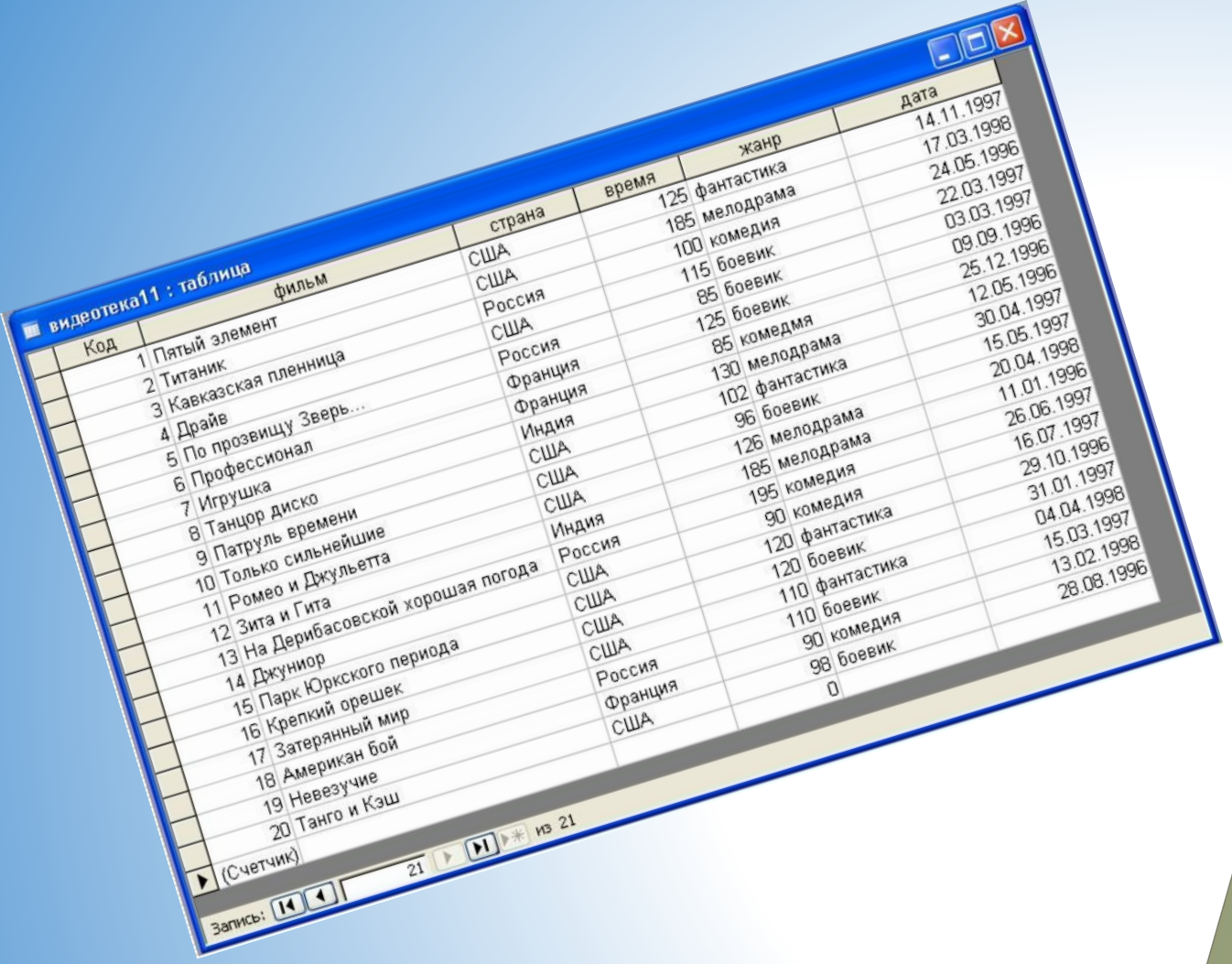


«Основные понятия БД и СУБД»

Тема урока: «Реляционная база данных»

Презентацию подготовил учитель информатики
ГБОУ «Школы № 1905» Панин Геннадий Геннадьевич

Начать показ презентации

06.05.2017

Содержание

База данных

В узком смысле слова, база данных — это некоторый набор данных, необходимых для работы.

Однако данные — это абстракция; никто никогда не видел "просто данные"; они не возникают и не существуют сами по себе. Данные суть отражение объектов реального мира.

В широком смысле слова база данных — это совокупность описаний объектов реального мира и связей между ними, актуальных для конкретной прикладной области.

Возврат в содержание

Основные понятия БД и СУБД

- Восприятие реального мира строится как последовательность разных, иногда и взаимосвязанных, явлений. Еще в древности люди различными способами описывали эти явления, часто даже не понимая их сущности. Сегодня это описание называется данными. К данным относятся: факты, явления, события, идеи или предметы.
- Фиксация данных традиционно происходит через конкретные средства общения - естественного языка или изображений, на конкретном носителе: камне или бумаге. Учитывая тот факт, что естественный язык обладает достаточной гибкостью, данные и их интерпретация (семантика) фиксируются совместно. Например, рассмотрим утверждение "Стоимость билета на электричку 147". Здесь "147" - данное, а "Стоимость билета на электричку" - его семантика.
- Очень часто данные и интерпретация разделяются. Например, "Расписание движения поездов" представляется в виде таблицы, в которой в верхней части отдельно от данных будет приведена их интерпретация, а это затрудняет работу с данными и приводит к сложности получения сведения из нижней части таблицы.
- Разделение данных и интерпретации становиться еще более ощутимым, когда ЭВМ применяется для ввода и обработки данных. Это происходит потому, ЭВМ может иметь дело только с данными. Большая часть интерпретирующей информации не фиксируются в явной форме (ЭВМ не "понимает", является ли "22.50" стоимостью авиабилета или временем вылета). Почему так происходит?
- Существуют как минимум две исторические причины, способствующие тому, что активное использование ЭВМ способствовало тому, что произошло разделение данных и интерпретации:
- - ЭВМ не имело достаточных возможностей, чтобы обрабатывать тексты на естественном языке - основном языке интерпретации данных;
- высокая стоимость памяти ЭВМ.
- Память использовали для того, чтобы хранить сами данные, а интерпретацией занимались непосредственно пользователи. Процесс выглядел следующим образом, интерпретацию данных закладывали в программу, которая "понимала", например, что пятое вводимое значение связано с временем прибытия поезда, а пятое - с временем его убытия. Такая последовательность действий делала программу незаменимой, потому что без интерпретации данные всего лишь совокупность битов на запоминающем устройстве.
- Все серьезные проблемы, которые возникают при введении данных, связаны с тем, что между данными и использующими их программами существует очень жесткая связь.
- Как показывает практика, совместное использование одних и тех же данных, приносит массу проблем. Например, очень часто бывает так, что при использовании одной и той же ЭВМ пользователями создаются и используются в программах разные наборы данных, содержащие сходную информацию. Это можно объяснить тем, что пользователь просто не имеет информации о том, что сотрудник, который работает рядом, давно ввел в ЭВМ нужные данные.
- Очень удобно при использовании, когда разработчики прикладных программ, размещают нужные им данные в файлах. Надо учитывать, что одинаковые данные в разных приложениях отличаются организацией, то есть обладают разной последовательностью размещения в записи, разные форматы одних и тех же полей и т.п. Поэтому обобщить все данные очень сложно. Это связано с тем, что если один разработчик производит изменение структуры записи файла, то и другой разработчик должен произвести изменения в программах, использующих записи этого файла.

Возврат в содержание

Модели данных

Рассмотрим инфологическую модель, отображающую законы реального мира в различные концепции, доступные для понимания человеком. Они абсолютно независимы от параметров среды хранения данных. На сегодняшний день, можно назвать множество существующих подходов к построению таких моделей. Например, графовые модели, семантические сети, модель "сущность-связь" и т.д. Самой распространенной является модель "сущность-связь".

Ее суть в том, что инфологическую модель отображают в компьютероориентированную даталогическую модель, которая "понятна" СУБД. СУБД, которые способны поддерживать различные даталогические модели создавались в процессе интенсивного развития теории и практического использования баз данных, в том числе и средств вычислительной.

Изначально, использовались иерархические даталогические модели, которые были простыми в организации, имели заранее заданные связи между сущностями. Достаточно высокую производительность иерархических СУБД на медленных ЭВМ с весьма ограниченными объемами памяти обеспечивало сходство с физическими моделями данных. При этом если данные не имели древовидной структуры возникало много проблем.

Для мало ресурсных ЭВМ так же создавались сетевые модели. Они представляют собой сложные структуры, которые в свою очередь состоят из наборов" (поименованных двухуровневых деревьев), которые соединяются с помощью "записей-связок", образуя цепочки и т.д. Чтобы увеличить производительность СУБД при разработке сетевых моделей придумали множество "маленьких хитростей". Но они существенно усложнили СУБД. Чтобы ими успешно пользоваться программист должен знать массу терминов, изучить несколько внутренних языков СУБД, детально представлять логическую структуру базы данных для осуществления навигации среди различных экземпляров, наборов, записей и т.п.

Это привело к необходимости поиска иных способов практического использования иерархических и сетевых СУБД. Новые СУБД появились в конце 60-х годов и отличались от предыдущих простотой организации и наличием весьма удобных языков манипулирования данными. Главным недостатком этих СУБД стало ограничение количества файлов для хранения данных, количества связей между ними, длины записи и количества ее полей.

Эксплуатационные характеристики БД зависят от физической организации данных, поэтому разработчики СУБД пытаются создать наиболее производительные физические модели данных. Каждому пользователю предлагается тот или иной инструментарий для под настройки модели под конкретную БД. Так как на сегодняшний день очень много разнообразных способов корректировки физических моделей промышленных СУБД , что не позволяет рассмотреть их в этом разделе.

Возврат в содержание

Классификация СУБД от модели данных

Традиционно все СУБД классифицируются в зависимости от модели данных, которая лежит в их основе. Принято выделять:

- Иерархическую модель данных
- Сетевую модель данных
- Реляционную модель данных

Можно добавлять данные на основе инвертированных списков.

Возврат в содержание

Реляционная модель данных

Реляционной считается такая база данных, в которой все данные представлены для пользователя в виде прямоугольных таблиц значений данных, и все операции над базой данных сводятся к манипуляциям с таблицами. Таблица состоит из строк и столбцов и имеет имя, уникальное внутри базы данных. Таблица отражает тип объекта реального мира, а каждая ее строка — конкретный объект.

Возврат в содержание

Основные понятия базы данных

Так, таблица Деталь содержит сведения о всех деталях, хранящихся на складе, а ее строки являются наборами значений атрибутов конкретных деталей. Каждый столбец таблицы — это совокупность значений конкретного атрибута объекта. Так, столбец Материал представляет собой множество значений "Сталь", "Олово", "Цинк", "Никель". В столбце Количество содержатся целые неотрицательные числа. Значения в столбце Вес — вещественные числа, равные весу детали в килограммах.

Эти значения не появляются из воздуха. Они выбираются из множества всех возможных значений атрибута объекта, которое называется доменом. Так, значения в столбце материал выбираются из множества имен всех возможных материалов — пластмасс, древесины, металлов и т.д. Следовательно, в столбце Материал принципиально невозможно появление значения, которого нет в соответствующем домене, например, "вода" или "песок".

Каждый столбец имеет имя, которое обычно записывается в верхней части таблицы. Оно должно быть уникальным в таблице, однако различные таблицы могут иметь столбцы с одинаковыми именами. Любая таблица должна иметь по крайней мере один столбец; столбцы расположены в таблице в соответствии с порядком следования их имен при ее создании. В отличие от столбцов, строки не имеют имен; порядок их следования в таблице не определен, а количество логически не ограничено.

Возврат в содержание

Основные понятия базы данных

Номер детали	Название детали	Количество	Вес	Материал
1	Втулка	12000	0,8	Сталь
2	Педаль	10000	1,0	Сталь
3	Ступица	5000	0,5	Сталь
4	Передний тормоз	11000	0,5	Алюминий
5	Крыло	300	0,7	Пластмасса

Возврат в содержание

Взаимосвязь таблиц базы данных

Взаимосвязь таблиц является важнейшим элементом реляционной модели данных. Она поддерживается внешними ключами.

Рассмотрим пример, в котором база данных хранит информацию о рядовых служащих (таблица Служащий) и руководителях (таблица Руководитель) в некоторой организации. Первичный ключ таблицы Руководитель — столбец Номер. Столбец Фамилия не может выполнять роль первичного ключа, так как в одной организации могут работать два руководителя с одинаковыми фамилиями. Любой служащий подчинен единственному руководителю, что должно быть отражено в базе данных. Таблица Служащий содержит столбец Номер руководителя, и значения в этом столбце выбираются из столбца Номер таблицы Руководитель. Столбец Номер Руководителя является внешним ключом в таблице Служащий.

Возврат в содержание

Взаимосвязь таблиц базы данных

Возврат в содержание

Таблицы невозможно хранить и обрабатывать, если в базе данных отсутствуют "данные о данных", например, описатели таблиц, столбцов и т.д. Их называют обычно метаданными. Метаданные также представлены в табличной форме и хранятся в словаре данных.

Помимо таблиц, в базе данных могут храниться и другие объекты, такие как экранные формы, отчеты, представления и даже прикладные программы, работающие с базой данных.

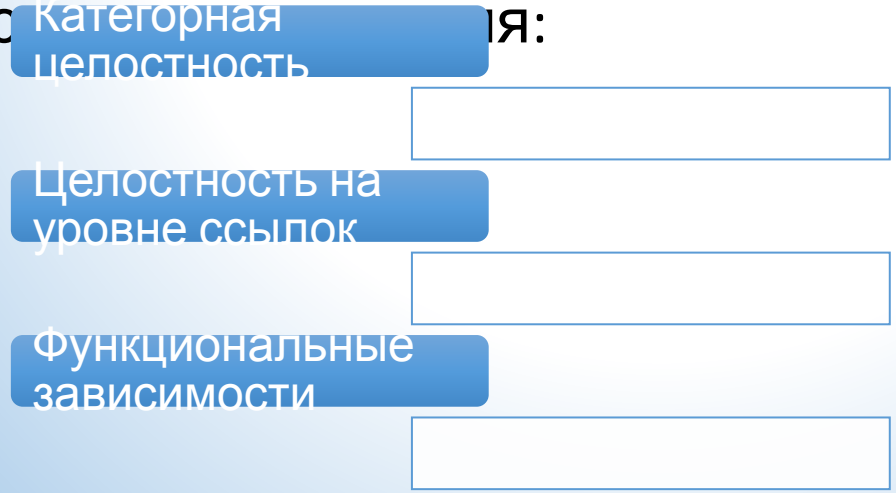
Для пользователей информационной системы недостаточно, чтобы база данных просто отражала объекты реального мира. Важно, чтобы такое отражение было однозначным и непротиворечивым. В этом случае говорят, что база данных удовлетворяет условию целостности.

Для того, чтобы гарантировать корректность и взаимную непротиворечивость данных, на базу данных накладываются некоторые ограничения, которые называют ограничениями целостности.

Возврат в содержание

Ограничительные условия, поддерживающие целостность

В реляционной модели Кодда есть несколько ограничительных условий, используемых для проверки данных в базе данных, а также для придания осмысленности структуре данных. Принято выделять следую ия:



Возврат в содержание

Целостность категории и ссылок

В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД.

Первое требование называется требованием целостности сущности.

Второе требование называется требованием целостности по ссылкам, является более сложным

Возврат в содержание

Операции над реляционными данными

Множество операций над реляционными данными образуют реляционную алгебру. Каждая операция использует одну или две таблицы. Основных операций восемь, которые разбиты на две группы.

Возврат в содержание

Традиционные операции

1) Объединение двух отношений (C1 = A ∪ B) предполагает, что на входе задано два односхемных отношения A и B. Результат объединения есть построенное по той же схеме отношение C, содержащее все кортежи A и все кортежи отношения B.
2) Пересечение двух отношений (C2=A ∪ B) предполагает на входе два односхемных отношения A и B. На выходе создается отношение по той же схеме, содержащее только те кортежи отношения A, которые есть в отношении B.
3) Вычитание двух отношений (C3=A-B). Все три отношения строятся по одной схеме. В результирующее отношение C3 включаются только те кортежи из A, которых нет в отношении B.
4) Декартово произведение (C4=A × B). Ее важное отличие от предшествующих состоит в том, что отношения A и B могут быть построены по разным схемам, а схема отношения C4 включает все атрибуты отношении A и B.

Возврат в содержание

Специальные операции

1) Операция селекция выполняется по строкам. На входе операции используется одно отношение. Результат выборки есть новое отношение, построенное по той же схеме, содержащее подмножество кортежей исходного отношения, удовлетворяющих условию выборки.
2) Операция проекция. На входе операции используется одно отношение. Результирующее отношение включает подмножество атрибутов исходного. Каждому кортежу исходного отношения соответствует такой кортеж в результирующем отношении, что значения одинаковых атрибутов этих двух кортежей совпадают. Но при этом в результирующем отношении кортежи-дубликаты устраняются, в связи с чем мощность результирующего отношения может быть меньше мощности исходного.
3) Операция соединение естественное. На входе операции используется два отношения. В каждом из отношений выделен атрибут, по которому будет осуществляться соединение. Оба атрибута должны быть определены на одном и том же домене. Схема результирующего отношения включает все атрибуты двух отношений. Допускается, чтобы в схеме результирующего отношения вместо двух атрибутов, по которым выполняется соединение, был представлен только один. Операция соединение похожа на декартово произведение.
4) Операция деление. На входе операции используется два отношения A и B. Пусть отношение A, называемое делимым, содержит атрибуты ($A_1, A_2, \ldots, A_n$). Отношение B – делитель -содержит подмножество атрибутов A; положим, ($A_1, A_2, \ldots, A_k$), где ($k<n$). Результирующее отношение C определено на атрибутах отношения A, которых нет и B, т.е. $A_{k+1}, A_{k+2}, \ldots, A_n$. Кортеж включается в результирующее отношение только, если его декартово произведение с отношением B содержится в делимом-отношении A.

Возврат в содержание

Операции реляционной модели данных предоставляют возможность произвольно манипулировать отношениями, позволяя обновлять БД, а также выбирать подмножества хранимых данных и представлять их в нужном виде.

Рассмотренные нами операции реляционной алгебры или алгебры отношений, позволяют пошагово описать процесс получения результирующего отношения.

Возврат в содержание

Нормализация отношений

Одна из важнейших проблем проектирования схемы БД заключается в выделении типов записей, определении состава их атрибутов. Группировка атрибутов должна быть рациональной, т.е. минимизирующей дублирование данных и упрощающей процедуры их обработки и обновления.

Сначала эти вопросы решались интуитивно. Однако интуиция может подвести даже опытного специалиста, поэтому Коддом был разработан в рамках реляционной модели данных аппарат, называемый нормализацией отношений. И хотя идеи нормализации сформулированы в терминологии реляционной модели данных, они в равной степени применимы и для других моделей данных.

Коддом выделено три нормальных формы отношений. Самая совершенная из них - третья. Предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме. В процессе таких преобразований могут выделяться новые отношения.

Возврат в содержание

Первая нормальная форма

Отношение называется нормализованным или приведенным к первой нормальной форме (1НФ), если все его атрибуты простые.

Ненормализованное отношение легко сделать нормализованным. Такое преобразование может привести к увеличению мощности отношения и изменению ключа.

Функциональная зависимость. Пусть X и Y - два атрибута некоторого отношения, Говорят, что Y функционально зависит от X, если в любой момент времени каждому значению X соответствует не более чем одно значение атрибута Y. Функциональную зависимость можно обозначить так: X>Y.

Полная функциональная зависимость. Говорят, что не ключевой атрибут функционально полно зависит от составного ключа, если он функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.

Возврат в содержание

Вторая нормальная форма

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый не ключевой атрибут функционально полно зависит от составного ключа.

Чтобы отношение привести ко второй нормальной форме, необходимо:

- построить его проекцию, исключив атрибуты, которые не находятся в полной функциональной зависимости от составного ключа;
- построить дополнительно одну или несколько проекций на часть составного ключа и атрибуты, функционально зависящие от этой части ключа.

Транзитивная зависимость. Пусть X, Y, Z - три атрибута некоторого отношения. При этом X>Y и Y>Z, но обратное соответствие отсутствует, т. е. Z не> или Y не>X. Тогда говорят, что Z транзитивно зависит от X.

Возврат в содержание

Третья нормальная форма

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Рассматриваемая версия третьей нормальной формы часто называется нормальной формой Бойса-Кодда (НФБК).

Возврат в содержание

Другие нормальные формы

Первая нормальная форма запрещает таблицам иметь неатомарные, или многозначные атрибуты. Однако существует множество ситуаций моделирования, требующих многозначных атрибутов. Например, преподаватель в вузе отвечает за несколько дисциплин. Существует несколько решений, каждое из которых имеет определенные недостатки. Все они требуют лишней памяти из-за наличия пустых значений, либо из-за необходимости вводить избыточные данные. Те из них, в которых есть пустые значения, нарушают категорийную целостность, поскольку все атрибуты вместе составляют ключ таблицы. Эти кажущиеся связи между независимыми атрибутами можно исключить, потребовав, чтобы каждое значение атрибута сочеталось с каждым значением другого атрибута как минимум в одной строке. Условие, обеспечивающее независимость атрибутов путем обязательного повторения значений, называется многозначной зависимостью. Многозначная зависимость является таким же ограничительным условием, как функциональная зависимость. Очевидно, что поскольку они требуют огромного числа повторений значений данных, важный этап процесса нормализации состоит в избавлении от многозначных зависимостей.

Таблица имеет четвертую нормальную форму (4НФ), если она имеет 3НФ и не содержит многозначных зависимостей.

Для избавления от некоторых других аномалий были предложены еще несколько нормальных форм: пятая нормальная форма (5НФ), нормальная форма область/ключ (НФОК) и т.д. Однако они имеют очень ограниченное практическое использование.

Возврат в содержание

Заключение

Можно сказать, что презентация обозначает проблему и объясняет, как можно решить в общем виде эти проблемы. Для того чтобы дать практические рекомендации необходимо придерживаться следующих пунктов:

1. Выбор концептуальной модели, с помощью которой будет построена концептуальная схема;
2. Построение точного описания семантических ограничений, поддерживаемых выбранной СУБД;
3. Построить отображение выбранной концептуальной модели в модель данных, поддерживаемую СУБД.
4. Определить, что такое хорошая схема и описать методику ее построения.

Возврат