13. Внутренняя организация реляционных СУБД

Особенности данных СУБД:

- 2 уровня представления - уровень, на котором производится непосредственное управление данными во внешней памяти (единицами являются физические сектора, кластеры, буферы или цепочки буферов ОЗУ) и языковой уровень - язык SQL (единица отношения, кортежи, атрибуты);

- наличие специальных отношений - каталогов, которые содержат информацию об именовании объектов, пути доступа к объекту и конкретные средства этих объектов (владелец, ключ индекса и т.д.);

- регулярность структур данных;

- необходимо поддерживать избыточность данных: резервные копии для поддержания безопасности работы (журнальные структуры) и восстановлении при сбоях.

В СУБД можно выделить несколько структур (объектов во внешней памяти):

- стоки отношений - основная часть БД, видимая пользователю;

- управляющие структуры - индексы:

1) по инициативе пользователя;

2) по инициативе верхнего уровня для быстрого доступа;

3) по инициативе системы нижнего уровня для размещения данных и автоматической поддержки индексов при изменении данных(!);

- журнальная информация, фиксирующая все изменения в БД;

- служебная информация для обеспечения работы нижнего уровня (о свободном дисковом пространстве).

**Хранение отношений**

существует 2 подхода к физической организации хранения:

1. кортежное хранение(!) - более распространённое;

2. поатрибутное хранение.

При 1 обеспечивается быстрый доступ к целому кортежу, но т.к. в разных кортежах значения атрибутов могут дублироваться, то такая БД может создать лишнюю дублируемую информацию, и если требуется часть кортежа, то может потребоваться лишнее обращение к памяти. При 2 столбец хранится с исключёнными дубликатами  требуется меньше внешней памяти; при операциях соединения есть преимущества; но требуются дополнительные операции, когда требуется целый кортеж.

**Индексы**  
главное назначение - обеспечить эффективный прямой доступ к кортежу по ключу. Обычно определяется для одного отношения и строится по значению какого-либо атрибута. Если этот атрибут является ключом отношения, то индекс должен быть уникальным, т.е. не содержать дублированных значений ключа.  
Помимо таких индексов рассматриваются индексы, связанные с несколькими отношениями (мультииндексы). В них ключ индекса является составным, по отдельным составляющим его составляются кортежи из разных отношений. Такие мультииндексы используются для сложных операций (эквисоединение).  
Общая идея организации - хранение упорядоченного списка ключей с привязкой к каждому ключу идентификатора кортежа или списка идентификаторов кортежа, если ключи индекса не уникальны.

**Журнальная информация**

**В журнале должны хранится все** изменения, происходящие в БД. С технической точки зрения журнал должен представлять файл последовательного доступа с записями переменного размера. Конкретная структура записей является частным делом каждой реализации. Журнал недоступен пользователю, а доступен самой СУБД. Из соображения надёжности существует несколько копий журнала на разных носителях.

5 Служебная информация

Служебная информация необходима для корректной работы с внешней памятью. В число файлов служебной информации входят:

- внутренние каталоги, которые описывают свойства объектов БД;

- описатели свободной и занятой памяти в терминах страницы;

- информация, которая связывает страницы одного отношения (если в одном файле внешней памяти располагается несколько отношений, то страницы каждого отношения должны быть связаны между собой).

14. Индексы БД

Индексы - это специальные структуры в базах данных, которые позволяют ускорить поиск и сортировку по определенному полю или набору полей в таблице, а также используются для обеспечения уникальности данных. Проще всего индексы сравнить с указателями в книгах. Если нет указателя, то нам придется просмотреть всю книгу, чтобы найти нужное место, а с указателем то же действие можно выполнить намного быстрее.

Обычно чем больше индексов, тем больше производительность запросов к базе данных. Однако при излишнем увеличении количества индексов падает производительность операций изменения данных (вставка/изменение/удаление), увеличивается размер БД, поэтому к добавлению индексов следует относиться осторожно.

Некоторые общие принципы, связанные с созданием индексов:

* индексы необходимо создавать для столбцов, которые используются в джойнах, по которым часто производится поиск и операции сортировки. При этом необходимо учесть, что индексы всегда автоматически создаются для столбцов, на которые накладывается ограничение primary key. Чаще всего они создаются и для столбцов с foreign key (в Access - автоматически);
* индекс обязательно в автоматическом режиме создается для столбцов, на которые наложено ограничение уникальности;
* лучше всего индексы создавать для тех полей, в которых - минимальное число повторяющихся значений и данные распределены равномерно. В Oracle есть специальные битовые индексы для столбцов с большим количеством повторяющихся значений, в SQL Server и Access такой разновидности индексов не предусмотрено;
* если поиск постоянно производится по определенному набору столбцов (одновременно), то в этом случае, возможно, есть смысл создать композитный индекс (только в SQL Server) - один индекс для группы столбцов;
* при внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу. В результате индекс может быть сильно фрагментирован, что сказывается на производительности. Периодически следует проверять степень фрагментации индексов и дефрагментировать их. При загрузке большого количества данных иногда есть смысл вначале удалить все индексы, а после завершения операции создать их заново;
* индексы можно создавать не только для таблиц, но и для представлений (только в SQL Server). Преимущества - возможность вычислять поля не в момент запроса, а в момент появления новых значений в таблицах.

В SQL Server предусмотрено два типа индексов: **кластерные** и **некластерные**.

**Кластерный индекс** в таблице может быть только один. Проще всего сравнить таблицу, на которую наложен такой индекс, с телефонным справочником: все записи в данной таблице упорядочены по кластерному индексу. Относиться к выбору поля для кластерного индекса следует очень осторожно - например, если в эту таблицу часто производится вставка данных, а кластерный индекс наложен не на поле с автоприращением, то вполне может получиться так, что нам часто придется вставлять новые записи в середину таблицы. Результат - большое количество операций page split, фрагментация таблицы и, как следствие, серьезное падение производительности (за счет фрагментации и за счет того, что само по себе page split - достаточно ресурсоемкая операция. По умолчанию кластерный индекс создается для поля первичного ключа, и, учитывая это, лучше делать первичный ключ числовым полем с автоприращением.

**Некластерный индекс** больше всего похож на указатель в конце книги. Для таблицы можно создавать таких индексов очень много (можно даже по нескольку для каждого столбца, но большой пользы это не приносит).

15. Связи между отношениями БД

Между отношениями БД могут быть установлены *связи* или ассоциации, показывающие, каким образом отношения соотносятся или взаимодействуют между собой.

*Степень связи* – количество типов отношений, которые охвачены данной связью.

Между отношениями предметной области могут иметь место:

−    бинарные связи (между двумя отношениями или между отношением  и ем же самим - рекурсивная связь);

−    тренарные связи (между тремя отношениями);

−    и в общем случае - n-арные связи.

На практике чаще всего встречаются связи со степенью два, то есть бинарные связи.

*Рекурсивная связ*ь – связь, в которой одни и те же отношения участвуют несколько раз в разных ролях.

         Каждый ЧЕЛОВЕК является сыном одного и только одного ЧЕЛОВЕКА;

         Каждый ЧЕЛОВЕК может являться отцом для одного или более ЛЮДЕЙ ("ЧЕЛОВЕКОВ").

Cреди бинарных связей существуют три фундаментальных вида связи:

−    один к одному (1:1);

−    один ко многим (1:M);

−    многие ко многим (M:N).

*Связь один к одному* (1:1) существует, когда один экземпляр одного отношения связан с единственным экземпляром другого отношения (студент может не "получать" стипендию, а может получать).

*Связь один ко многим* (1:M) существует, когда один экземпляр одного отношения связан с одним или более экземпляром другого отношения и каждый экземпляр второго отношения связан только с одним экземпляром первого отношения (квартира может пустовать, в ней может жить один или несколько жильцов).

Связь многие ко многим (М:N) существует, когда один экземпляр одного отношения связан с одним или более экземпляром другого отношения и каждый экземпляр второго отношения связан с одним или более экземпляром первого отношения.

В связи между данными, хранимыми в разных отношениях, в реляционной БД устанавливаются с помощью ключей.

*Первичный ключ* (primary key) - это столбец или группа столбцов, однозначно определяющие кортеж (запись). *Ключ* или *возможный ключ* – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр отношения. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать кортеж по оставшимся.

***Внешний ключ (***foreign key) - это столбец или группа столбцов, ссылающиеся на столбец или группу столбцов другой таблицы. Таблица, в которой находится внешний ключ называется *дочерней*. Таблица, на которую ссылается внешний ключ, называется *родительской таблицей*, а столбцы, на которые ссылается внешний ключ - родительским ключом. Родительский ключ должен быть первичным или уникальным ключом, значения же внешнего ключа могут повторяться хоть сколько раз. То есть с помощью внешних ключей поддерживаются связи "один ко многим". Типы данных (а в некоторых СУБД и размерности) соответствующих столбцов внешнего и родительского ключа должны совпадать.

И самое главное. Все значения внешнего ключа должны совпадать с каким-либо из значений родительского ключа.

16. Классификация ограничений целостности

Ограничения целостности можно классифицировать несколькими способами:

* По способам реализации.
* По времени проверки.
* По области действия.

Каждая система обладает своими средствами поддержки ограничений целостности. Различают два способа реализации:

Декларативная поддержка ограничений целостности.

Процедурная поддержка ограничений целостности.

**Декларативная поддержка ограничений целостности** заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности (если они имеются в СУБД) определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений (ASSERTION).

***Процедурная поддержка ограничений целостности*** заключается в использовании триггеров и хранимых процедур.

**Классификация ограничений целостности по времени проверки**

По времени проверки ограничения делятся на:

* Немедленно проверяемые ограничения.
* Ограничения с отложенной проверкой.

*Определение 6*. ***Немедленно проверяемые ограничения*** проверяются непосредственно в момент выполнения операции, могущей нарушить ограничение. Например, проверка уникальности потенциального ключа проверяется в момент вставки записи в таблицу. Если ограничение нарушается, то такая операция отвергается. Транзакция, внутри которой произошло нарушение немедленно проверяемого утверждения целостности, обычно откатывается.

***Ограничения с отложенной проверкой*** проверяется в момент фиксации транзакции оператором COMMIT WORK. Внутри транзакции ограничение может не выполняться. Если в момент фиксации транзакции обнаруживается нарушение ограничения с отложенной проверкой, то транзакция откатывается. Примером ограничения, которое не может быть проверено немедленно является ограничение из примера 1. Это происходит оттого, что транзакция, заключающаяся во вставке нового сотрудника в таблицу PERSON, состоит не менее чем из двух операций - вставки строки в таблицу PERSON и обновления строки в таблице DEPART. Ограничение, безусловно, неверно после первой операции и становится верным после второй операции.

#### Классификация ограничений целостности по области действия

По области действия ограничения делятся на:

* Ограничения домена
* Ограничения атрибута
* Ограничения кортежа
* Ограничения отношения
* Ограничения базы данных

***Ограничения целостности домена*** представляют собой ограничения, накладываемые только на допустимые значения домена. Фактически, ограничения домена обязаны являться частью определения домена

***Ограничение целостности атрибута*** представляют собой ограничения, накладываемые на допустимые значения атрибута вследствие того, что атрибут основан на каком-либо домене. Ограничение атрибута в точности совпадают с ограничениями соответствующего домена. Отличие ограничений атрибута от ограничений домена в том, что ограничения атрибута *проверяются*.

***Ограничения целостности кортежа*** представляют собой ограничения, накладываемые на допустимые значения *отдельного* кортежа отношения, и *не являющиеся* ограничением целостности атрибута. Требование, что ограничение относится к *отдельному* кортежу отношения, означает, что для его проверки *не требуется* никакой информации о других кортежах отношения.

***Ограничения целостности отношения*** представляют ограничения, накладываемые только на допустимые значения *отдельного* отношения, и *не являющиеся* ограничением целостности кортежа. Требование, что ограничение относится к отдельному отношению, означает, что для его проверки не требуется информации о других отношениях (в том числе не требуется ссылок *по внешнему ключу* на кортежи *этого же* отношения).

***Ограничения целостности базы данных*** представляют ограничения, накладываемые на значения двух или более связанных между собой отношений (в том числе отношение может быть связано само с собой).

17. Общая интерпретация реляционных операций

Основная идея реляционной алгебры состоит в том, что коль скоро отношения являются множествами, то средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для баз данных.

Набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса - теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

* объединения отношений;
* пересечения отношений;
* взятия разности отношений;
* прямого произведения отношений.

Специальные реляционные операции включают:

* ограничение (выборка) отношения;
* проекцию отношения;
* соединение отношений;
* деление отношений.

**Выборка**Возвращает отношение, содержащее все кортежи из заданного отношения, которые удовлетворяют указанным условиям. Операцию выборки также иногда называют операцией ограничения.

**Проекция**Возвращает отношение, содержащее все кортежи (подкортежи) заданного отношения, которые остались в этом отношении после исключения из него некоторых атрибутов.

**Произведение**Возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям.

**Объединение**Возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим.

**Пересечение**Возвращает отношение, содержащее все кортежи, которые принадлежат одновременно двум заданным отношениям.

**Разность**Возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух заданных отношений ине принадлежат второму.

**Соединение**Возвращает отношение, содержащее все возможные кортежи, которые представляют собой комбинацию атрибутов двух кортежей принадлежащих двум заданным отношениям, при условии, что в этих двух комбинируемых кортежах присутствуют одинаковые значения в одном или нескольких общих для исходных отношений атрибутах (причем эти общие значения в результирующем кортеже появляются один раз, а не дважды)

**Деление**Для заданных двух унарных отношений и одного бинарного возвращает отношение, содержащее все кортежи из первого унарного отношения, которые содержатся также в бинарном отношении и соответствуют всем кортежам во втором унарном отношении.

18. Транзакции в реляционных БД. Модели транзакций

Транзакция - это неделимая, с точки зрения воздействия на СУБД, последовательность операций манипулирования данными. Для пользователя транзакция выполняется по принципу "*все или ничего*", т.е. либо транзакция выполняется целиком и переводит базу данных из одного *целостного состояния* в другое *целостное состояние*, либо, если по каким-либо причинам, одно из действий транзакции невыполнимо, или произошло какое-либо нарушение работы системы, база данных возвращается в исходное состояние, которое было до начала транзакции (происходит откат транзакции). С этой точки зрения, транзакции важны как в многопользовательских, так и в однопользовательских системах. В однопользовательских системах транзакции - это логические единицы работы, после выполнения которых база данных остается *в целостном состоянии*. Транзакции также являются *единицами восстановления*данных после сбоев - восстанавливаясь, система ликвидирует следы транзакций, не успевших успешно завершиться в результате программного или аппаратного сбоя. Эти два свойства транзакций определяют атомарность (неделимость) транзакции. В многопользовательских системах, кроме того, транзакции служат для обеспечения *изолированной* работы отдельных пользователей - пользователям, одновременно работающим с одной базой данных, кажется, что они работают как бы в однопользовательской системе и не мешают друг другу.

Плоские, или традиционные, транзакции, характеризуются четырьмя классическими свойствами: атомарности, согласованности, изолированности, долговечности (прочности) — *ACID* (Atomicity, Consistency, *Isolation*, *Durability*). Иногда традиционные транзакции называют *ACID-транзакциями*. Упомянутые выше свойства означают следующее:

* *Свойство атомарности*(Atomicity) выражается в том, что транзакция должна быть выполнена в целом или не выполнена вовсе.
* *Свойство согласованности*(Consistency) гарантирует, что по мере выполнения транзакций данные переходят из одного согласованного состояния в другое — транзакция не разрушает взаимной согласованности данных.
* *Свойство изолированности*(*Isolation*) означает, что конкурирующие за доступ к базе данных транзакции физически обрабатываются последовательно, изолированно друг от друга, но для пользователей это выглядит так, как будто они выполняются параллельно.
* *Свойство долговечности*(*Durability*) трактуется следующим образом: если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах (даже в случае последующих ошибок).

Две модели транзакций, используемые в большинстве коммерческих СУБД: модель автоматического выполнения транзакций и модель управляемого выполнения транзакций, обе основаны на инструкциях языка SQL – COMMIT и ROLLBACK.

Автоматическое выполнение транзакций.

В стандарте ANSI/ISO зафиксировано, что транзакция автоматически начинается с выполнения пользователем или программой первой инструкции SQL. Далее происходит последовательное выполнение инструкций до тех пор, пока транзакция не завершается одним из двух способов:

• инструкцией COMMIT, которая выполняет завершение транзакции: изменения, внесенные в БД, становятся постоянными, а новая транзакция начинается сразу после инструкции COMMIT;

• инструкцией ROLLBACK, которая отменяет выполнение текущей транзакции и возвращает БД к состоянию начала транзакции, новая транзакция начинается сразу после инструкции ROLLBACK.

Такая модель создана на основе модели, принятой в СУБД DB2.

Управляемое выполнение транзакций.

Отличная от модели ANSI/ISO модель транзакций используется в СУБД Sybase, где применяется диалект Transact-SQL, в котором для обработки транзакций служат четыре инструкции:

• инструкция BEGIN TRANSACTION сообщает о начале транзакции, т.е. начало транзакции задается явно;

• инструкция COMMIT TRANSACTION сообщает об успешном выполнении транзакции, но при этом новая транзакция не начинается автоматически;

• инструкция SAVE TRANSACTION позволяет создать внутри транзакции точку сохранения и присвоить сохраненному состоянию имя точки сохранения, указанное в инструкции;

• инструкция ROLLBACK отменяет выполнение текущей транзакции и возвращает БД к состоянию, где была выполнена инструкция SAVE TRANSACTION (если в инструкции указана точка сохранения – ROLLBACK TO имя\_точки\_сохранения), или к состоянию начала транзакции.

19. Проблемы параллелизма транзакций

**Транзакция** – это логическая единица работы, которая лежит в основе проблемы параллелизма.

**Параллелизм**- возможность одновременной обработки в СУБД нескольких транзакций (параллельное выполнение транзакции). В связи с тем, что могут возникнуть проблемы в параллельных транзакциях, в системе существует механизм, позволяющий управлять транзакциями.

Главная задача управления параллелизмом заключается в следующем:− или все изменения, сделанные транзакцией, фиксируются и делаются постоянными (успешное завершение транзакции), или транзакция никак не влияет на состояние БД (ошибки при выполнении транзакции, откат транзакции).

**Основные проблемы, возникающие при параллелизме:**

**1. Пропавшие обновления.**Пусть выполняются параллельно две транзакции, каждая из которых выполняет извлечение некоторой записи, ее анализ и последующее обновление. В зависимости от того, какая из двух транзакций завершится позднее, будут утеряны изменения, сделанные ранее завершенной транзакцией.

**2. Чтение «грязных» данных**. Проблема незафиксированной зависимости. Здесь можно, в свою очередь, выделить две ситуации, каждая из которых определяется тем, что одна из двух параллельно выполняющихся транзакций завершается откатом.

а) Чтение грязных данных - Транзакция 1 прочитала данные, измененные транзакцией 2, но в последующем транзакция 2 отменила свои изменения.

б) Отсутствие потерянных изменений - Транзакция 1 изменяет данные после того, как эти данные были изменены транзакцией 2. Если транзакция 2 аннулирует свои изменения, состояние БД будет восстановлено таким, каким оно было перед началом выполнения транзакции 2. А значит, изменения, сделанные транзакцией 1, также будут аннулированы.

**3. Чтение несогласованных данных.** Проблема несовместимого анализа. Здесь также можно рассмотреть две ситуации:

а) Отсутствие неповторяющихся чтений - Транзакция 2 изменяет данные, прочитанные транзакцией 1. Если после этого транзакция 1 повторно читает те же данные, она получит уже другие результаты.

б) **Строки-призраки** Фантомы - Транзакция 1 считывает несколько записей, удовлетворяющих некоторому условию, после чего транзакция 2 добавляет новую запись, удовлетворяющую этому условию. Если после этого транзакция 1 повторно прочитает записи, удовлетворяющие тому же условию, она получит другой набор записей – в нем появились дополнительные записи-фантомы.

20. Способы и механизмы решений проблем параллелизма транзакций

**Стандартом SQL-92 предусмотрено 4 уровня изоляции:**

· незафиксированное чтения (READ UNCOMITTED);

· зафиксированное чтения (READ COMITTED);

· повторяющееся чтения (REPETABLE READ);

· сериализация (SERIALIZABLE).

Если в транзакции установлен определенный уровень изоляции, то для нее гарантировано невозникновение определенных эффектов.

**Сериализация транзакций** – последовательное выполнение. Это механизм выполнения транзакций по такому плану, когда результат совместного выполнения транзакций эквивалентен результату их последовательного выполнения.

Для обеспечения сериализации транзакций. применяются методы **«захвата»** и **«освобождения»** объекта. Транзакция захватывает объект – это приводит к его **блокировке** для другой транзакции. И освобождает его только при своем завершении. При этом совместимытолько захваты несколькими транзакциями на чтение объекта.

**Выделяют 2 основных режима захватов (блокировки):**

**1.**совместный режим **S-режим** (блокировка с доступом) – отвечающий за разделяемый захват объекта, чтобы можно было выполнять чтение.

**2.**монопольный **X-режим** (блокировка без взаимного доступа) – необходим для выполнения операции записи, удаления, модификации.

К методу управления параллелизмом относится **блокировка**. Она нужна для объекта, чтобы предотвратить его непредсказуемые изменения. Если проанализировать приведенные выше проблемы, можно увидеть, что конфликты возникают, когда транзакции параллельно пытаются выполнить операции чтения или записи данных.

Для устранения таких конфликтов используется механизм блокировок.

Протокол двухфазной блокировки устанавливает следующие правила наложения и снятия блокировок:

1. Перед выполнением каких-либо операций с некоторым объектом транзакция должна установить на этот объект соответствующую блокировку.

2. После снятия блокировки транзакция не должна накладывать никаких других блокировок.

Таким образом, транзакция, подчиняющаяся протоколу двухфазной блокировки, имеет две фазы: фаза наложения блокировки – обычно при выполнении соответствующей команды SQL, фаза снятия блокировки – обычно при выполнении команд COMMIT или ROLLBACK.

21. Язык реляционных баз данных SQL. Основные возможности и функции

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году и которая будет использоваться в текущем руководстве.

В зависимости от задачи, которую выполняет команда SQL, он может принадлежать к одному из следующих типов:

* **DDL** (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

* + **CREATE**: создает объекты базы данных (саму базу даных, таблицы, индексы и т.д.)
  + **ALTER**: изменяет объекты базы данных
  + **DROP**: удаляет объекты базы данных
  + **TRUNCATE**: удаляет все данные из таблиц
* **DML** (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды на выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

К этому типу относятся следующие команды:

* + **SELECT**: извлекает данные из БД
  + **UPDATE**: обновляет данные
  + **INSERT**: добавляет новые данные
  + **DELETE**: удаляет данные
* **DCL** (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:
  + **GRANT**: предоставляет права для доступа к данным
  + **REVOKE**: отзывает права на доступ к данным
* В SQL используются следующие основные типы данных, форматы которых могут несколько различаться для разных СУБД:
* INTEGER  - целое число (обычно до 10 значащих цифр и знак);
* SMALLINT  - "короткое целое" (обычно до 5 значащих цифр и знак);
* DECIMAL(p,q)  - десятичное число, имеющее p цифр (0 < p < 16) и знак; с помощью q задается число цифр справа от десятичной точки (q < p, если q = 0, оно может быть опущено);
* FLOAT  - вещественное число с 15 значащими цифрами и целочисленным порядком, определяемым типом СУБД;
* CHAR(n)  - символьная строка фиксированной длины из n символов (0 < n < 256);
* VARCHAR(n)  - символьная строка переменной длины, не превышающей n символов (n > 0 и разное в разных СУБД, но не меньше 4096);
* DATE  - дата в формате, определяемом специальной командой (по умолчанию mm/dd/yy); поля даты могут содержать только реальные даты, начинающиеся за несколько тысячелетий до н.э. и ограниченные пятым-десятым тысячелетием н.э.;
* TIME  - время в формате, определяемом специальной командой, (по умолчанию hh.mm.ss);
* DATETIME  - комбинация даты и времени;
* MONEY  - деньги в формате, определяющем символ денежной единицы ($, руб, ...) и его расположение (суффикс или префикс), точность дробной части и условие для показа денежного значения.

22. Реляционная алгебра. Операции объединения и пересечения отношений

*пределение 2*. ***Объединением***двух совместимых по типу отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif называется отношение с тем же заголовком, что и у отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif, и телом, состоящим из кортежей, принадлежащих или http://citforum.ru/database/dblearn/image104.gif, или http://citforum.ru/database/dblearn/image35.gif, или обоим отношениям.

Синтаксис операции объединения:

http://citforum.ru/database/dblearn/image144.gif

Замечание. Объединение, как и любое отношение, не может содержать одинаковых кортежей. Поэтому, если некоторый кортеж входит и в отношение http://citforum.ru/database/dblearn/image104.gif, и отношение http://citforum.ru/database/dblearn/image35.gif, то в объединение он входит один раз.

**Пример 2**. Пусть даны два отношения http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif с информацией о сотрудниках:

|  |  |  |
| --- | --- | --- |
| ***Табельный номер*** | **Фамилия** | **Зарплата** |
| *1* | Иванов | 1000 |
| *2* | Петров | 2000 |
| *3* | Сидоров | 3000 |

**Таблица 1 Отношение A**

|  |  |  |
| --- | --- | --- |
| ***Табельный номер*** | **Фамилия** | **Зарплата** |
| *1* | Иванов | 1000 |
| *2* | Пушников | 2500 |
| *4* | Сидоров | 3000 |

**Таблица 2 Отношение B**

Объединение отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif будет иметь вид:

|  |  |  |
| --- | --- | --- |
| **Табельный номер** | **Фамилия** | **Зарплата** |
| 1 | Иванов | 1000 |
| 2 | Петров | 2000 |
| 3 | Сидоров | 3000 |
| 2 | Пушников | 2500 |
| 4 | Сидоров | 3000 |

**Таблица 3 Отношение A UNION B**

Замечание. Как видно из приведенного примера, потенциальные ключи, которые были в отношениях http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif *не наследуются*объединением этих отношений. Поэтому, в объединении отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif атрибут "Табельный номер" может содержать дубликаты значений. Если бы это было не так, и ключи наследовались бы, то это противоречило бы понятию объединения как "объединение множеств". Конечно, объединение отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif имеет, как и любое отношение, потенциальный ключ, например, состоящий из всех атрибутов.

**Пересечение**

*Определение 3*. ***Пересечением***двух совместимых по типу отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif называется отношение с тем же заголовком, что и у отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif.

Синтаксис операции пересечения:

http://citforum.ru/database/dblearn/image145.gif

**Пример 3**. Для тех же отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif, что и в предыдущем примере пересечение имеет вид:

|  |  |  |
| --- | --- | --- |
| **Табельный номер** | **Фамилия** | **Зарплата** |
| 1 | Иванов | 1000 |

**Таблица 4 Отношение A INTERSECT B**

Замечание. Казалось бы, что в отличие от операции объединения, потенциальные ключи могли бы наследоваться пересечением отношений. Однако это не так. Вообще, *никакие реляционные операторы не передают результатирующему отношению никаких данных о потенциальных ключах*. В качестве причины этого можно было бы привести тривиальное соображение, что так получается более просто и симметрично - все операторы устроены одинаково. На самом деле причина более глубока, и заключается в том, что потенциальный ключ - семантическое понятие, отражающее различимость объектов предметной области. Наличие потенциальных ключей *не выводится*из структуры отношения, а явно задается для каждого отношения, исходя из его смысла. Реляционные же операторы являются *формальными*операциями над отношениями и выполняются одинаково, независимо от смысла данных, содержащихся в отношениях. Поэтому, реляционные операторы ничего не могут "знать" о смысле данных. Трактовка результата реляционных операций - дело пользователя.

23. Реляционная алгебра. Операция вычитания отношений. Декартово произведение

*Определение 4*. ***Вычитанием***двух совместимых по типу отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif называется отношение с тем же заголовком, что и у отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif, и телом, состоящим из кортежей, принадлежащих отношению http://citforum.ru/database/dblearn/image104.gif и не принадлежащих отношению http://citforum.ru/database/dblearn/image35.gif.

Синтаксис операции вычитания:

http://citforum.ru/database/dblearn/image146.gif

Пусть даны два отношения http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif с информацией о сотрудниках:

|  |  |  |
| --- | --- | --- |
| ***Табельный номер*** | **Фамилия** | **Зарплата** |
| *1* | Иванов | 1000 |
| *2* | Петров | 2000 |
| *3* | Сидоров | 3000 |

**Таблица 1 Отношение A**

|  |  |  |
| --- | --- | --- |
| ***Табельный номер*** | **Фамилия** | **Зарплата** |
| *1* | Иванов | 1000 |
| *2* | Пушников | 2500 |
| *4* | Сидоров | 3000 |

**Таблица 2 Отношение B**

Для тех же отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif, что и в предыдущем примере вычитание имеет вид:

|  |  |  |
| --- | --- | --- |
| ***Табельный номер*** | **Фамилия** | **Зарплата** |
| *2* | Петров | 2000 |
| *3* | Сидоров | 3000 |

**Таблица 5 Отношение A MINUS B**

**Декартово произведение**

*Определение 5*. ***Декартовым произведением***двух отношений http://citforum.ru/database/dblearn/image147.gif и http://citforum.ru/database/dblearn/image148.gif называется отношение, заголовок которого является ***сцеплением заголовков***отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif:

http://citforum.ru/database/dblearn/image149.gif,

а тело состоит из кортежей, являющихся ***сцеплением кортежей***отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif:

http://citforum.ru/database/dblearn/image150.gif,

таких, что http://citforum.ru/database/dblearn/image151.gif, http://citforum.ru/database/dblearn/image152.gif.

Синтаксис операции декартового произведения:

http://citforum.ru/database/dblearn/image153.gif

Пусть даны два отношения http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif с информацией о поставщиках и деталях:

|  |  |
| --- | --- |
| ***Номер поставщика*** | **Наименование поставщика** |
| *1* | Иванов |
| *2* | Петров |
| *3* | Сидоров |

**Таблица 6 Отношение A (Поставщики)**

|  |  |
| --- | --- |
| ***Номер детали*** | **Наименование детали** |
| *1* | Болт |
| *2* | Гайка |
| *3* | Винт |

**Таблица 7 Отношение B (Детали)**

Декартово произведение отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif будет иметь вид:

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер поставщика** | **Наименование поставщика** | **Номер детали** | **Наименование детали** |
| 1 | Иванов | 1 | Болт |
| 1 | Иванов | 2 | Гайка |
| 1 | Иванов | 3 | Винт |
| 2 | Петров | 1 | Болт |
| 2 | Петров | 2 | Гайка |
| 2 | Петров | 3 | Винт |
| 3 | Сидоров | 1 | Болт |
| 3 | Сидоров | 2 | Гайка |
| 3 | Сидоров | 3 | Винт |

24. Реляционная алгебра. Проекция отношения. Операция выборки

***Проекцией***отношения http://citforum.ru/database/dblearn/image104.gif по атрибутам http://citforum.ru/database/dblearn/image163.gif, где каждый из атрибутов принадлежит отношению http://citforum.ru/database/dblearn/image104.gif, называется отношение с заголовком http://citforum.ru/database/dblearn/image164.gif и телом, содержащим множество кортежей вида http://citforum.ru/database/dblearn/image165.gif, таких, для которых в отношении http://citforum.ru/database/dblearn/image104.gif найдутся кортежи со значением атрибута http://citforum.ru/database/dblearn/image166.gifравным http://citforum.ru/database/dblearn/image167.gif, значением атрибута http://citforum.ru/database/dblearn/image168.gif равным http://citforum.ru/database/dblearn/image94.gif, …, значением атрибута http://citforum.ru/database/dblearn/image169.gif равным http://citforum.ru/database/dblearn/image170.gif.

Синтаксис операции проекции:

http://citforum.ru/database/dblearn/image171.gif

Замечание. Операция проекции дает "*вертикальный срез*" отношения, в котором удалены все возникшие при таком срезе дубликаты кортежей.

**Пример 7**. Пусть дано отношение http://citforum.ru/database/dblearn/image104.gifс информацией о поставщиках, включающих наименование и месторасположение:

|  |  |  |
| --- | --- | --- |
| ***Номер поставщика*** | **Наименование поставщика** | **Город поставщика** |
| *1* | Иванов | Уфа |
| *2* | Петров | Москва |
| *3* | Сидоров | Москва |
| *4* | Сидоров | Челябинск |

**Таблица 11 Отношение A (Поставщики)**

Проекция http://citforum.ru/database/dblearn/image172.gif будет иметь вид:

|  |
| --- |
| **Город поставщика** |
| Уфа |
| Москва |
| Челябинск |

**Таблица 12 Отношение A[Город поставщика]**

25. Реляционная алгебра. Соединение отношений. Тэта-соединение, эквисоединение

#### Соединение

Операция соединения отношений, наряду с операциями выборки и проекции, является одной из наиболее важных реляционных операций.

Обычно рассматривается несколько разновидностей операции соединения:

* Общая операция соединения
* http://citforum.ru/database/dblearn/image156.gif-соединение (тэта-соединение)
* Экви-соединение
* Естественное соединение

Наиболее важным из этих частных случаев является операция естественного соединения. Все разновидности соединения являются частными случаями общей операции соединения.

#### Общая операция соединения

*Определение 8*. ***Соединением***отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif по условию http://citforum.ru/database/dblearn/image154.gif называется отношение

http://citforum.ru/database/dblearn/image173.gif

http://citforum.ru/database/dblearn/image154.gif представляет собой логическое выражение, в которое могут входить атрибуты отношений http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif и (или) скалярные выражения.

Таким образом, операция соединения есть результат последовательного применения операций декартового произведения и выборки. Если в отношениях http://citforum.ru/database/dblearn/image104.gif и http://citforum.ru/database/dblearn/image35.gif имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать.

#### Тэта-соединение

*Определение 9*. Пусть отношение http://citforum.ru/database/dblearn/image104.gif содержит атрибут http://citforum.ru/database/dblearn/image166.gif, отношение http://citforum.ru/database/dblearn/image35.gif содержит атрибут http://citforum.ru/database/dblearn/image174.gif, а http://citforum.ru/database/dblearn/image156.gif - один из операторов сравнения (http://citforum.ru/database/dblearn/image157.gif и т.д.). Тогда http://citforum.ru/database/dblearn/image156.gif-***соединением***отношения http://citforum.ru/database/dblearn/image104.gif по атрибуту http://citforum.ru/database/dblearn/image166.gif с отношением http://citforum.ru/database/dblearn/image35.gif по атрибуту http://citforum.ru/database/dblearn/image174.gif называют отношение

http://citforum.ru/database/dblearn/image175.gif

Это частный случай операции общего соединения.

Иногда, для операции http://citforum.ru/database/dblearn/image156.gif-соединения применяют следующий, более короткий синтаксис:

http://citforum.ru/database/dblearn/image176.gif

**Пример 8**. Рассмотрим некоторую компанию, в которой хранятся данные о поставщиках и поставляемых деталях. Пусть поставщикам и деталям присвоен некий статус. Пусть бизнес компании организован таким образом, что поставщики имеют право поставлять только те детали, статус которых не выше статуса поставщика (смысл этого может быть в том, что хороший поставщик с высоким статусом может поставлять больше разновидностей деталей, а плохой поставщик с низким статусом может поставлять только ограниченный список деталей, важность которых (статус детали) не очень высока).

|  |  |  |
| --- | --- | --- |
| *Номер поставщика* | Наименование поставщика | **X**  **(Статус поставщика)** |
| *1* | Иванов | 4 |
| *2* | Петров | 1 |
| *3* | Сидоров | 2 |

**Таблица 13 Отношение A (Поставщики)**

|  |  |  |
| --- | --- | --- |
| ***Номер детали*** | **Наименование детали** | **Y**  **(Статус детали)** |
| *1* | Болт | 3 |
| *2* | Гайка | 2 |
| *3* | Винт | 1 |

**Таблица 14 Отношение B (Детали)**

Ответ на вопрос "какие поставщики имеют право поставлять какие детали?" дает http://citforum.ru/database/dblearn/image156.gif-соединение http://citforum.ru/database/dblearn/image177.gif:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер поставщика** | **Наименование поставщика** | **X**  **(Статус поставщика)** | **Номер детали** | **Наименование детали** | **Y**  **(Статус детали)** |
| 1 | Иванов | 4 | 1 | Болт | 3 |
| 1 | Иванов | 4 | 2 | Гайка | 2 |
| 1 | Иванов | 4 | 3 | Винт | 1 |
| 2 | Петров | 1 | 3 | Винт | 1 |
| 3 | Сидоров | 2 | 2 | Гайка | 2 |
| 3 | Сидоров | 2 | 3 | Винт | 1 |

**Таблица 15 Отношение "Какие поставщики поставляют какие детали"**

**Экви-соединение**

Наиболее важным частным случаем http://citforum.ru/database/dblearn/image156.gif-соединения является случай, когда http://citforum.ru/database/dblearn/image156.gif есть просто равенство.

Синтаксис ***экви-соединения***:

http://citforum.ru/database/dblearn/image178.gif

**Пример 9**. Пусть имеются отношения http://citforum.ru/database/dblearn/image179.gif, http://citforum.ru/database/dblearn/image180.gif и http://citforum.ru/database/dblearn/image181.gif, хранящие информацию о поставщиках, деталях и поставках соответственно (для удобства введем краткие наименования атрибутов):

|  |  |
| --- | --- |
| ***Номер поставщика***  ***PNUM*** | **Наименование поставщика**  **PNAME** |
| *1* | Иванов |
| *2* | Петров |
| *3* | Сидоров |

**Таблица 16 Отношение P (Поставщики)**

|  |  |
| --- | --- |
| ***Номер детали***  ***DNUM*** | **Наименование детали**  **DNAME** |
| *1* | Болт |
| *2* | Гайка |
| *3* | Винт |

**Таблица 17 Отношение D (Детали)**

|  |  |  |
| --- | --- | --- |
| ***Номер поставщика***  ***PNUM*** | ***Номер детали***  ***DNUM*** | **Поставляемое количество**  **VOLUME** |
| *1* | *1* | 100 |
| *1* | *2* | 200 |
| *1* | *3* | 300 |
| *2* | *1* | 150 |
| *2* | *2* | 250 |
| *3* | *1* | 1000 |

**Таблица 18 Отношение PD (Поставки)**

Ответ на вопрос, какие детали поставляются поставщиками, дает экви-соединение http://citforum.ru/database/dblearn/image182.gif. На самом деле, т.к. в отношениях имеются одинаковые атрибуты, то требуется сначала переименовать атрибуты, а потом выполнить экви-соединение. Запись становится более громоздкой:

http://citforum.ru/database/dblearn/image183.gif

Обычно, такой сложной формой записи не пользуются. Но как бы то ни было, в результате имеем отношение:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер поставщика**  **PNUM1** | **Наименование поставщика**  **PNAME** | **Номер поставщика**  **PNUM2** | **Номер детали**  **DNUM** | **Поставляемое количество**  **VOLUME** |
| 1 | Иванов | 1 | 1 | 100 |
| 1 | Иванов | 1 | 2 | 200 |
| 1 | Иванов | 1 | 3 | 300 |
| 2 | Петров | 2 | 1 | 150 |
| 2 | Петров | 2 | 2 | 250 |
| 3 | Сидоров | 3 | 1 | 1000 |

**Таблица 19 Отношение "Какие детали поставляются какими поставщиками"**

Недостатком экви-соединения является то, что если соединение происходит по атрибутам с одинаковыми наименованиями (а так чаще всего и происходит!), то в результатирующем отношении появляется два атрибута с одинаковыми значениями. В нашем примере атрибуты PNUM1 и PNUM2 содержат дублирующие данные. Избавиться от этого недостатка можно, взяв проекцию по всем атрибутам, кроме одного из дублирующих. Именно так действует естественное соединение.

**Экви-соединение**

Наиболее важным частным случаем http://citforum.ru/database/dblearn/image156.gif-соединения является случай, когда http://citforum.ru/database/dblearn/image156.gif есть просто равенство.

Синтаксис ***экви-соединения***:

http://citforum.ru/database/dblearn/image178.gif

**Пример 9**. Пусть имеются отношения http://citforum.ru/database/dblearn/image179.gif, http://citforum.ru/database/dblearn/image180.gif и http://citforum.ru/database/dblearn/image181.gif, хранящие информацию о поставщиках, деталях и поставках соответственно (для удобства введем краткие наименования атрибутов):

|  |  |
| --- | --- |
| ***Номер поставщика***  ***PNUM*** | **Наименование поставщика**  **PNAME** |
| *1* | Иванов |
| *2* | Петров |
| *3* | Сидоров |

**Таблица 16 Отношение P (Поставщики)**

|  |  |
| --- | --- |
| ***Номер детали***  ***DNUM*** | **Наименование детали**  **DNAME** |
| *1* | Болт |
| *2* | Гайка |
| *3* | Винт |

**Таблица 17 Отношение D (Детали)**

|  |  |  |
| --- | --- | --- |
| ***Номер поставщика***  ***PNUM*** | ***Номер детали***  ***DNUM*** | **Поставляемое количество**  **VOLUME** |
| *1* | *1* | 100 |
| *1* | *2* | 200 |
| *1* | *3* | 300 |
| *2* | *1* | 150 |
| *2* | *2* | 250 |
| *3* | *1* | 1000 |

**Таблица 18 Отношение PD (Поставки)**

Ответ на вопрос, какие детали поставляются поставщиками, дает экви-соединение http://citforum.ru/database/dblearn/image182.gif. На самом деле, т.к. в отношениях имеются одинаковые атрибуты, то требуется сначала переименовать атрибуты, а потом выполнить экви-соединение. Запись становится более громоздкой:

http://citforum.ru/database/dblearn/image183.gif

Обычно, такой сложной формой записи не пользуются. Но как бы то ни было, в результате имеем отношение:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер поставщика**  **PNUM1** | **Наименование поставщика**  **PNAME** | **Номер поставщика**  **PNUM2** | **Номер детали**  **DNUM** | **Поставляемое количество**  **VOLUME** |
| 1 | Иванов | 1 | 1 | 100 |
| 1 | Иванов | 1 | 2 | 200 |
| 1 | Иванов | 1 | 3 | 300 |
| 2 | Петров | 2 | 1 | 150 |
| 2 | Петров | 2 | 2 | 250 |
| 3 | Сидоров | 3 | 1 | 1000 |

**Таблица 19 Отношение "Какие детали поставляются какими поставщиками"**

Недостатком экви-соединения является то, что если соединение происходит по атрибутам с одинаковыми наименованиями (а так чаще всего и происходит!), то в результатирующем отношении появляется два атрибута с одинаковыми значениями. В нашем примере атрибуты PNUM1 и PNUM2 содержат дублирующие данные. Избавиться от этого недостатка можно, взяв проекцию по всем атрибутам, кроме одного из дублирующих. Именно так действует естественное соединение.

**Экви-соединение**

Наиболее важным частным случаем http://citforum.ru/database/dblearn/image156.gif-соединения является случай, когда http://citforum.ru/database/dblearn/image156.gif есть просто равенство.

Синтаксис ***экви-соединения***:

http://citforum.ru/database/dblearn/image178.gif

**Пример 9**. Пусть имеются отношения http://citforum.ru/database/dblearn/image179.gif, http://citforum.ru/database/dblearn/image180.gif и http://citforum.ru/database/dblearn/image181.gif, хранящие информацию о поставщиках, деталях и поставках соответственно (для удобства введем краткие наименования атрибутов):

|  |  |
| --- | --- |
| ***Номер поставщика***  ***PNUM*** | **Наименование поставщика**  **PNAME** |
| *1* | Иванов |
| *2* | Петров |
| *3* | Сидоров |

**Таблица 16 Отношение P (Поставщики)**

|  |  |
| --- | --- |
| ***Номер детали***  ***DNUM*** | **Наименование детали**  **DNAME** |
| *1* | Болт |
| *2* | Гайка |
| *3* | Винт |

**Таблица 17 Отношение D (Детали)**

|  |  |  |
| --- | --- | --- |
| ***Номер поставщика***  ***PNUM*** | ***Номер детали***  ***DNUM*** | **Поставляемое количество**  **VOLUME** |
| *1* | *1* | 100 |
| *1* | *2* | 200 |
| *1* | *3* | 300 |
| *2* | *1* | 150 |
| *2* | *2* | 250 |
| *3* | *1* | 1000 |

**Таблица 18 Отношение PD (Поставки)**

Ответ на вопрос, какие детали поставляются поставщиками, дает экви-соединение http://citforum.ru/database/dblearn/image182.gif. На самом деле, т.к. в отношениях имеются одинаковые атрибуты, то требуется сначала переименовать атрибуты, а потом выполнить экви-соединение. Запись становится более громоздкой:

http://citforum.ru/database/dblearn/image183.gif

Обычно, такой сложной формой записи не пользуются. Но как бы то ни было, в результате имеем отношение:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер поставщика**  **PNUM1** | **Наименование поставщика**  **PNAME** | **Номер поставщика**  **PNUM2** | **Номер детали**  **DNUM** | **Поставляемое количество**  **VOLUME** |
| 1 | Иванов | 1 | 1 | 100 |
| 1 | Иванов | 1 | 2 | 200 |
| 1 | Иванов | 1 | 3 | 300 |
| 2 | Петров | 2 | 1 | 150 |
| 2 | Петров | 2 | 2 | 250 |
| 3 | Сидоров | 3 | 1 | 1000 |

**Таблица 19 Отношение "Какие детали поставляются какими поставщиками"**

Недостатком экви-соединения является то, что если соединение происходит по атрибутам с одинаковыми наименованиями (а так чаще всего и происходит!), то в результатирующем отношении появляется два атрибута с одинаковыми значениями. В нашем примере атрибуты PNUM1 и PNUM2 содержат дублирующие данные. Избавиться от этого недостатка можно, взяв проекцию по всем атрибутам, кроме одного из дублирующих. Именно так действует естественное соединение.