Содержание:

Введение

Основные идеи современных информационных технологий основаны на концепции, согласно которой данные должны быть организованы в базы данных, чтобы адекватно отражать изменяющийся реальный мир и удовлетворять информационные потребности пользователей. Эти базы данных создаются и работают под управлением специальных программных систем, называемых системами управления базами данных (СУБД).

Увеличение объема и структурная сложность хранимых данных, расширение круга пользователей информационных систем привели к широкому использованию наиболее удобных и относительно простых для понимания реляционных (табличных) СУБД. Для обеспечения одновременного доступа к данным множества пользователей, зачастую расположенных достаточно далеко друг от друга и от места хранения баз данных, были созданы сетевые многопользовательские версии баз данных на основе реляционной структуры. Так или иначе, они решают специфические проблемы параллельных процессов, целостности (корректности) и безопасности данных, а также авторизации доступа.

Цели данной работы:

-дать основные понятия о базах данных, описать архитектуру СУБД, модели данных;

- раскрыть модель сущность-взаимосвязь, описать характеристики взаимосвязей, классификацию сущностей, структуру первичных и внешних ключей, определить понятие целостности данных;
- описывать реляционную структуру данных, реляционные базы данных и способы управления ими.

Глава 1. Основные понятия БД и СУБД.

1.1 Данные и компьютеры

Восприятие реального мира может быть соотнесено с последовательностью различных, хотя иногда и взаимосвязанных, явлений. С древних времен люди пытались описать эти явления (даже когда не могли их понять). Это описание называется данными.

Традиционно сбор данных осуществляется с использованием определенных средств коммуникации (например, с использованием естественного языка или изображений) на определенном носителе (например, камне или бумаге). Обычно данные (факты, явления, события, идеи или объекты) и их интерпретация (семантика) фиксируются вместе, поскольку естественный язык достаточно гибок, чтобы представлять и то, и другое. Примером может служить выписка «Цена билета 128». Здесь «128» - заданное, а «Цена авиабилета» - его семантика.

Часто данные и интерпретация разделены. Например, «Расписание ВС» можно представить в виде таблицы, вверху которой (отдельно от данных) дается их интерпретация. Такое разделение затрудняет работу с данными (сложно быстро получить информацию из нижней части таблицы).

Использование компьютеров для хранения и обработки данных обычно приводит к еще большему разделению данных и интерпретации. Компьютер в основном имеет дело с данными как таковыми. Большая часть интерпретирующей информации вообще не записывается явно (компьютер не «знает», является ли «21,50» ценой авиабилета или временем вылета). Почему это случилось?

Есть как минимум две исторические причины, по которым использование компьютеров привело к разделению данных и интерпретации. Во-первых, компьютеры не обладали достаточными возможностями для обработки текстов на естественном языке - основном языке интерпретации данных. Во-вторых, стоимость компьютерной памяти изначально была очень высокой. Для хранения самих данных использовалась память, и интерпретация традиционно предоставлялась пользователю. Пользователь поместил интерпретацию данных в

свою программу, которая, например, «знала», что шестое входное значение связано со временем прибытия самолета, а четвертое - со временем его вылета. Это значительно повысило роль программы, поскольку вне интерпретации данные представляют собой не что иное, как набор битов на устройстве памяти.

Жесткая зависимость между данными и использующими их программами создает серьезные проблемы в обслуживании данных и делает их использование менее гибким.

Пользователи одного и того же компьютера нередко создают и используют в своих программах разные наборы данных, содержащие схожую информацию. Иногда это связано с тем, что пользователь не знает (или не хочет знать), что в соседней комнате или за соседним столиком сидит сотрудник, который давно ввел в компьютер необходимые данные. Чаще потому, что при совместном использовании одних и тех же данных возникает множество проблем.

Разработчики прикладных программ (написанных, например, на BASIC, Pascal или C) размещают нужные им данные в файлах, организуя их наиболее удобным для себя способом. При этом одни и те же данные могут иметь совершенно разную организацию в разных приложениях (разная последовательность размещения в записи, разные форматы одних и тех же полей и т. Д.). Публиковать такие данные крайне сложно: например, любое изменение структуры записи файла, сделанное одним из разработчиков, приводит к необходимости для других разработчиков изменять те программы, которые используют записи этого файла.

1.2 Архитектура СУБД

СУБД должна предоставлять доступ к данным любым пользователям, в том числе тем, кто практически не имеет и (или) не хочет знать о:

· Физическое расположение данных и их описания в памяти;

· Механизмы поиска запрашиваемых данных;

· Проблемы, возникающие из-за одновременного запроса одних и тех же данных многими пользователями (прикладными программами); · Способы обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа;

Поддержание баз данных в актуальном состоянии и многие другие функции СУБД.

При выполнении основной из этих функций СУБД должна использовать разные описания данных. Как вы создаете эти описания?

Естественно, начинать проект базы данных следует с анализа предметной области и определения требований к ней отдельных пользователей (сотрудников организации, для которой создается база данных). Более подробно этот процесс будет рассмотрен ниже, но здесь отметим, что проектирование обычно поручается человеку (группе лиц) - администратору базы данных (DBA). Это может быть как специальный сотрудник организации, так и будущий пользователь базы данных, хорошо знакомый с машинной обработкой данных.

Объединив личные представления о содержимом базы данных, полученные в результате интервью с пользователями, и свои представления о данных, которые могут потребоваться в будущих приложениях, администратор баз данных сначала создает обобщенное неформальное описание создаваемой базы данных. Это описание, составленное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем, кто работает над проектированием баз данных, называется инфологической моделью данных

Эта ориентированная на человека модель полностью не зависит от физических параметров среды хранения. В конце концов, этой средой может быть человеческая память, а не компьютер. Следовательно, инфологическая модель не должна изменяться до тех пор, пока некоторые изменения в реальном мире не потребуют изменения какого-либо определения в нем, чтобы эта модель продолжала отражать предметную область.

Остальные модели, показанные на рис. 1, ориентированы на компьютер. С их помощью СУБД позволяет программам и пользователям получать доступ к сохраненным данным только по их именам, не беспокоясь о физическом расположении этих данных. Требуемые данные ищутся СУБД на внешних устройствах хранения в соответствии с физической моделью данных.

Поскольку указанный доступ осуществляется с использованием конкретной СУБД, модели должны быть описаны на языке описания данных этой СУБД. Такое описание, созданное администратором баз данных с использованием инфологической модели данных, называется даталогической моделью данных.

Трехуровневая архитектура (инфологический, логический и физический уровни) позволяет гарантировать независимость хранимых данных от программ, использующих их. Администратор баз данных может при необходимости перезаписать сохраненные данные на другие носители и (или) реорганизовать их физическую структуру, изменив только физическую модель данных. Администратор базы данных может подключать к системе любое количество новых пользователей (новых приложений), добавляя, при необходимости, логическую модель. Указанные изменения в физической и даталогической моделях не будут замечены существующими пользователями системы (они будут «прозрачными» для них), так же как не будут замечены новые пользователи. Таким образом, независимость данных позволяет системе баз данных развиваться без нарушения работы существующих приложений.

1.3 Модели данных

Инфологическая модель отображает реальный мир в виде некоторых понятных человеку концепций, которые полностью не зависят от параметров среды хранения данных. Существует множество подходов к построению таких моделей: модели графов, семантические сети, модель сущность-связь и т. Д. Самым популярным из них была модель сущность связи.

Инфологическая модель должна быть отображена в компьютерно-ориентированную даталогическую модель, «понятную» СУБД. В процессе развития теории и практического использования баз данных, а также вычислительной техники были созданы СУБД, поддерживающие различные логические модели.

Во-первых, они начали использовать иерархические даталогические модели. Простота организации, наличие предопределенных отношений между сущностями и сходство с физическими моделями данных позволило добиться приемлемой производительности иерархической СУБД на медленных компьютерах с очень ограниченными объемами памяти. Но, если данные не имели древовидной структуры, то возникало множество сложностей при построении иерархической модели и желании добиться желаемой производительности.

Также были созданы сетевые модели для компьютеров с низким уровнем ресурсов. Это довольно сложные конструкции, состоящие из «множеств» - двухуровневых деревьев. «Наборы» соединяются с помощью «связанных записей» для формирования цепочек и так далее. При разработке сетевых моделей было придумано множество «маленьких уловок», которые увеличивают производительность СУБД, но значительно усложняют последнюю. Программист приложения должен знать множество терминов, изучить несколько внутренних языков СУБД и подробно понимать логическую структуру базы данных для навигации между различными экземплярами, наборами, записями и т. Д. Один из разработчиков операционной системы UNIX. Система сказала: «Сетевая база - самый верный способ потерять данные».

Сложность практического использования иерархических и сетевых СУБД заставила искать другие способы представления данных. В конце 60-х появились СУБД на основе инвертированных файлов, отличающиеся простотой организации и наличием очень удобных языков манипулирования данными. Однако такие СУБД имеют ряд ограничений на количество файлов для хранения данных, количество связей между ними, длину записи и количество ее полей.

Физическая организация данных оказывает большое влияние на производительность базы данных. Разработчики СУБД стараются создать максимально производительные физические модели данных, предлагая пользователям тот или иной инструментарий для настройки модели под конкретную базу данных. Многообразие методов коррекции физических моделей современных промышленных СУБД не позволяет рассматривать их в этом разделе.

Глава 2. Инфологическая модель данных «Сущность-связь»

2.1 Основные понятия

Цель инфологического моделирования - предоставить человеку наиболее естественные способы сбора и представления информации, которая должна

храниться в создаваемой базе данных. Поэтому они пытаются построить инфологическую модель данных по аналогии с естественным языком (последний не может использоваться в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка). Основными конструктивными элементами инфологических моделей являются сущности, отношения между ними и их свойства (атрибуты).

Сущность - это любой различимый объект (объект, который мы можем отличить от другого), информация о котором должна храниться в базе данных. Сущностями могут быть люди, места, самолеты, полеты, вкус, цвет и т. Д. Необходимо различать такие понятия, как тип сущности и экземпляр сущности. Тип сущности относится к набору однородных индивидов, объектов, событий или идей, которые действуют как единое целое. Экземпляр сущности относится к определенной вещи в коллекции. Например, типом объекта может быть ГОРОД, а экземпляром - Москва, Киев и т. Д.

Атрибут - это именованная характеристика объекта. Его имя должно быть уникальным для определенного типа сущности, но оно может быть одинаковым для разных типов сущностей (например, ЦВЕТ может быть определен для многих сущностей: DOG, CAR, SMOKE и т. Д.). Атрибуты используются для определения того, какую информацию следует собирать об объекте. Примеры атрибутов для объекта ТРАНСПОРТ: ТИП, БРЕНД, НОМЕРНАЯ ТАБЛИЧКА, ЦВЕТ и т. Д. Здесь также существует различие между типом и экземпляром. Тип атрибута COLOR имеет много экземпляров или значений: красный, синий, банановый, ночной белый и т. Д., Но каждому экземпляру объекта присваивается только одно значение атрибута.

Абсолютного различия между типами сущностей и атрибутами нет. Атрибут таков только по отношению к типу сущности. В другом контексте атрибут может действовать как независимый объект. Например, для автомобильного завода цвет - это только атрибут производимого продукта, а для завода по производству красок цвет - это тип объекта.

Ключ - это минимальный набор атрибутов, значения которых можно использовать для однозначного поиска требуемого экземпляра сущности. Минимальность означает, что исключение любого атрибута из набора не позволяет идентифицировать сущность остальными. Для объекта Schedule ключом является

атрибут Flight_number или набор: Departure_point, Departure_time и Destination_point (при условии, что один самолет вылетает из точки в точку за раз).

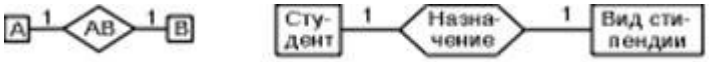
Отношения - это объединение двух или более объектов. Если бы цель базы данных заключалась только в хранении отдельных, несвязанных данных, то ее структура могла бы быть очень простой. Однако одним из основных требований к организации базы данных является обеспечение возможности поиска одних сущностей по значениям других, для чего необходимо установить определенные связи между ними. А поскольку реальные базы данных часто содержат сотни или даже тысячи объектов, теоретически между ними может быть установлено более миллиона связей. Наличие такого набора связей определяет сложность инфологических моделей.

2.2 Описание ссылок и язык моделирования

При построении инфологических моделей можно использовать язык ERдиаграмм (от английского Entity-Relationship, то есть сущность-связь). В них объекты изображены в виде отмеченных прямоугольников, ассоциации - отмечены ромбами или шестиугольниками, атрибуты - отмечены овалами, а связи между ними - в виде ненаправленных ребер, по которым определяется степень связи (1 или буква, заменяющая слово «многие») и можно поставить необходимое объяснение.

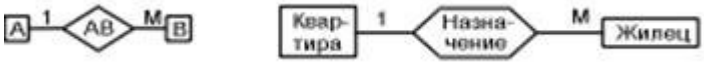
Между двумя объектами, например A и B, возможны четыре типа отношений.

Первый тип - это отношение ОДИН К ОДНОМ (1: 1): в каждый момент времени каждый представитель (экземпляр) объекта A соответствует 1 или 0 представителям объекта B:



Студент не может «зарабатывать» стипендию, получать обычную стипендию или одну из увеличенных стипендий.

Второй тип - это отношение ОДИН К МНОГИМ (1: М): один представитель объекта А соответствует 0, 1 или нескольким представителям объекта В.



Квартира может быть пустой, в ней могут проживать один или несколько жильцов.

Поскольку между двумя сущностями возможны отношения в обоих направлениях, существует еще два типа отношений МНОГИЕ К ОДНОМУ (М: 1) и МНОГИМ К МНОГИМ (М: N).

Пример. Если отношения между сущностями МУЖЧИНА и ЖЕНЩИНА называются БРАКОМ, то есть четыре возможных представления таких отношений:

Характер отношений между организациями не ограничивается перечисленными. Есть и более сложные связи: Множество связей между одними и теми же объектами

(пациент, имеющий одного лечащего врача, может также иметь несколько врачей-консультантов; врач может быть лечащим врачом нескольких пациентов и может одновременно консультировать нескольких других пациентов);

(врач может назначить несколько пациентов на несколько тестов, анализ может быть назначен несколькими врачами нескольким пациентам, а пациент может быть назначен на несколько тестов несколькими врачами);

· Связи более высокого порядка, семантика (значение) которых иногда бывает очень сложной.

2.3 Классификация сущностей

Существует три основных класса сущностей: стержневые, ассоциативные и характеристические, а также подкласс ассоциативных сущностей - обозначения.

Ядро (core) - это независимый объект (более подробно он будет определен ниже).

В рассмотренных выше примерах стержнями являются «Студент», «Квартира», «Мужчины», «Доктор», «Женитьба» и другие, названия которых заключены в прямоугольники. Ассоциативный объект (ассоциация) - это отношение «многие ко многим» («ко многим» и т. Д.) Между двумя или более объектами или экземплярами объектов. Ассоциации считаются полноценными организациями:

- они могут участвовать в других ассоциациях и обозначениях таким же образом, как и основные организации;
- могут иметь свойства, то есть иметь не только набор ключевых атрибутов, необходимых для обозначения взаимосвязей, но также любое количество других атрибутов, характеризующих взаимосвязь.

Характеристический объект (характеристика) - это отношение «многие-кодному» или «один-к-одному» между двумя объектами (особый случай ассоциации). Единственная цель характеристики в рамках рассматриваемой предметной области - описать или уточнить какую-то другую сущность. Потребность в них

возникает из-за того, что объекты реального мира иногда обладают многозначными свойствами. У мужа может быть несколько жен, книга - несколько характеристик переиздания (переработанное, дополненное, исправленное, ...) и т. Д.

Существование характеристики полностью зависит от характеризуемой сущности: женщины лишаются своего статуса жен, если их муж умирает.

Назначающий объект или обозначение - это отношения «многие к одному» или «один к одному» между двумя организациями, которые отличаются от характеристики тем, что не зависят от обозначенного объекта.

В заключение рассмотрим пример построения инфологической модели базы данных «Питание», в которой должна храниться информация о блюдах, их ежедневном потреблении, продуктах, из которых эти блюда готовятся, и поставщиках этих продуктов. Информацию будут использовать шеф-повар и менеджер небольшого заведения общественного питания, а также его посетители.

С помощью этих пользователей определены следующие объекты и характеристики проектируемой базы:

1. Блюда, которые необходимо описать с использованием данных, включенных в их рецепты: номер блюда (например, из книги рецептов), название блюда, тип блюда (закуска, суп, горячее и т. Д.), Рецепт (технология приготовления), выход (вес порции), название, калорийность и вес каждого продукта, входящего в блюдо.
2. Для каждого поставщика товаров: наименование, адрес, наименование поставляемого товара, дата доставки и цена на момент доставки.
3. Ежедневное потребление (потребление) пищи: блюдо, количество порций, дата.

Анализ объектов позволяет выделить:

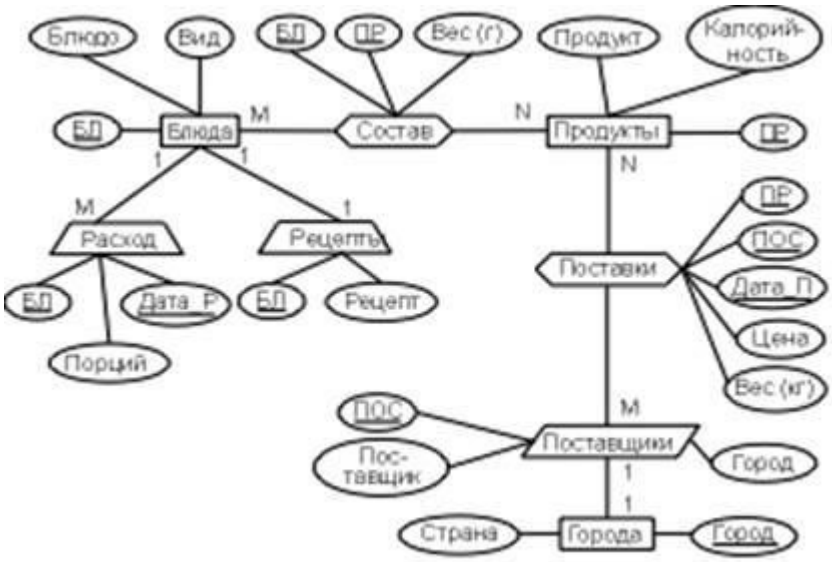
· Жезлы Блюда, Еды и Города;

Состав ассоциаций (связывает блюда с продуктами) и

Поставки (связывает Поставщиков с Продуктами);

· Обозначение Поставщиков;

· Характеристики рецептов и потребления.



2.4 Первичный и внешний ключи

Напомним, что ключ или возможный ключ - это минимальный набор атрибутов, который можно использовать для однозначного поиска требуемого экземпляра объекта. Минимальность означает, что исключение любого атрибута из набора не позволяет идентифицировать сущность остальными. У каждой сущности есть хотя бы один возможный ключ. Один из них принимается за первичный ключ. При выборе первичного ключа предпочтение следует отдавать несоставным ключам или ключам, состоящим из минимального количества атрибутов. Также нецелесообразно использовать ключи с длинными текстовыми значениями (предпочтительны целочисленные атрибуты). Таким образом, для идентификации учащегося можно использовать либо уникальный номер в зачетной книжке, либо набор из фамилии, имени, отчества, номера группы, а также могут быть дополнительные атрибуты, поскольку возможно, что два учащегося (а чаще девушки) с одинаковыми фамилиями, именами и отчествами. Также плохо использовать в качестве ключа не номер блюда, а его название, например, «Сыр плавленый Дружба с ветчиной и солеными огурцами» или «Кролик в сметане с картофельными крокетами и салатом из красной капусты».

Первичный ключ основной сущности (любой атрибут, участвующий в первичном ключе) не может принимать неопределенное значение. В противном случае возникнет противоречивая ситуация: появится неиндивидуализированный и,

следовательно, несуществующий экземпляр базовой сущности. По тем же причинам необходимо убедиться, что первичный ключ уникален.

Теперь о внешних ключах:

Если объект C связывает объекты A и B, то он должен включать внешние ключи, соответствующие первичным ключам объектов A и B.

Если объект B обозначает объект A, то он должен включать внешний ключ, соответствующий первичному ключу объекта A.

Таким образом, при рассмотрении проблемы выбора способа представления ассоциаций и обозначений в базе данных необходимо ответить на главный вопрос: «Что такое внешние ключи?» И далее, для каждого внешнего ключа необходимо решить три вопроса:

1. Может ли данный внешний ключ принимать нулевые значения (значения NULL)? Другими словами, может ли существовать некоторый экземпляр объекта данного типа, для которого целевая сущность, указанная внешним ключом, неизвестна? В случае отгрузки это, вероятно, невозможно - отгрузка от неизвестного поставщика или отгрузка неизвестного продукта не имеет смысла. Но в случае сотрудников такая ситуация, однако, может иметь смысл - вполне возможно, что какой-либо сотрудник в настоящее время вообще не зарегистрирован ни в каком отделе. Обратите внимание, что ответ на этот вопрос не зависит от прихоти разработчика базы данных, но определяется фактическим курсом действий, предпринятым в той части реального мира, которая должна быть представлена в рассматриваемой базе данных. Подобные комментарии относятся к вопросам, обсуждаемым ниже.

1. Что должно произойти при попытке УДАЛИТЬ целевой объект, на который ссылается внешний ключ? Например, при удалении поставщика, осуществившего хотя бы одну доставку. Есть три возможности:

КАСКАДЫ

Операция удаления выполняется "каскадно", чтобы также удалить поставки от этого поставщика.

ОГРАНИЧЕНО

Удаляются только те поставщики, которые еще не осуществили поставки. В противном случае операция удаления отклоняется.

УСТАНОВЛЕНЫ

Для всех поставок удаляемого поставщика значение NULL внешнего ключа устанавливается равным null, а затем поставщик удаляется. Этот вариант, конечно, неприменим, если данный внешний ключ не должен содержать значений NULL.

3. Что должно произойти, если вы попытаетесь ОБНОВИТЬ первичный ключ целевой сущности, на которую ссылается некоторый внешний ключ? Например, может быть сделана попытка обновить номер поставщика, для которого существует по крайней мере одна соответствующая поставка. Для определенности рассмотрим этот случай более подробно. Доступны те же три варианта, что и для удаления:

КАСКАДЫ

Операция обновления выполняется "каскадно", чтобы обновить внешний ключ и в поставках этого поставщика.

ОГРАНИЧЕНО

Только те поставщики, которые еще не поставили, обновляют первичные ключи. В противном случае операция обновления отклоняется.

УСТАНОВЛЕНЫ

Для всех поставок такого поставщика значение NULL внешнего ключа устанавливается равным NULL, а затем обновляется первичный ключ поставщика. Этот вариант, конечно, неприменим, если данный внешний ключ не должен содержать значений NULL.

Таким образом, для каждого внешнего ключа в проекте разработчик базы данных должен указать не только поле или комбинацию полей, составляющих этот внешний ключ и целевую таблицу, которая определяется этим ключом, но и ответы на вышеуказанные вопросы (три ограничения, относящиеся к этому внешнему

ключу).

Наконец, о характеристиках - обозначающих сущности, существование которых зависит от обозначаемого типа.

2.5 Ограничения целостности

Целостность (от англ. Integrity - неприкосновенность,

неприкосновенность, сохранность, целостность) - понимается как верность данных в любое время. Но эта цель может быть достигнута только в определенных пределах: СУБД не может контролировать правильность каждого отдельного значения, введенного в базу данных (хотя каждое значение можно проверить на достоверность). Например, вы не можете найти, что входное значение 5 (представляющее номер дня недели) на самом деле должно быть 3. С другой стороны, значение 9, очевидно, будет неправильным, и СУБД должна его отклонить. Однако для этого ей нужно сказать, что номера дней недели должны принадлежать набору

(1,2,3,4,5,6,7).

Поддержание целостности базы данных можно рассматривать как защиту данных от неправильных изменений или уничтожения (не путать с незаконными изменениями и уничтожением, которые представляют собой проблему безопасности). Современные СУБД имеют ряд инструментов управления целостностью (а также инструменты управления безопасностью).

Есть три группы правил честности:

1. Целостность по сути.
2. Целостность ссылок.

3. Целостность, определяемая пользователем.

В разделе 2.3 обсуждается обоснование двух правил целостности, общих для любой реляционной базы данных.

1. Не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение.
2. Значение внешнего ключа должно:

- быть равным значению первичного ключа цели;
- быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе, должно быть неопределенным.

Для любой конкретной базы данных существует ряд дополнительных конкретных правил, которые применяются только к ней и определяются разработчиком.

Глава 3. Реляционный подход.

3.1 Реляционная структура данных

В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных, т.е. возможности использования привычных и естественных способов представления данных. Наиболее значительной из них была статья сотрудника фирмы IBM д-ра Э.Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), где, вероятно, впервые был применен термин "реляционная модель данных".

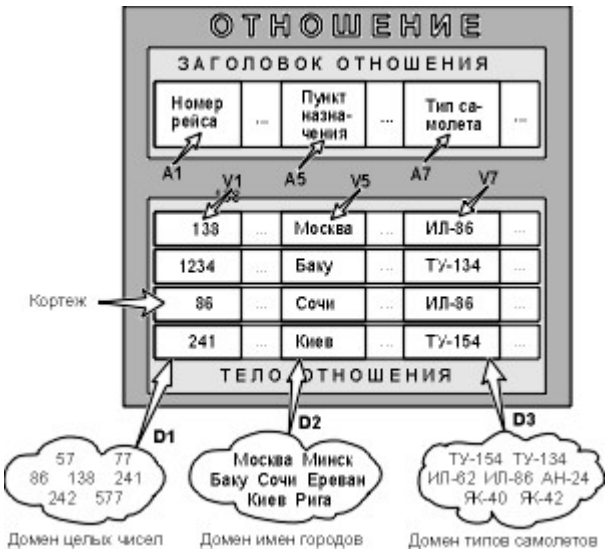
Будучи математиком по образованию Э.Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как *отношение* – relation (англ.).

Наименьшая единица данных реляционной модели – это отдельное *атомарное* (неразложимое) для данной модели значение данных. Так, в одной предметной

области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой – как три различных значения.

Доменом называется множество атомарных значений одного и того же типа. Смысл доменов состоит в следующем. Если значения двух атрибутов берутся из одного и того же домена, то, вероятно, имеют смысл сравнения, использующие эти два атрибута (например, для организации транзитного рейса можно дать запрос "Выдать рейсы, в которых время вылета из Москвы в Сочи больше времени прибытия из Архангельска в Москву"). Если же значения двух атрибутов берутся из различных доменов, то их сравнение, вероятно, лишено смысла: стоит ли сравнивать номер рейса со стоимостью билета? Отношение на доменах D1, D2, ..., Dn (не обязательно, чтобы все они были различны) состоит из заголовка и тела. На рис. 3 приведен пример отношения для расписания движения самолетов.

Заголовок состоит из такого фиксированного множества атрибутов A1, A2, ..., An, что существует взаимно однозначное соответствие между этими атрибутами Ai и определяющими их доменами Di (i=1,2,...,n).



Тело состоит из изменяющегося во времени набора кортежей, где каждый кортеж, в свою очередь, состоит из набора пар атрибут-значение (Ai: Vi), (i = 1,2, ..., n), одна такая пара для каждого атрибута Ai в заголовке. Для любой данной пары атрибут-значение (Ai: Vi) Vi - это значение из одного домена Di, связанного с атрибутом Ai.

Степень связи - это количество ее атрибутов. Отношение первой степени называется унарным, второй степени - двоичным, третьей степени - тернарным, ..., а степени n - n-арным.

Кардинальное число или мощность отношения - это количество его кортежей. Кардинальное число отношения меняется со временем, а не его степень.

Поскольку отношение - это набор, а наборы, по определению, не содержат совпадающих элементов, то никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольный данный момент времени. Пусть R - отношение с атрибутами A1, A2, ..., An. Набор атрибутов K = (Ai, Aj, ..., Ak) отношения R называется возможным ключом R тогда и только тогда, когда выполняются два независимых от времени условия:

1. Единственность: в произвольный данный момент времени никакие два разных набора R не имеют одинакового значения для Ai, Aj, ..., Ak.

1. Минимальность: ни один из атрибутов Ai, Aj, ..., Ak не может быть исключен из K без нарушения единственности.

Каждое отношение имеет по крайней мере один возможный ключ, поскольку по крайней мере комбинация всех его атрибутов удовлетворяет условию уникальности. Один из возможных ключей (выбранный случайным образом) является его первичным ключом. Остальные возможные ключи, если таковые имеются, называются альтернативными ключами.

Вышеупомянутые и некоторые другие математические концепции явились теоретической основой для создания реляционных СУБД, разработки соответствующих языковых инструментов и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных. Однако для обычного пользователя реляционных СУБД неформальные эквиваленты этих концепций могут быть успешно использованы:

Взаимосвязь - таблица (иногда файл),

Кортеж - строка (иногда запись),

Атрибут - Столбец, Поле.

3.2 Реляционная база данных

Реляционная база данных - это набор отношений, содержащий всю информацию, которая должна храниться в базе данных. Однако пользователи могут воспринимать такую базу данных как набор таблиц.

1. Каждая таблица состоит из строк одного типа и имеет уникальное имя.
2. Строки имеют фиксированное количество полей (столбцов) и значений (несколько полей и повторяющиеся группы не допускаются). Другими словами, в каждой позиции ta отличаются друг от друга по крайней мере одним значением: столбцы на пересечении строки и столбца всегда имеют ровно одно значение или ничего.
3. Строки таблицы должны быть уверены, что однозначно идентифицируют любую строку такой таблицы.
4. Имена однозначно присваиваются столбцам таблицы, и в каждый из них помещаются однородные значения данных (даты, фамилии, целые числа или денежные суммы).
5. Полное информационное наполнение базы данных представлено в виде явных значений данных, и этот способ представления является единственным. В частности, нет специальных «ссылок» или указателей, соединяющих одну таблицу с другой. Итак, связь между строкой с BL = 2 таблицы «Блюда» на рис. 4 и строкой с PR = 7 продуктов таблицы (рис нужен для приготовления Харчо) представлена не с помощью указателей, но из-за наличия в таблице «Состав» строки, в которой номер блюда равен 2, а номер продукта - 7.
6. При выполнении операций с таблицей ее строки и столбцы могут обрабатываться в любом порядке независимо от их информативности.

Этому способствует наличие имен таблиц и их столбцов, а также возможность выбора любой из их строк или любого набора строк с заданными характеристиками.

Блюда

БЛ	Блюдо
1	Лобио
2	Харчо
3	Шашлык
4	Кофе

Вид
Закуска
Суп
Горячее
Десерт

Расход

Продукты

ПР	Продукт	Калор.
1	Фасоль	3070
2	Лук	450
3	Масло	7420
4	Зелень	180
5	Мясо	1660
6	Томаты	240
7	Рис	3340
8	Кофе	2750

Рецепты

Состав

БЛ	ПР	Вес (г)
1	1	200
1	2	40
1	3	30
1	4	10
2	5	80
2	2	30
2	6	40
2	7	50
2	3	15
2	4	15
3	5	180
3	6	100
3	2	40
3	4	20

БЛ	Рецепт
1	Ломаную очищ
...	...

БЛ	Порций	Дата_Р
1	158	1/9/94
2	144	1/9/94
3	207	1/9/94

4 235 1/9/ 94

... ...

Поставщики

ПОС	Поставщик
1	"Полесье"
2	"Нат алка"
3	"Хуа нхэ"
4	"Лай ма"
5	"Юрмала"
6	"Даугава"

Города

Город
Киев
Пекин
Рига

Поставки

ПОС	ПР	Вес (кг)	Цена	Дата_П
1	6	1 20	0.45	27/8/94
1	3	5 0	1.82	27/8/94
1	2	5 0	0.61	27/8/94
2	2	1 00	0.52	27/8/94
2	5	1 00	2.18	27/8/94
2	4	10	0.88	27/8/94
3	1	2 50	0.37	24/8/94

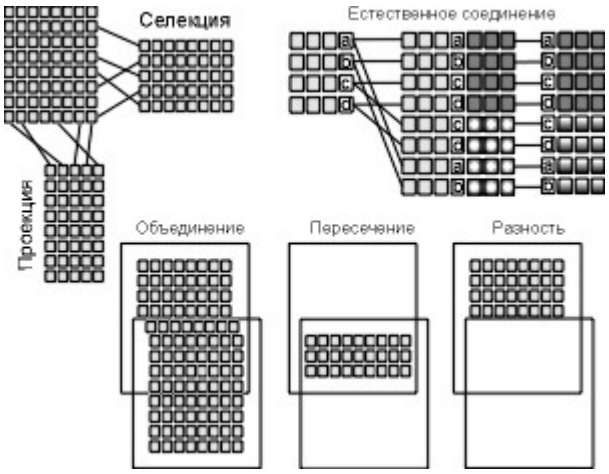
Код	Город	Страна
Киев	Киев	Украина
Пекин	Пекин	Китай
Рига	Рига	Латвия

3	7	7 5		0 .44	2 4/8/ 94			
						4 0	2 .87	2 4/8/ 94
			3	8				
			4	3	7 0		1 .56	3 0/8/ 94
			5	5	2 00		2 .05	3 0/8/ 94
			6	6	1 5		0 .99	3 0/8/ 94

3.3 Управление реляционными данными

Стремление минимизировать количество таблиц для хранения данных может привести к различным проблемам при их обновлении, и будут даны рекомендации по разделению некоторых больших таблиц на несколько маленьких. Но как сформировать требуемый ответ, если необходимые для этого данные хранятся в разных таблицах?

Предложив реляционную модель данных, Э.Ф. Кодд также создал инструмент для удобной работы с отношениями - реляционную алгебру. Каждая операция этой алгебры использует одну или несколько таблиц (отношений) в качестве операндов и в результате создает новую таблицу, т.е. позволяет вам «вырезать» или «склеивать» таблицы.



Созданы языки манипулирования данными, позволяющие реализовать все операции реляционной алгебры и практически любую их комбинацию. Среди них наиболее распространенными являются SQL (язык структурированных запросов) и QBE (Quere-By-Example). Оба являются языками очень высокого уровня, с помощью которых пользователь указывает, какие данные следует получить, не уточняя процедуру их получения.

Заключение

Сегодня реляционные базы данных остаются наиболее

распространенными из-за их простоты и ясности как в процессе создания, так и на уровне пользователя.

Главное преимущество реляционных баз данных - совместимость с наиболее популярным языком запросов SQL. С помощью одного запроса на этом языке вы можете объединить несколько таблиц во временную таблицу и вырезать из нее необходимые строки и столбцы (выделение и проекция). Поскольку табличная структура реляционной базы данных интуитивно понятна для пользователей, язык SQL прост и легок для изучения. Реляционная модель имеет прочную теоретическую основу, на которой были основаны эволюция и реализация реляционных баз данных. На волне популярности, вызванной успехом реляционной

модели, SQL стал основным языком для реляционных баз данных.

В процессе анализа представленной информации были выявлены следующие недостатки рассматриваемой модели базы данных: - поскольку все поля одной таблицы должны содержать постоянное количество полей предопределенных типов, необходимо создавать дополнительные таблицы, учитывающие индивидуальные характеристики элементов с помощью внешних ключей. Такой подход очень затрудняет создание сложных отношений в базе данных;

- высокая сложность манипулирования информацией и изменения связей.

Литература

1. Дейт К. Руководство по реляционной СУБД DB2. - М.: Финансы и статистика, 1988. - 320 с.
2. Кириллов В.В. Основы проектирования реляционных баз данных .Учебное пособие. - СПб.: ИТМО, 1994. - 90 с.
3. Мейер М. Теория реляционных баз данных. -М.: Мир, 1987. - 608 с.

1. Ульман Дж. Базы данных на Паскале. -М.: Машиностроение, 1990. - 386 с.
2. http://www.citforum.ru/database/sql_kg/index.shtml “ Основы проектирования реляционных баз данных ”