КОНСПЕКТ ЗАНЯТИЯ № 1 ВТОРОЙ НЕДЕЛИ КУРСА «БАЗЫ ДАННЫХ»

1. МОДЕЛИ ДАННЫХ. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ: ДОПУСТИМЫЕ СТРУКТУРЫ

Каждая система БД реализует ту или иную модель данных. Модель данных определяет правила порождения допустимых для системы видов структур данных, возможные операции над такими структурами и системы ограничений целостности данных.

Рассмотрим список моделей данных, которые были разработаны за всю историю существования баз данных:

- иерархическая (*англ.* hierarchical), конец 1960-х и 1970-е;
- сетевая (*англ.* network), 1970-е;
- реляционная (*англ.* relational), 1970-е и начало 1980-х;
- сущность-связь (*англ.* entity-relationship), 1970-е;
- расширенная реляционная (*англ.* extended relational), 1980-е;
- семантическая (*англ.* semantic), конец 1970-х и 1980-е;
- объектно-ориентированная (*англ.* object-oriented), конец 1980-х – начало 1990-х;
- объектно-реляционная (*англ.* object-relational), конец 1980-х – начало 1990-х;
- полуструктурированная (*англ.* semi-structured), конец 1990-х.

В данном параграфе рассматривается реляционная модель данных, основанная на математической теории множеств. Такой фундамент обеспечивает математическую строгость реляционной модели данных. На сегодняшний день она является самой распространенной, и поэтому будет рассматриваться наиболее подробно.

1.1. ОБЩАЯ ХАРАКТЕРИСТИКА РЕЛЯЦИОННОЙ МОДЕЛИ

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии **отношение** (relation).

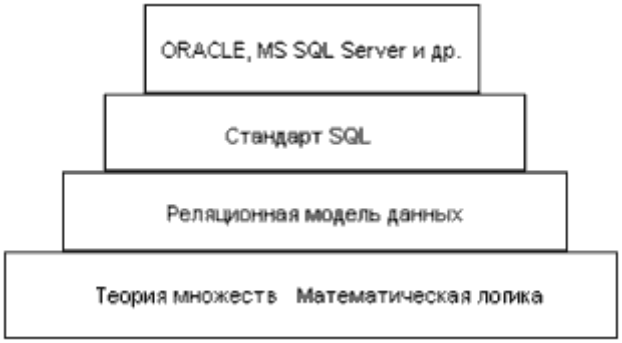
Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица, где строкам таблицы соответствуют **кортежи**, а столбцам – **атрибуты отношения**.

Именно простота и понятность для пользователя явились основной причиной их широкого использования.

Примерами реляционных СУБД являются следующие: dBaselll Plus (Aston-Tate), FoxBase (Fox Software), Paradox (Borland), Visual FoxPro (Microsoft), Access (Microsoft), Oracle (Oracle), HyTech (МИФИ).

Различные фирмы, производители СУБД, предлагают свои реализации языка SQL. Эти реализации отличаются как друг от друга, так и от стандартизованного языка SQL. Это и хорошо и плохо. Хорошо это тем, что конкретная реализация языка, может включать в себя более широкие возможности по сравнению со стандартизованными SQL, например, больше типов данных, большее количество команд, больше дополнительных опций имеющихся команд. Такие возможности делают работу с конкретной СУБД более эффективной. Кроме того, такие нестандартные возможности языка проходят практическую апробацию и со временем могут быть включены в стандарт. Плохо же это тем, что различия в синтаксисе реализаций SQL затрудняют перенос приложений из одной системы в другую. Например, если приложение было написано для базы данных MS SQL Server с использованием своего диалекта SQL - языка Transact-SQL, то при переносе системы в базу данных ORACLE, не все конструкции языка будут понятны соответствующему диалекту SQL - языку PL/SQL.

Взаимосвязь реляционной модели данных, стандарта языка SQL и различных его реализаций можно условно изобразить в виде следующей пирамиды:



Каждый более высокий уровень основывается на понятиях, определенных на более низком уровне. На каждом из уровней

используется своя терминология. Например, на уровне теории множеств мы говорим "множество", "подмножество декартового произведения", "кортеж". На уровне реляционной модели используем термины "домен", "отношение", "кортеж". На уровне стандарта SQL и конкретных реализаций используем термины "тип данных", "таблица", "строка таблицы". И каждый раз речь идет об одном и том же.

Реляционная модель состоит из трех частей:

- Структурной части.
- Целостной части.
- Манипуляционной части.

Структурная часть описывает, какие объекты рассматриваются реляционной моделью. Постулируется, что единственной структурой данных, используемой в реляционной модели, являются нормализованные отношения.

Целостная часть описывает ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных. Это ***целостность сущностей*** и ***целостность внешних ключей***.

Манипуляционная часть описывает два эквивалентных способа манипулирования реляционными данными - ***реляционную алгебру*** и ***реляционное исчисление.***

Реляционная алгебра и реляционное исчисление будут рассмотрены на следующей неделе. На этой – мы разберем структурную часть.

1.2. ОСНОВНЫЕ ТЕРМИНЫ РЕЛЯЦИОННОЙ МОДЕЛИ

1.2.1. Типы данных

Вспомним, какие вообще типы данных обычно рассматриваются в программировании. Как правило, типы данных делятся на три группы:

- Простые типы данных.
- Структурированные типы данных.
- Ссылочные типы данных.

ВАЖНО: Реляционная модель требует, чтобы типы используемых данных были простыми.

Простые, или атомарные[1], типы данных не обладают внутренней структурой. Данные такого типа называют ***скалярами***. К простым типам данных относятся следующие типы:

- Логический.
- Строковый.
- Численный.

Различные языки программирования могут расширять и уточнять этот список, добавляя такие типы как:

- Целый.
- Вещественный.
- Дата.
- Время.
- Денежный.
- Перечислимый.
- Интервальный и тд.

Обычно в современных реляционных базах данных допускается хранение символьных, числовых данных (точных и приблизительных), специализированных числовых данных (таких, как «деньги»), а также специальных «темпоральных» данных (дата, время, временной интервал). Кроме того, в реляционных системах поддерживается возможность определения пользователями собственных типов данных.

1.2.2. Структура

В реляционной модели единственной допустимой структурой данных считается отношение, от английского названия которого – relation – модель и получила своѐ имя. На бытовом уровне математическое понятие «отношение» часто отождествляют с двумерной таблицей (см. рис. 3.1).

[1] Конечно, понятие атомарности довольно относительно. Так, строковый тип данных можно рассматривать как одномерный массив символов, а целый тип данных - как набор битов. Важно лишь то, что при переходе на такой низкий уровень теряется семантика (смысл) данных. Если строку, выражающую, например, фамилию сотрудника, разложить в массив символов, то при этом теряется смысл такой строки как единого целого.

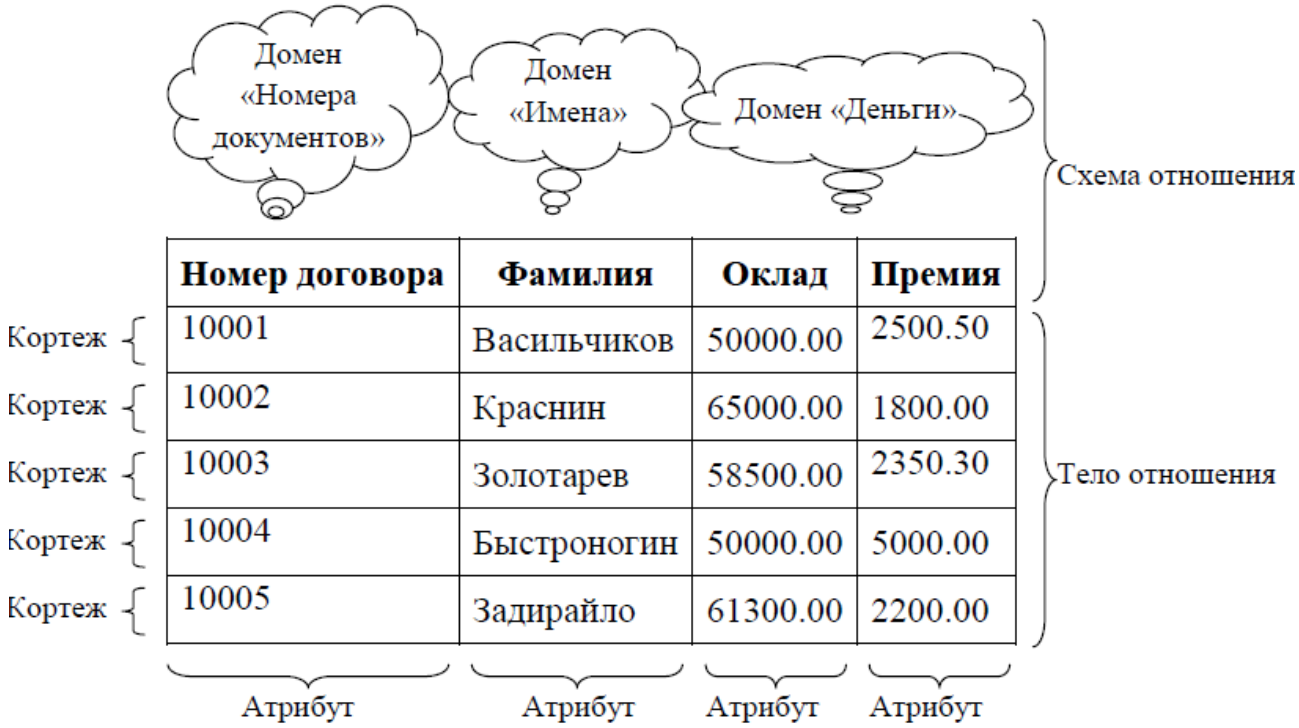


Рисунок 3.1. Пример таблицы,
соответствующей некоторому реляционному отношению

1.2.3. Атрибут отношения

Это характеристика (свойство) некоторого объекта или явления реального мира.

Например, если в качестве объекта рассмотрения взять сотрудника некоторой организации, для него можно будет выделить такие атрибуты-характеристики как: номер договора с работодателем, фамилия, размер оклада и размер премии. Можно заметить, что перечисленные характеристики соответствуют столбцам таблицы, изображѐнной на рисунке 3.1.

1.2.4. Домен

Доменом называется множество однотипных скалярных значений. Домен можно рассматривать как подмножество значений некоторого типа данных имеющих определенный смысл. Домен характеризуется следующими свойствами:

- Домен имеет *уникальное имя* (в пределах базы данных).
- Домен может иметь некоторое *логическое условие*, позволяющее описать подмножество данных, допустимых для данного домена.
- Домен несет определенную *смысловую нагрузку*.

Например, домен D, имеющий смысл "возраст сотрудника" можно описать как следующее подмножество множества натуральных чисел:

$$D = \{n \in \mathbb{N} : n \geq 18 \text{ and } n \leq 60\}$$

Отличие домена от понятия подмножества состоит именно в том, что *домен отражает семантику*, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущие различный смысл. Например, домены "Вес детали" и "Имеющееся количество" можно одинаково описать как множество неотрицательных целых чисел, но смысл этих доменов будет различным, и это будут *различные* домены.

Основное значение доменов состоит в том, что *домены ограничивают сравнения*. Некорректно, с логической точки зрения, сравнивать значения из различных доменов (атрибуты «Оклад» и «Номер отдела» определены на разных доменах), даже если они имеют одинаковый тип. В этом проявляется смысловое ограничение доменов.

Понятие домена можно также проиллюстрировать следующей формулой:

домен = скалярный тип данных + (опционально) некоторое дополнительное ограничение диапазона значений указанного типа.

Например, атрибуты «Оклад» и «Премия» на рисунке 3.1 определены на домене «Деньги», состоящем из дробных чисел с двумя разрядами в дробной части и пятью разрядами в целой, а атрибут «Номер договора» – на домене «Номера документов», состоящем из целых чисел в диапазоне от 10000 до 99999.

Замечание. Не всегда очевидно, как задать логическое условие, ограничивающее возможные значения домена. К примеру, условие на строковый тип данных, задающий домен "Фамилия сотрудника". Ясно, что строки, являющиеся фамилиями не должны начинаться с цифр, служебных символов, с мягкого знака и т.д. Но вот является ли допустимой фамилия "Гггггыыыыы"? Почему бы нет? Очевидно, нет! А может кто-то назло так себя назовет. Трудности такого рода возникают потому, что смысл реальных явлений далеко не всегда можно формально

описать. Просто мы, как все люди, интуитивно понимаем, что такое фамилия, но никто не может дать такое формальное определение, которое отличало бы фамилии от строк, фамилиями не являющимися. Выход из этой ситуации простой - положиться на разум сотрудника, вводящего фамилии в компьютер.

1.2.5. Кортеж

В базе данных кортежу соответствует строка или запись таблицы.

Любой кортеж содержит утверждение о некотором объекте (явлении) реального мира, обладающем набором свойств, совпадающих с атрибутами отношения. Например, каждый кортеж, соответствующий строке таблицы, изображѐнной на рисунке 3.1, характеризует конкретного сотрудника организации.

В математическом виде каждый кортеж представляет собой множество пар вида <Имя_атрибута : Значение_атрибута>:

$$(< A_1 : Val_1 >, < A_2 : Val_2 >, \ldots, < A_n : Val_n >)$$

таких что значение Val_i атрибута A_i принадлежит домену D_i

Основываясь на наличии кортежа, состоящего из пар {<Номер договора : 10001>, <Фамилия : Васильчиков>, <Оклад : 50000.00>, <Премия : 2500.50>}, мы можем утверждать, что сотрудник Васильчиков работает в организации по договору с номером 10001, получает оклад 50000 рублей в месяц и премию две тысячи пятьсот рублей пятьдесят копеек.

1.2.6. Отношение и его составляющие

После введения терминов «атрибут», «домен» и «кортеж» – становится проще объяснить суть фундаментального понятия для реляционной модели данных – «отношение» (оно отражено в названии модели – реляционная/relation – англ. «отношение»).

Отношение – это совокупность заголовка отношения и тела отношения.

1. ***Заголовок отношения*** (в некоторых изданиях используют термин «схема отношения») – это совокупность всех атрибутов

отношения вместе с доменами, на которых они определены. Более формально **схемой отношения** называется конечное множество пар вида:

$$\{<A_1 : D_1>, <A_2 : D_2>, \ldots, <A_n : D_k>\}, \text{ где}$$

$k \geq 0$ – общее количество доменов,
$n \geq 0$ – общее количество атрибутов (**степень отношения**),
$k \leq n$, т.к. на одном домене может быть определено несколько атрибутов.

2. Тело отношения – это совокупность всех кортежей (в базе данных телу отношения соответствует набор всех строк таблицы).

Каждая реляционная модель характеризуется двумя критериями:
- степенью[2],
- мощностью[3].

1.3. МАТЕМАТИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ ОТНОШЕНИЯ

Возвращаясь к математическому понятию отношения можно сделать следующие выводы:

1. Заголовок отношения описывает декартово произведение доменов, на котором задано отношение. Заголовок не статичен – допускается изменение, добавление или удаление атрибутов. Это принято называть эволюцией схемы[4] базы данных.

2. Тело отношения представляет собой набор кортежей, т.е. подмножество декартового произведения доменов. Тело отношения может изменяться во время работы с базой данных – кортежи могут изменяться, добавляться и удаляться.

Таким образом, заголовок отношения – это запись вида $D_1 x D_2 x \ldots x D_k$ (декартово произведение доменов). А тело отношения – это результат записи $D_1 x D_2 x \ldots x D_k$ (подмножество результата декартового произведения доменов).

---

[2] Число атрибутов в отношении называют ***степенью*** (или ***-арностью***) отношения.

[3] Общее количество кортежей отношения называют ***мощностью*** отношения.

[4] В классических реляционных базах данных после определения схемы базы данных могли изменяться только значения кортежей. Атрибуты оставались статичны.

Вспомним, что собой представляет это математическое действие. **Декартовым произведением множеств** называется набор всех возможных упорядоченных сочетаний элементов множеств.

Допустим, в нашем случае таблица состоит из атрибутов:

- Номер договора (определен на домене D_1 – номера документов);
- Фамилия (определен на домене D_2 - имена);
- Оклад (определен на домене D_3 - деньги);
- Премия (также определен на домене D_3 - деньги).

Предположим, что множество каждого домена состоит из двух элементов:

D1 = {10001, 10002}
D2 = {Васильчиков, Краснин}
D3 = {50000, 2500}

Так как отношение состоит из четырех атрибутов, необходимо рассмотреть декартово произведение четырех множеств: D1, D2 и дважды D3. В результате мы получим следующий набор четверок:

$D_1 \times D_2 \times D_3 \times D_3$ = {
(10001, Васильчиков, 50000, 50000),
(10001, Васильчиков, 50000, 2500),
(10001, Васильчиков, 2500, 50000),
(10001, Васильчиков, 2500, 2500),
(10002, Васильчиков, 50000, 50000),
(10002, Васильчиков, 50000, 2500),
(10002, Васильчиков, 2500, 50000),
(10002, Васильчиков, 2500, 2500),
(10001, Краснин, 50000, 50000),
(10001, Краснин, 50000, 2500),
(10001, Краснин, 2500, 50000),
(10001, Краснин, 2500, 2500),
(10002, Краснин, 50000, 50000),
(10002, Краснин, 50000, 2500),
(10002, Краснин, 2500, 50000),
(10002, Краснин, 2500, 2500)
}

Каждая четверка соответствует потенциально возможному кортежу отношения, общее количество таких четверок будет равно произведению мощностей перемножаемых множеств: 2 x 2 x 2 x 2 = 16.

Очевидно, что не все полученные сочетания отвечают действительности, поэтому тело отношения в итоге составит такое подмножество четверок-кортежей, которое соответствует истинным утверждениям о сотрудниках организации.

1.4. СВОЙСТВА ОТНОШЕНИЙ

Термины, которыми оперирует реляционная модель данных, имеют соответствующие "табличные" синонимы:

Реляционный термин	**Соответствующий "табличный" термин**
База данных	Набор таблиц
Схема базы данных	Набор заголовков таблиц
Отношение	Таблица
Заголовок отношения	Заголовок таблицы
Тело отношения	Тело таблицы
Атрибут отношения	Наименование столбца таблицы
Кортеж отношения	Строка таблицы
Степень (-арность) отношения	Количество столбцов таблицы
Мощность отношения	Количество строк таблицы
Домены и типы данных	Типы данные в ячейках таблицы

Хотя любое отношение можно изобразить в виде таблицы, нужно четко понимать, что *отношения не являются таблицами*. Это близкие, но не совпадающие понятия.

1. *В отношении нет одинаковых кортежей*. Действительно, тело отношения есть *множество* кортежей и, как всякое множество, не может содержать неразличимые элементы. Таблицы в отличие от отношений могут содержать одинаковые строки.
2. *Кортежи не упорядочены*. Действительно, несмотря на то, что мы изобразили отношение "Сотрудники" в виде таблицы, нельзя сказать, что сотрудник Иванов "предшествует" сотруднику Петрову. Причина та же - тело отношения есть множество, а множество не

упорядочено. Это вторая причина, по которой нельзя отождествить отношения и таблицы - строки в таблицах упорядочены. (Упорядоченность – требования, предъявляемые к таблице со стороны организации).

3. *Все значения атрибутов атомарны*. Это следует из того, что лежащие в их основе атрибуты имеют атомарные значения. Это четвертое отличие отношений от таблиц - в ячейки таблиц можно поместить что угодно - массивы, структуры, и даже другие таблицы.