Понятие модели данных. Обзор разновидностей моделей данных

С.Д. Кузнецов. Базы данных. Тема 2.

План (1)

- Модель данных
- Ранние модели данных
  - Модель данных инвертированных таблиц
    - Структуры данных
    - Манипулирование данными
    - Ограничения целостности
  - Иерархическая модель данных
    - Иерархические структуры данных
    - Манипулирование данными
    - Ограничения целостности
  - Сетевая модель данных
    - Сетевые структуры данных
    - Манипулирование данными
    - Ограничения целостности

План (2)

- Неформальное введение в реляционную модель данных
  - Реляционные структуры данных
  - Манипулирование реляционными данными
  - Целостность в реляционной модели данных

План (3)

- Современные модели данных
  - Объектно-ориентированная модель данных
    - Типы и структуры данных объектной модели
    - Манипулирование данными в объектной модели
    - Ограничения целостности в объектной модели
  - Модель данных SQL
    - Типы и структуры данных SQL
    - Манипулирование данными в SQL
    - Ограничения целостности в модели SQL
  - Истинная реляционная модель
    - Типы и структуры данных истинной реляционной модели
    - Манипулирование данными в истинной реляционной модели
    - Ограничения целостности в истинной реляционной модели

Модель данных (1)

- В модели данных описывается некоторый набор родовых понятий и признаков, которыми должны обладать все конкретные СУБД и управляемые ими базы данных, если они основываются на этой модели.
- Наличие модели данных позволяет сравнивать конкретные реализации, используя один общий язык.
- Хотя понятие модели данных было введено Коддом, наиболее распространенная трактовка модели данных, по-видимому, принадлежит Кристоферу Дейту.
- Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода:
  - структурной части,
  - манипуляционной части
  - и целостной части.

Модель данных (2)

- В структурной части модели данных фиксируются основные логические структуры данных, которые могут применяться на уровне пользователя при организации БД, соответствующих данной модели.
  - Например, в модели данных SQL основным видом структур базы данных являются таблицы,
  - а в объектной модели данных – объекты ранее определенных типов.
- Манипуляционная часть модели данных содержит спецификацию одного или нескольких языков, предназначенных для написания запросов к БД.
  - Эти языки могут быть абстрактными, не обладающими точно проработанным синтаксисом –
    - это свойственно языками реляционной алгебры и реляционного исчисления, используемым в реляционной модели данных,
  - или законченными производственными языками,
    - как в случае модели данных SQL.
  - Основное назначение манипуляционной части модели данных – обеспечить эталонный «модельный» язык БД, уровень выразительности которого должен поддерживаться в реализациях СУБД, соответствующих данной модели.

Модель данных (3)

- В целостной части модели данных (которая явно выделяется не во всех известных моделях) специфицируются механизмы ограничений целостности, которые обязательно должны поддерживаться во всех реализациях СУБД, соответствующих данной модели.
- Например, в целостной части реляционной модели данных категорически требуется поддержка ограничения первичного ключа в любой переменной отношения,
  - а аналогичное требование к таблицам в модели данных SQL отсутствует.
- Применим понятие модели данных для обзора как подходов, предшествовавших появлению реляционных баз данных, так и для подходов, которые возникли позже.
- Не будем касаться особенностей каких-либо конкретных систем;
  - это привело бы к изложению многих технических деталей, которые, хотя и интересны, находятся несколько в стороне от основной цели курса.

Ранние модели данных (1)

- Начнем с рассмотрения общих подходов к организации трех типов ранних систем, а именно,
  - систем, основанных на инвертированных списках,
  - иерархических
  - и сетевых систем управления базами данных.
- Эти системы активно использовались в течение многих лет, задолго до появления работоспособных реляционных СУБД.
  - 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.

Ранние модели данных (2)

- 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.
  - Абстрактные представления ранних систем появились позже на основе анализа и выявления общих признаков у различных конкретных систем.
- 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.

Ранние модели данных (3)

- Можно считать, что уровень средств ранних СУБД соотносится с уровнем файловых систем примерно так же, как уровень языка Cobol соотносится с уровнем языков ассемблера.
  - Заметим, что при таком взгляде уровень реляционных систем соответствует уровню языков Ада или APL.
- Навигационная природа ранних систем и доступ к данным на уровне записей заставляли пользователей самих 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.

Ранние модели данных (4)
Модель данных инвертированных таблиц (1)

- К числу наиболее известных и типичных представителей систем, в основе которых лежит эта модель данных, относятся
  - СУБД Datacom/DB, выведенная на рынок в конце 1960-х гг. компанией Applied Data Research, Inc. (ADR) и принадлежащая в настоящее время компании Computer Associates,
  - и Adabas (**Ada**ptable **Da**tabase **S**ystem), которая была разработана компанией Software AG в 1971 г. и до сих пор является ее основным продуктом.
- Организация доступа к данным на основе инвертированных таблиц используется практически во всех современных реляционных СУБД, но в этих системах пользователи не имеют непосредственного доступа к инвертированным таблицам (индексам).
  - Когда мы будем рассматривать внутренние интерфейсы реляционных СУБД, можно будет увидеть, что они очень близки к пользовательским интерфейсам систем, основанных на инвертированных таблиц.

Ранние модели данных (5)

Модель данных инвертированных таблиц (2). Структуры данных

- База данных в модели инвертированных таблиц похожа на БД в модели SQL, но с тем отличием, что
  - пользователям видны и хранимые таблицы, и пути доступа к ним.
- Строки таблиц упорядочиваются системой в некоторой физической, видимой пользователям последовательности.
- Физическая упорядоченность строк всех таблиц может определяться и для всей БД (так делается, например, в Datacom/DB).
- Для каждой таблицы можно определить произвольное число ключей поиска, для которых строятся индексы.
  - Эти индексы автоматически поддерживаются системой, но явно видны пользователям.

Ранние модели данных (6)

Модель данных инвертированных таблиц (3). Манипулирование данными (1)

- Поддерживаются два класса операций:

1. Операции, устанавливающие адрес записи и разбиваемые на два подкласса:
   - прямые поисковые операторы (например, установить адрес первой записи таблицы по некоторому пути доступа);
   - операторы, устанавливающие адрес записи при указании относительной позиции от предыдущей записи по некоторому пути доступа.
2. Операции над адресуемыми записями.

Ранние модели данных (7)

Модель данных инвертированных таблиц (4). Манипулирование данными (2)

- Типичный набор операций:
  - **LOCATE FIRST** – найти первую запись таблицы ***T*** в физическом порядке;
    - ✓ возвращается адрес записи;
  - **LOCATE FIRST WITH SEARCH KEY EQUAL** – найти первую запись таблицы ***T*** с заданным значением ключа поиска ***k***;
    - ✓ возвращается адрес записи;
  - **LOCATE NEXT** – найти первую запись, следующую за записью с заданным адресом в заданном пути доступа; возвращается адрес записи;
  - **LOCATE NEXT WITH SEARCH KEY EQUAL** – найти следующую запись таблицы ***T*** в порядке пути поиска с заданным значением ***k***;
    - ✓ должно быть соответствие между используемым способом сканирования и ключом ***k***;
    - ✓ возвращается адрес записи;

Ранние модели данных (8)

Модель данных инвертированных таблиц (5). Манипулирование данными (3)

- **LOCATE FIRST WITH SEARCH KEY GREATER** – найти первую запись таблицы ***T*** в порядке ключа поиска ***k*** со значением ключевого поля, большим заданного значения ***k***;
  - ✓ возвращается адрес записи;
- **RETRIVE** – выбрать запись с указанным адресом;
- **UPDATE** – обновить запись с указанным адресом;
- **DELETE** – удалить запись с указанным адресом;
- **STORE** – включить запись в указанную таблицу;
  - ✓ операция генерирует и возвращает адрес записи.

Ранние модели данных (9)

Модель данных инвертированных таблиц (6). Ограничения целостности

- Общие правила определения целостности БД отсутствуют.
- В некоторых системах поддерживаются ограничения уникальности значений некоторых полей,
  - но в основном вся поддержка целостности данных возлагается на прикладную программу.

Ранние модели данных (10)
Иерархическая модель данных (1)

- Типичным представителем (наиболее известным и распространенным) является СУБД IMS (Information Management System) компании IBM.
- Первая версия системы появилась в 1968 году.

Ранние модели данных (11)

Иерархическая модель данных (2). Структуры данных (1)

- Иерархическая БД состоит из упорядоченного набора *деревьев*;
  - более точно, из упорядоченного набора нескольких экземпляров одного *типа дерева*.
- Тип дерева состоит из одного «корневого» *типа записи* и упорядоченного набора из нуля или более типов поддеревьев,
  - каждое из которых является некоторым типом дерева.
- Тип дерева в целом представляет собой иерархически организованный набор типов записи.

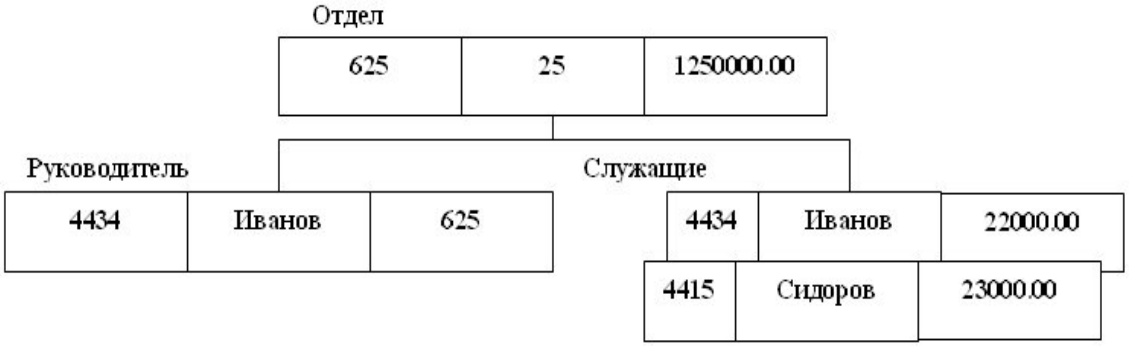
Ранние модели данных (12)

Иерархическая модель данных (3). Структуры данных (2)

- Пример типа дерева (схемы иерархической БД).
  - Тип записи **Отдел** является предком для типов записи Руководитель и Служащие,
  - а **Руководитель** и **Служащие** – потомки типа записи **Отдел**.
  - Поле **Рук_Отдел** типа записи **Руководитель** содержит номер отдела, в котором работает служащий, являющийся данным руководителем (предполагается, что он работает не обязательно в том же отделе, которым руководит).
- Между типами записи поддерживаются связи (правильнее сказать, *типы* связей, поскольку реальные связи появляются в экземплярах типа дерева).

Ранние модели данных (13)

Иерархическая модель данных (4). Структуры данных (3)



- База данных с такой схемой
  - мы показываем один экземпляр дерева.
- Все экземпляры данного типа потомка с общим экземпляром типа предка называются близнецами.
- Для иерархической базы данных определяется полный порядок обхода дерева: сверху-вниз, слева-направо.

Ранние модели данных (14)

Иерархическая модель данных (5). Манипулирование данными

- Примеры типичных операций манипулирования иерархически организованными данными:
  - найти указанный экземпляр типа дерева БД (например, отдел 310);
  - перейти от одного экземпляра типа дерева к другому;
  - перейти от экземпляра одного типа записи к экземпляру другого типа записи внутри дерева (например, перейти от отдела к первому сотруднику);
  - перейти от одной записи к другой в порядке обхода иерархии;
  - вставить новую запись в указанную позицию;
  - удалить текущую запись.

Ранние модели данных (15)

Иерархическая модель данных (6). Ограничения целостности

- В иерархической модели данных автоматически поддерживается целостность ссылок между предками и потомками.
  - Основное правило: *никакой потомок не может существовать без своего родителя.*
- Аналогичная поддержка целостности по ссылкам между записями без связи «предок-потомок», не обеспечивается.
  - Примером такой «внешней» ссылки является содержимое поля **Рук_Отдел** в экземпляре типа записи **Руководитель**.

Ранние модели данных (16)
Сетевая модель данных (1)

- Типичным представителем систем, основанных на сетевой модели данных, является СУБД IDMS (Integrated Database Management System),
  - разработанная компанией Cullinet Software, Inc. и изначально ориентированная на использования на мейнфреймах компании IBM.
- Архитектура системы основана на предложениях Data Base Task Group (DBTG) организации CODASYL (**Co**nference on **Da**ta **Sy**stems **L**anguages),
  - которая отвечала за определение языка программирования COBOL.
- Отчет DBTG был опубликован в 1971 г., и вскоре после этого появилось несколько систем, поддерживающих архитектуру CODASYL, среди которых присутствовала и СУБД IDMS.
- В настоящее время IDMS принадлежит компании Computer Associates.

Ранние модели данных (17)

Сетевая модель данных (2) Сетевые структуры данных (1)

- Сетевой подход к организации данных является расширением иерархического подхода:
  - в иерархических структурах запись-потомок должна иметь в точности одного предка;
  - в сетевой структуре данных у потомка может иметься любое число предков.
- Сетевая БД состоит из набора записей и набора связей между этими записями,
  - более точно, из набора экземпляров каждого *типа* из заданного в схеме БД набора *типов записи* и набора экземпляров каждого *типа* из заданного набора *типов связи*.

Ранние модели данных (18)

Сетевая модель данных (3) Сетевые структуры данных (2)

- 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:
  - каждый экземпляр типа записи ***P*** является предком только в одном экземпляре типа связи ***L***;
  - каждый экземпляр типа записи ***C*** является потомком не более чем в одном экземпляре типа связи ***L***.

Ранние модели данных (19)

Сетевая модель данных (4) Сетевые структуры данных (3)

- На формирование типов связи не накладываются особые ограничения; возможны, например, следующие ситуации:
  - тип записи потомка в одном типе связи ***L1*** может быть типом записи предка в другом типе связи ***L2*** (как в иерархии);
  - данный тип записи ***P*** может быть типом записи предка в любом числе типов связи;
  - данный тип записи ***P*** может быть типом записи потомка в любом числе типов связи;

Ранние модели данных (20)

Сетевая модель данных (5) Сетевые структуры данных (4)

- может существовать любое число типов связи с одним и тем же типом записи предка и одним и тем же типом записи потомка;
  - и если ***L1*** и ***L2*** – два типа связи с одним и тем же типом записи предка ***P*** и одним и тем же типом записи потомка ***C***, то правила, по которым образуется родство, в разных связях могут различаться;
- типы записи ***X*** и ***Y*** могут быть предком и потомком в одной связи и потомком и предком - в другой;
- предок и потомок могут быть одного типа записи.

Ранние модели данных (21)

Сетевая модель данных (6) Сетевые структуры данных (5)

- Простой пример схемы сетевой БД.
- Три типа записей: **Отдел**, **Служащие** и **Руководитель** и три типа связи: **Состоит из служащих**, **Имеет руководителя** и **Является служащим.**
- В типе связи **Состоит из служащих** типом записи-предком является **Отдел**, а типом записи-потомком – **Служащие**
  - экземпляр этого типа связи связывает экземпляр типа записи **Отдел** со многими экземплярами типа записи **Служащие**, соответствующими всем служащим данного отдела.

Ранние модели данных (22)

Сетевая модель данных (7) Сетевые структуры данных (6)

- В типе связи **Имеет руководителя** типом записи-предком является **Отдел**, а типом записи-потомком – **Руководитель**
  - экземпляр этого типа связи связывает экземпляр типа записи **Отдел** с одним экземпляром типа записи **Руководитель**, соответствующим руководителю данного отдела.
- В типе связи **Является служащим** типом записи-предком является **Руководитель**, а типом записи-потомком – **Служащие**
  - экземпляр этого типа связи связывает экземпляр типа записи **Руководитель** с одним экземпляром типа записи **Служащие**, соответствующим тому служащему, которым является данный руководитель.

Ранние модели данных (23)

Сетевая модель данных (8) Манипулирование данными (1)

- Примерный набор операций манипулирования данными:
  - найти конкретную запись в наборе однотипных записей (например, служащего с именем Иванов);
  - перейти от предка к первому потомку по некоторой связи (например, к первому служащему отдела 625);
  - перейти к следующему потомку в некоторой связи (например, от Иванова к Сидорову);

Ранние модели данных (24)

Сетевая модель данных (9) Манипулирование данными (2)

- перейти от потомка к предку по некоторой связи
  - например, найти отдел, в котором работает Сидоров;
- создать новую запись;
- уничтожить запись;
- модифицировать запись;
- включить в связь;
- исключить из связи;
- переставить в другую связь и т.д.

Ранние модели данных (25)

Сетевая модель данных (10) Ограничения целостности

- Имеется (необязательная) возможность потребовать для конкретного типа связи отсутствие потомков, не участвующих ни в одном экземпляре этого типа связи
  - как в иерархической модели.

Неформальное введение в реляционную модель данных (1)

- Основные идеи реляционной модели данных были предложены Эдгаром Коддом в 1969 г.
  - E. F. Codd. Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks . Заново опубликовано в ACM SIGMOD Record, March 2009 (Vol. 38, No. 1).
  - Имеется русский перевод: Э.Ф. Кодд. “Выводимость, избыточность и согласованность отношений, хранимых в крупных банках данных”, http://citforum.ru/database/classics/first_rel_paper/
- Несмотря на общепризнанную значимость этой и последующих работ Кодда, эти работы
  - писались на идейном уровне,
  - не были (по теперешним меркам) глубоко технически проработанными,
  - во многих важных местах допускали неоднозначное толкование.
- Поэтому эти работы невозможно было использовать как непосредственное руководство для реализации СУБД, поддерживающей реляционную модель.
- За прошедшие десятилетия реляционная модель развивалась в двух направлениях.
- Первое направление заложил экспериментальный проект компании IBM System R.
- В этом проекте возник язык SQL, изначально основанный на идеях Кодда, но нарушающий некоторые принципиальные предписания реляционной модели.
- К настоящему времени в действующем стандарте языка SQL, по сути, специфицирована некоторая собственная, законченная модель данных.

Неформальное введение в реляционную модель данных (2)

- Второе направление, начиная с 1990-х гг., возглавляет Кристофер Дейт, к которому позже примкнул Хью Дарвен.
- Оба этих ученых также работали в компании IBM и до 1990-х гг. внесли большой вклад в развитие языка SQL.
- Однако в 1990-е гг. Дейт и Дарвен пришли к выводу, что искажения реляционной модели данных, свойственные языку SQL, достигли настолько высокого уровня, что пришло время предложить альтернативу,
  - опирающуюся на неискаженные идеи Эдгара Кодда и обеспечивающую все возможности
    - как SQL,
    - так и объектно-ориентированного подхода к организации баз данных и СУБД.

Неформальное введение в реляционную модель данных (3)

- Новые идеи Дейта и Дарвена были впервые изложены в их *Третьем манифесте*, а позже на основе этих идей была специфицирована модель данных.
- Авторы считают, что они приводят всего лишь современную и полную интерпретацию идей Кодда.
- С этим можно соглашаться или спорить, но бесспорен один факт – Кодд не участвовал в написании этих материалов и никогда не писал что-либо подобное.
- Тем не менее, далее при обсуждении реляционной модели мы будем использовать, в основном, интерпретацию Дейта и Дарвена.

Неформальное введение в реляционную модель данных (4)
Реляционные структуры данных (1)

- Основная идея Кодда состояла в том, чтобы выбрать в качестве родовой логической структуры хранения данных структуру, которая, с одной стороны, была бы достаточно удобной для большинства приложений и, с другой стороны, допускала бы возможность выполнения над базой данных ненавигационных операций.
- Иерархические и, в особенности, сетевые структуры данных являются навигационными по своей природе.
- Ненавигационному использованию таблиц мешает упорядоченность их столбцов и, в особенности, строк.
- По сути, Кодд предложил использовать в качестве родовой структуры БД «таблицы», в которых и столбцы, и строки не являются упорядоченными

Неформальное введение в реляционную модель данных (5)
Реляционные структуры данных (2)

- Такая «таблица» со множеством столбцов $\{A_1, A_2, \ldots, A_n\}$, в которой каждый столбец A_i может содержать значения из множества $T_i = \{v_{i1}, v_{i2}, \ldots, v_{im}\}$ (все множества конечны), в математическом смысле представляет собой отношение над множествами $\{T_1, T_2, \ldots, T_n\}$.
- В математике отношением над множествами $\{T_1, T_2, \ldots, T_n\}$ называется подмножество декартова произведения этих множеств, т.е. некоторое множество кортежей $\{\{v_1, v_2, \ldots, v_n\}\}$, где $v_i \in T_i$.
- Поэтому для обозначения родовой структуры Кодд стал использовать термин *отношение (relation)*, а для обозначения элементов отношения – термин *кортеж*.
- Соответственно, модель данных получила название *реляционной модели*.

Неформальное введение в реляционную модель данных (6)
Реляционные структуры данных (3)

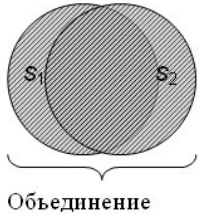
- Схема БД в реляционной модели данных – это набор именованных *заголовков отношений* вида $H_i = \{<A_i^1, T_i^1>, < A_i^2, T_i^2 >, \ldots, < A_i^{ni}, T_i^{ni} >\}$.
- T_i называется *доменом* атрибута A_i.
- По Кодду, каждый домен T_i является подмножеством значений некоторого базового типа данных T_i^+, а значит, к его элементам применимы все операции этого базового типа
  - в конце 1960-х гг. базовыми типами данных считались типы данных распространенных тогда языков программирования;
  - в IBM наиболее популярными языками были PL1 и COBOL.

Неформальное введение в реляционную модель данных (7)
Реляционные структуры данных (4)

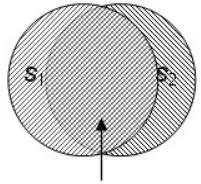
- Реляционная база данных в каждый момент времени представляет собой набор именованных отношений, каждое из которых обладает заголовком, таким как он определен в схеме БД, и телом.
- Имя отношения R_i совпадает с именем заголовка этого отношения H_{Ri}.
- Тело отношения B_{Ri} – это множество кортежей вида {<A_i^1, T_i^1, v_i^1>, < A_i^2, T_i^2, v_i^2 >, …, < A_i^{ni}, T_i^{ni}, v_i^{ni}>}, где $v_i^j \in T_i^j$.
- Во время жизни БД тела отношений могут изменяться, но все содержащиеся в них кортежи должны соответствовать заголовкам соответствующих отношений.

Неформальное введение в реляционную модель данных (8)
Манипулирование реляционными данными (1)

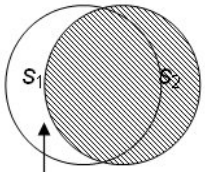
- К отношениям, вообще говоря, применимы обычные теоретико-множественные операции: объединение, пересечение, вычитание, взятие декартова произведения.
- Для двух множеств S_1 $\{s_1\}$ и S_2 $\{s_2\}$ результатом операции объединения этих двух множеств **S_1 UNION S_2** является множество S $\{s\}$ такое, что $s \in S_1$ *или* $s \in S_2$.
- Результатом операции пересечения **S_1 INTERSECT S_2** является множество S $\{s\}$ такое, что $s \in S_1$ *и* $s \in S_2$.
- Результатом операции вычитания **S_1 MINUS S_2** является множество S $\{s\}$ такое, что $s \in S_1$ *и* $s \notin S_2$



Объединение



Пересечение



Вычитание

Неформальное введение в реляционную модель данных (9)
Манипулирование реляционными данными (2)

- Понятно, что эти операции применимы к любым телам отношений, но результатом не будет являться отношение, если у отношений-операндов не совпадают заголовки.
- Кодд предложил в качестве средства манипулирования реляционными базами данных специальный набор операций, которые гарантированно производят отношения.
- Этот набор операций принято называть *реляционной алгеброй Кодда*, хотя он и не является *алгеброй* в математическом смысле этого термина, поскольку некоторые бинарные операции этого набора применимы не к произвольным парам отношений.

Неформальное введение в реляционную модель данных (10)
Манипулирование реляционными данными (3)

- В алгебре Кодда имеется десять операций:
  - объединение (**UNION**),
  - пересечение (**INTERSECT**),
  - вычитание (**MINUS**),
  - взятие расширенного декартова произведения (**TIMES**),
  - переименование атрибутов (**RENAME**),
  - проекция (**PROJECT**),
  - ограничение (**WHERE**),
  - соединение (***Θ*-JOIN**),
  - деление (**DIVIDE BY**)
  - и присваивание.
- Если не вдаваться в некоторые тонкости, то почти все операции этого набора обладают очевидной и простой интерпретацией.

Неформальное введение в реляционную модель данных (11)
Манипулирование реляционными данными (4)

- При выполнении операции *объединения* (**UNION**) двух отношений с одинаковыми заголовками производится отношение, включающее все кортежи, входящие хотя бы в одно из отношений-операндов.
- Операция *пересечения* (**INTERSECT**) двух отношений с одинаковыми заголовками производит отношение, включающее все кортежи, входящие в оба отношения-операнда.
- Отношение, являющееся *разностью* (**MINUS**) двух отношений с одинаковыми заголовками, включает все кортежи, входящие в отношение-первый операнд, такие, что ни один из них не входит в отношение, являющееся вторым операндом.

Неформальное введение в реляционную модель данных (12)
Манипулирование реляционными данными (5)

- При выполнении *декартова произведения* (**TIMES**) двух отношений, пересечение заголовков которых пусто, производится отношение, кортежи которого производятся путем объединения кортежей первого и второго операндов.
- Операция *переименования* (**RENAME**) производит отношение, тело которого совпадает с телом операнда, но имена атрибутов изменены;
  - эта операция позволяет выполнять первые три операции над отношениями с «почти» совпадающими заголовками (совпадающими во всем, кроме имен атрибутов) и выполнять операцию **TIMES** над отношениями, пересечение заголовков которых не является пустым.

Неформальное введение в реляционную модель данных (13)
Манипулирование реляционными данными (6)

- Результатом *ограничения* (**WHERE**) отношения по некоторому условию является отношение, включающее кортежи отношения-операнда, удовлетворяющее этому условию.
- При выполнении *проекции* (**PROJECT**) отношения на заданное подмножество множества его атрибутов производится отношение, кортежи которого являются соответствующими подмножествами кортежей отношения-операнда.
- При *Θ-соединении* (***Θ*-JOIN**) двух отношений по некоторому условию (*Θ*) образуется результирующее отношение, кортежи производятся путем объединения кортежей первого и второго отношений и удовлетворяют этому условию.

Неформальное введение в реляционную модель данных (14)
Манипулирование реляционными данными (7)

- У операции *реляционного деления* (**DIVIDE BY**) два операнда – бинарное и унарное отношения.
  - Результирующее отношение состоит из унарных кортежей, включающих значения первого атрибута кортежей первого операнда таких, что множество значений второго атрибута (при фиксированном значении первого атрибута) включает множество значений второго операнда.
- Операция *присваивания* (:=) позволяет сохранить результат вычисления реляционного выражения в существующем отношении БД.

Неформальное введение в реляционную модель данных (15)
Целостность в реляционной модели данных (1)

- Кодд предложил два декларативных механизма поддержки целостности реляционных баз данных, которые затвержены в реляционной модели данных и должны поддерживаться в любой реализующей ее СУБД:
  - *ограничение целостности сущности*, или *ограничение первичного ключа* и
  - *ограничение ссылочной целостности*, или *ограничение внешнего ключа.*

Неформальное введение в реляционную модель данных (16)
Целостность в реляционной модели данных (2)

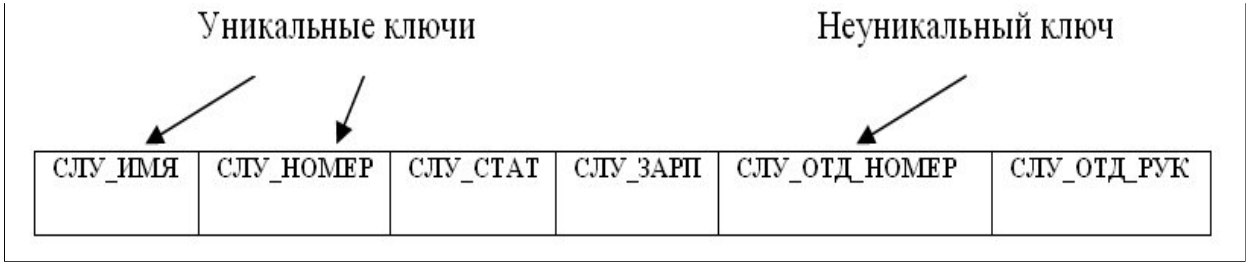
- *Ограничение целостности сущности* звучит следующим образом: для заголовка любого отношения базы данных должен быть явно или неявно определен *первичный ключ*, являющийся таким минимальным подмножеством заголовка отношения, что в любом теле этого отношения, которое может появиться в базе данных, значение первичного ключа в любом кортеже этого тела является уникальным, т.е. отличается от значения первичного ключа в любом другом кортеже.
- Под минимальностью первичного ключа понимается то, что если из множества атрибутов первичного ключа удалить хотя бы один атрибут, то ограничение целостности изменится, т.е. в БД смогут появляться тела отношений, которые не допускались исходным первичным ключом.

Неформальное введение в реляционную модель данных (17)
Целостность в реляционной модели данных (3)

- Если первичный ключ не объявляется явно, то в качестве первичного ключа отношения принимается весь его заголовок.
- Поскольку по определению любое тело отношения с заданным заголовком является множеством, следовательно, в нем отсутствуют дубликаты, и первичный ключ, совпадающий с заголовком отношения, всегда обладает свойством уникальности.
- Должно быть понятно, что в этом случае определение первичного ключа не задает никакого ограничения целостности.

Неформальное введение в реляционную модель данных (18)
Целостность в реляционной модели данных (4)

- Чтобы пояснить смысл *ограничения ссылочной целостности*, нужно сначала ввести понятие *внешнего ключа*.
- В принципе при использовании реляционной модели данных можно хранить все данные, соответствующие предметной области в одной таблице.



- Такой подход приводит к избыточности хранения (данные об отделе повторяются в каждой записи о служащем этого отдела) и усложняет выполнения некоторых операций.

Неформальное введение в реляционную модель данных (19)
Целостность в реляционной модели данных (5)

- Два файла; в одном данные, индивидуальные для каждого служащего, а во втором – данные об отделах.
- В файле **СЛУЖАЩИЕ** – поле **СЛУ_ОТД_НОМЕР**, для каждого служащего его уникальный номер отдела.
- В файле **ОТДЕЛЫ** – поле **ОТД_НОМЕР**, являющееся уникальным ключом этого файла.
- Введя файлы **СЛУЖАЩИЕ** и **ОТДЕЛЫ**, а также обеспечив связь между ними с помощью полей **СЛУ_ОТД_НОМЕР** и **ОТД_НОМЕР**, мы смогли обеспечить табличное представление иерархии **ОТДЕЛ-СЛУЖАЩИЕ**.
- В терминах реляционной модели данных в отношении **ОТДЕЛЫ** поле **ОТД_НОМЕР** является *первичным ключом*, а в отношении **СЛУЖАЩИЕ** поле **СЛУ_ОТД_НОМЕР** является *внешним ключом*, ссылающимся на **ОТДЕЛЫ**.

Неформальное введение в реляционную модель данных (20)
Целостность в реляционной модели данных (6)

- Более точно, *внешним ключом отношения R_1, ссылающимся на отношение R_2*, называется подмножество заголовка H_{R1}, которое совпадает с первичным ключом отношения R_2 (с точностью до имен атрибутов).
- Тогда ограничение ссылочной целостности реляционной модели данных можно сформулировать следующим образом:
  - в любом теле отношения R_1, которое может появиться в базе данных, для «не пустого» значения внешнего ключа, ссылающегося на отношение R_2, в любом кортеже этого тела должен найтись кортеж в теле отношения R_2, которое содержится в базе данных, с совпадающим значением первичного ключа.
- Легко заметить, что это почти то же самое ограничение, о котором говорилось в связи с иерархической моделью данных:
  - *никакой потомок не может существовать без своего родителя*, но немного уточненное –
  - *ссылки на родителя должны быть корректными.*

Современные модели данных (1)

- История современных моделей данных началась с 1989 г., когда группа известных специалистов в области языков программирования баз данных опубликовала статью под названием «Манифест систем объектно-ориентированных баз данных» *(Первый манифест).*
  - Malcolm Atkinson, Francois Bancilhon, David DeWitt, Klaus Dittrich, David Maier, and Stanley Zdonik: “The Object-Oriented Database System Manifesto”, Proc. 1st International Conference on Deductive and Object-Oriented Databases, Kyoto, Japan (1989). New York, N.Y.: Elsevier Science (1990).
  - Имеется русский перевод: М. Аткинсон и др. “Манифест систем объектно-ориентированных баз данных”, *СУБД,* No. 4, 1995, http://citforum.ru/database/classics/oo_manifesto/
- К этому времени уже существовало несколько реализаций объектно-ориентированных СУБД (ООСУБД), но каждая из них опиралась на некоторое расширение объектной модели какого-либо объектно-ориентированного языка программирования (Smalltalk, Object Lisp, C++),
  - отсутствовали какие-либо общие подходы.

Современные модели данных (2)

- В *Первом манифесте* не предлагалась единая объектно-ориентированная модель данных, но выделялся набор требований к ООСУБД.
- Базовыми требованиями являлось:
  - преодоление несоответствия между типами данных, используемыми в языках программирования, и типами данных, поддерживаемыми в набравших к тому времени силу реляционных (вернее, SQL-ориентированных) СУБД,
  - а также придание СУБД возможностей хранить в БД данные произвольно сложной структуры.
- Эти требования сопровождались утверждениями об ограниченности реляционной модели данных и языка SQL и потребности использовать более развитые модели данных.
- Под влиянием *Первого манифеста* в 1991 г. возник консорциум ODMG (Object Database Management Group), задачей которого была разработка стандарта объектно-ориентированной модели данных.
- В течение более чем десятилетнего существования ODMG опубликовала три базовых версии стандарта, последняя из которых называется ODMG 3.0.
- На этот документ мы и будем опираться в дальнейшем изложении.

Современные модели данных (3)

- В ответ на публикацию *Первого манифеста* группа исследователей, близких к индустрии баз данных, в 1990 г. опубликовала документ «Манифест систем баз данных третьего поколения» (*Второй манифест*),
  - который во многом направлен на защиту инвестиций крупных компаний-производителей программного обеспечения SQL-ориентированных СУБД.
  - M. Stonebraker, L. Rowe, B. Lindsay, J. Gray, M. Carey, M. Brodie, Ph. Bernstein, D. Beech. “Third-Generation Data Base System Manifesto”. Proc. IFIP WG 2.6 Conf. on Object-Oriented Databases, July 1990, *ACM SIGMOD Record 19,* No. 3 (September 1990).
  - Имеется русский перевод: Стоунбрейкер М. и др. “Системы баз данных третьего поколения: Манифест”, *СУБД,* No. 2, 1996, http://citforum.ru/database/classics/manifest/
- Соглашаясь с авторами *Первого манифеста* относительно потребности обеспечения развитой системы типов данных в СУБД, авторы *Второго манифеста* утверждали, что можно добиться аналогичных результатов,
  - не производя революцию в области технологии баз данных,
  - а эволюционно развивая технологию SQL-ориентированных СУБД.

Современные модели данных (4)

- За публикацией *Второго манифеста* последовало появление *объектно-реляционных* продуктов ведущих компаний-поставщиков SQL-ориентированных СУБД (Informix Universal Server, Oracle8, IBM DB2 Universal Database).
- В 1999 г. был принят стандарт языка SQL (SQL:1999), в котором был зафиксирован ряд новых черт языка, придающих ему черты полноценной модели данных.
- В последнем принятом стандарте SQL:2003 эта модель уточнена и расширена.
- В пятой части курса мы достаточно подробно обсудим стандарт SQL, а здесь остановимся лишь на некоторых особенностях модели данных SQL, отличающих ее от реляционной модели данных.

Современные модели данных (5)

- Итак, в начале 1990-х гг. были провозглашены два манифеста, каждый из которых претендовал на роль программы будущего развития технологии баз данных.
  - В первом манифесте реляционная модель данных отвергалась полностью,
  - а во втором заменялась еще незрелой к тому времени моделью данных SQL, которая уже тогда была далека от реляционной модели.
- На защиту реляционной модели данных в ее первозданном виде встали Кристофер Дейт и Хью Дарвен, опубликовавшие в 1995 г. статью, под названием «Третий манифест»
  - Hugh Darwen and C. J. Date: *The Third Manifesto. ACM SIGMOD Record 24,* No. 1 (March 1995).
  - Имеется русский перевод: Х. Дарвин, К. Дейт. “Третий манифест”, *СУБД,* No. 1, 1996, http://citforum.ru/database/classics/third_manifesto/

Современные модели данных (6)

- *Третий манифест* являлся одновременно наиболее консервативным и наиболее радикальным.
- Консервативность *Третьего манифеста* заключается в том, что его авторы всеми силами утверждают необходимость и достаточность использования в системах базах данных следующего поколения классической реляционной модели данных.
- Радикальность состоит в том, что
  - авторы полностью отрицают подходы, предлагаемые в первых двух манифестах, расценивая их как необоснованные, плохо проработанные, избыточные и даже вредные (за исключением одной общей идеи о потребности обеспечения развитой системы типов);
  - фактически, авторы полностью отбрасывают технологию, созданную индустрией баз данных за последние 25 лет, и предлагают вернуться к истокам реляционной модели данных, т.е. начальным статьям Э. Кодда.

Современные модели данных (7)

- Позже Дейт и Дарвен написали книгу, первое издание которой вышло в 1998 г. под названием
  - «Foundation for Object/Relational Databases: The Third Manifesto»,
- второе – в 2000 г. под названием
  - «Foundation for Future Database Systems: The Third Manifesto» (имеется перевод второго издания на русский язык)
- и третье – под названием
  - «Databases, Types and the Relational Model: The Third Manifesto» в 2006 г.
- В этих книгах очень подробно излагается подход авторов к построению СУБД на основе, как они утверждают, истинных идей Эдгара Кодда, изложенных им в своих первых статьях про реляционную модель данных.
- Некоторые более поздние идеи Кодда относительно той же реляционной модели авторами отвергаются.
- В любом случае, Кодд и Дарвен предлагают некоторый современный вариант реляционной модели данных
  - далее для определенности мы будем называть ее истинной реляционной моделью,
- который, безусловно, заслуживает внимания и изучения.
- В этом курсе мы ограничимся только кратким очерком основных черт этой модели.

Современные модели данных (8)
Объектно-ориентированная модель данных (1)

- Если не обращать внимания на особенности объектно-ориентированной терминологии, то объектно-ориентированная модель данных отличается от других двух моделей, описываемых в этом разделе, прежде всего, в одном принципиальном аспекте.
  - В модели данных SQL и истинной реляционной модели данных база данных представляет собой набор именованных контейнеров данных одного родового типа: таблиц или отношений соответственно.
  - В объектно-ориентированной модели данных база данных – это набор объектов (контейнеров данных) произвольного типа.

Современные модели данных (9)

ОО-модель данных (2). Типы и структуры данных (1)

- В объектной модели данных вводятся две разновидности типов: *литеральные* и *объектные* типы.
- Литеральные типы данных – это обычные типы данных, принятые в традиционных языках программирования.
- Они подразделяются на
  - базовые скалярные числовые типы,
  - символьные и булевские типы (*атомарные литералы*),
  - конструируемые типы записей (*структур*) и *коллекций.*

Современные модели данных (10)

ОО-модель данных (3). Типы и структуры данных (2)

- Литеральный тип записи – это традиционный определяемый пользователем структурный тип, подобный структурному типу языка C или типу записи языка Pascal.
- Отличие состоит лишь в том, что в объектной модели атрибут типа записи может определяться не только на литеральном, а на объектном типе,
  - т.е. значение литерального типа записи может в качестве компонентов включать объекты.
- Это звучит странно, но здесь все странности проистекают из особенностей объектно-ориентированной терминологии.
  - У любого существующего объекта имеется одно и только одно местоположение, характеризующееся его идентификатором (OID).
  - Когда в модели говорится, что некоторое структурное значение в качестве компонента имеет некоторый объект, то, конечно, имеется в виду OID этого объекта, являющийся всего лишь аналогом указательного значения в традиционных языках программирования.

Современные модели данных (11)

ОО-модель данных (4). Типы и структуры данных (3)

- Имеются четыре вида типов коллекций:
  - типы множеств,
  - мультимножеств (неупорядоченные наборы элементов, возможно, содержащие дубликаты),
  - списков (упорядоченные наборы элементов, возможно, содержащие дубликаты)
  - и словарей (множества пар <ключ, значение>, причем все ключи в этих парах должны быть различными).
- Типом элемента любой коллекции может являться любой скалярный или объектный тип, кроме того же типа коллекции.

Современные модели данных (12)

ОО-модель данных (5). Типы и структуры данных (4)

- Объектные типы в объектной модели данных по смыслу ближе всего к понятию *класса* в объектно-ориентированных языках программирования.
  - У каждого объектного типа имеется операция создания и инициализации нового объекта этого типа.
  - Эта операция возвращает значение OID нового объекта, который можно хранить в любом месте, где допускается хранение объектов данного типа, и использовать для обращения к *операциям* объекта, определенным в его объектном типе.
- Имеются два вида объектных типов.
- Первый из них называется атомарным объектным типом.
  - Нестрого говоря, при определении атомарного объектного типа указывается его внутренняя структура (набор свойств – атрибутов и связей) и набор операций, которые можно применять к объектам этого типа.
  - Для определения атомарного объектного типа можно использовать механизм наследования, расширяя набор свойств и/или переопределяя существующие и добавляя новые операции.

Современные модели данных (13)

ОО-модель данных (6). Типы и структуры данных (5)

- *Атрибутами* называются свойства объекта, значение которых можно получить по OID объекта.
  - Значениями атрибутов могут быть и литералы, и объекты (т.е. OID), но только тогда, когда не требуется обратная ссылка.
- *Связи* – это инверсные свойства.
- В этом случае значением свойства может быть только объект.
- Связи определяются между атомарными объектными типами.
- В объектной модели ODMG поддерживаются только бинарные связи, т.е. связи между двумя типами.
- Связи могут быть разновидностей «один-к-одному», «один-ко-многим» и «многие-ко-многим» в зависимости от того, сколько экземпляров соответствующего объектного типа может участвовать в связи.

Современные модели данных (14)

ОО-модель данных (7). Типы и структуры данных (6)

- Связи явно определяются путем указания *путей обхода*.
- Пути обхода указываются парами, по одному пути для каждого направления обхода связи.
- Например, в базе данных **СЛУЖАЩИЕ-ОТДЕЛЫ** служащий работает (**works**) в одном отделе, а отдел состоит (**consists of**) множества служащих.
  - Тогда путь обхода **consists_of** должен быть определен в объектном типе **ОТДЕЛ**, а путь обхода **works** – в типе **СЛУЖАЩИЙ**.
- Тот факт, что пути обхода относятся к одной связи, указывается в разделе **inverse** обоих объявлений пути обхода.
- Это связь «один-ко-многим».
  - Путь обхода **consists_of** ассоциирует объект типа **ОТДЕЛ** с литеральным множеством объектов типа **СЛУЖАЩИЙ**,
  - а путь обхода works ассоциирует объект типа **СЛУЖАЩИЙ** с объектом типа **ОТДЕЛ**.
- Пути обхода, ведущие к коллекциям объектов, могут быть упорядоченными или неупорядоченными в зависимости от вида коллекции, указанного в объявлении пути обхода.

Современные модели данных (15)

ОО-модель данных (8). Типы и структуры данных (7)

- Хотя связь является модельным понятием, другие понятия модели наталкивают на мысль, что единственным способом реализации связей является хранение в объекте OID или коллекции OID связанных объектов в зависимости от вида связи.
- Это можно сделать и с использованием должным образом типизированных атрибутов.
- Однако явное определение связи обеспечивает системе дополнительную информацию, которая используется в объектной модели как ограничение целостности.

Современные модели данных (16)
ОО-модель данных (9). Типы и структуры данных (8)

- Второй вид – это объектные типы коллекций.
  - Как и в случае использования литеральных типов коллекций, можно определять объектные типы
    - множеств,
    - мультимножеств,
    - списков
    - и словарей.
  - Типом элемента объектного типа коллекции может быть любой литеральный или объектный тип, кроме самого того типа коллекции.
  - У объектных типов коллекций имеется предопределенные наборы операций.
  - В отличие от литеральных типов коллекций, которые, как и все литеральные типы являются множествами значений, объектные типы коллекций обладают операцией создания объекта, обладающего, как и все объекты, собственным OID.

Современные модели данных (17)
ОО-модель данных (10). Типы и структуры данных (9)

- Интересен и важен один специальный случай неявного использования объектов типа множества.
- При определении атомарного объектного типа можно в качестве одного из дополнительных свойств этого типа указать, что для него должен быть создан объект типа множества, элементами которого являются объекты данного атомарного типа
  - *экстент* объектного структурного типа.
- Поскольку такой объект создается неявно, его OID неизвестен, но зато у него имеется имя, явно задающееся в определении совпадающее с именем атомарного объектного типа.
- Наличие этой возможности позволяет создавать объектные базы данных, состоящие из именованных контейнеров объектов однотипных типов, содержащих в действительности OID этих объектов.

Современные модели данных (18)
ОО-модель данных (11). Манипулирование данными (1)

- В стандарте ODMG в качестве базового средства манипулирования объектными базами данных предлагается язык OQL (Object Query Language).
  - Небольшой, но достаточно сложный язык запросов.
- Разработчики в целом характеризуют его следующим образом:
  - OQL опирается на объектную модель ODMG (имеется в виду, что в нем поддерживаются средства доступа ко всем возможным структурам данных, допускаемых в структурной части модели).
  - OQL очень близок к SQL/92.
  - Расширения относятся к объектно-ориентированным понятиям, таким как
    - сложные объекты,
    - объектные идентификаторы,
    - путевые выражения,
    - полиморфизм,
    - вызов операций и
    - отложенное связывание.
  - В OQL обеспечиваются высокоуровневые примитивы для работы с множествами объектов, но, кроме того, имеются настолько же эффективные примитивы для работы со структурами, списками и массивами.

Современные модели данных (19)
ОО-модель данных (12). Манипулирование данными (2)

- OQL является функциональным языком, допускающим неограниченную композицию операций, если операнды не выходят на пределы системы типов.
  - Это является следствием того факта, что результат любого запроса обладает типом, принадлежащим к модели типов ODMG, и поэтому к результату запроса может быть применен новый запрос.
- OQL не является вычислительно полным языком. Он представляет собой простой язык запросов.
- Операторы языка OQL могут вызываться из любого языка программирования, для которого в стандарте ODMG определены правила связывания.
  - И, наоборот, в запросах OQL могут присутствовать вызовы операций, запрограммированных на этих языках.
- В OQL не определяются явные операции обновления, а используются вызовы операций, определенных в объектах для целей обновления.
- В OQL обеспечивается декларативный доступ к объектам. По этой причине OQL-запросы могут хорошо оптимизироваться.
- Можно легко определить формальную семантику OQL.

Современные модели данных (20)
ОО-модель данных (13). Манипулирование данными (3)

- Получить номера руководителей отделов и тех служащих их отделов, зарплата которых превышает 20000 руб.
- ```
  SELECT DISTINCT STRUCT ( ОТД_РУК: D.ОТД_РУК,
                           СЛУ:     ( SELECT E
                                      FROM D.CONSISTS_OF AS E
                                      WHERE E.СЛУ_ЗАРП > 20000.00 ) )
  FROM ОТДЕЛЫ D
  ```
- Предполагается, что для атомарного объектного типа **ОТДЕЛ** определен экстент типа множества с именем **ОТДЕЛЫ**.
- Перебираются все существующие объекты типа **ОТДЕЛ**, и для каждого такого объекта происходит переход по связи к литеральному множеству объектов типа **СЛУЖАЩИЙ**, соответствующих служащим, которые работают в данном отделе.
- На основе этого множества формируется «усеченное» множество объектов типа **СЛУЖАЩИЙ**, в котором остаются только объекты-служащие с зарплатой, большей 20000.00.
- Результатом запроса является литеральное значение-множество, элементами которого являются значения-структуры с двумя литеральными значениями,
  - первое из которых есть атомарное литеральное значение типа INTEGER,
  - а второе – литеральное значение-множество с элементами-объектами типа EMP.
- Более точно, результат запроса имеет тип **set < struct { integer ОТД_РУК; bag < СЛУЖАЩИЙ > СЛУ } >.**
  ```

Современные модели данных (21)
ОО-модель данных (14). Манипулирование данными (4)

- В совокупности результатом допустимых в OQL выражений запросов могут являться:
  - коллекция объектов;
  - индивидуальный объект;
  - коллекция литеральных значений;
  - индивидуальное литеральное значение.

Современные модели данных (22)

ОО-модель данных (15). Ограничения целостности (1)

- В соответствии с общей идеологией объектно-ориентированного подхода в модели ODMG два объекта считаются совпадающими в том и только в том случае, когда являются одним и тем же объектом, т.е. имеют один и тот же OID.
  - Объекты одного объектного типа с разными OID считаются разными, даже если обладают полностью совпадающими состояниями.
- Поэтому в объектной модели отсутствует аналог ограничения целостности сущности реляционной модели данных.
- Интересно, что при определении атомарного объектного типа можно объявить ключ – набор свойств объектного класса, однозначно идентифицирующий состояние каждого объекта, входящего в экстент этого класса.
  - Для класса может быть объявлено несколько ключей, а может не быть объявлено ни одного ключа даже при наличии определения экстента.
  - Но при этом определение ключа не трактуется в модели как ограничение целостности;
    - утверждается, что объявление ключа способствует повышению эффективности выполнения запросов.

Современные модели данных (23)

ОО-модель данных (16). Ограничения целостности (2)

- Что же касается ссылочной целостности, то она поддерживается, если между двумя атомарными объектными типами определяется связь вида «один-ко-многим».
- В этом случае объекты на стороне связи «один» рассматриваются как предки, а объекты на стороне связи «многие» – как потомки,
  - и ООСУБД обязана следить за тем, чтобы не образовывались потомки без предков.

Современные модели данных (24)

Модель данных SQL (1). Типы и структуры данных (1)

- SQL-ориентированная база данных представляет собой набор таблиц, каждая из которых в любой момент времени содержит некоторое мультимножество строк, соответствующих заголовку таблицы.
  - В этом состоит первое и наиболее важное отличие модели данных SQL от реляционной модели данных.
- Вторым существенным отличием является того, что для таблицы поддерживается порядок столбцов, соответствующий порядку их определения.
- Другими словами, таблица – это вовсе не отношение, хотя во многом они похожи.

Современные модели данных (25)

Модель данных SQL (2). Типы и структуры данных (2)

- Имеется две основных разновидности таблиц, хранимых в базе данных:
  - традиционная таблица
  - и типизированная таблица.
- Традиционная таблица определяется как множество столбцов с указанными типами данных.
- В SQL поддерживаются следующие категории типов данных: точные числовые типы; приближенные числовые типы; типы символьных строк; типы битовых строк; типы даты и времени; типы временных интервалов; булевский тип; типы коллекций; анонимные строчные типы; типы, определяемые пользователем; ссылочные типы.
- Здесь мы ограничимся только пояснениями наименее очевидных случаев.

Современные модели данных (26)

Модель данных SQL (3). Типы и структуры данных (3)

- Булевский тип в SQL содержит три значения – *true*, *false* и *uknown*.
  - Это связано с интенсивным использованием в SQL так называемого неопределенного значения (NULL), которое разрешается использовать вместо значения любого типа данных.
- Допускается объявление двух видов типов коллекций:
  - типы массива
  - и типы мультимножества.
  - Элементы типа коллекции могут быть любого типа данных, определенного к моменту определения данного типа коллекции.
  - При объявлении типа мультимножества можно явно запретить наличие в его значениях элементов-дубликатов, что фактически приводит к объявлению типа множества.
- Анонимный строчный тип – это безымянный структурный тип, значения которого являются строками, состоящими из элементов ранее определенных типов.

Современные модели данных (27)

Модель данных SQL (4). Типы и структуры данных (4)

- Поддерживается два вида типов данных, определяемых пользователями: индивидуальные и структурные типы.
- Индивидуальный тип – это именованный тип данных, основанный на единственном предопределенном типе.
  - Индивидуальный тип не наследует от своего опорного типа набор операций над значениями.
  - Чтобы выполнить некоторую операцию базового типа над значениями определенного над ним индивидуального типа, требуется явно сообщить системе, что с этими значениями нужно обращаться как со значениями базового типа.
  - Имеется также возможность явного определения методов, функций и процедур, связанных с данным индивидуальным типом.

Современные модели данных (28)

Модель данных SQL (5). Типы и структуры данных (5)

- Структурный тип данных – это именованный типы данных, включающий один или более атрибутов любого из допустимых в SQL типа данных, в том числе другого структурного типа, типа коллекций, анонимного строчного типа и т. д.
  - Дополнительные механизмы определяемых пользователями методов, функций и процедур позволяют определить поведенческие аспекты структурного типа.
  - При определении структурного типа можно использовать механизм наследования от ранее определенного структурного типа.

Современные модели данных (28)

Модель данных SQL (5). Типы и структуры данных (5)

- При определении типизированной таблицы указывается ранее определенный структурный тип, и если в нем содержится *n* атрибутов, то в таблице образуется *n+1* столбец, из которых
  - последние *n* столбцов с именами и типами данных, совпадающими именам и типам атрибутов структурного типа,
  - а первый столбец, имя которого явно задается, называется «самоссылающимся» и содержит типизированные уникальные идентификаторы строк, которые могут
    - генерироваться системой при вставке строк в типизированную таблицу,
    - явно указываться пользователями
    - или состоять из комбинации значений других столбцов.
  - Типом «самоссылающегося» столбца является ссылочный тип, ассоциированный со структурным типом типизированной таблицы.
  - Способ генерации значений ссылочного типа указывается при определении соответствующего структурного типа и подтверждается при определении типизированной таблицы.

Современные модели данных (29)
Модель данных SQL (6). Типы и структуры данных (6)

- При определении типизированных таблиц можно использовать механизм наследования.
  - Можно определить подтаблицу типизированной подтаблицы, если структурный тип подтаблицы является непосредственным подтипом структурного типа супертаблицы.
  - Подтаблица наследует у супертаблицы способ генерации значений ссылочного типа и все ограничения целостности, которые были специфицированы в определении супертаблицы.
  - Дополнительно можно определить ограничения, затрагивающие новые столбцы.

Современные модели данных (30)
Модель данных SQL (7). Типы и структуры данных (7)

- С типизированной таблицей можно обращаться, как с традиционной таблицей,
  - считая, что у нее имеются неявно определенные столбцы,
- а можно относиться к строкам типизированной таблицы,
  - как к объектам структурного типа, OID которых содержатся в «самоссылающемся» столбце.
- Ссылочный тип можно использовать для типизации столбцов традиционных таблиц и атрибутов структурных типов, на которых потом определяются типизированные таблицы.
- В последнем случае можно считать, что значениями атрибутов соответствующих объектов являются объекты структурного типа, с которым ассоциирован данный ссылочный тип.

Современные модели данных (31)

Модель данных SQL (8). Манипулирование данными (1)

- Выборка данных производится из одной или нескольких таблиц, указываемых в разделе **FROM** запроса.
  - В последнем случае на первом этапе выполнения оператора **SELECT** образуется одна общая таблица, получаемая из исходных таблиц путем операции расширенного декартова умножения.
- Таблицы могут быть как базовыми, реально хранимыми в базе данных (традиционными или типизированными), так и порожденными, т.е. задаваемыми в виде некоторого оператора **SELECT**.
- Это допускается, поскольку результатом выполнения оператора **SELECT** в его базовой форме является традиционная таблица.
- Кроме того, в разделе **FROM** можно указывать выражения соединения базовых и/или порожденных таблиц, результатами которых опять же являются традиционные таблицы.

Современные модели данных (32)

Модель данных SQL (9). Манипулирование данными (2)

- На следующем шаге общая таблица, полученная после выполнения раздела, подвергается фильтрации путем вычисления для каждой ее строки логического выражения, заданного в разделе **WHERE** запроса.
  - В отфильтрованной таблице остаются только те строки общей таблицы, для которых значением логического выражения является *true*.
- Если в операторе отсутствует раздел **GROUP BY**, то после этого происходит формирование результирующей таблицы запроса путем вычисления выражений, заданных в списке выборки оператора **SELECT**.
  - В этом случае список выборки вычисляется для каждой строки отфильтрованной таблицы, и в результирующей таблице появится ровно столько же строк.

Современные модели данных (33)

Модель данных SQL (10). Манипулирование данными (3)

- При наличии раздела GROUP BY из отфильтрованной таблицы получается сгруппированная таблица, в которой каждая группа состоит из кортежей отфильтрованной таблицы с одинаковыми значениями столбцов группировки, задаваемых в разделе GROUP BY.
- Если в запросе отсутствует раздел HAVING, то результирующая таблица строится прямо на основе сгруппированной таблицы.
  - Иначе образуется отфильтрованная сгруппированная таблица, содержащая только те группы, для которых значением логического выражения, заданного в разделе HAVING, является *true*.
- Результирующая таблица на основе сгруппированной или отфильтрованной сгруппированной таблицы строится путем вычисления списка выборки для каждой группы.
- Тем самым, в результирующей таблице появится ровно столько строк, сколько групп содержалось в сгруппированной или отфильтрованной сгруппированной таблице.

Современные модели данных (34)
Модель данных SQL (11). Манипулирование данными (4)

- Если в запросе присутствует ключевое слово DISTINCT, то из результирующей таблицы устраняются строки-дубликаты, т.е. запрос вырабатывает не мультимножество, а множество строк.
- Наконец, в запросе может присутствовать еще и раздел ORDER BY.
  - В этом случае результирующая таблица сортируется в порядке возрастания или убывания в соответствии со значениями ее столбцов, указанных в разделе ORDER BY.
  - Результатом такого запроса является не таблица, а отсортированный список, который нельзя сохранить в базе данных.
  - Сам же запрос, содержащий раздел ORDER BY, нельзя использовать в разделе FROM других запросов.
- Приведенная характеристика средств манипулирования данными языка SQL является не вполне точной и полной.
- Кроме того, она отражает *семантику* оператора SQL, а не то, как он обычно исполняется в SQL-ориентированных СУБД.

Современные модели данных (35)

Модель данных SQL (12). Ограничения целостности (1)

- Наиболее важным отличием модели данных SQL от реляционной модели данных является то, что таблицы SQL могут содержать мультимножества строк.
  - Из этого, в частности, следует, что в модели SQL отсутствует обязательное предписание об ограничении целостности сущности.
  - В базе данных могут существовать таблицы, для которых не определен первичный ключ.
- С другой стороны, если для таблицы определен первичный ключ, то для нее ограничение целостности сущности поддерживается точно так же, как это требуется в реляционной модели данных.

Современные модели данных (36)

Модель данных SQL (13). Ограничения целостности (2)

- Ссылочная целостность в модели данных SQL поддерживается в обязательном порядке, но в трех разных вариантах, лишь один из которых полностью соответствует реляционной модели.
  - Это связано с уже упоминавшимся в этом разделе интенсивным использованием в SQL неопределенных значений.
- Кроме того, в SQL имеются развитые возможности явного определения ограничений целостности
  - на уровне столбцов таблиц,
  - на уровне таблиц целиком
  - и на уровне базы данных.

Современные модели данных (37)

Истинная РМД (1). Типы и структуры данных (1)

- Кристофер Дейт и Хью Дарвен поставили перед собой трудную задачу:
  - показать, что на основе идей Эдгара Кодда можно реализовать СУБД, обеспечивающие возможности по части представления и хранения данных сложной структуры, не меньшие тех, которые обеспечивают объектные и SQL-ориентированные СУБД.
- Этому мешал, прежде всего, тезис Кодда о нормализации отношений:
  - в реляционной базе данных должны содержаться только отношения с атрибутами, определенными на «доменах, элементы которых являются атомарными (не составными) значениями»

Современные модели данных (38)

Истинная РМД (2). Типы и структуры данных (2)

- Дейт пишет:
  - «Я согласен с Коддом, что желательно оставаться в рамках логики первого порядка, если это возможно. В то же время я отвергаю идею "атомарных значений", по крайней мере, в смысле абсолютной атомарности. В Третьем манифесте мы допускаем наличие доменов, содержащих значения произвольной сложности. (Они могут быть даже отношениями.) Тем не менее, мы остаемся в рамках логики первого порядка.»
- Если учесть, что цитировалась первая официальная публикация Кодда по поводу реляционной модели данных, то трудно сказать, что Дейт очень уж строго следует всем его заветам.
- Те постулаты Кодда, которые вредят достижению цели Третьего манифеста, просто отвергаются.

Современные модели данных (39)

Истинная РМД (3). Типы и структуры данных (3)

- В истинно реляционной модели очень большое внимание уделяется типам данных.
- Предлагаются три категории типов данных:
  - скалярные типы,
  - кортежные типы
  - и типы отношений.

Современные модели данных (40)

Истинная РМД (4). Типы и структуры данных (4)

- Скалярный тип данных – это привычный инкапсулированный тип, реальная внутренняя структура которого скрыта от пользователей.
- Предлагаются механизмы определения новых скалярных типов и операций над ними.
- Типом атрибута определяемого скалярного типа может являться любой определенный к этому моменту
  - скалярный тип,
  - любой кортежный тип
  - и тип отношения.
- Некоторые базовые скалярные типы данных должны быть предопределены в системе.
  - В число этих типов должен входить тип **truth value**
    - так Дейт и Дарвен называют булевский тип
  - ровно с двумя значениями *true* и *false*.

Современные модели данных (41)

Истинная РМД (5). Типы и структуры данных (5)

- Кортежный тип – это безымянный тип данных, определяемый с помощью генератора типа **TUPLE** с указанием множества пар **<имя_атрибута, тип_атрибута>** (заголовка кортежа).
- Типом атрибута кортежного типа может являться любой определенный к этому моменту
  - скалярный тип,
  - любой кортежный тип
  - и тип отношения.
- Значением кортежного типа является кортеж, представляющий собой множество триплетов **<имя_атрибута, тип_атрибута, значение_атрибута>**, которое соответствует заголовку кортежа этого кортежного типа.

Современные модели данных (42)

Истинная РМД (6). Типы и структуры данных (6)

- Тип отношения – это безымянный тип данных, определяемый с помощью генератора типа **RELATION** с указанием некоторого заголовка кортежа.
- Значением типа отношения является
  - заголовок отношения, совпадающий с заголовком кортежа этого типа отношения,
  - и тело отношения, представляющее собой множество кортежей, соответствующих этому заголовку.
- Кортежные типы и типы отношений не являются инкапсулированными: имеется возможность прямого доступа к атрибутам.

Современные модели данных (43)

Истинная РМД (7). Типы и структуры данных (7)

- Для всех разновидностей типов данных разработана модель множественного наследования, позволяющая определять новые типы данных на основе уже определенных типов.
- Модель наследования по Дейту и Дарвену не является частью истинной реляционной модели данных.

Современные модели данных (44)

Истинная РМД (8). Типы и структуры данных (8)

- При таких определениях значениями атрибутов отношения могут быть не только значения произвольно сложных скалярных типов, типами атрибутов которых могут быть, в частности, отношения, но и просто отношения.
- Тем не менее, Дейт и Дарвен говорят:
  - «Каждый кортеж в [отношении] ***R*** содержит в точности одно значение ***v*** для каждого атрибута ***A*** в [заголовке отношения] ***H***. Иными словами, ***R*** находится в *первой нормальной форме*, 1NF.»
- Это хорошее и понятное определение первой нормальной формы, но трудно сказать, согласился бы с ним Кодд.

Современные модели данных (45)

Истинная РМД (9). Типы и структуры данных (9)

- База данных в истинной реляционной модели – это набор долговременно хранимых именованных *переменных отношений*, каждая из которых определена на некотором типе отношений.
- В каждый момент времени каждая переменная отношения базы данных содержит некоторое значение отношения соответствующего типа.

Современные модели данных (46)

Истинная РМД (10). Манипулирование данными

- Вообще говоря, в качестве эталонного средства манипулирования данными в истинной реляционной модели можно использовать реляционную алгебру Кодда.
- Однако Дейт и Дарвен предложили новую реляционную алгебру, названную ими Алгеброй *A*, которая основывается на реляционных аналогах булевских операций *конъюнкции*, *дизъюнкции* и *отрицания*.
- Позже мы опишем эту алгебру и покажем, что через ее операции выражаются все операции алгебры Кодда.

Современные модели данных (47)

Истинная РМД (11). Ограничения целостности (1)

- В число обязательных требований истинной реляционной модели входит требование определения хотя бы одного возможного ключа для каждой переменной отношения
  - возможный ключ – это одно из подмножеств заголовка переменной отношения, обладающее свойствами первичного ключа.
- Кроме того, говорится, что
  - «любое условное выражение, которое является (или логически эквивалентно) замкнутой правильно построенной формулой (WFF) реляционного исчисления, должно быть допустимо в качестве спецификации ограничения целостности».

Современные модели данных (48)

Истинная РМД (12). Ограничения целостности (2)

- Средства поддержки декларативной ссылочной целостности фигурируют только в разделе рекомендуемых возможностей:
  - «В **D**
    - [конкретную реализацию истинной реляционной модели]
  - следует включить некоторую декларативную сокращенную форму для выражения ссылочных ограничений (называемых также ограничениями внешнего ключа)».

Заключение

- Кратко рассмотрены особенности трех ранних моделей данных:
  - модели инвертированных таблиц,
  - иерархической модели
  - и сетевой модели данных.
- Представлена исходная реляционная модель данных, определенная Эдгаром Коддом.
- Описаны основные черты трех современных моделей данных, системы типов данных которых позволяют сохранять в базе данных и обрабатывать данные произвольно сложной структуры:
  - объектно-ориентированная модель данных,
  - модель данных SQL
  - и истинно реляционная модель данных.