Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра информационных систем

ТЕОРИЯ БАЗ ДАННЫХ

Конспект лекций
для студентов специальности 5В060200 -Информатика

Алматы 2014

Составитель: А.Г. Бычков. Теория баз данных. Конспект лекций для студентов специальности 5В060200 –Информатика. - Алматы: АУЭС, 2014.- 57 с.

Настоящий конспект лекций составлен в соответствии с программой курса «Теория баз данных» для студентов специальности 5В060200 – «Информатика».

Ил.39, библиогр. – 6 назв.

Рецензент: доцент кафедры РТ А.А.Куликов

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2014 г.

Содержание

Введение 4
1 Лекция №1. Назначение и основные компоненты системы баз данных 5
2 Лекция №2. Модели БД 8
3 Лекция №3. Реляционная модель и ее характеристики. Целостность в реляционной модели 11
4 Лекция №4. Реляционная алгебра 17
5 Лекция №5. Язык SQL 24
6 Лекция №6. Проектирование реляционной БД. Функциональные зависимости 29
7 Лекция №7 Процедура нормализации 32
8 Лекция №8. Нормальные формы отношений 36
9 Лекция №9. Проектирование БД с использованием метода сущность-связь 41
10 Лекция №10. Правила формирования отношений 46
11 Лекция №11. Физическая организация базы данных: структуры хранения и методы доступа 50
12 Лекция №12. Защита баз данных 53
Список литературы 57

Введение

Исторически сложившееся развитие вычислительных систем обусловило необходимость хранения в электронном (машиночитаемом) виде все большего количества информации. Сложности, возникшие при решении на практике задач структурированного хранения и эффективной обработки возрастающих объемов информации, стимулировали исследования в соответствующих областях. Задачи хранения и обработки данных были формализованы. Была создана теоретическая база для решения задач такого класса, результатом реализации на практике которой стали системы, предназначенные для организации обработки, хранения и предоставления доступа к информации. Позже такие системы стали называть системами баз данных.

По мере развития систем баз данных, менялись принципы организации данных в них: первоначально данные представлялись на основе иерархической, а впоследствии сетевой модели. В конце 1970-х – начале 1980-х годов начали появляться первые реляционные продукты. В настоящее время системы баз данных на основе реляционной модели занимают лидирующее положение. Кроме того, многие коммерческие реляционные системы приобретают объектно-ориентированные черты. На основании этого можно предположить, что в будущем объектно-ориентированные системы будут постепенно вытеснять реляционные.

Целью курса «Теория баз данных» является изучение основ методологии проектирования баз данных: концептуальному, логическому и физическому проектированию на примере иерархических, сетевых и реляционных баз данных; знакомство с основами языков описания, манипулирования базами данных, а также языков создания; формирование представлений об архитектуре, основных подходах к проектированию и областях применения систем баз данных, о перспективных моделях баз данных.

В настоящих методических указаниях рассматриваются основные понятия, история развития и базовые модели теории баз данных; реляционная модель баз данных, основные методы проектирования; реляционная алгебра и основы языка SQL.

1 Лекция №1. Назначение и основные компоненты системы баз данных

Цель лекции: изучение назначения, основных компонентов и функций системы баз данных.

Содержание лекции: понятие БД и СУБД; уровни абстракции в СУБД; представления; функции СУБД.

1.1 Понятие БД и СУБД

Система баз данных – это компьютеризированная система, основная задача которой – хранение информации и предоставление доступа к ней по требованию.

Система баз данных включает в себя:

a) данные, непосредственно сохраняемые в базе данных;

b) аппаратное обеспечение;

c) программное обеспечение;

d) пользователей;

e) прикладные программисты;

f) конечные пользователи;

g) администраторы баз данных.

Данные в базе данных являются интегрированными и, как правило, общими. Понятие интегрированных данных подразумевает возможность представления базы данных как объединение нескольких отдельных файлов данных, полностью или частично не перекрывающихся.

К аппаратному обеспечению системы относятся накопители для хранения информации, вместе с устройствами ввода-вывода, контролерами устройств и т.д.; вычислительная техника, используемая для поддержки работы ПО системы.

Программное обеспечение является промежуточным слоем между физической базой данных и пользователями системы и называется диспетчером базы данных или системой управления базами данных (СУБД). Все запросы пользователей обрабатываются СУБД.

Таким образом, СУБД – это специализированное программное обеспечение, предоставляющее пользователю базы данных возможность работать с ней, не вникая в детали хранения информации на уровне программного обеспечения.

Прикладные программисты отвечают за написание прикладных программ, использующих базу данных.

Конечные пользователи работают с базой данных непосредственно, через рабочую станцию или терминал. Конечный пользователь может получить доступ к базе данных, используя соответствующее прикладное ПО.

Администраторы базы данных – технические специалисты, осуществляющие создание баз данных, технический контроль работы СУБД

и др. операции.

База данных (БД) состоит из некоторого набора постоянных данных, которые используются прикладными системами какого-либо предприятия. Использование баз данных для хранения информации позволяет организовать централизованное управление данными, что обеспечивает следующие преимущества:

a) возможность сокращения избыточности;
b) возможность устранения (до некоторой степени) противоречивости;
c) возможность общего доступа к данным;
d) возможность соблюдения стандартов;
e) возможность введения ограничений для обеспечения безопасности;
f) возможность обеспечения целостности данных;
g) возможность обеспечения независимости данных.

Поскольку программное обеспечение отделяется от данных, хранимых БД, изменения, вносимые в структуру БД, в большинстве случаев не приводят к необходимости внесения радикальных изменений в программное обеспечение.

1.2 Уровни абстракции в СУБД

Существует 3 уровня архитектуры СУБД:

a) внутренний уровень - наиболее близкий к физическому хранению, он связан со способами хранения информации на физических устройствах хранения;

b) внешний уровень - наиболее близкий к пользователям, он связан со способами представления данных для отдельных пользователей;

c) концептуальный уровень - является промежуточным между двумя первыми (этот уровень связан с обобщенными представлениями пользователей, в отличие от внешнего уровня, связанного с индивидуальными представлениями пользователей).

1.3 Представления

Соответственно трем уровням архитектуры выделяют три уровня абстракции данных в СУБД.

1.3.1 Внешний уровень – внешнее представление.

Внешний уровень - индивидуальный уровень пользователя. Каждый пользователь имеет свой язык общения с СУБД. Для программиста - это какой-либо язык программирования, для пользователя - язык запросов или язык, основанный на формах и меню.

Представление отдельного пользователя о БД на внешнем уровне архитектуры называют внешним представлением. Таким образом, внешнее представление - это содержимое БД, каким его видит отдельный пользователь. В общем случае внешнее представление состоит из множества

экземпляров каждого типа внешней записи, которые не обязательно совпадают с хранимыми записями. Каждое внешнее представление определяется средствами внешней схемы, которая, в основном, состоит из определений каждого типа записей во внешнем представлении.

1.3.2 Концептуальный уровень – концептуальное представление.

Концептуальное представление - это представление всей информации БД в несколько более абстрактной форме по сравнению с физическим способом хранения данных. Концептуальное представление представляет данные такими, какими они есть на самом деле, а не такими, какими их вынужден видеть пользователь в рамках определенного языка. Концептуальное представление состоит из множества экземпляров каждого типа концептуальной записи. Концептуальное представление определяется средствами концептуальной схемы, которая состоит из определений каждого типа концептуальных записей. Концептуальное представление, таким образом, обеспечивает независимость данных от способа их хранения.

1.3.3 Внутренний уровень – внутреннее представление.

Внутреннее представление - это представление нижнего уровня всей БД. Оно состоит из множества экземпляров каждого типа внутренней записи. Внутренняя запись соответствует хранимой записи. Внутреннее представление не связано с физическим уровнем и в нем не рассматриваются физические записи. Внутреннее представление предполагает существование бесконечного линейного адресного пространства. Подробности отображения этого пространства на физические устройства хранения не включены в общую архитектуру из-за сильной зависимости от системы.

Внутреннее представление описывается с помощью внутренней схемы, которая описывается с помощью внутреннего языка определения данных.

Между тремя уровнями представлений имеются два уровня отображений. Отображения концептуального уровня на внутренний и внешнего уровня на концептуальный. Отображения сохраняют независимость данных случае внесения в структуру БД изменений.

1.4 Функции СУБД

1. Определение данных. СУБД должна допускать определения данных (внешние схемы, концептуальную и внутреннюю схемы, соответствующие отображения). Для этого СУБД включает в себя языковый процессор для различных языков определений данных.

2. Обработка данных. СУБД должна обрабатывать запросы пользователя на выборку, а также модификацию данных. Для этого СУБД включает в себя компоненты процессора языка обработки данных.

3. Безопасность и целостность данных. СУБД должна контролировать запросы и пресекать попытки нарушения правил безопасности и целостности.

4. Восстановление данных и дублирование. СУБД должна обеспечить

восстановление данных после сбоев.

5. Словарь данных. СУБД должна обеспечить функцию словаря данных. Сам словарь можно считать системной базой данных, которая содержит данные о данных пользовательской БД, т.е. содержит определения других объектов системы. Словарь интегрирован в определяемую им БД и поэтому содержит описание самого себя.

2 Лекция №2. Модели БД

Цель лекции: изучение различных моделей баз данных.

Содержание лекции: обзор ранних (дореляционных) СУБД; иерархическая модель; сетевая модель; основные достоинства и недостатки ранних СУБД.

2.1 Обзор ранних (дореляционных) СУБД

Рассмотрим некоторые наиболее общие характеристики ранних систем.

Эти системы активно использовались в течение многих лет, дольше, чем используется многие из реляционных СУБД. На самом деле некоторые из ранних систем используются даже в наше время, накоплены громадные базы данных, и одной из актуальных проблем информационных систем является использование этих систем совместно с современными системами.

Все ранние системы не основывались на каких-либо абстрактных моделях. Понятие модели данных фактически вошло в обиход специалистов в области БД только вместе с реляционным подходом.

В ранних системах доступ к БД производился на уровне записей. Пользователи этих систем осуществляли явную навигацию в БД, используя языки программирования, расширенные функциями СУБД. Интерактивный доступ к БД поддерживался только путем создания соответствующих прикладных программ с собственным интерфейсом.

Навигационная природа ранних систем и доступ к данным на уровне записей заставляли пользователя самого производить всю оптимизацию доступа к БД, без какой-либо поддержки системы.

После появления реляционных систем большинство ранних систем было оснащено «реляционными» интерфейсами. Однако в большинстве случаев это не сделало их по-настоящему реляционными системами, поскольку оставалась возможность манипулировать данными в естественном для них режиме.

2.2 Иерархическая модель

Типичным представителем (наиболее известным и распространенным) является Information Management System (IMS) фирмы IBM. Первая версия появилась в 1968 г. До сих пор поддерживается много баз данных, что создает существенные проблемы с переходом как на новую технологию БД, так и на

новую технику. Иерархическая БД состоит из упорядоченного набора деревьев; более точно, из упорядоченного набора нескольких экземпляров одного типа дерева.

Тип дерева (см. рисунок 2.1) в целом представляет собой иерархически организованный набор типов записи.

Рисунок 2.1 - Пример типа дерева (схема иерархической БД)

Здесь (см. рисунок 2.1) Группа является предком для Куратора и Студенты, а Куратор и Студенты – потомки Группы. Между типами записи поддерживаются связи.

Все экземпляры данного типа потомка с общим экземпляром типа предка называются близнецами. Для БД определен полный порядок обхода – сверху-вниз, слева-направо.

Примерами типичных операторов манипулирования иерархически организованными данными могут быть следующие:

a) найти указанное дерево БД;

b) перейти от одного дерева к другому;

c) перейти от одной записи к другой внутри дерева (например, от отдела - к первому сотруднику);

Автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя. Заметим, что аналогичное поддержание целостности по ссылкам между записями, не входящими в одну иерархию, не поддерживается (примером такой «внешней» ссылки может быть содержимое поля Каф_Номер в экземпляре типа записи Куратор).

2.3 Сетевая модель

Типичным представителем является Integrated Database Management System (IDMS) компании Cullinet Software, Inc., предназначенная для использования на машинах основного класса фирмы IBM под управлением большинства операционных систем. Архитектура системы основана на предложениях Data Base Task Group (DBTG) Комитета по языкам программирования Conference on Data Systems Languages (CODASYL), организации, ответственной за определение языка программирования Кобол.

Отчет DBTG был опубликован в 1971 г., а в 70-х годах появилось

несколько систем, среди которых IDMS.

Сетевой подход к организации данных является расширением иерархического. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков.

Сетевая БД состоит из набора экземпляров каждого типа записи и набора экземпляров каждого типа связи (см. рисунок 2.4).

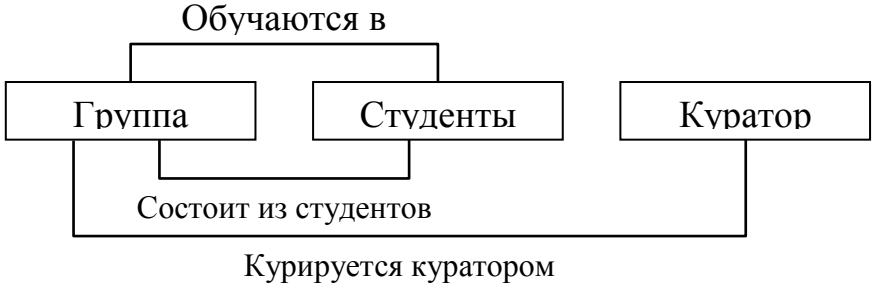

Рисунок 2.4 - Простой пример сетевой схемы БД

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

a) каждый экземпляр типа P является предком только в одном экземпляре L;

b) каждый экземпляр C является потомком не более чем в одном экземпляре L.

На формирование типов связи не накладываются особые ограничения; возможны, например, следующие ситуации:

a) тип записи потомка в одном типе связи L1 может быть типом записи предка в другом типе связи L2 (как в иерархии);

b) данный тип записи P может быть типом записи предка в любом числе типов связи;

c) данный тип записи P может быть типом записи потомка в любом числе типов связи.

Примерный набор операций может быть следующим:

a) найти конкретную запись в наборе однотипных записей;

b) перейти от предка к первому потомку по некоторой связи;

c) перейти от потомка к предку по некоторой связи и т.д.

Ограничения целостности в принципе не требуется, но иногда требуется целостность по ссылкам (как в иерархической модели).

2.4 Основные достоинства и недостатки ранних СУБД

Сильные места ранних СУБД:

a) развитые средства управления данными во внешней памяти на низком уровне;

b) возможность построения вручную эффективных прикладных систем;

c) возможность экономии памяти за счет разделения подобъектов (в сетевых системах).

Недостатки:

a) слишком сложно пользоваться;

b) фактически необходимы знания о физической организации;

c) прикладные системы зависят от этой организации;

d) их логика перегружена деталями организации доступа к БД.

3 Лекция №3. Реляционная модель и ее характеристики. Целостность в реляционной модели

Цель лекции: изучение характеристик реляционной модели и понятия целостности данных.

Содержание лекции: представление информации в реляционной БД; домены; отношения, свойства и виды отношений; целостность реляционных данных; потенциальные и первичные ключи; внешние ключи; ссылочная целостность.

3.1 Представление информации в реляционных БД

Реляционный подход является наиболее распространенным в настоящее время. К числу достоинств реляционного подхода можно отнести:

a) наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;

b) наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации баз данных;

c) возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

Однако реляционные системы далеко не сразу получили широкое распространение. В то время, как основные теоретические результаты в этой области были получены еще в 70-х, и тогда же появились первые прототипы реляционных СУБД, долгое время считалось невозможным добиться эффективной реализации таких систем. Однако отмеченные выше преимущества и постепенное накопление методов и алгоритмов организации реляционных баз данных и управления ими привели к тому, что уже в середине 80-х годов реляционные системы практически вытеснили с мирового

рынка ранние СУБД.

В настоящее время основным предметом критики реляционных СУБД является не их недостаточная эффективность, а следующие недостатки:

a) присущая этим системам некоторая ограниченность (прямое следствие простоты) при использовании в так называемых нетрадиционных областях (наиболее распространенными примерами являются системы автоматизации проектирования), в которых требуются предельно сложные структуры данных;

b) невозможность адекватного отражения семантики предметной области (другими словами, возможности представления знаний о семантической специфике предметной области в реляционных системах очень ограничены).

В реляционной модели рассматриваются три аспекта данных:

a) структура данных (объекты данных);

b) целостность данных;

c) обработка данных (операторы).

Рассмотрим специальную терминологию, применяемую в рамках аспекта «структура данных» (см. рисунок 3.1).

3.2 Домены

Домен является наименьшей семантической единицей данных, которая предполагается отдельным значением данных (таким как номер студента, фамилия студента и т.д.). Такие значения называют скалярами. Скалярные значения представляют собой наименьшую семантическую единицу данных в том смысле, что они являются атомарными: в реляционной модели у них отсутствует внутренняя структура.

Таким образом, домен – именованное множество скалярных значений одного типа. Например, домен городов - это множество всех возможных названий городов. Домены являются общими совокупностями значений, из которых берутся реальные значения атрибутов.

Основное значение доменов в том, что домены ограничивают сравнения**.** Сравнение будет иметь смысл для атрибутов, основанных на оценке, полученной студентом на экзамене, и то, и другое - целые числа, однако такое сравнение будет лишено смысла.

3.3 Отношения. Свойства и виды отношений

Вокруг понятия «отношение» сложилась некоторая двусмысленность из-за отсутствия четкого разграничения между переменными отношений и значениями отношений. Переменная отношения – это обычная переменная, такая же, как и в языках программирования, т.е. именованный объект, значение которого может изменяться со временем. А значение этой переменной в любой момент будет значением отношения**.**

Отношение R, определенное на множестве доменов D1, D2, …, Dn (не обязательно различных), содержит две части – заголовок и тело. Заголовок содержит фиксированное множество атрибутов или точнее пар на одном и том же домене. Например, можно сравнивать числовой код студента <имя-атрибута : имя-домена>:

{<A1:D1>, <A2:D2>, …, <An:Dn>},

причем каждый атрибут Aj соответствует одному и только одному из лежащих в основе доменов Dj (j=1,2, …, n). Все имена атрибутов A1, A2, …, An разные.

Тело состоит из множества кортежей. Каждый кортеж в свою очередь содержит множество пар <имя-атрибута : значение-атрибута>:

{<A1:Vi1>, <A2:Vi2>, …, <An:Vin>},

где i=1, 2, …, m; где m - количество кортежей в этом множестве).

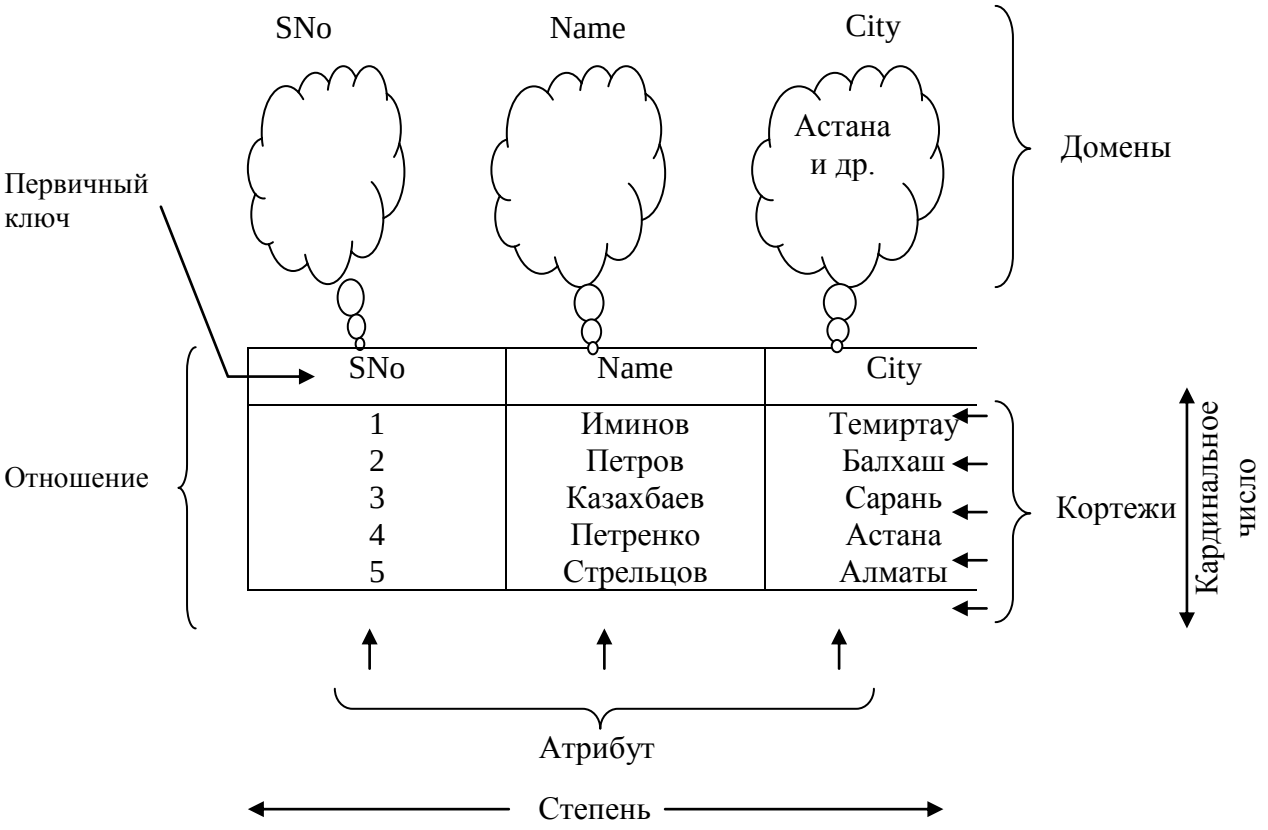

Рисунок 3.1 - Отношение

В каждом таком кортеже есть одна такая пара <имя-атрибута : значение-атрибута>, т.е. <Aj:Vij>, для каждого атрибута Aj в заголовке. Для любой такой пары <Aj:Vij> Vij является значением из уникального домена Dj, который связан с атрибутом Aj.

Значения m и n называются соответственно кардинальным числом и степенью отношения R.

3.3.1Свойства отношений.

1. В отношении отсутствуют одинаковые кортежи.

Это свойство следует из того факта, что тело отношения – это математическое множество (кортежей), а множества в математике по определению не содержат одинаковых элементов. Это свойство служит иллюстрацией различия между отношением и таблицей, т.к. таблица в общем случае может содержать одинаковые строки. Важным следствием того, что не существует одинаковых строк, является то, что всегда существует первичный ключ. Кортежи не упорядочены сверху вниз.

Это свойство также следует из того, что тело отношения – это математическое множество, а простые множества в математике не упорядочены.

2. Атрибуты не упорядочены слева направо.

И это свойство следует из того, что заголовок отношения определен как множество атрибутов. Аналогично второму свойству можно заметить отличия между таблицей и отношением – в таблице столбцы упорядочены слева направо.

3. Все значения атрибутов атомарные.

Это свойство является следствием того, что все лежащие в основе домены содержат только атомарные значения. Отношение, удовлетворяющее этому условию, называется нормализованным, или представленном в первой нормальной форме. Это означает, что с точки зрения реляционной модели все отношения нормализованы.

3.3.2 Виды отношений.

Определим некоторые виды отношений, встречающиеся в реляционных системах.

Именованное отношение – это переменная отношения, определенная в СУБД посредством операторов создания отношений.

Базовым отношением называется именованное отношение, которое не является производным, т.е. базовое отношения является автономным.

Производным отношением называется отношение, определенное (посредством реляционного выражения) через другие именованные выражения и, в конечном счете, через базовые отношения.

Выражаемое отношение – отношение, которое можно получить из набора именованных отношений посредством некоторого реляционного выражения.

Представление – это именованное производное отношение. Представления, как и базовые отношения, являются переменными отношений. Представления виртуальны – они представлены в системе исключительно через определение в терминах других именованных отношений.

Снимки – это именованные производные отношения, в отличие от представлений являются реальными и представлены в системе не только в виде определений в терминах других именованных отношений, но и своими данными.

Результатом запроса называется неименованное производное отношение, служащее результатом некоторого определенного запроса.

Промежуточным результатом называется неименованное производное отношение, являющееся результатом некоторого реляционного выражения, вложенного в другое, большее выражение.

Хранимое отношение – отношение, которое поддерживается непосредственно в физической памяти.

3.4 Целостность реляционных данных

Большинство БД подчиняются множеству правил целостности. В любой момент времени любая база данных содержит некую определенную конфигурацию значений данных, и предполагается, что эта конфигурация отображает действительность. В реляционной модели есть два общих особых правил целостности. Эти правила относятся к потенциальным (и первичным) ключам и к внешним ключам.

3.5 Потенциальные и первичные ключи

Пусть R – некоторое отношение. Тогда потенциальный ключ**,** скажем, K для R – это подмножество множества атрибутов R, обладающее следующими свойствами:

a) свойство уникальности – нет двух разных кортежей в отношении R с одинаковым значением K;

b) свойство не избыточности – никакое из подмножеств K не обладает свойством уникальности.

Потенциальные ключи определены как множества атрибутов. Потенциальный ключ, состоящий из нескольких атрибутов, называется составным, состоящий из одного атрибута – простым.

Если определить «потенциальный ключ», не являющийся не избыточным, системе не будет известно об этом, и она не сможет обеспечить должным образом соответствующее ограничение целостности. Например: в отношении с данными о студентах можно определить избыточный потенциальный ключ, состоящий из уникального кода студента StNo и названия города, в котором он проживает City. В таком случае система не сможет соблюдать ограничение, обеспечивающее уникальность номера студента в глобальном смысле, вместо этого будет выполняться более слабое ограничение, обеспечивающее уникальность номера студента в пределах города.

Отношение может иметь более одного потенциального ключа. В этом случае в реляционной модели, один из них выбирается в качестве первичного в базовом отношении, а остальные потенциальные ключи, если они есть, будут называться альтернативными ключами.

Если множество потенциальных ключей содержит более одного элемента, выбор первичного ключа, в общем случае, осуществляется произвольно.

3.6 Внешние ключи

Пусть R2 – базовое отношение. Тогда внешний ключ – FK в отношении R2 – это подмножество множества атрибутов R2 такое, что:

a) существует базовое отношение R1 (R1 и R2 не обязательно различны) с потенциальным ключом FK;

b) каждое значение FK в текущем значении R2 всегда совпадает со значением FK некоторого кортежа в текущем значении R1.

Следует отметить, что:

a) внешние ключи, как и потенциальные, определены как множества атрибутов;

b) данный внешний ключ будет составным (т.е., будет состоять из более чем одного атрибута) тогда и только тогда, когда соответствующий потенциальный ключ также будет составным;

c) он будет простым тогда и только тогда, когда соответствующий потенциальный ключ также будет простым;

d) каждый атрибут, входящий в данный внешний ключ, должен быть определен на том же домене, что и соответствующий атрибут соответствующего потенциального ключа;

e) для внешнего ключа не требуется, чтобы он был компонентом первичного ключа или какого-либо потенциального ключа в содержащем его отношении;

f) значение внешнего ключа представлено ссылкой к кортежу, содержащему соответствующее значение потенциального ключа (Проблема обеспечения того, что БД не включает никаких неверных значений внешних ключей, известна как проблема ссылочной целостности. Ограничение, по которому значения данного внешнего ключа должны быть адекватны значениям соответствующего потенциального ключа, называют ссылочным ограничением. Отношение, содержащее внешний ключ, называют ссылающимся отношением, а отношение, которое содержит соответствующий потенциальный ключ, – ссылочным или целевым отношением);

g) связи, существующие в базе данных, отображают с помощью ссылочных диаграмм - Groups ← Students → Cities;

h) отношение может быть и ссылочным, и ссылающимся одновременно - Students → Groups → Department;

i) определении внешних ключей сказано, что R1 и R2 не обязательно различны (то есть отношение может ссылаться само на себя - такие отношения иногда называют самоссылающимися);

j) самоссылающиеся отношения представляют собой частный случай ситуации, когда могут возникнуть ссылочные циклы, которые можно отобразить на ссылочной диаграмме следующим образом:

$$Rn \rightarrow R(n-1) \rightarrow R(n-2) \rightarrow \ldots \rightarrow R2 \rightarrow R1 \rightarrow Rn.$$

3.7 Ссылочная целостность

База данных не должна содержать несогласованных значений внешних ключей. Несогласованное значение внешнего ключа – это такое значение внешнего ключа, для которого не существует отвечающего ему значения соответствующего потенциального ключа в соответствующем целевом отношении.

3.7.1 Правила внешних ключей.

Для того чтобы избежать некорректных состояний для каждого внешнего ключа, необходимо установить правила, на основании которых СУБД будет действовать при попытке удалить объект ссылки внешнего ключа или обновить потенциальный ключ, на который ссылается внешний ключ. Для каждого из этих случаев можно предусмотреть, по меньшей мере, две возможности:

a) при попытке удалить объект ссылки внешнего ключа:

1) ограничить – приостановить операцию удаления, до момента, когда не будет существовать ссылающихся объектов;

2) каскадировать – каскадировать операцию удаления, удалив соответствующие ссылающиеся объекты;

b) при попытке обновить потенциальный ключ, на который ссылается внешний ключ:

1) ограничить – приостановить операцию обновления, до момента, когда не будет существовать ссылающихся объектов;

2) каскадировать – каскадировать операцию обновления, обновив значение внешнего ключа в соответствующих ссылающихся объектах.

3.7.2 Значения NULL и поддержка ссылочной целостности.

Значения NULL используются для обозначения факта отсутствия информации. При этом следует учесть, что значения NULL отличаются от числового значения 0 или символьных пробелов.

Возможность присутствия в отношении значений NULL приводит к необходимости формирования правила целостности объектов. Целостность объектов – ни один элемент первичного ключа не может содержать значения NULL.

4 Лекция №4. Реляционная алгебра

Цель лекции: изучение основ реляционной алгебры.

Содержание лекции: понятие реляционной алгебры; замкнутость в реляционной алгебре; традиционное операции над множествами; свойства основных операций реляционной алгебры; специальные реляционные операции.

4.1 Понятие реляционной алгебры

Реляционная алгебра в основном состоит из набора операторов,

использующих отношения в качестве операндов и возвращающих отношения в качестве результата.

Реляционная алгебра, определенная Коддом, состоит из восьми операторов, составляющих две группы, по четыре оператора в каждой:

a) традиционные операции над множествами: объединение, пересечение, вычитание и декартово произведение (модифицированные с учетом того, что их операндами являются отношения, а не произвольные множества);

b) специальные реляционные операции: выборка, проекция, соединение и деление.

4.2 Замкнутость в реляционной алгебре

Результат каждой операции над отношением (или реляционной операции) также является отношением. Это реляционное свойство называется свойством замкнутости. Поскольку результат любой операции имеет тот же тип, что и исходные объекты (отношения), то результат одной операции может использоваться в качестве исходных данных для другой.

4.3 Традиционные операции над множествами

4.3.1 Объединение.

Объединение в реляционной алгебре - это особая форма объединения, в которой требуется, чтобы два исходных отношения были совместимы по типу.

Будем говорить, что два отношения совместимы по типу, если у них идентичные заголовки, а точнее:

a) если каждое из них имеет одно и то же множество имен атрибутов (они должны иметь одну и ту же степень);

b) если соответствующие атрибуты (т.е. атрибуты с теми же самыми именами в двух отношениях) определены на одном и том же домене;

c) операции объединения, пересечения и вычитания требуют от операндов совместимости по типу.

Объединением двух совместимых по типу отношений A и B (A UNION B) называется отношение с тем же заголовком, как и в отношениях A и B, и с телом, состоящим из множества всех кортежей, принадлежащих A или B или обоим отношениям.

Пример операции объединения отношений на рисунке 4.1 и рисунке 4.2.

4.3.2 Пересечение.

Пересечением двух совместимых по типу отношений A и B (A INTERSECT B) называется отношение с тем же заголовком, как и в отношениях A и B, и с телом, состоящим из множества всех кортежей, принадлежащих одновременно обоим отношениям A и B. Пример операции пересечения отношений приведен на рисунке 4.1 и рисунке 4.3.

4.3.3 Вычитание.

Вычитанием двух совместимых по типу отношений A и B (A MINUS B) называется отношение с тем же заголовком, как и в отношениях A и B, и с телом, состоящим из множества всех кортежей, принадлежащих отношению A и не принадлежащих отношению B.

Пример операции вычитания отношений приведен на рисунке 4.1 и рисунке 4.4.

A

CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

B

CityNo	CityName	RgNo
2	Сарань	1
3	Балхаш	1
4	Алматы	2

Рисунок 4.1 - Исходные отношения

A UNION B

CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1
4	Алматы	2

Рисунок 4.2 - Результат объединения отношений A и B

A INTERSECT B

CityNo	CityName	RgNo
2	Сарань	1
3	Балхаш	1

Рисунок 4.3. - Результат операции пересечения отношений A и B

A MINUS B

CityNo	CityName	RgNo
1	Темиртау	1

B MINUS A

CityNo	CityName	RgNo
4	Алматы	2

Рисунок 4.4 - Результат операции вычитания отношений A минус B и B минус A

4.3.4 Произведение.

В математике декартово произведение (или для краткости произведение) двух множеств является множеством всех таких упорядоченных пар элементов, что первый элемент в каждой паре берется из первого множества, а второй элемент в каждой паре берется из второго множества.

Декартово произведение двух отношений A и B (A TIMES B), где A и B

не имеют общих имен атрибутов, определяется как отношение с заголовком, который представляет собой сцепление двух заголовков исходных отношений A и B, и телом, состоящим из множества всех кортежей t таких, что t представляет собой сцепление кортежа a, принадлежащего отношению A, и кортежа b, принадлежащего отношению B. Кардинальное число результата равняется произведению кардинальных чисел исходных отношений A и B, а степень равняется сумме их степеней. Пример операции декартова произведения представлена на рисунке 4.5

A

CityNo	CityName	A_RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

B

B_RgNo	RgName
1	Карагандинская
2	Алматинская

A TIMES B

CityNo	CityName	A_RgNo	B_RgNo	RgName
1	Темиртау	1	1	Карагандинская
1	Темиртау	1	2	Алматинская
2	Сарань	1	1	Карагандинская
2	Сарань	1	2	Алматинская
3	Балхаш	1	1	Карагандинская
3	Балхаш	1	2	Алматинская

Рисунок 4.5 - Результат операции декартово произведение отношений A и B

Явное использование операции декартова произведения требуется только для очень сложных запросов. Эта операция включена в реляционную алгебру главным образом по концептуальным соображениям. Декартово произведение требуется как промежуточный шаг при определении операции Θ-соединения, которая используется довольно часто.

4.4 Свойства основных операций реляционной алгебры

Операции объединения, пересечения и декартова произведения (но не вычитания) обладают свойствами ассоциативности и коммутативности.

Пусть A, B и C – произвольные реляционные выражения (дающие совместимые по типу отношения). Тогда операция объединения (A UNION B) UNION C эквивалентна операции A UNION (B UNION C) (свойство ассоциативности), а операция объединения A UNION B эквивалентна операции B UNION A (свойство коммутативности). Аналогично свойства ассоциативности и коммутативности определяются для остальных операций.

Свойство ассоциативности позволяет записывать последовательные операторы объединения (пересечения и декартова произведения) без

использования круглых скобок; таким образом, выражение из предыдущего примера можно однозначно упростить:

A UNION B UNION C.

4.5 Специальные реляционные операции

4.5.1 Выборка.

Выборка – это сокращенное название Θ-выборки, где Θ обозначает любой скалярный оператор сравнения (=, ≠, >, ≥, ≤, <). Θ-выборкой из отношения A по атрибутам X и Y (в этом порядке):

A WHERE X Θ Y

называется отношение, имеющее тот же заголовок, что и отношение A, и тело, содержащее множество всех кортежей отношения A, для которых проверка условия X Θ Y дает значение истина. Атрибуты X и Y должны быть определены на одном и том же домене, а оператор должен иметь смысл для этого домена. На рисунке 4.6 приведен пример операции выборки.

A

CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1
4	Алматы	2

A WHERE RgNo = 1

CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

Рисунок 4.6 - Исходное отношение A и результат операции выборки кортежей из отношения A по условию RGNo = 1

4.5.2 Проекция.

Проекцией отношения A по атрибутам X, Y,..., Z, где каждый из атрибутов принадлежит отношению A:

A [X, Y, …, Z],

называется отношение с заголовком {X, Y,..., Z} и телом, содержащим множество всех кортежей {X:x, Y:y,..., Z:z}, таких, для которых в отношении A значение атрибута X равно x, атрибута Y равно y, ..., атрибута Z равно z.

Таким образом, с помощью оператора проекции получено «вертикальное» подмножество данного отношения, т.е. подмножество, получаемое исключением всех атрибутов, не указанных в списке атрибутов, и последующим исключением дублирующих кортежей (см. рисунок 4.7).

Никакой атрибут не может быть указан в списке атрибутов более одного раза. Действие такой операции эквивалентно указанию списка всех атрибутов исходного отношения, т.е. такая операция представляет собой тождественную проекцию. Другими словами, имя отношения является допустимым реляционным выражением.

A

CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1
4	Алматы	2

A [CityName]

CityName
Темиртау
Сарань
Балхаш
Алматы

Рисунок 4.7 - Исходное отношение A и результат операции проекции отношения A по атрибуту CityName

4.5.3 Естественное соединение.

Пусть отношения A и B имеют заголовки {Xl, X2, ..., Xm, Y1, Y2, ..., Yn} и {Yl, Y2, ..., Yn, Zl, Z2, ..., Zp} соответственно; т.е. атрибуты Yl, Y2, ..., Yn (и только они) - общие для двух отношений; X1, X2, ... ,Xm – остальные атрибуты отношения A; Zl, Z2, ..., Zp - остальные атрибуты отношения B. Предположим также, что соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене. Рассматривать выражения {X1, X2, ..., Xm}, {Y1, Y2, ..., Yn} и {Zl, Z2, ..., Zp} как три составных атрибута X, Y и Z соответственно. Тогда естественным соединением отношений A и B (A JOIN B) называется отношение с заголовком {X, Y, Z} и телом, содержащим множество всех кортежей {X:x, Y:y, Z:z}, таких, для которых в отношении A значение атрибута X равно x, а атрибута Y равно y, и в отношении B значение атрибута Y равно y, а атрибута Z равно z.

Пример операции естественного соединения приведен на рисунке 4.8.

A

CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

B

RgNo	RgName
1	Карагандинская
2	Алматинская

A JOIN B

CityNo	CityName	A_RgNo	RgName
1	Темиртау	1	Карагандинская
2	Сарань	1	Карагандинская
3	Балхаш	1	Карагандинская

Рисунок 4.8 - Исходные отношения A и B и результат операции естественного соединения

Соединение обладает свойствами ассоциативности и коммутативности.

Отсюда следует, что выражения:

(A JOIN B) JOIN C и A JOIN (B JOIN C) могут быть однозначно упрощены к следующему:

A JOIN B JOIN C.

Кроме того, выражения A JOIN B и B JOIN A эквивалентны.

4.5.4 Θ-соединение.

Операция Θ-соединения предназначается для тех случаев когда нам нужно соединить вместе два отношения на основе некоторых условий, отличных от эквивалентности.

Пусть отношения A и B не имеют общих имен атрибутов (как и в рассмотренной выше операции декартова произведения) и Θ определяется так же, как и в операции выборки.

Тогда Θ-соединением отношения A по атрибуту X с отношением B по атрибуту Y называется результат вычисления выражения:

(A TIMES B) WHERE X Θ Y.

Θ-соединение, таким образом, это отношение с тем же заголовком, что и при декартовом произведении отношений A и B, и с телом, содержащим множество кортежей, принадлежащих этому декартову произведению и вычисление условия XΘY дает значение истина для этого кортежа. Атрибуты X и У должны быть определены на одном и том же домене, а операция должна иметь смысл для этого домена.

4.5.5 Деление.

Пусть отношения A и B имеют заголовки {X1, X2,..., Xm, Y1, Y2, ..., Yn} и {Y1, Y2, ..., Yn} соответственно, т.е. атрибуты Y1, Y2,..., Yn — общие для двух отношений, и отношение A имеет дополнительные атрибуты X1, X2, ... ,Xm, а отношение B не имеет дополнительных атрибутов. (Отношения A и B представляют соответственно делимое и делитель.)

Предположим также, что соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене. Пусть теперь выражения {X1, X2, ..., Xm} и {Y1, Y2, ..., Yn} обозначают два составных атрибута X и Y соответственно.

Тогда делением отношений A на B (A DIVIDEBY B) называется отношение с заголовком {X} и телом, содержащим множество всех кортежей {X:x}, таких, что существует кортеж {X:x, Y:y}, который принадлежит отношению A для всех кортежей {Y:y}, принадлежащих отношению B.

Нестрого это можно сформулировать так: результат содержит такие X-значения из отношения A, для которых соответствующие Y-значения (из A) включают все Y-значения из отношения B.

Пример операции деления приведен на рисунке 4.9.

Отношение M является проекцией отношения Marks, а отношение S – проекцией отношения Subjects. Результат операции деления M DIVIDE BY S фактически содержит номера студентов, которые сдавали дисциплины с номерами 1 и 5.

M

StNo	SubjNo
1	1
1	5
2	1
2	5
3	1
3	5
4	1
5	1

S

SubjNo
1
5

M DIVIDE BY S

StNo
1
2
3

Рисунок 4.9 - Пример операции деления

5 Лекция №5. Язык SQL

Цель лекции: изучение основных понятий языка SQL.

Содержание лекции: история создания и развития SQL; основные понятия SQL; запросы на чтение данных, оператор SELECT; многотабличные запросы на чтение (объединения).

5.1 История создания и развития SQL

Язык для взаимодействия с БД SQL появился в середине 70-х и был разработан в рамках проекта экспериментальной реляционной СУБД System R. На сегодняшний день SQL является единственным стандартным языком запросов. Язык SQL обладает следующими достоинствами:

a) независимость от конкретных СУБД (если при создании БД не использовались нестандартные возможности языка SQL, предоставляемые некоторой СУБД, то такую БД можно без изменений перенести на СУБД другого производителя);

b) реляционная основа (реляционная модель имеет солидный теоретический фундамент);

c) SQL обладает высокоуровневой структурой, напоминающей английский язык.

Официальный стандарт языка SQL опубликован ANSI и ISO в 1989 году и значительно расширен в 1992 году.

5.2 Основные понятия SQL

5.2.1 Операторы.

В SQL используется приблизительно тридцать операторов, каждый из которых «просит» СУБД выполнить определенное действие, например, прочитать данные, создать таблицу или добавить в таблицу новые данные.

Все операторы SQL имеют одинаковую структуру, которая показана на рисунке 5.1.

Каждый оператор SQL начинается с глагола, т.е. ключевого слова, описывающего действие, выполняемое оператором. Типичными глаголами являются SELECT (выбрать), CREATE (создать), INSERT (добавить), DELETE (удалить), COMMIT(завершить). После глагола идет одно или несколько предложений. Предложение описывает данные, с которыми работает оператор, или содержит уточняющую информацию о действии, выполняемом оператором. Каждое предложение также начинается с ключевого слова такого, как WHERE (где), FROM (откуда), INTO (куда) и HAVING (имеющий). Одни предложения в операторе являются обязательным, а другие – нет.

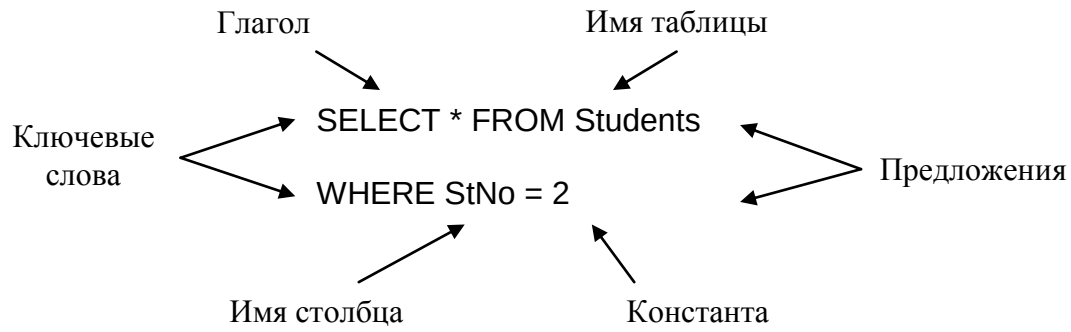

Рисунок 5.1 - Структура оператора SQL

5.2.2 Имена.

В соответствии со стандартом ANSI/ISO, в SQL имена должны содержать от 1 до 18 символов, начинаться с буквы и не содержать пробелы или специальные символы пунктуации. В стандарте SQL2 максимальное число символов в имени увеличено до 128.

Полное имя таблицы состоит из имени владельца таблицы и собственно ее имени, разделенных точкой (.). Например, полное имя таблицы Students, владельцем которой является пользователь по имени Admin, имеет следующий вид:

Admin.Students

Полное имя столбца состоит из имени таблицы, содержащей столбец, и имени столбца (простого имени), разделенных точкой (.). Например, полное имя столбца StName из таблицы Students имеет следующий вид:

Students.StName

5.3 Запросы на чтение данных. Оператор SELECT

Оператор SELECT применяется для построения выборок данных и имеет следующий синтаксис:

SELECT [ALL | DISTINCT] возвращаемый_столбец, … | *

FROM спецификатор_таблицы, …
WHERE условие_поиска
GROUP BY имя_столбца, …
HAVING условие_поиска
ORDER BY спецификатор_сортировки, … .

5.3.1 Предложение SELECT.

В предложении SELECT, с которого начинаются все операторы SELECT, необходимо указать элементы данных, которые будут возвращены в результате запроса. Эти элементы задаются в виде списка возвращаемых столбцов, разделенных запятыми. Для каждого элемента из этого списка в таблице результатов запроса будет создан один столбец. Столбцы в таблице результатов будут расположены в том же порядке, что и элементы списка возвращаемых столбцов. Возвращаемый столбец может представлять собой:

a) имя столбца, идентифицирующее один из столбцов, содержащихся в таблицах, которые перечислены в предложении FROM (когда в качестве возвращаемого столбца указывается имя столбца таблицы базы данных, SQL просто берет значение этого столбца для каждой из строк таблицы и помещает его в соответствующую строку таблицы результатов запроса);

b) константа, показывающая, что в каждой строке результатов запроса должно содержаться одно и то же значение;

c) выражение, показывающее, что SQL должен вычислять значение, помещаемое в результаты запроса, по формуле, определенной в выражении.

5.3.2 Предложение FROM.

Предложение FROM состоит из ключевого слова FROM, за которым следует список спецификаторов таблиц, разделенных запятыми. Каждый спецификатор таблицы идентифицирует таблицу, содержащую данные, которые считывает запрос. Такие таблицы называются исходными таблицами запроса (и оператора SELECT), поскольку все данные, содержащиеся в таблице результатов запроса, берутся из них.

Результатом SQL-запроса на чтение всегда является таблица, содержащая данные и ничем не отличающаяся от таблиц базы данных

Кроме столбцов, значения которых считываются непосредственно из базы данных, SQL-запрос на чтение может содержать вычисляемые столбцы, значения которых определяются на основании значений, хранящихся в базе данных. Чтобы получить вычисляемый столбец, в списке возвращаемых столбцов необходимо указать выражение.

Иногда требуется получить содержимое всех столбцов таблицы. С учетом этого в SQL разрешается использовать вместо списка возвращаемых столбцов символ звездочки (*), который означает, что требуется прочитать все столбцы**:**

Показать все данные, содержащиеся в таблице Students.

SELECT *
FROM Students

Повторяющиеся строки из таблицы результатов запроса можно удалить, если в операторе SELECT перед списком возвращаемых столбцов указать ключевое слово DISTINCT.

5.3.3 Отбор строк (предложение WHERE).

Обычно требуется выбрать из таблицы несколько строк и включить в результаты запроса только их.

Чтобы указать, какие строки требуется отобрать, следует использовать предложение WHERE.

Предложение WHERE состоит из ключевого слова WHERE, за которым следует условие поиска, определяющее, какие именно строки требуется прочитать.

Если условие поиска имеет значение TRUE, строка будет включена в результаты запроса.

Если условие поиска имеет значение FALSE или NULL, то строка исключается из результатов запроса.

5.3.4 Условия поиска.

Сравнение (=, <>, <, <=, >, >=). При сравнении SQL вычисляет и сравнивает значения двух выражений для каждой строки данных.

Выражения могут быть как очень простыми, например, содержать одно имя столбца или константу, так и более сложными – арифметическими выражениями.

Ниже приведен синтаксис оператора сравнения:

Выражение1 = | <> | < | <= | > | >= Выражение2.

Проверка на принадлежность диапазону значений (BETWEEN). При этом проверяется, находится ли значение данных между двумя определенными значениями:

-проверяемое_выражение [NOT] BETWEEN нижнее_выражение AND верхнее_выражение.

При проверке на принадлежность диапазону верхний и нижний пределы считаются частью диапазона, поэтому входят в результаты запроса.

Поверка на членство в множестве (IN). В этом случае проверяется, соответствует ли значение данных какому-либо значению из заданного списка:

-проверяемое_выражение [NOT] IN (константа, …).

Например: вывести список фамилий студентов, которые учатся в группах с кодами 1, 3, 5 и 10:

```
SELECT StName
FROM Students
WHERE GrNo IN (1, 3, 5, 10).
```

Проверка на соответствие шаблону (LIKE). Поверка на соответствие шаблону (ключевое слово LIKE) позволяет определить, соответствует ли значение данных в столбце некоторому шаблону**.** Шаблон представляет собой строку, в которую может входить один или более подстановочных знаков**.** Эти знаки интерпретируются особым образом:

- имя_столбца [NOT] LIKE шаблон [ESCAPE символ_пропуска].

Подстановочные знаки. Подстановочный знак % совпадает с любой последовательностью из нуля или более символов. Например, следующий запрос выведет информацию о студентах, чья фамилия начинается с «Иван»:

SELECT *

FROM StudentsWHERE StName LIKE 'Иван%'.

Подстановочный знак «_» (символ подчеркивания) совпадает с любым отдельным символом. Например, если вы уверены, что имя студентки либо «Наталья», либо «Наталия», то можете воспользоваться следующим запросом:

SELECT *

FROM Students WHERE StName LIKE 'Натал_я'.

5.3.5 Сортировка результатов запроса (предложение ORDER BY).

Строки результатов запроса, как и строки таблицы базы данных, не имеют определенного порядка. Включив в оператор SELECT предложение ORDER BY, можно отсортировать результаты запроса. Это предложение состоит из ключевых слов ORDER BY, за которыми следует список имен столбцов, разделенных запятыми:

ORDER BY имя_столбца [ASC | DESC], … .

По умолчанию, данные сортируются в порядке возрастания. Чтобы сортировать их по убыванию, следует включить в предложение сортировки ключевое слово DESC.

Например, нужно вывести список фамилий студентов учащихся в группе с кодом 1 в обратном алфавитном порядке:

SELECT StName

FROM Students ORDER BY DESC StName .

5.4 Многотабличные запросы на чтение (объединения)

На практике многие запросы считывают данные сразу из нескольких таблиц базы данных.

5.4.1 Запросы с использованием отношения предок/потомок.

Среди многотабличных запросов наиболее распространены запросы к двум таблицам, связанными с помощью отношения предок/потомок. Запрос о студентах, учащихся в группе с некоторым названием является примером такого запроса. У каждого студента (потомка) есть соответствующая ему группа (предок), и каждая группа (предок) может иметь много студентов (потомков). Пары строк, из которых формируются результаты запроса, связаны отношением предок/потомок.

Например, нужно вывести список фамилий студентов с названием группы, в которой он учится:

SELECT StName, GrName

FROM Students, Groups

WHERE Students.GrNo = Groups.GrNo .

6 Лекция №6. Проектирование реляционной БД. Функциональные зависимости.

Цель лекции: изучение проектирования реляционной базы данных и функциональных зависимостей.

Содержание лекции: понятие проектирования БД; функциональные зависимости; тривиальные и нетривиальные зависимости; неприводимое множество зависимостей.

6.1 Понятие проектирования БД

Как в некоторой базе данных, для заданного набора данных выбрать подходящую логическую структуру? Иначе говоря, нужно решить вопрос, какие базовые отношения и с какими атрибутами следует задать. Следует заметить, что речь здесь пойдет о логическом, а не физическом макете.

Прежде всего создать логический (т.е. реляционный) макет, а затем в виде отдельного шага отобразить этот логический макет на некоторые физические структуры, поддерживаемые СУБД.

Физический макет по определению является специфическим для данной СУБД. Логический макет, наоборот, совершенно независим от СУБД, и для его реализации могут быть использованы строгие теоретические принципы.

Необходимо, чтобы макет был стабильным, т.е. оставался работоспособным даже при возникновении в приложении новых (т.е. неизвестных на момент создания исходного макета) требований к данным.

Следуя этим допущениям, нужно создать концептуальную схему, т.е. абстрактный логический макет, не зависящий от аппаратного обеспечения, операционной системы, СУБД, языка программирования, пользователя и т.д.

6.2 Функциональные зависимости

Для демонстрации будем использовать несколько измененную версию отношения Students из учебной БД, которое в дополнение к обычным атрибутам StNo, GrNo, StName, CityNo будет содержать также атрибут RgNo. Далее это измененное отношение будет называться SR. В виде таблицы оно представлено на рисунке 6.1.

SR				
StNo	GrNo	StName	CityNo	RgNo
1	1	Иминов	3	1
2	1	Петров	3	1
3	1	Сидоров	3	1
4	2	Стрельцов	1	1
5	2	Казахбаев	4	2

Рисунок 6.1 - Данные отношения SR

Следует четко различать значение этого отношения (т.е. значение переменной отношения) в определенный момент времени и набор всех возможных значений**,** которые данное отношение (переменная) может принимать в различные моменты времени.

При рассмотрении переменных отношения, например базовых отношений, интерес представляют не столько функциональные зависимости для определенного в некоторый момент времени значения, сколько функциональные зависимости, выполняющиеся для всех возможных значений данной переменной.

Пусть R является переменной отношения, а X и Y – произвольными подмножествами множества атрибутов отношения R. Тогда Y функционально зависимо от X, что в символическом виде записывается как X→Y (и читается либо как «X функционально определяет Y» , либо как «X стрелка Y»), тогда и только тогда, когда для любого допустимого значения отношения R каждое значение X связано в точности с одним значением Y.

Иначе говоря, для любого допустимого значения отношения R, когда бы два кортежа отношения R ни совпадали по значению X, они также совпадают и по значению Y. Далее термин «функциональная зависимость» будет использоваться в последнем безотносительном ко времени смысле.

Например, в случае отношения SR функциональная зависимость {StNo}→{GrNo} выполняется для всех возможных значений SR, поскольку в любой момент времени данному студенту соответствует одна группа; таким образом, любые два кортежа отношения SR в один и тот же момент времени и с одним и тем же номером студента должны соответствовать одной и той же группе. Практически утверждение, что данная функциональная зависимость выполняется «всегда» (т.е. для всех возможных значений SR), является ограничением целостности для отношения SR, поскольку при этом накладываются определенные ограничения на все допустимые значения.

Ниже перечислено несколько безотносительных ко времени функциональных зависимостей для переменной отношения SR:

a) {StNo}→{GrNo};
b) {StNo}→{StName};
c) {StNo}→{CityNo} ;
d) {StNo}→{RgNo};
e) {StNo}→{GrNo, StName};
f) {StNo, GrNo}→ {StName} и другие.

Левая и правая стороны символической записи функциональной зависимости иногда называются детерминантом и зависимой частью соответственно. Как говорится в определении, детерминант и зависимая часть являются множествами атрибутов. Когда множество содержит только один атрибут, он называется одноэлементным множеством, скобки опускаются и символическая запись принимает вид:

StNo→GrNo .

Обратите внимание, в частности, на функциональные зависимости, которые выполняются для таблицы на рисунке 6.1, но не выполняются «всегда»:

$$GrNo \rightarrow CityNo \ .$$

Следует отметить, что если X является потенциальным ключом отношения R, например, X является первичным ключом, то все атрибуты Y отношения R должны быть обязательно функционально зависимы от X (это следует из определения потенциального ключа). В обычном отношении студентов Students, например, необходимо, чтобы выполнялась зависимость:

$$StNo \rightarrow \{GrNo, StName, CityNo\} \ .$$

Фактически, если отношение R удовлетворяет функциональной зависимости $A \rightarrow B$ и A не является потенциальным ключом, то R будет характеризоваться некоторой избыточностью**.** Например, в случае отношения SR сведения о том, что каждый данный город находится в данной области, будут повторяться много раз (это хорошо видно на рисунке 6.1).

На практике важно сократить множество ФЗ до компактных размеров, поскольку функциональные зависимости являются ограничениями целостности, поэтому при каждом обновлении данных в СУБД все они должны быть проверены.

6.3 Тривиальные и нетривиальные зависимости

Очевидным способом сокращения размера множества ФЗ было бы исключение тривиальных зависимостей, т.е. таких, которые не могут не выполняться. В качестве примера приведем тривиальную ФЗ для отношения SR:

$$\{StNo, GrNo\} \rightarrow \{StNo\} \ .$$

Фактически ФЗ тривиальна тогда и только тогда, когда правая часть символической записи данной зависимости является подмножеством (не обязательно собственным подмножеством) левой части.

6.4 Неприводимое множество зависимостей

Множество ФЗ называется неприводимым тогда и только тогда, когда выполняются перечисленные ниже свойства:

a) правая часть (зависимая часть) каждой ФЗ множества S содержит только один атрибут (т.е. является одноэлементным множеством);

b) левая часть (детерминант) каждой ФЗ множества S является неприводимой, т.е. ни один атрибут не может быть опущен из детерминанта (без конвертирования множества S в некоторое множество, не эквивалентное множеству S), в таком случае ФЗ является неприводимой слева;

c) ни одна функциональная зависимость в S не может быть опущена из S без конвертирования множества S в некоторое множество, не эквивалентное множеству S.

Множество зависимостей t, которое неприводимо и эквивалентно некоторому другому множеству зависимостей S, называется неприводимым покрытием множества S. Таким образом, с тем же успехом в системе вместо исходного множества зависимостей S может быть использовано его неприводимое покрытие t.

7 Лекция №7. Процедура нормализации

Цель лекции: изучение нормализации отношений.

Содержание лекции: нормальные формы - основные понятия;декомпозиция без потерь и функциональные зависимости; диаграммы функциональных зависимостей; первая нормальная форма, возможные недостатки отношений в 1НФ.

7.1 Нормальные формы – основные понятия

Процесс дальнейшей нормализации, который далее будет упоминаться просто как нормализация, основывается на концепции нормальных форм. Говорят, что отношение находится в некоторой нормальной форме, если удовлетворяет заданному набору условий. Например, отношение находится в первой нормальной форме, или сокращенно в 1 НФ, тогда и только тогда, когда оно содержит только скалярные значения.

Отсюда следует, что каждое нормализованное отношение находится в первой нормальной форме. Иначе говоря, термины «нормализованное» и «1НФ» означают одно и то же.

Все нормализованные отношения находятся в 1НФ. Некоторые отношения 1НФ находятся также в 2НФ и некоторые отношения 2НФ находятся также в ЗНФ. Мотивом для введения дополнительных определений было то, что вторая нормальная форма «более желательна», чем первая, а третья, в свою очередь, «более желательна», чем вторая. Процедуру нормализации можно охарактеризовать как последовательное приведение данного набора отношений к некоторой более желательной форме.

Эта процедура обратима, т.е. всегда можно использовать ее результат (например, множество отношений, находящихся в ЗНФ) для обратного преобразования (в исходное отношение, находящееся в 2НФ). Возможность выполнения обратного преобразования является весьма важной характеристикой, поскольку означает, что в процессе нормализации информация не утрачивается.

7.2 Декомпозиция без потерь и функциональные зависимости

Процедура нормализации включает разбиение, или декомпозицию данного отношения на другие отношения, причем декомпозиция должна быть обратимой, т.е. выполняться без потерь информации. Иначе говоря, интерес

представляет только те операции, которые выполняются без потерь информации.

Рассмотрим отношение Students из учебной базы данных, с атрибутами {StNo, GrNo, StName, CityNo} (см. рисунок 7.2).

В первом случае информация не утрачивается, поскольку отношения SGN и SC все еще содержат информацию о том, что Иванов живет в городе с кодом 1, Петров – 2. Соединение этих отношений позволяет восстановить исходное отношение Students, иначе говоря, первая декомпозиция является декомпозицией без потерь. Ниже приведены две возможные декомпозиции этого отношения (см. рисунок 7.3).

Во втором случае информация о городе, в котором проживает студент утрачивается, поскольку студенты, учащиеся в группе с кодом 1, живут в разных городах, и, зная код группы, невозможно однозначно определить код города, в котором проживает студент.

Students			
StNo	GrNo	StName	CityNo
1	1	Иминов	1
2	1	Петров	3

Рисунок 7.2 - Отношение Students

1. SGN

StNo	GrNo	StName
1	1	Иминов
2	1	Петров

SC

StNo	CityNo
1	1
2	3

2. SGN

StNo	GrNo	StName
1	1	Иминов
2	1	Петров

GC

GrNo	CityNo
1	1
1	3

Рисунок 7.3 - Возможные декомпозиции отношения Students

Следует отметить, что процесс, который до сих пор назывался «декомпозицией», на самом деле называется проецированием, т.е. каждое из показанных выше отношений SGN, SC и GC в действительности являются проекциями исходного отношения Students. Таким образом, оператор декомпозиции в процедуре нормализации фактически является оператором проецирования.

Исходное отношение при этом равно соединению его проекций. Для выполнения декомпозиции отношения без потерь необходимо знать, какие условия должны быть соблюдены для того, чтобы при обратном соединении гарантировать получение исходного отношения. Ответ на этот вопрос

содержится в теореме Хеза.

Теорема Хеза. Пусть R{A, B, C} является отношением, где A, B, C – атрибуты этого отношения. Если R удовлетворяет зависимости A→B, то отношение R равно соединению его проекций {A, B} и {A, C}.

7.3 Диаграммы функциональных зависимостей

Некоторое неприводимое множество зависимостей отношения R можно представить в виде диаграммы функциональных зависимостей (диаграммы ФЗ).

На рисунках 7.4 и 7.5 показаны диаграммы ФЗ для некоторых отношений из учебной БД. Как видно из рисунка 7.4 и 7.5, каждая стрелка начинается с потенциального ключа (на самом деле с первичного ключа) соответствующего отношения. По определению стрелки должны начинаться с каждого потенциального ключа, поскольку одному значению такого ключа всегда соответствует, по крайней мере, еще одно какое-то значение. Некоторые стрелки следовало бы исключить ввиду того, что они вызывают определенные трудности, но стрелки, начинающиеся с потенциальных ключей, никогда не могут быть исключены.

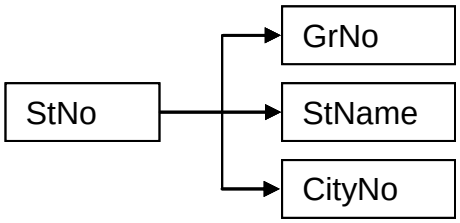

Рисунок 7.4 -Диаграмма ФЗ для таблицы Students

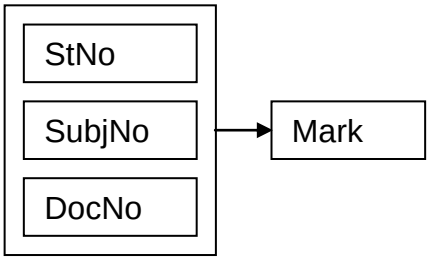

Рисунок 7.5 - Диаграмма ФЗ для таблицы Marks

7.4 Первая нормальная форма. Возможные недостатки отношения в 1НФ

Пусть каждое отношение имеет в точности один потенциальный ключ, который является первичным ключом. Отношение находится в первой нормальной форме тогда и только тогда, когда все используемые домены содержат только скалярные значения.

В этом определении всего лишь утверждается, что любое нормализованное отношение находится в 1НФ. Однако отношение, которое находится только в 1 НФ (т.е. не находится ни во второй, ни в третьей нормальной форме) обладает структурой, по некоторым причинам не совсем желательной. Для иллюстрации этого факта допустим, что информация о студентах и оценках содержится не в 2-х отношениях Students и Marks, а в одном, назовем его SM:

SM{StNo, CityNo, GrNo, SubjNo, DocNo, Mark},
PRIMARY KEY (StNo, SubjNo, DocNo).

Диаграмма функциональных зависимостей этого отношения будет выглядеть, как показано на рисунке 8.1.

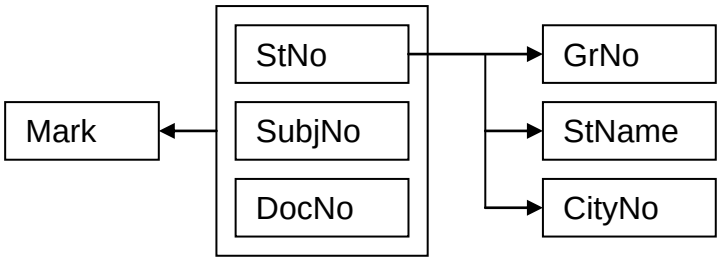

Рисунок 8.1 -Диаграмма функциональных зависимостей отношения SM

Обратите внимание, что диаграммы ФЗ отношения SM «сложнее», чем диаграммы ФЗ отношений Students и Marks, из которых оно образовано. В диаграммах ФЗ отношений Students и Marks все стрелки начинаются только от первичных ключей, тогда как в диаграмме ФЗ отношения SM появляются дополнительные стрелки. Ниже приведена таблица данных для отношения SM (см. рисунок 8.2).

SM						
StNo	StName	GrNo	CityNo	SubjNo	DocNo	Mark
1	Иминов	1	1	1	127	5
1	Иминов	1	1	5	128	4
2	Петров	1	3	1	127	3

Рисунок 8.2 - Данные отношения SM

Несмотря на то что отношение SM, как и Students и Marks, находится в 1й НФ, оно очевидно обладает избыточностью, поскольку, например, в каждом кортеже для студента Иминова указан его номер «1», код его группы – «1» и код города, в котором он проживает – «1». Аналогичная ситуация с другими студентами.

Избыточность в отношении SM приводит к разным аномалиям обновления**,** т.е. к трудностям при выполнении операций обновления типа INSERT (вставка), DELETE (удаление) и UPDATE (обновление). Для начала рассмотрим избыточность типа студент—код города студента, соответствующую функциональной зависимости StNo →CityNo, и перечисленные ниже проблемы с операциями обновления.

Операция вставки (INSERT). Нельзя вставить данные о студенте, проживающем в некотором городе, не указывая хотя бы одну, полученную этим студентом, оценку.

Операция удаления (DELETE). Если удалить единственный кортеж отношения SM для некоторого студента, будет удалена не только информация

о соответствующей оценке, но и информация о студенте и городе, в котором он проживает.

В действительности проблема заключается в том, что в отношении SM содержится очень много совместной информации, поэтому при удалении некоторого кортежа приходится удалять слишком иного другой информации. Для решения этой проблемы нужно разделить информацию на несколько частей, т.е. разместить информацию о студентах в одном отношении, а об оценках – в другом. Таким образом, неформально процедуру нормализации можно охарактеризовать как процедуру разбиения логически несвязанной информации по отдельным отношениям.

Операция модификации (UPDATE). Фамилия студента и код города, в котором он проживает, повторяется в отношении SM множество раз, и это приводит к возникновению проблем при обновлении. Если студент меняет фамилию или переезжает в другой город, то возникает проблема, связанная либо с поиском в отношении SM всех кортежей, в которых присутствует информация об этом студенте.

Для решения проблемы избыточности, которая характерна для отношения SM, достаточно заменить его двумя другими:

Students{StNo, GrNo, StName, CityNo}

и

Marks{StNo, SubjNo, DocNo, Mark} .

Переработанная таким образом структура позволяет преодолеть все перечисленные ранее проблемы, связанные с операциями обновления.

Операция вставки (INSERT). Теперь с помощью вставки соответствующего кортежа в отношение Students можно включить информацию о студенте и городе, в котором он проживает, даже если он в настоящий момент не получил ни одной оценки.

Операция удаления (DELETE). Теперь можно исключить информацию об оценке, удаляя соответствующий кортеж из отношения Marks, при этом информация о студенте и городе, в котором он проживает, не утрачивается.

Операция модификации (UPDATE). В переработанной структуре фамилия студента и информация о городе, в котором он проживает, появляется всего один раз, поскольку существует только один кортеж для данного студента в отношении Students (атрибут StNo является первичным ключом для такого отношения). Иначе говоря, избыточность данных StNo-StName-StCity устранена.

8 Лекция №8. Нормальные формы отношений

Цель лекции: изучение основных свойств нормальных форм.

Содержание лекции: вторая нормальная форма, возможные недостатки отношений в 2НФ; третья нормальная форма, возможные недостатки отношений в 3НФ; нормальная форма Бойса-Кодда.

8.1 Вторая нормальная форма. Возможные недостатки отношения во 2НФ

Определим 2НФ при условии, что существует только один потенциальный ключ, который является первичным ключом.

Отношение находится во второй нормальной форме тогда и только тогда, когда оно находится в первой нормальной форме, и каждый неключевой атрибут неприводимо зависим от первичного ключа.

Оба отношения, Students и Marks находятся во второй нормальной форме с первичными ключами StNo и {StNo, SubjNo, DocNo} соответственно, а отношение SM не находится в ней. Всякое отношение, которое находится в 1НФ и не находится в 2НФ, всегда можно свести к эквивалентному набору отношений, находящихся в 2НФ.

Рассмотрим другой пример. Предположим, информация о коде города, названии города и области, в которой этот город расположен находятся в одной таблице CNR{CityNo, CityName, RgNo, RgName} (см. рисунок 8.3).

CNR			
CityNo	CityName	RgNo	RgName
1	Темиртау	1	Карагандинская
2	Сарань	1	Карагандинская
3	Балхаш	1	Карагандинская
4	Алматы	2	Алматинская

Рисунок 8.3 - Данные отношения CNR

Диаграмма ФЗ отношения CNR выглядит следующим образом (см. рисунок 8.4).

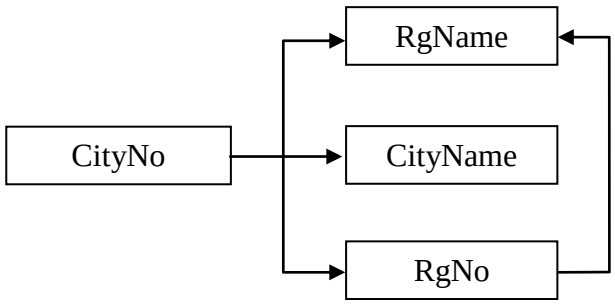

Рисунок 8.4 - Функциональные зависимости в отношении CNR

Эта диаграмма ФЗ «сложнее» диаграмм ФЗ отношений Cities и Regions. Несмотря на то что отношение CNR находится во 2НФ, оно обладает некоторой избыточностью, связанной с наличием транзитивной ФЗ между атрибутами CityNo и RgName. Транзитивная зависимость приводит к следующим аномалиям обновления.

Операция вставки (INSERT). Нельзя включить данные о некоторой области до тех пор, пока не появиться запись о городе, находящемся в данной области.

Операция удаления (DELETE). При удалении из отношения CNR последнего кортежа для некоторого города будет удалена не только информация о данном городе, но также информация о том, в какой области этот город находился. Вновь причиной этих неприятностей является совместная информация: отношение CNR содержит информацию о городах вместе с информацией об областях. Для разрешения этой ситуации следует поступить так, как и раньше, т.е. «разобрать» всю эту информацию и перенести в одно отношение сведения об областях, а в другое – сведения о городах.

Операция модификации (UPDATE). В отношении CNR код и название области для каждого города повторяется несколько раз (поэтому оно характеризуется некоторой избыточностью). Таким образом, при изменении кода области возникнет либо проблема необходимости поиска в отношении CNR всех кортежей для этой области (для внесения соответствующих изменений), либо проблема получения несовместимого результата.

Для решения этих проблем необходимо заменить отношение CNR двумя проекциями:

Cities{CityNo, CityName, RgNo},
Regions{RgNo, RgName} .

Переработанная таким образом структура отношений позволит преодолеть все описанные проблемы с операциями обновления.

8.2 Третья нормальная форма. Возможные недостатки отношения в 3НФ

Отношение находится в третьей нормальной форме тогда и только тогда, когда оно находится во второй нормальной форме и каждый неключевой атрибут нетранзитивно зависит от первичного ключа. (Под «нетранзитивной зависимостью» подразумевается отсутствие какой-либо взаимной зависимости в изложенном выше смысле.)

Отношения Cities и Regions находятся в третьей нормальной форме. Таким образом, вторым этапом нормализации является создание проекций для исключения транзитивных зависимостей.

8.2.1 Сохранение зависимости.

В процессе приведения отношений часто возникают ситуации, когда данное отношение может быть подвергнуто операции декомпозиции разными способами. Рассмотрим снова приведенное выше отношение CNR с функциональными зависимостями CityNo→CityName, CityNo→RgNo, CityNo→RgName, RgNo→RgName и, следовательно, транзитивной зависимостью CityNo→RgName (на рисунке 8.5 транзитивная зависимость показана пунктирной стрелкой).

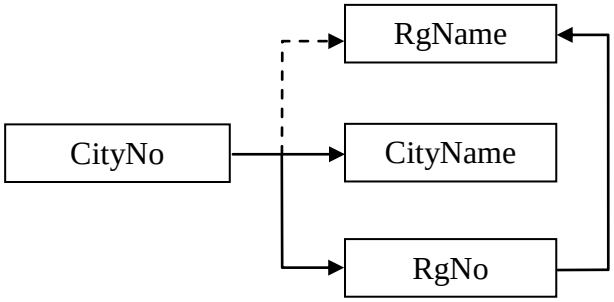

Рисунок 8.5 - Функциональные зависимости в отношении CNR

Выше отмечалось, что аномалии обновления, которые сопровождают отношение CNR, можно преодолеть с помощью декомпозиции с заменой этого отношения двумя проекциями в ЗНФ:

Cities{CityNo, CityName, RgNo} и Regions{RgNo, RgName} .

Назовем эту декомпозицию просто «декомпозицией №1», имея в виду, что для нее существует альтернативная «декомпозиция №2»:

Cities{CityNo, CityName, RgNo} и Regions{CityNo, RgName} .

При этом обе проекции Cities одинаковы как для №1, так и для №2. Декомпозиция №2 происходит также без потери информации, а обе ее проекции находятся в ЗНФ. Однако по некоторым причинам декомпозиция №2 менее желательна, чем декомпозиция №1. Например, после выполнения декомпозиции №2 все еще невозможно вставить информацию о том, что некоторая область имеет определенный код, без указания города, который находится в этой области.

Концепция независимых проекций, таким образом, обеспечивает критерий выбора одной из нескольких возможных декомпозиции. Декомпозиция с независимыми проекциями в приведенном выше общем смысле предпочтительнее той, в которой проекции зависимы.

Отношение, которое не может быть подвергнуто декомпозиции с получением независимых проекций, называется атомарным. Однако это не значит, что любое неатомарное отношение может быть разбито на атомарные компоненты. Идея нормализации с декомпозицией на независимые проекции называется декомпозицией с сохранением зависимости.

8.3 Нормальная форма Бойса-Кодда

Определение Кодда для ЗНФ не совсем подходит для отношений с перечисленными ниже условиями:

a) отношение имеет два (или более) потенциальных ключа;

b) два потенциальных ключа являются сложными.

c) они перекрываются (т.е. имеют, по крайней мере, один общий атрибут).

Отношение находится в нормальной форме Бойса-Кодда тогда и только тогда, когда каждая нетривиальная и неприводимая слева ФЗ обладает

потенциальным ключом в качестве детерминанта.

Комбинация вышеперечисленных условий не часто встречается на практике, и для отношения без этих условий ЗНФ и НФБК эквивалентны.

Менее формальное определение имеет другую формулировку: отношение находится в нормальной форме Бойса-Кодда тогда и только тогда, когда детерминанты являются потенциальными ключами.

Иначе говоря, на диаграмме ФЗ стрелки будут начинаться только с потенциальных ключей. Согласно данному определению никакие другие стрелки не допускаются.

Примером отношения, которое находится в НФБК, может служить отношение Students, в которое добавлен атрибут IdCode – идентификационный код:

Students {StNo, IdCode, GrNo, StName, CityNo} (см**.** рисунок 8.6).

В этом отношении детерминанты являются потенциальными ключами, а все стрелки начинаются с потенциальных ключей. Рассмотрим отношение, не находящееся в НФБК.

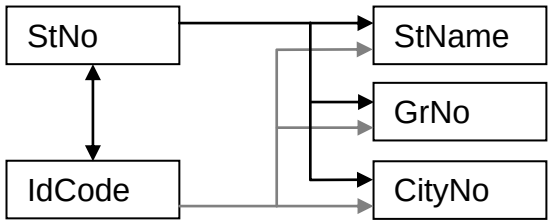

Рисунок 8.6 - Диаграмма ФЗ расширенного отношения, Students, находящегося в НФБК

Предположим, что информация об идентификационных кодах студентов хранится в отношении Marks. Назовем модифицированное отношение MI {StNo, IdCode, SubjNo, DocNo, Mark} (см. рисунок 8.7).

MI

StNo	IdCode	SubjNo	DocNo	Mark
1	2895764537	1	127	5
1	2895764537	5	128	4
2	3094769520	1	127	3
2	3094769520	5	128	3
3	2984267527	1	127	5

Рисунок 8.7 - Данные отношения MI

В этом отношении присутствуют 2 потенциальных ключа {StNo, SubjNo, DocNo} и {IdCode, SubjNo, DocNo}. Отношение находится в 3-й НФ, но не находится в НФБК, так как содержит два детерминанта, которые не являются потенциальными ключами этого отношения (StNo и IdCode детерминанты, поскольку они определяют друг друга). Как видно, в

отношении MI присутствует доля избыточности, которая имелась и в ранее рассмотренных отношениях (SM и CNR), поэтому оно характеризуется такими же аномалиями обновления. Для решения этой проблемы отношение MI следует разбить на две проекции:

SI {StNo, IdCode} и Marks {StNo, SubjNo, DocNo, Mark}

или другим способом:

SI {StNo, IdCode} и Marks {IdCode, SubjNo, DocNo, Mark} .

Таким образом, присутствуют две, в одинаковой мере допустимые декомпозиции, причем все проекции отношения MI находятся в НФБК. Исходя из соображений здравого смысла, первая декомпозиция лучше, поскольку в учебной БД для идентификации студента используется его код StNo.

9 Лекция №9. Проектирование БД с использованием метода сущность-связь

Цель лекции: изучение метода сущность-связь.

Содержание лекции: возникновение семантического моделирования; основные понятия метода; диаграммы ER-экземпляров и ER-типа.

9.1 Возникновение семантического моделирования

Проектирование реляционной базы данных в терминах отношений на основе механизма нормализации часто представляет собой очень сложный и неудобный для проектировщика процесс. Потребности проектировщиков баз данных в более удобных и мощных средствах моделирования предметной области вызвали к жизни направление семантических моделей данных.

Метод проектирования «сущность-связь» называют также методом «ER-диаграмм»: во-первых, ER –аббревиатура от слов Essence (сущность) и Relation (связь)**,** во-вторых, метод основан на использовании диаграмм, называемых соответственно диаграммами ER-экземпляров и диаграммами ER-типа.

9.2 Основные понятия метода

Основными понятиями метода «сущность-связь» являются следующие:

a) сущность – представляет собой объект, информация о котором хранится в БД (экземпляры сущности отличаются друг от друга и однозначно идентифицируются, названиями сущностей являются, как правило, существительные**,** например: ПРЕПОДАВАТЕЛЬ, ДИСЦИПЛИНА, ГРУППА);

b) атрибут сущности – представляет собой свойство сущности (это понятие аналогично понятию атрибута в отношении; так атрибутами сущности ПРЕПОДАВАТЕЛЬ может быть его Фамилия, Должность, Стаж (преподавательский) и т. д.);

c) ключ сущности – атрибут или набор атрибутов, используемый для идентификации экземпляра сущности (как видно из определения, понятие ключа сущности аналогично понятию ключа отношения);

d) связь между сущностями (связь двух или более сущностей предполагает зависимость между атрибутами этих сущностей, название связи обычно представляется глаголом; примерами связей между сущностями являются следующие: ПРЕПОДАВАТЕЛЬ ВЕДЕТ ДИСЦИПЛИНУ (Иванов ВЕДЕТ «Организацию БД и знаний»), ПРЕПОДАВАТЕЛЬ ПРЕПОДАЕТ В ГРУППЕ (Иванов ПРЕПОДАЕТ В 256 группе));

e) степень связи – является характеристикой связи между сущностями, которая может быть следующих видов: 1:1, 1:M, M:1, M:M.;

f) класс принадлежности (КП) экземпляров сущности **(**КП сущности может быть: обязательным и необязательным - класс принадлежности сущности является обязательным**,** если все экземпляры этой сущности обязательно участвуют в рассматриваемой связи, в противном случае класс принадлежности сущности является необязательным);

g) диаграммы ER-экземпляров;

h) диаграммы ER-типа.

9.3 Диаграммы ER-экземпляров и ER-типа

С целью повышения наглядности и удобства проектирования для представления сущностей, экземпляров сущностей и связей между ними следующие графические средства:

a) диаграммы ER-экземпляров;

b) диаграммы ER-типа, или ER-диаграммы.

На рисунке 9.1 приведена диаграмма ER-экземпляров для сущностей ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА со связью ВЕДЕТ.

Диаграмма ER-экземпляров показывает, какую конкретно дисциплину (СУБД, C++ и т.д.) ведет каждый из преподавателей. На рисунке 9.2 представлена диаграмма ER-типа, соответствующая рассмотренной диаграмме ER-экземпляров.

На начальном этапе проектирования БД выделяются атрибуты, составляющие ключи сущностей.

На основе анализа диаграмм ER - типа формируются отношения проектируемой БД. При этом учитывается степень связи сущностей и класс их принадлежности, которые, в свою очередь, определяются на основе анализа диаграмм ER-экземпляров соответствующих сущностей.

Варьируя классом принадлежности сущностей для каждого из названных типов связи, можно получить несколько вариантов диаграмм ER-типа. Рассмотрим примеры некоторых из них.

9.3.1 Связи типа 1:1 и необязательный класс принадлежности.

В приведенной на рисунке 9.2 диаграмме степень связи между сущностями 1:1, а класс принадлежности обеих сущностей необязательный.

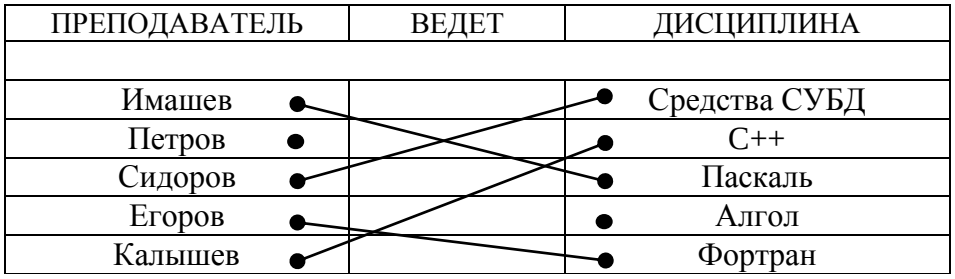

Рисунок 9.1 - Диаграмма ER-экземпляров

Рисунок 9.2 - Диаграмма ER-типа

Действительно, из рисунка 9.1 видно, что каждый преподаватель ведет не более одной дисциплины, а каждая дисциплина ведется не более чем одним преподавателем (степень связи 1:1); некоторые преподаватели не ведут ни одной дисциплины и имеются дисциплины, которые не ведет ни один из преподавателей (класс принадлежности обеих сущностей необязательный).

9.3.2 Связи типа 1:1 и обязательный класс принадлежности.

На рисунке приведены диаграммы, у которых степень связи между сущностями 1:1, а класс принадлежности обеих сущностей обязательный.

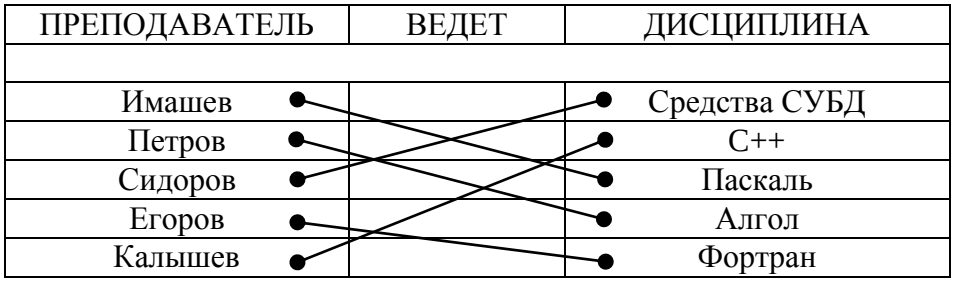

Рисунок 9.3 - Диаграмма ER-экземпляров для связи 1:1 и обязательным КП обеих сущностей

Рисунок 9.4 - Диаграмма ER-типа для связи 1:1 и обязательным КП обеих сущностей

В этом случае каждый преподаватель ведет одну дисциплину и каждая дисциплина ведется одним преподавателем.

Возможны два промежуточных варианта с необязательным классом принадлежности одной из сущностей.

Диаграммы ER-типа графически изображаются следующим образом:

a) обязательное участие в связи экземпляров сущности отмечается блоком с точкой внутри, смежным с блоком этой сущности (см. рисунок 9.4);

b) необязательное участие экземпляров сущности в связи – дополнительный блок к блоку сущности не пристраивается, а точка размещается на линии связи (см. рисунок 9.2);

c) символы на линии связи указывают на степень связи;

d) под каждым блоком, соответствующим некоторой сущности, указывается ее ключ, выделяемый подчеркиванием;

e) многоточие за ключевыми атрибутами означает, что возможны другие атрибуты сущности, но ни один из них не может быть частью ее ключа, эти атрибуты выявляются после формирования отношений.

На практике степень связи и класс принадлежности сущностей при проектировании БД определяется спецификой предметной области. Рассмотрим примеры вариантов со степенью связи 1:M или M:1.

Связь типа 1:M – каждый преподаватель может вести несколько дисциплин, но каждая дисциплина ведется одним преподавателем,

Связи типа M:1 – каждый преподаватель может вести одну дисциплину, но каждую дисциплину могут вести несколько преподавателей.

Примеры с типом связи 1:M или M:1 могут иметь ряд вариантов, отличающихся классом принадлежности одной или обеих сущностей. Обозначим обязательный класс принадлежности символом «O», а необязательный - символом «H», тогда варианты для связи типа 1:M условно можно представить как: O–O, O–H, H–O, H–H. Для связи типа M:1 также имеются 4 аналогичных варианта.

9.3.3 Связи типа 1:M вариант H-O.

Каждый преподаватель может вести несколько дисциплин ИЛИ ни одной, но каждая дисциплина ведется одним преподавателем (см. рисунки 9.5 и 9.6).

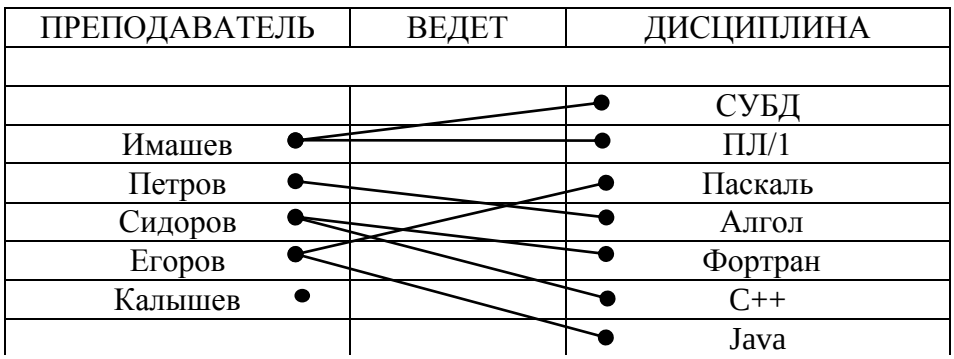

Рисунок 9.5 - Диаграмма ER-экземпляров для связи типа 1:M варианта H-O

Рисунок 9.6 - Диаграмма ER-типа для связи типа 1:M варианта H-O

По аналогии легко составить диаграммы и для остальных вариантов.

Связи типа M:M – каждый преподаватель может вести несколько дисциплин, а каждая дисциплина может вестись несколькими преподавателями. Как и в случае других типов связей, для связи типа M:M возможны 4 варианта, отличающиеся классом принадлежности сущностей.

9.3.4 Связи типа M:M и вариант класса принадлежности O-H.

Допустим, что каждый преподаватель ведет не менее одной дисциплины, а дисциплина может вестись более чем одним преподавателем, есть и такие дисциплины, которые никто не ведет. Соответствующие этому случаю диаграммы приведены на рисунке 9.7.

Выявление сущностей и связей между ними, а также формирование на их основе диаграмм ER-типа выполняется на начальных этапах метода «сущность-связь». Рассмотрим этапы реализации метода.

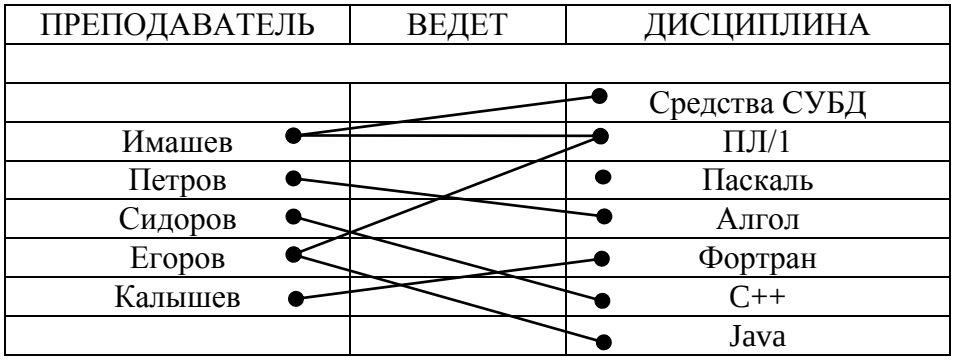

Рисунок 9.7 - Диаграмма ER-экземпляров для связи типа M:M и вариант класса принадлежности O-H.

Рисунок 9.8 - Диаграмма ER-типов для связи типа M : M и варианта O-H

Процесс проектирования базы данных является итерационным – допускающим возврат к предыдущим этапам для пересмотра ранее принятых решений и включает следующие этапы:

а) выделение сущностей и связей между ними;

b) построение диаграмм er-типа с учетом всех сущностей и их связей;

c) формирование на основе построенных ранее диаграмм er-типа набора предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения;

d) добавление не ключевых атрибутов в отношения;

e) приведение предварительных отношений к нормальной форме Бойса-Кодда, например, с помощью метода нормальных форм;

f) пересмотр er-диаграмм в некоторых случаях (некоторые отношения не приводятся к нормальной форме Бойса-Кодда, некоторым атрибутам не находится логически обоснованных мест в предварительных отношениях).

После преобразования ER-диаграмм осуществляется повторное выполнение предыдущих этапов проектирования (возврат к этапу a)).

Одним из узловых этапов проектирования является этап формирования отношений. Рассмотрим процесс формирования предварительных отношений, составляющих первичный вариант схемы БД.

В рассмотренных выше примерах связь ВЕДЕТ всегда соединяет две сущности и поэтому является бинарной**.** Сформулированные ниже правила формирования отношений из диаграмм ER-типа распространяются именно на бинарные связи. Поэтому, когда речь идет о связях, слово «бинарные» далее опускается.

10 Лекция №10. Правила формирования отношений

Цель лекции: изучение правила формирования отношений основе диаграмм ER-типа.

Содержание лекции: правила формирования отношений.

10.1 Правила формирования отношений

Правила формирования отношений основываются на учете следующего:

a) степени связи между сущностями (1:1, 1:M, M:1, M:M);

b) класса принадлежности экземпляров сущностей (обязательный и необязательный).

Рассмотрим формулировки шести правил формирования отношений на основе диаграмм ER-типа.

10.2 Степень связи 1:1, класс принадлежности обеих сущностей обязательный

Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей обязательный, то формируется одно отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

На рисунке 10.1 приведены диаграмма ER-типа и отношение, сформированное на ее основе.

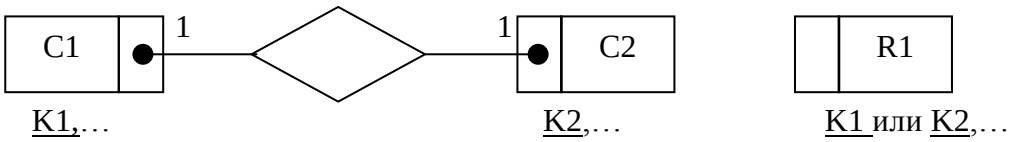

Рисунок 10.1 - Диаграмма и отношение

На рисунке 10.1 используются следующие обозначения -Cl, C2 – сущности 1 и 2; Kl, K2 – ключи первой и второй сущности соответственно; Rl – отношение 1, сформированное на основе первой и второй сущностей; Kl, K2,... означает, что ключом сформированного отношения может быть либо К1, либо К2.

10.3 Степень связи 1:1, класс принадлежности одной сущности обязательный, а второй – необязательный

Если степень связи 1:1 и класс принадлежности одной сущности обязательный, а второй – необязательный, то под каждую из сущностей формируется по отношению с первичными ключами, являющимися ключами соответствующих сущностей. Далее к отношению, сущность которого имеет обязательный КП, добавляется в качестве атрибута ключ сущности с необязательным КП.

На рисунке 10.2 приведены диаграмма ER-типа и отношения, сформированные на ее основе.

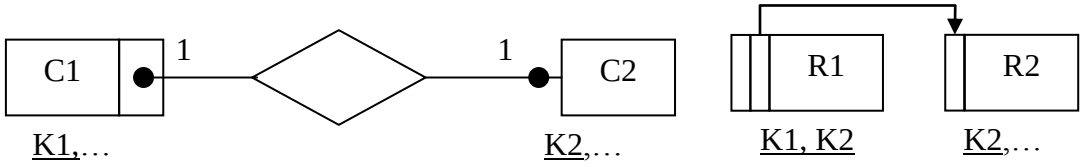

Рисунок 10.2 - Диаграмма и отношения

10.4 Степень связи 1:1, класс принадлежности обеих сущностей – необязательный

Если степень связи 1:1 и класс принадлежности обеих сущностей является необязательным, то необходимо использовать три отношения. Два отношения соответствуют связываемым сущностям, ключи которых являются первичными в этих отношениях. Третье отношение является связным между первыми двумя, поэтому его ключ объединяет ключевые атрибуты связываемых отношений. На рисунке 10.3 приведены диаграмма ER-типа и отношения, сформированные на ее основе.

Сформулируем аналогичные два правила для вариантов, степень связи между сущностями которых 1:M. Если две сущности C1 и C2 связаны как

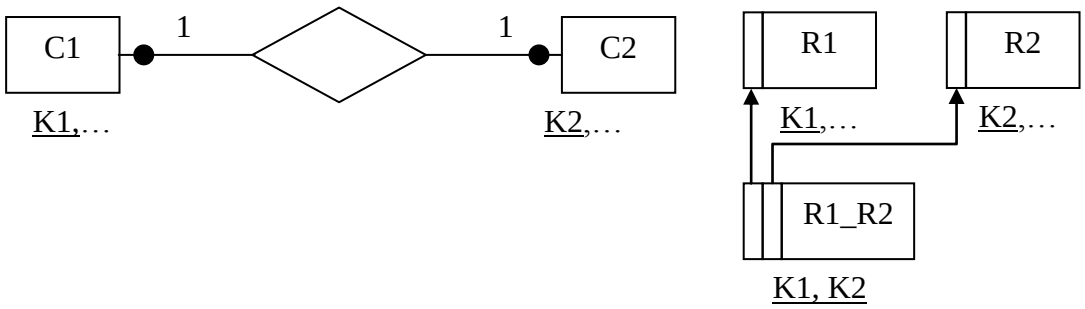

Рисунок 10.3 - Диаграмма и отношения

1:M, сущность C1 будем называть односвязной (1-связной), а сущность C2- многосвязной (M-связной). Определяющим фактором при формировании отношений, связанных этим видом связи, является класс принадлежности M- связной сущности. Так, если класс принадлежности M-связной сущности обязательный, то в результате применения правила получим два отношения, если необязательный – три отношения. Класс принадлежности односвязной сущности не влияет на результат**.**

10.5 Степень связи между сущностями 1:M (или M:1), класс принадлежности M-связной сущности обязательный

Если степень связи между сущностями 1:M (или M:1) и класс принадлежности M-связной сущности обязательный, то достаточно формирование двух отношений (по одному на каждую из сущностей). При этом первичными ключами этих отношений являются ключи их сущностей. Кроме того, ключ 1-связной сущности добавляется как атрибут (внешний ключ) в отношение, соответствующее M-связной сущности.

На рисунке 10.4 приведены диаграмма ER-типа и отношения, сформированные на ее основе.

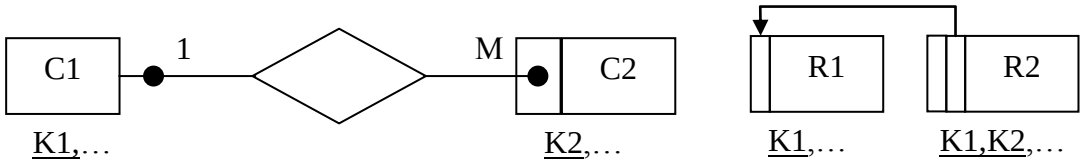

Рисунок 10.4 - Диаграмма и отношения

10.6 Степень связи 1:M (M:1)и класс принадлежности M-связной сущности – необязательный

Если степень связи 1:M (M:1) и класс принадлежности M-связной сущности является необязательным, то необходимо формирование трех отношений (см. рисунок 10.5).

Два отношения соответствуют связываемым сущностям, ключи которых

являются первичными в этих отношениях. Третье отношение является связным между первыми двумя (его ключ объединяет ключевые атрибуты связываемых отношений).

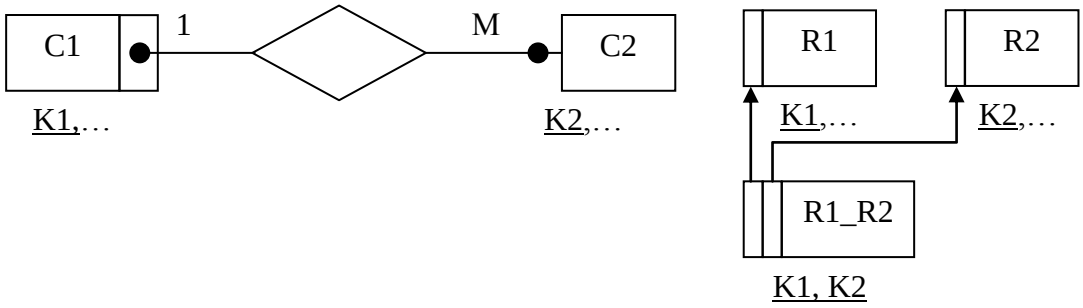

Рисунок 10.5 - Диаграмма и отношения

При наличии связи M:M между двумя сущностями необходимо три отношения независимо от класса принадлежности любой из сущностей. Использование одного или двух отношений в этом случае не избавляет от пустых полей или избыточно дублируемых данных.

10.7 Степень связи M:M, независимо от класса принадлежности сущностей

Если степень связи M:M, то независимо от класса принадлежности сущностей формируются три отношения Два отношения соответствуют связываемым сущностям, и их ключи являются первичными ключами этих отношений. Третье отношение является связным между первыми двумя, а его ключ объединяет ключевые атрибуты связываемых отношений.

На рисунке 10.6 приведены диаграмма ER-типа и отношения, сформированные по правилу 10.6. В конспекте показан вариант с классом принадлежности сущностей H-H, хотя, он может быть произвольным.

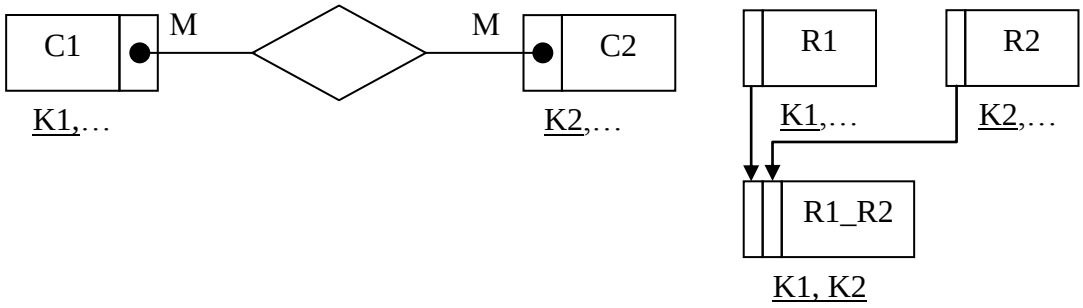

Рисунок 10.6 - Диаграмма и отношения

Аналогичные результаты получаются и для трех других вариантов, различающихся классами принадлежности их сущностей.

11 Лекция №11. Физическая организация базы данных: структуры хранения и методы доступа

Цель лекции: изучение структуры хранения и методов доступа к БД.

Содержание лекции: доступ к базе данных; кластеризация; индексирование; хеширование.

11.1 Доступ к базе данных

Основным способом повышения производительности является минимизация числа дисковых операций ввода-вывода данных**.**

Любое упорядочение (расположение) данных на диске называется структурой хранения. Можно организовать самые разные структуры хранения, обладающие различной производительностью и оптимальные для различных способов использования. Совершенная СУБД должна содержать несколько разных структур хранения для различных частей системы.

Работа СУБД построена следующим образом и включает три основных этапа:

a) сначала в СУБД определяется искомая запись, а затем для ее извлечения запрашивается диспетчер файлов**;**

b) диспетчер файлов определяет страницу**,** на которой находится искомая запись, а затем для извлечения этой страницы запрашивает диспетчер дисков;

c) наконец, диспетчер дисков определяет физическое положение искомой страницы на диске и посылает соответствующий запрос на ввод-вывод данных.

Для выполнения этих операций необходимо знать значения физических адресов на диске. Например, если диспетчер файлов запрашивает некоторую страницу, диспетчеру дисков необходимо знать, где конкретно находится страница на физическом диске. Однако «пользователю» диспетчера дисков, т.е., диспетчеру файлов, не обязательно знать физические адреса. Вместо этого диспетчеру файлов достаточно рассматривать диск как набор страниц фиксированного размера (т.е. набор «ячеек» памяти одинакового размера) с уникальным идентификационным номером набора страниц. Каждая страница, в свою очередь, обладает уникальным внутри данного набора идентификационным номером страницы, причем наборы не имеют общих страниц.

Соответствие физических адресов на диске и номеров страниц достигается с помощью диспетчера дисков. Главным (и не единственным) преимуществом такой организации является изоляция программного кода, зависящего от конкретного устройства диска, внутри одного из компонентов системы, а именно: внутри диспетчера дисков. В таком случае все компоненты высокого уровня, в частности, диспетчера файлов, могут быть аппаратно независимыми.

При работе с диском как набором хранимых файлов диспетчер файлов использует все имеющиеся средства диспетчера дисков согласно определенному в СУБД способу. При этом каждый набор страниц может содержать один или несколько хранимых файлов.

11.2 Кластеризация

В основе технологии кластеризации данных лежит принцип как можно более близкого физического размещения на диске логически связанных между собой и часто используемых данных. Физическая кластеризация данных – чрезвычайно важное условие высокой производительности.

Внутрифайловую и межфайловую кластеризацию СУБД может осуществлять, размещая логически связанные записи на одной странице (если это возможно) или на смежных страницах (в противном случае).

Кластеризация внутри СУБД возможна только в том случае, если администратор базы данных организует ее. В совершенных СУБД часто предусмотрено задание нескольких различных типов кластеризации данных из разных файлов.

11.3 Индексирование

Рассмотрим в качестве примера таблицу с данными о студентах, а также часто используемый и потому очень важный запрос типа: «Найти всех студентов учащихся в группе X», где X – некий параметр.

Для поиска всех студентов из конкретной группы можно применить следующую стратегию: найти в файле групп данную группу, а затем согласно указателям извлечь все соответствующие записи из файла студентов.

Такая стратегия будет более эффективной по сравнению с поиском в файле с данными студентов, поскольку, СУБД известна физическая последовательность записей в файле групп. Кроме того, даже если придется просмотреть файл групп полностью, для такого поиска потребуется гораздо меньше операций ввода-вывода, поскольку физический размер файла групп меньше, чем размер файла с данными студентов из-за меньшего размера записей.

В рассматриваемом примере файл групп называется индексным файлом или индексом по отношению к файлу студентов, и наоборот, файл студентов индексирован (называется индексированным файлом) по отношению к файлу групп.

Индексный файл – это хранимый файл особого типа, в котором каждая запись состоит из двух значений, а именно: данных и указателя. Данные соответствуют некоторому полю (индексному полю) из индексированного файла, а указатель служит для связывания с соответствующей записью индексированного файла. Индексное поле также называется индексным ключом (index key).

Основным преимуществом использования индексов является значительное ускорение процесса выборки или извлечения данных, а основным недостатком – замедление процесса обновления данных, поскольку при каждом добавлении новой записи в индексированный файл потребуется также добавить новый индекс в индексный файл.

Индексы часто называют инвертированными списками. Дело в том, что если файл студентов имеет традиционную структуру списка набора значений полей для каждой записи, то индекс содержит список набора записей для каждого значения индексированного поля.

11.4 Хеширование

Хешированием, хеш-адресацией или хеш-индексированием называется технология быстрого прямого доступа к хранимой записи на основе было заданного значения некоторого поля. При этом не обязательно, чтобы поле ключевым.

Ниже перечислены основные черты этой технологии:

a) каждая хранимая запись базы данных размещается по адресу, который вычисляется с помощью специальной хеш-функции на основе значения некоторого поля данной записи, т.е. хеш-поля, или хеш-ключа. Вычисленный адрес называется хеш-адресом;

b) для сохранения записи в СУБД сначала вычисляется хеш-адрес новой записи, а затем диспетчер файлов помещает эту запись по вычисленному адресу;

c) для извлечения нужной записи по заданному значению хеш-поля в СУБД сначала вычисляется хеш-адрес, а затем диспетчеру файлов посылается запрос для извлечения записи по вычисленному адресу.

Недостатком хеширования является возможность возникновения коллизий**,** т.е. ситуаций, когда две или более различных записи имеют одинаковые адреса.

Допустим, что файл студентов (с записями 100, 200 и т.д.) содержит также запись с номером 1400. При использовании хеш-функции h = StNo mod 13 возникнет коллизия (по адресу 9) с записью 100.

Для разрешения таких коллизий можно использовать значения хеш-функции в качестве адреса не какой-либо записи с данными, а точки привязки. Точка привязки – это начальный адрес цепочки указателей (цепочки коллизий), связывающей вместе все записи или все страницы с записями, которые вызывают коллизии по адресу.

Один из недостатков описанного выше способа хеширования – возрастание числа коллизий с увеличением размера хранимого файла. Это, в свою очередь, может привести к значительному увеличению среднего времени доступа.

12 Лекция №12. Защита баз данных

Цель лекции: изучение методов защиты баз данных.

Содержание лекции: защита баз данных, целостность и сохранность баз данных; методы обеспечения безопасности; избирательное управление доступом; шифрование данных; контрольный след выполняемых операций.

12.1 Защита баз данных. Целостность и сохранность баз данных

Термин безопасность относится к защите данных от несанкционированного доступа, изменения или разрушения данных, а целостность – к точности или истинности данных.

Как при обеспечении безопасности, так и при обеспечении целостности система вынуждена проверить, не нарушают ли выполняемые пользователем действия некоторых правил**.** Эти правила должны быть заданы (обычно администратором базы данных) и сохранены в системном каталоге. Среди многочисленных аспектов проблемы безопасности необходимо отметить следующие:

a) правовые, общественные и этические аспекты (имеет ли право некоторое лицо получить запрашиваемую информацию, например, об оценках студента);

b) физические условия (например, закрыт ли данный компьютер или терминальная комната или защищен каким-либо другим образом);

c) организационные вопросы (например, как в рамках предприятия, обладающего некой системой, организован доступ к данным);

d) вопросы реализации управления (например, если используется метод доступа по паролю, то как организована реализация управления и как часто меняются пароли);

e) аппаратное обеспечение (обеспечиваются ли меры безопасности на аппаратном уровне, например, с помощью защитных ключей или привилегированного режима управления);

f) безопасность операционной системы (например, затирает ли базовая операционная система содержание структуры хранения и файлов с данными при прекращении работы с ними).

12.2 Методы обеспечения безопасности

В современных СУБД поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных, а именно: избирательный подход или обязательный подход.

В случае избирательного управления пользователь обладает различными правами (привилегиями или полномочиями) при работе с разными объектами.

В случае обязательного управления, наоборот, каждому объекту данных

присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

Для того чтобы разобраться, какие правила безопасности к каким запросам доступа применяются, в системе должны быть предусмотрены способы опознания источника этого запроса, т.е. опознания запрашивающего пользователя**.** Поэтому в момент входа в систему от пользователя обычно требуется ввести не только его идентификатор (например, имя или должность), но также и пароль (чтобы подтвердить свои права на заявленные ранее идентификационные данные).

12.2.1 Избирательное управление доступом.

Избирательное управление доступом поддерживается многими СУБД. Избирательное управление доступом поддерживается в языке SQL.В общем случае система безопасности таких СУБД базируется на трех компонентах.

Пользователи. СУБД выполняет любое действия с БД от имени какого-то пользователя. Каждому пользователю присваивается идентификатор – короткое имя, однозначно определяющее пользователя в СУБД. Для подтверждения того, что пользователь может работать с введенным идентификатором используется пароль. Таким образом, с помощью идентификатора и пароля производится идентификация и аутентификация пользователя.

Объекты БД**.** По стандарту SQL2 защищаемыми объектами в БД являются таблицы, представления, домены и определенные пользователем наборы символов. Большинство коммерческих СУБД расширяет список объектов, добавляя в него хранимые процедуры и др. объекты.

Привилегии. Привилегии показывают набор действий, которые возможно производить над тем или иным объектом. Например, пользователь имеет привилегию для просмотра таблицы.

12.2.2 Обязательное управление доступом.

Методы обязательного управления доступом применяются к базам данных, в которых данные имеют достаточно статичную или жесткую структуру, свойственную, например, правительственным или военным организациям. Как уже отмечалось, основная идея заключается в том, что каждый объект данных имеет некоторый уровень классификации**,** например: секретно, совершенно секретно, для служебного пользования и т.д., а каждый пользователь имеет уровень допуска с такими же градациями, что и в уровне классификации. Тогда на основе этих сведений можно сформулировать два очень простых правила безопасности:

a) пользователь имеет доступ к объекту, только если его уровень допуска больше или равен уровню классификации объекта.

b) пользователь может модифицировать объект, только если его уровень допуска равен уровню классификации объекта.

12.3 Шифрование данных

До сих пор подразумевалось, что предполагаемый нелегальный пользователь пытается незаконно проникнуть в базу данных с помощью обычных средств доступа, имеющихся в системе. Теперь следует рассмотреть случай, когда такой пользователь пытается проникнуть в базу данных, минуя систему, т.е. физически перемещая часть базы данных или подключаясь к коммуникационному каналу. Наиболее эффективным методом борьбы с такими угрозами является шифрование данных**,** т.е. хранение и передача особо важных данных в зашифрованном виде.

Для обсуждения основных концепций кодирования данных следует ввести некоторые новые понятия. Исходные (незакодированные) данные называются открытым текстом. Открытый текст шифруется с помощью специального алгоритма шифрования. В качестве входных данных для такого алгоритма выступают открытый текст и ключ шифрования, а в качестве выходных – зашифрованная форма открытого текста, которая называется зашифрованным текстом. Если детали алгоритма шифрования могут быть опубликованы или, по крайней мере, могут не утаиваться, то ключ шифрования обязательно хранится в секрете. Именно зашифрованный текст, который непонятен тем, кто не обладает ключом шифрования, хранится в базе данных и передается по коммуникационному каналу.

12.4 Контрольный след выполняемых операций

Не бывает неуязвимых систем безопасности, поскольку настойчивый потенциальный нарушитель всегда сможет найти способ преодоления всех систем контроля. Поэтому при работе с очень важными данными или при выполнении критических операций возникает необходимость регистрации контрольного следа выполняемых операций. Если, например, противоречивость данных приводит к подозрению, что совершено несанкционированное вмешательство в базу данных, то контрольный след должен быть использован для прояснения ситуации и подтверждения того, что все процессы находятся под контролем. Если это не так, то контрольный след поможет обнаружить нарушителя.

Для сохранения контрольного следа обычно используется особый файл, в котором система автоматически записывает все выполненные пользователями операции при работе с обычной базой данных.

12.5 Поддержка мер обеспечения безопасности в языке SQL

В действующем стандарте языка SQL предусматривается поддержка только избирательного управления доступом. Она основана на двух более или менее независимых частях SQL. Одна из них называется механизмом представлений, который (как говорилось выше) может быть использован для скрытия очень важных данных от несанкционированных пользователей. Другая называется подсистемой полномочий и наделяет одних пользователей

правом избирательно и динамично задавать различные полномочия другим пользователям, а также отбирать такие полномочия в случае необходимости.

Механизм представлений языка SQL позволяет различными способами разделить базу данных на части таким образом, чтобы некоторая информация была скрыта от пользователей, которые не имеют прав для доступа к ней. Однако этот режим задается не с помощью параметров операций, на основе которых санкционированные пользователи выполняют те или иные действия с заданной частью данных. Эта функция выполняется с помощью директивы GRANT.

Создатель объекта также получает право предоставить привилегии доступа какому-нибудь другому пользователю с помощью оператора GRANT. Ниже приводится синтаксис утверждения GRANT:

GRANT {SELECT|INSERT|DELETE|(UPDATE столбец, …)}, …

ON таблица TO {пользователь | PUBLIC} [WITH GRANT OPTION] .

Список литературы

1. Кузнецов С.Д. Основы современных баз данных. Центр Информационных Технологий. http://www.citforum.ru /database/osbd/contents.shtml

2. Кузнецов С.Д. Основы баз данных. 1-е изд. - М.: «Интернет-университет информационных технологий - ИНТУИТ.ру», 2005.

3. Дейт К.Дж. Введение в системы баз данных. - М.: Издательский дом «Вильямс», 2008.

4. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. - Базы данных. Учебник для вузов. – М.: Корона-Принт, 2004.

5. Джоунс Э., Стивенз Р., Плю Р., Гарретт Р., Кригель А. Функции SQL. Справочник программиста. – М.: Диалектика, 2006.

6. Харрингтон Дж. Разработка баз данных. – М.: ДМК Пресс, 2005.

Сводный план 2014г. поз.366

Андрей Григорьевич Бычков

ТЕОРИЯ БАЗ ДАННЫХ

Конспект лекций
для студентов специальности 5В060200 –Информатика

Редактор Н.М.Голева
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать __.__.__.
Тираж 30 экз.
Объем 3,6 уч.-изд. л.

Формат 60Х84 1/16
Бумага типографическая №1
Заказ _____. Цена 1750 т.

Копировально-множительное бюро
Некоммерческого акционерного общества
Алматинского университета энергетики и связи
050013 Алматы, Байтурсынова, 126