывается реляционная теория.

*Объединением* двух совместных по типу (отношения, которые имеют равное множество атрибутов и определены на одних и техже доменах) отношений А и В называется отношение одинаковым заголовком, как и в отношениях А и В, и с телом, состоящим из множества всех кортежей, которые относятся к А или В, или обоим отношениям за исключением повторяющихся.[12]

Иначе говоря, объединение возвращает отношение, которое включает в себя все кортежи и они относятся к одному или обоим из заданных отношений.

Данная операция отношения А и В имеет вид:

A union B

В результате объединения заголовок будет совпадать с заголовками отношений, тело содержит в себе кортежи этих отношений.

Пример данной операции можно посмотреть в пункте 2.1 (Пример 5).

*Пересечение* двух подходящих по типу отношений А и В называется отношение с идентичным заголовком, как и в отношениях А и В, и с телом, состоящим из множества кортежей, которые относятся одновременно обоим отношениям А и В.[12]

Оно возвращает отношение, которое включает в себя кортежи, принадлежащие одновременно к двум заданным отношениям.

При пересечении, как и при объединении, операнды должны совмещаться по типу.

Операция пересечения отношения А и В обозначена как:

A intersect B

При пересечении заголовок совмещает заголовки изначальных отношений, а тело содержит кортежи, которые включены в оба этих отношения.

Пример данной операции можно посмотреть в пункте 2.1 (Пример 7).

*Разность* двух подходящих по типу отношений А и В – это отношение с идентичным заголовком, как и в отношениях А и В, и с телом, состоящим из множества всех кортежей, которые относятся к отношению А и не принадлежащих отношению В.[12]

Другими словами разность возвращает отношение, которое содержит все кортежи, принадлежащие только одному из заданных отношений.

Операнды владеют равное множество атрибутов и определены на равных доменах.

Разность имеет вид:

A minus B

В итоге заголовок будет совпадать с заголовками начальных отношений, в тело включаются кортежи первого отношения, которые не включены во второе.

Пример данной операции можно посмотреть в пункте 2.1 (Пример 6).

*Декартово произведение* двух отношений А и В, где А и В не имеют схожих имен атрибутов, определяется как отношение с заголовком, который представляет собой сцепление двух заголовков исходных отношений А и В, и телом, состоящим из множества всех кортежей t, таких, что t представляет собой сцепление кортежа а, принадлежащего отношению А, и кортежа b, принадлежащего отношению В. Кроме того:

* Степень результирующего отношения равна сумме степеней исходных отношений;
* Мощность результирующего отношения равна произведению мощностей исходных отношений. [12]

Иначе говоря, оно возвращает отношение, которое включает в себя все кортежи, представляющие собой сочетание двух кортежей, и они должны принадлежать заданным отношениям.

Одна из особенностей в том, что атрибуты изначальных отношений могут совпадать в именах, но при этом следует заранее переименовать один из атрибутов.

В этой операции заголовок организован соединением заголовков исходных отношений, тело – соединением кортежей этих отношений.

Операция выглядит так:

A times B

Декартовое произведение не нуждается в совместимости по типу исходных отношений. Для того чтобы уклониться от неоднозначности имен, которые возникают в результирующих отношениях, следует переименовать атрибут одного из отношений.

Применение этой операции обычно происходит в сочетании с реляционными операциями.

Пример данной операции можно посмотреть в пункте 2.1 (Пример 8)

Теперь стоит изучить операции, входящие в реляционные.

*Выборка или ограничение отношения* – это сокращенной название θ-выборки, где θ обозначает любой скалярный оператор сравнения (=, <>, >, >= и т.д.). θ-выборкой из отношения А по атрибутам X и Y (в этом порядке) [12]

A where X θ Y (обозначение выборки)

называется отношение, которое имеет заголовок отношения А, и тело, которое содержит множество кортежей t отношения А, и для них условие «X θ Y» принимает истинной значение.

Другими словами, выборка возвращает отношение, которое включает в себя кортежи заданного отношения, при этом они должны соответствовать заданным условиям.

Стоит иметь ввиду, что атрибуты X и Y должны определяться на одном домене, и для него оператор должен быть не бессмысленным.

Выражение выборки where может включать в себя не только простые сравнения, но и произвольное число логических сочетаний этих сравнений (т.е. может пользоваться такими логическими связками как and, or, not и ограничение в виде круглых скобок). Это выполнимо благодаря свойству замкнутости.

Пример данной операции можно посмотреть в пункте 2.1 (Пример 3).

*Операция проекции* применяется к одному отношению и определяет новое отношение, содержащее вертикальное подмножество отношения-операнда, т.е. подмножество, получаемое исключением всех атрибутов, не указанных в списке атрибутов, с последующим уничтожением кортежей-дубликатов из того, что осталось.[12]

Другими словами, она возвращает отношение, которое включает в себя кортежи заданного отношения из тех, что остались в отношении после исключения из него некоторых атрибутов.

Операция проекции выглядит так:

А[W,X, ,Y,Z]

где А – отношение, (W, X, ,Y,Z) – множество атрибутов.

Пример данной операции можно посмотреть в пункте 2.1 (Пример 4).

*Соединение* возвращает отношение, где кортежи – это сочетание двух кортежей, которые принадлежат двум определенным отношениям и обладают общими значениями для одного или нескольких общих атрибутов этих двух отношений (при этом в результате общие значения появляются только единожды).[12]

В общем случае имеет вид:

(A times B) where XθY

Результатом операции соединения из двух исходных отношений создается новое отношение.

Данная операция заимствует начало от операции декартова произведения. И она обладает разными типами, полезность которых зависит от ситуации.

Рассмотрим *тета-соединение.*

Пусть отношения А и В не имеют общих имен атрибутов и θ обозначает всякий скалярный оператор сравнения (=, <>, >, >=, <, <=). Тогда тета-соединением отношения А по атрибуту X c отношением В по атрибуту Y называется отношение с подобным заголовком, как и при декартовом произведении отношений А и В, и с телом, содержащим множество кортежей t, таких что t принадлежит этому декартовому произведению и вычисление условия «X θ Y» дает значение истина для этого кортежа. Атрибуты X и Y должны принадлежать одному домену, а операция должна быть не бессмысленной для этого домена.[12]

Степень тета-соединениея – это сумма степеней операндо-отношений А и В.

Тета-соединение имеет название эквисоединение (соединение по эквивалентности) в том случае, когда θ обозначает «равно».

Пример данной операции можно посмотреть в пункте 2.1 (Пример 9).

*Естественным соединением* называется соединение по эквивалентности двух отношений А и В, выполненное по всем общим атрибутам, из результатов которого исключается по одному экземпляру каждого общего атрибута.[12]

Степень соединения – это сумма степеней операндов-отношений А и В не включая число общих атрибутов.

Данная операция выглядит как:

A join B

Пример данной операции можно посмотреть в пункте 2.1 (Пример 10).

*Делением* отношений А и В называется отношение с заголовком {X} и телом, которое включает в себя множество всех кортежей {X:x}, таких что существует кортеж {X:x,Y:y}, принадлежащий отношению А для всех кортежей {Y:y}, принадлежащих отношению В.[12]

Другими словами, в результат входят X-значения из отношения А, и для них актуально Y-значения входящих в отношение А и они включают в себя Y-значения из отношения В.

Деление представляет собой:

A divideby B

Данная операция очень актуальна для таких запросов, где на обычном языке входит слово «все».

Пример данной операции можно посмотреть в пункте 2.1 (Пример 11).

Изучив операции реляционной алгебры можно сделать небольшие выводы. Точнее можно выделить основные операции, к ним относится: объединение, разность, декартово произведение, проекция и выборка. Другие же операции (пересечение, соединение и деление) можно получить благодаря основным. И всё-таки все они будут обладать самостоятельными значениями. Так же можно наблюдать то, что РБД как хранит данные, так и в удобном для пользователя выдает результат, который можно представить в виде несложной реляционной операции или их комбинаций.

# 1.6. Обзор реляционного исчисления

Хотя реляционная алгебра является полезным в анализе оценки запросов SQL фактически основан на другой язык запросов: реляционное исчисление (прикладное исчисление предикатов).

Исчислением предикатов называется формальный язык, который основан на ветви математической логики.

Исчислением можно воспользоваться при разработке запросов на всех базах данных, которые состоят из конечного набора отношений в простой нормальной форме.

Если сравнить реляционное исчисление и реляционную алгебру, то они будут являться двумя противоположными подходами. Их различие в том, что алгебра представляет собой набор операций с помощью которых сообщается системе как строить необходимые отношения в базе данных, тогда как в исчисление для получения результата следует уточнять свойства отношения, которое ищем без уточнения процедуры получения. Другими словами исчисление обеспечивает более процедурный путь к определению запросов.

Но это лишь внешние отличия (которые заключаются в том, что реляционная алгебра больше принадлежит к языкам программирования, а реляционное исчисление больше относится к естественному языку), так как в принципе эти понятия логически эквивалентны (т.е. каждому выражению в реляционной алгебре совпадает равное выражение в реляционном исчислении и наоборот), что подразумевает взаимно-однозначное соответствие.

Реляционная алгебра и реляционное исчисление логически эквивалентны по существу: для любого алгебраического выражения, существует эквивалентное выражение в исчислении, и наоборот. То есть запрос к базе данных можно сформулировать в одном языке, если и только если оно может быть выражено в другом. Этот результат известен как теорема Кодда.

Преимущество исчисления представляет собой упрощенную работу пользователя, т.е. уже не требуется самостоятельно составлять алгоритмы выполнения запросов. И помогает в этом программы СУБД, при условии, что она в меру интеллектуальна и сама составляет действующий алгоритм.

В основе реляционного исчисления находится механизм исчисления предикатов первого порядка. Оно является непроцедурным, имеет ту же выразительную силу, как реляционная алгебра, то есть реляционно полная.

Реляционное исчисление – это система обозначений для получения необходимого отношения в терминах данных отношений.[15]

Механизм исчисления замкнут относительно понятия отношения. Это означает, что формулы реляционного исчисления определяются над отношениями РБД и результатом вычисления также являются отношения. В результате любое выражение может пониматься как отношение.

Ключевые понятия – переменная с некоторой областью допустимых значений и правильно построенная формула (WFF). Они полагаются на такие понятия, как предикат, квантор и переменная.

Правильно построенная формула (Well-Formed Formula) служит для выражения условий, накладываемых на кортежные переменные. Ее основой являются простые условия, представляющие собой операции сравнения скалярных значений (значений атрибутов переменных или литерально заданных констант).[1]

Реляционное исчисление состоит из двух исчислений: реляционного исчисления кортежей и доменного реляционного исчисления, которые являются частью реляционной модели для баз данных и обеспечивают декларативный способ определить запросы базы данных.

Рассмотрим понятия реляционного исчисления с переменными кортежей и доменов. Они отличаются областью определения переменной.

В исчислении кортежей чтобы описать отношения стоит использовать переменные с допустимым значением, которым является кортеж некоторого отношение базы данных.

В исчислении доменов используют элементы домена, где значение некоторого домена являются допустимым значением переменных.

В своей работе более подробно будет рассмотрено реляционное исчисление с переменными кортежей и коротко описаны основные особенности исчисления доменов.

# 1.6.1. Обзор реляционного исчисления с переменными кортежей

Исчисление кортежа – это исчисление, которое было представлено Эдгаром Ф. Коддом как часть реляционной модели, чтобы обеспечить декларативный язык запросов базы данных для этой модели данных. Это сформировало вдохновение для языков запросов базы данных QUELL и SQL, которого последний, несмотря на то, что намного менее верный исходной реляционной модели и исчислению, является теперь фактическим стандартным языком запросов базы данных; диалект SQL используется почти каждой системой управления реляционными базами данных. Lacroix и Pirotte предложили доменное исчисление, которое ближе к логике первого порядка и которое показало, что оба из этих вычислений (а также алгебра отношений) эквивалентны в выразительном питании. Впоследствии, языки запросов для реляционной модели вызвали реляционным образом завершенные, если они могли бы выразить, по крайне мере, все эти запросы.

Переменная кортежей (или область значения) определяется с помощью оператора RANGE, является синтаксисом языка QUEL.

Этот язык некоторое время создавал серьезную конкуренцию языку SQL, он очень схож с подъязыком данных ALPHA, который представил Кодд, он основан на реляционном исчисление, но так и не был реализован, зато оказал большое влияние на создание языка QUEL.

Данный оператор имеет вид:

RANGE OF T IS X1, X2, ..., Xn,

где T – определяемая переменная кортежей, а Xi (i=1, 2, ..., n) – либо имя отношения, либо выражение исчисление кортежей.[12]

Предположим, что Xi есть отношение Ri (i=1,2,...n). В этом случае отношения R1, R2, ... , Rn должны обладать одинаковыми заголовками, то есть они будут совместимы по типу. И тогда значение переменной Т будет некоторым текущем кортежем в какое-то заданное время, то есть данная переменная кортежа будет меняться на объединении отношений. В том случае, если идентификатор выражений имеет вид одно именованное R отношение, то Т примет значение текущих кортежей этого отношения R.

Заметим, что когда используются кортежные переменные в формулах возможно указывать на значение атрибута переменной (подобное можно наблюдать например, в программировании на языке С, в котором можно указывать на значение пол структурной переменной). Например, для того, чтобы указать на значение атрибута А переменной Т, следует воспользоваться конструкцией Т.А.

Если компоненты кортежа возможно определить благодаря их порядковой позиции, то возможно указать на них благодаря индексной ссылки T[i].

В пункте 1.3 упоминалось о правильно построенной формуле WFF. В основе у нее находятся простые условия, которые имеют вид операций сравнения скалярных значений. Более сложные варианты включают в себя логические связки такие как: AND, NOT, OR и IF...THEN. Они строятся с учетом обычных приоритетом операций (NOT>AND>OR), способностью расстановки скобок, а также используются кванторы EXISTS и FORALL. Первый квантор - квантор существования, второй – квантором общности.

Предположим, что *f* –является формулой WFF, в ней фигурирует переменная *х*, тогда кванторы будут являться допустимыми правильно построенными формулами и иметь вид: EXISTS *x(f)* и FORALL *x(f)*.

Значением данных формул будет: EXISTS *x(f)* – «Решение формулы *f* для переменной *x* принимает значение истина, когда существует **хотябы одно значение** этого *x*». FORALL *x(f)* – «Решение формулы *f* **для каждого** **значения** переменной *x* принимает значение истина».

Переменные, которые включены в правильно построенную формулу, могут быть связанными или свободными. То есть, связанными переменная *х* будет если при построении WFF, которая имеет вид EXISTS *x(f)* или FORALL *x(f)* переменная *х* применяется сразу после квантора. А свободными будут переменные, которые входят в правильно построенную формулу и при ее построении не используются кванторы. Другими словами, свободные это те кортежные переменные, которые могут быть включены в итоговое отношение, при условии, что расчет правильно построенной формулы принимает значение *true* для некоторого набора значений этих свободных кортежных переменных. А связанные это те переменные, которые не видны за пределами минимальной WFF, которая связывает эту переменную. В момент вычисления значения подобной WFF используется вся область определения значений связанных переменных.

Формула EXISTS T (*f* (T)) будет определена равносильно правильно построенной формуле как:

*false* OR (*f* (T1)) OR ... OR (*f* (Tm))

если R – является отношением с переменной кортежа Т, которая меняется на отношении R, а *f* (T) является формулой, где эта переменная кортежа применяется. Это означает, что формально формируется EXISTS как повторяющееся OR (ИЛИ).

Приведем пример:

Пусть применяется отношение R:

R (X, Y, Z)

$$\begin{matrix}3&4&5\\3&4&6\\3&5&6\end{matrix}$$

Тогда следующие выражения будут иметь значения:

EXISTS T (T.Z > 2) : *true*

EXISTS T (T.Y > 5) : *false*

EXISTS T (T.X > 3 OR T.Z >=6) : *true*

Тем временем FORALL формируется как повторяющееся AND (И). То есть, если R, T и *f* (T) имеют значения как и для EXISTS, то уже формула

FORALL T (*f* (T))

будет определена равнозначно формуле:

true AND (*f* (T1)) AND ... AND (*f* (Tm)).

Приведем пример: При условии, что отношение R включает в себя такие же кортежи, как и в предыдущем примере, тогда следующие выражения будут иметь значения:

FORALL T (T.X > 3) : false

FORALL T (T.Y > 3) : true

FORALL T (T.X = 3 AND T.Z > 4) : true

Таким образом, правильно построенные формулы из отношений баз данных снабжают средства формулировки условия выборки. Для того, чтобы применять исчисление в работе с базами данных, необходим компонент target\_list, который имеет название целевой список, и будет выявлять набор и имена столбцов итогового отношения.

Такой целевой список создается благодаря целевым элементам, каждый из которых может принимать вид:

* T.A [AS X], где T – является именем свободной переменной, которая соответствует правильно построенной формуле WFF, A – это имя атрибута отношения, где сформулирована переменная T, а X является именем атрибута результата решения элементов target\_list;
* T, что равнозначно наличию подсписка T.A1, T.A2, ..., T.An, в котором A1,A2, ..., An включает имена всех атрибутов определяющего отношения;
* N=T.A; где N – является новое имя подходящего атрибута итогового отношения. И он необходим тогда, когда в WFF применяются несколько свободных переменных, у которых идентичные области определения.

*Выражением* реляционного исчисления кортежей называется конструкция вида TARGET\_LIST where WFF. Значением выражения является отношение, тело которого определяется WFF, а набор атрибутов и их имена – целевым списком.[12]

В пункте 1.6 говорилось, что реляционное исчисление кортежей описывается с помощью синтаксиса реального языка запросов QUEL.

Рассмотрим с помощью этого языка основные операции реляционной алгебры для реляционного исчисления с переменными кортежей. Тогда следующие операции будут иметь вид:

Операция объединения:

RANGE OF t IS A,B

И читается это так: нужно получить множество всех кортежей t, которые будут принадлежать отношению или А, или В.

Операция разности:

RANGE OF t IS A;

RANGE OF r IS B;

t where t ≠ r

Читается: рассмотрим множества кортежей t, которое принадлежит А и не принадлежит В.

Операция декартово произведение:

RANGE OF r IS B;

RANGE OF v IS B;

(t,r) where EXISTS u EXISTS v (t[1]=u[1] AND ... AND t[k]=u[k] AND r[1]=v[1] AND ... AND r[m]=v[m])

Читается: рассмотрим множество кортежей арности k+m (где k-арность отношения А, а m-арность отношения В), таких, что существует кортеж u, который принадлежат отношению А, существует кортеж v, который принадлежит отношению В, причем k первых компонентов всех итоговых кортежей создают компоненты кортежа u, а последующие m компонентов итогового кортежа создают компоненты кортежа v.

Операция проекции:

RANGE OF u IS A;

t where EXISTS u (t[1]=u[i1] AND ... AND t[n]=u[in])

Операция выборки:

RANGE OF t IS A;

t where XθY

Читается: следует получить множество кортежей t, таких, чтобы выполнялось условие XθY, где все операнды, означающие компонент i, замещаются на t[i].

Остальные же операции возможно получить благодаря приведенным операциям, как и в реляционной алгебре.

# 1.6.2. Обзор реляционного исчисления с переменными доменами

В информатике доменное реляционное исчисление (DRC) – является исчислением, представленным Michel Lacroix и Alain Pirotte в качестве внешнего языка запросов базы данных для реляционной модели данных.

Данный внешний язык пользуется такими же операторами как и исчисления кортежа, логическими соединительными словами ∧ (and), ∨ (or) и ¬ (not). Для связки переменных возможно использовать квантор существования (∃) и квантор всеобщности (∀).

Разница в этих исчислениях (кортежей и доменов) заключается в том, что переменные получают свои значения в границах домена, а не кортежей. Формальное отличие этих исчислений – наличие добавочного набора предикатов, которые позволяют выражать условия членства (как можно сказать).

Если *R* - это *n*-арное отношение с атрибутами *t1*, *t2*, ..., *tn*, то условие членства будет выглядеть так

*R* *(pair, pair,…*),

где каждая пара *pair* имеет вид *t:v*, при этом *v –* это либо литерально задаваемая константа, либо имя доменной переменной.[12]

*R* *(pair, pair,…*) приобретает значение true лишь в том случае, когда в отношение R входит кортеж, обладающий значениями нужных атрибутов.

Если *v –* постоянна, то на атрибут *t* ставится жесткое условие, которое не будет подчиняться текущим значениям доменных переменных; если же *v –* имя доменной переменной, тогда условие членства имеет возможность приобритать совершенно различные значения при различных значениях этой переменной.

Приведем пример, вычисление выражения

ПД (ПНОМ:'П1', ДНОМ:'Д1')

данное выражение принимает значение *true*, только в том случае, когда имеется кортеж ПНОМ в отношении ПД, который равен П1, и значением ДНОМ, равным Д1. Подобно, условие членства

ПД (ПНОМ:ПНОМА, ДНОМ:ДНОМА)

данное выражение принимает значение *true*, только в том случае, когда имеется кортеж ПНОМ в отношении ПД, который равнозначен данному значению переменной домена ПНОМА (какому бы он не принадлежал), и значением ДНОМ, равнозначным данному значению переменной домена ДНОМА (какому бы он не принадлежал).

В других случаях формулы и выражения выглядят подобными между исчислением доменов и исчислением кортежей. Различаются только связанные и свободные вхождения переменных доменов.

Дальше предположим, что имеются переменные доменов, которые имеют имя, созданные с помощью добавления цифр к нужным именам доменов. С условием, что все атрибуты обладают подобными именами в БД поставщиков и деталей, как и должный ему домен, кроме атрибутов ПФАМ и ДНАЗВ, для них подобный домен имеет название ИМЯ.

Приведем пример: выражение

(ПНОМ1) where ПОСТАВЩИК (ПНОМ:ПНОМА, ГОРОД:`Москва`)

Будет означать подмножество всех номеров поставщиков из города Москва.

Пользуясь обычным синтаксисом языка предикатов для реляционного исчисления с переменными доменов имеет вид:

{*x*1*x*2*…x*k | ψ (*x*1, *x*2, *…*, *xk*)},

где ψ- формула, обладающая тем свойством, что только ее свободные переменные доменов являются различными переменными *x*1, *x*2, *…*, *xk*.[12]

Среди языков, которые основываются на табличных формах можно выделить язык QBE (Query-by-Example). Среди остальных он более интересный и был первым, который основывается на реляционном исчислении доменов.

# 1.7. Превосходство исчисления над алгеброй.

Кодд считал, что одним из преимуществ, которые могут быть заявлены для алгебраического подхода является его свобода от кванторов. Тем не менее, исчисление, кажется, превосходит алгебры в четырех отношениях.

1. Непринужденность увеличения – Реляционная полнота представляет собой очень простую выборочную силу, которая в большинстве практических условиях должна была бы быть увеличена. Наиболее естественный тип улучшения - введение способности призыва любого конечного множества функций библиотеки, оставаясь в пределах алгебраической структуры или структуры исчисления (в зависимости от выбранного). Проверка альфа-выражений показывает три различных местоположений в пределах таких выражений для возможного вызова библиотечных функций. Первое из этих местоположений находятся в пределах целого списка, чтобы обеспечить некоторое преобразование восстановленного отношения. Во-вторых, замена термина соединения функцией со знаком истины одного или нескольких переменных кортежа (возможно внесенный в указатель). В-третьих, замена индексированной переменной кортежа в пределах термина соединения функцией (приводящий к целому числу или строке символов, например). Такие усовершенствования легко вписываются в структуру исчисления. В алгебраической структуре, однако, все такие функции должны быть приделаны в форме отображений от отношений к отношениям. Это дает начало многословиям.
2. Область применения для поисковой оптимизации – реляционное исчисление позволяет пользователю запрашивать данные, которые он желает по своим свойствам. Это является идеальной отправной точкой для поисковой оптимизации. Алгебра, с другой стороны, требует, чтобы пользователь сформулировал последовательность алгебраических операций, которые будут генерировать требуемые данные из базы данных отношений. Для отличных от очень простых запросов, свойства требуемых данных, как правило, получают скрытые в определенной последовательности операций (один из многих возможных), который выбирает пользователь.

Поэтому, начиная с алгебраического языка оригинала, у каждого есть выбор на местном уровне оптимизации выполнение каждой операции (очень ограниченная форма оптимизации) или решения трудной проблемы анализа последовательностей таких операций для обнаружения предполагаемого определяющие свойства требуемых данных.

1. Возможность авторизации – очень разборчивым разрешение должно быть основано на определении свойств данных, запрошенных пользователем, а не на конкретном алгоритме, указанном пользователем для получения этих данных. Аргументы в пункте 2, таким образом, применяются с равной силой к возможности авторизации системы.
2. Близость с естественным языком – Очевидно, что большинство пользователей не должны изучать либо реляционное исчисление или алгебру, чтобы взаимодействовать с базами данных. Тем не менее, с просьбой представить данные по своим свойствам намного более естественное , чем разработка конкретного алгоритма или последовательности операций для его извлечения. Таким образом ориентированный на исчисление язык обеспечивает хороший выходной язык для более ориентированного пользователями исходного языка.