Лекция 8. Реляционная модель данных. Основные понятия реляционной модели: n-арное отношение, атрибуты, схемы отношений, кортежи, домены, ключи.

Реляционная модель данных, как и любая логическая модель данных, рассматривается с трех сторон: структура объектов, обеспечение целостности и манипулирование объектами.
Реляционная модель данных является формальной математически строгой моделью, которая была построена с использованием аппарата теории множеств, алгебры логики и исчисления предикатов.

В данной лекции рассмотрим структуру реляционных данных.

8.1 Структуры данных

Начнем с неформального определения терминов

Пример 1.

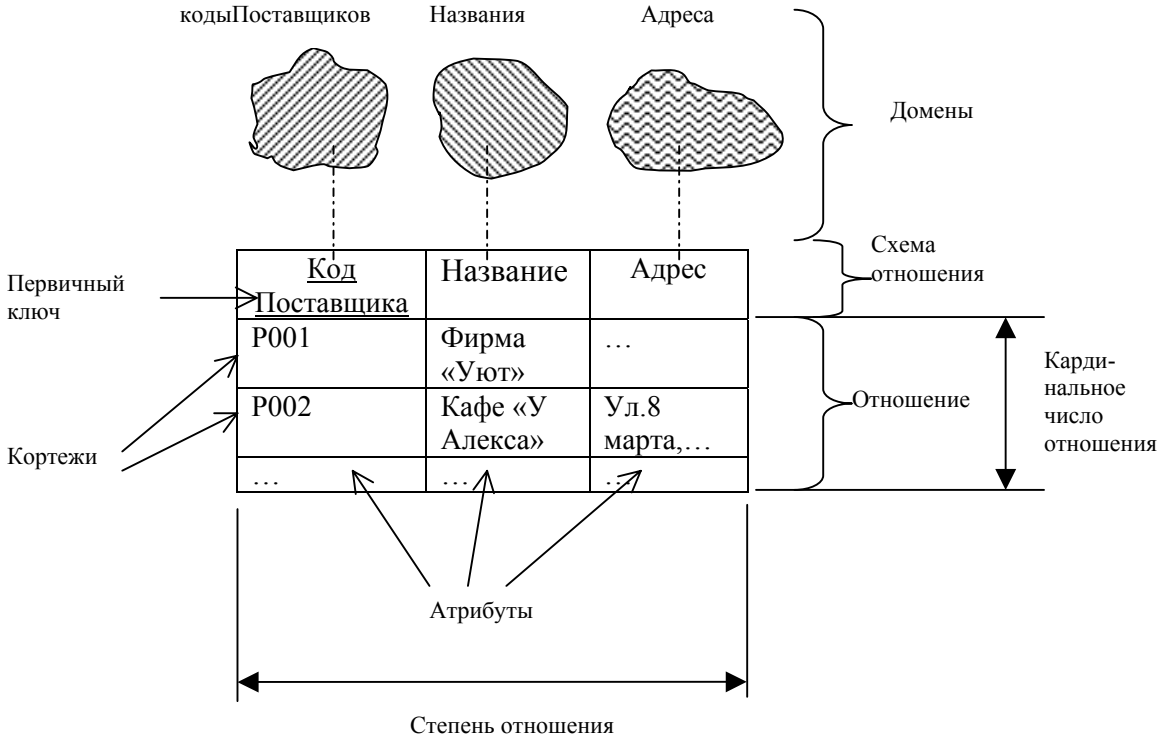

Рисунок 8.1-Реляционное отношение «ПОСТАВЩИКИ» очень подробно.

Отношение соответствует таблице, кортеж – строке таблицы, атрибут – столбцу таблицы.
Первичный ключ – уникальный идентификатор для таблицы, т.е. столбец или комбинация столбцов, для которых в любой момент времени не существует двух строк с одинаковыми значениями в этих столбцах.
Домен – множество допустимых значений атрибута.

Теперь дадим строгие математические определения этих элементов структуры.
Начнем с того, что минимальной единицей данных в реляционной модели данных является отдельное значение данных.
В реляционной модели данных все отдельные значения данных являются атомарными, т.е. у них нет внутренней структуры, и они неразложимы на составляющие.

> **Домен *D*** – конечное или счетное множество атомарных значений одного типа.

Пример 2. Домен категорий товаров, таких как «бытовая техника», «продукты питания» и т.п. может быть таким: строки русских символов длиной 20 символов.

Для создания домена в SQL (все предложения SQL даны по стандарту SQL92) есть оператор:

```
CREATE DOMAIN имя_домена  тип_данных
DEFAULT значение;
```

Пример 3. Определим домен для примера 2 как

```
CREATE DOMAIN Category  varchar(20);
```

Информация о доменах позволяет проводить сравнения значений атрибутов с равными доменами. Если значения атрибутов взяты из разных доменов, то очевидно, сравнение значений в общем случае не будет иметь смысла.

Пример 4. Сравнение значений атрибутов «кодЗаказа» и «кодПоставщика», даже если оба значения – числовые, не имеет смысла.

Предположим, что каждому домену D соответствует атрибут A. Тогда

> **Схема отношения, *R*** – конечное множество имен атрибутов $\{A_1,...,A_n\}$, где ***n*-**степень отношения.
> Каждому имени атрибута A_i ставится в соответствие домен D_i.

> **Кортеж *t* -** множество значений, взятых по одному для каждого имени атрибута из схемы отношения.
> Обозначается — $t(R)$, где R – схема отношения, или $t(R) = \{< A_1 : a_1 >,< A_2 : a_2 >,...,< A_n : a_n >\}$.

Каждый кортеж является отображением из схемы R во множество $D = D_1 \cup D_2 \cup ... \cup D_n$, т.е. $t(R) : R \to D$

> **Отношение *r* со схемой *R*** (обозначается r(R)) – конечное множество отображений, $\{t_1,...,t_p\}$ из R в D. Число p – кардинальное число отношения *r*.

Если кортеж $t \in r$, то значение этого кортежа на атрибуте A_i взято из D_i, $t(A_i) \in D_i, i = \overline{1,n}$

Следует различать переменные отношений (например, r(R))и значения отношений (например, таблица «Поставщики»). Переменная отношения – обычная переменная, значение которой и будет значением отношения. Значение отношения может изменяться со временем.

Свойства отношений (отличия реляционных отношений от таблиц):
1. Нет одинаковых кортежей.
2. Кортежи не упорядочены сверху вниз.
3. Атрибуты не упорядочены слева направо.
4. Все значения атрибутов – атомарные.

Исходя из таких свойств, можно сделать вывод о том, что отношение – это математическое множество (множество кортежей). Во множестве не может быть одинаковых элементов (иначе это не множество), и элементы неупорядочены.
Отношение $r(R) \in D_1 \times D_2 \times ... \times D_n$, т.е. подмножество декартова произведения соответствующих доменов.

На SQL оператор создания переменной отношения имеет вид:
(Примечание: в квадратных скобках указываются необязательные параметры.)

```
CREATE TABLE имя_таблицы
      (список имен атрибутов, через запятую)
      [список возможных ключей]
      [список внешних ключей];
```

где

```
список имен атрибутов имеет вид
имя_атрибута DOMAIN имя_домена [NOT NULL] [UNIQUE],
```

`NOT NULL` обозначает атрибут, значение которого должно быть указано в таблице обязательно.

`UNIQUE` обозначает атрибут, значение которого уникально в пределах одной таблицы, т.е. претендент на первичный ключ

Пример 5.

```
CREATE TABLE Suppliers
      (scode DOMAIN (scode) NOT NULL UNIQUE,
       sname DOMAIN (sname) NOT NULL,
       saddr DOMAIN (saddr) NOT NULL)
```

В данном случае единственным первичным ключом будет `scode`

Дадим строгое определение ключа реляционного отношения.

> **Ключ отношения r(R)** – подмножество $K \subseteq R$:
> $\forall t_1 \neq t_2,\ t_1, t_2 \in r \Rightarrow t_1(K) \neq t_2(K)$
> и ни одно собственное подмножество $K' \subset K$ не обладает такими свойствами.

(ключ — неизбыточный набор атрибутов, значения которых идентифицируют весь кортеж отношения)
Как правило, любое отношение может иметь несколько ключей, по крайней мере, один ключ – всегда (состоящий из всех атрибутов отношения).
Все ключи отношения называются возможными (candidate key). Один из возможных ключей выбирается в качестве первичного и называется PRIMARY KEY.

Ключ может быть простым и составным.

Пример 6.
Определение составного первичного ключа

```
PRIMARY KEY (scode, product_code, sclad_code)
```

8.2 Целостность реляционных данных

В любой момент времени любая база данных содержит некоторый набор значений данных. Предполагается, что база данных является моделью реального мира. Однако, набор значений не имеет смысла, если значения не представляют определенного состояния реального мира.

Пример7.
«Поставщик А сделал заказ на (-100) кондиционеров» - бессмысленное значение.

Следовательно, определение базы данных нуждается в расширении, включающем правила целостности. Суть правил состоит в том, чтобы сообщать СУБД об ограничениях на сочетания значений в реальном мире. Такие правила целостности называются **специализированными**, т.к. они применяются только в рамках конкретной базы данных.
Поэтому в реляционной модели данных есть и общие для всех реляционных баз данных правила целостности, и специализированные, относящиеся к конкретной базе данных.
Общих правил целостности всего два:

Первое правило: **В первичном ключе не должно быть атрибутов, которые могут принимать неизвестные (неопределенные) значения.**
В SQL для описания первичного ключа используется определение `PRIMARY KEY`

Второе правило: **База данных не должна содержать несогласованных значений внешних ключей .**
*Замечание: **Несогласованное значение** – это значение внешнего ключа, для которого не существует отвечающего ему значения первичного ключа.*
Определить внешний ключ можно оператором SQL:

```
FOREIGN KEY (список атрибутов внешнего ключа)
REFERENCES имя_таблицы с первичным ключом
            (список атрибутов первичного ключа)
```

Пример 8.
Рассмотрим ситуацию, когда в базе данных предприятия хранятся сведения о сотрудниках и о детях сотрудников.
Ответим на такие вопросы:

1. *Что произойдет, если удалить запись, на которую ссылается внешний ключ в таблице «ДЕТИ_СОТРУДНИКОВ» (например, удалить запись о сотруднике с Таб.№ 5052, если этот сотрудник имеет двух детей)?*
2. *Что произойдет при попытке обновить значение первичного ключа записи о сотруднике, на который ссылается внешний ключ?*

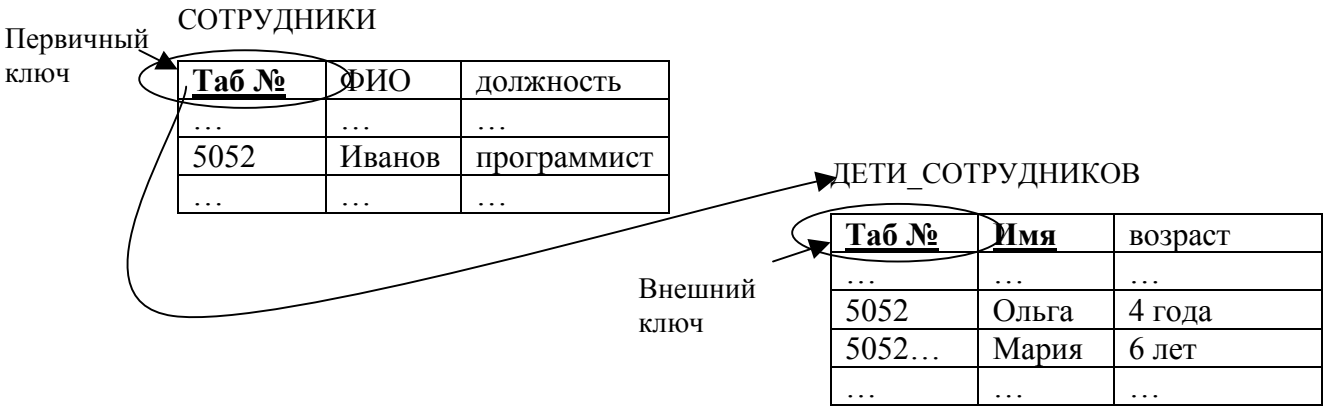

Рисунок 8.2 – Первичный и внешний ключи в таблицах

В общем, существует по крайней мере две возможности:

- Ограничить (в SQL - оператор `NO ACTION`) –

ограничить операцию «удаление»/«обновление» до того момента, когда уже не будет существовать соответствующее данному значению Таб № первичного ключа записей в отношении с внешним ключом (в противном случае операция запрещается)

- Каскадировать (в SQL - оператор `CASCADE`) –

каскадировать операцию «удаления»/«обновления», удаляя/обновляя также и внешние ключи.

Поэтому, в SQL задается ссылочная целостность в явном виде:

```
FOREIGN KEY (список атрибутов внешнего ключа)
REFERENCES имя_таблицы
     ON DELETE опция
     ON UPDATE опция
```

где задание поведения СУБД при удалении/обновлении ключа в родительской таблице задается опцией:

`CASCADE`
`NO ACTION`
вызов определенной процедуры (хранимой процедуры или процедуры триггеров)

Пример 9.
Вместо каскадирования или ограничения на изменение значений может вызываться некоторая процедура, например, при удалении записи информация заносится в архивную базу и т.п.

8.3 Целостность и неопределенные значения данных (null значения)

Проблема отсутствующей информации достаточно часто встречается на практике. Для решения проблемы создаются разные подходы. Часто используются «значения по умолчанию». Например, если цвет детали не известен, или не имеет значения в данной базе, то «значение по умолчанию» - «нет» иначе – название цвета.
Другой подход, предложенный Коддом, базируется на обозначении отсутствующего значения специальным **маркером**, который называется **null-значением**. Это значение

используется всегда, когда правильное значение для любого атрибута базы неизвестно или отсутствует.
Теория, изучающая null-значения, базируется на трехзначной логике (когда разрешаются три значения: «да», «нет», «UNK" – unknown - неизвестно)
Таким образом, становится очевидным правило целостности по сущностям, которые мы сформулировали в лекции 7:
«Ни один элемент первичного ключа не может иметь неопределенное, «null-значение»

Аналогично и для правила целостности по ссылкам:
«Всегда значения элементов внешнего ключа или являются null-значениями, или совпадают с некоторыми не null- значениями элементов первичного ключа»

Если СУБД разрешает использование null-значений, то к двум возможностям обработки ссылок по внешним ключам, добавляется третья.

Правило null-значений: **Для каждого внешнего ключа разработчик должен решить, может ли внешний ключ содержать null-значения.**

Как правило, можно задать разрешение на использование Null-значений на всю базу данных с помощью оператора SQL -

`NULLS ALLOWED` (разрешается использование Null-значений), либо
`NULLS NOT ALLOWED` (запрещается использование Null-значений)

Тогда, если Null-значения разрешаются, то добавляется новая опция: `SET NULL` (аннулировать) наряду с `CASCADE` и `NO ACTION`.
В соответствии с опцией `SET NULL`, при изменении записи о сотруднике внешний ключ устанавливается равным Null для всех записей о детях, а запись о сотруднике потом удаляется/обновляется.

В SQL92 к опциям `CASCADE` и `NO ACTION` добавляется SET NULL и SET DEFAULT:

```
FOREIGN KEY (список атрибутов внешнего ключа)
REFERENCES имя_таблицы
     ON DELETE опция
     ON UPDATE опция
```

где `опция` –
- либо `CASCADE`
- либо `NO ACTION`
- либо `SET NULL`
- либо `SET DEFAULT` (как установлено по умолчанию для всей базы данных)

Пример 10.

Допустим, мы разрешили существование Null-значений в таблице «ДЕТИ_СОТРУДНИКОВ». Примером может быть ситуация «сын полка», когда сотрудника увольняют, но о детях продолжают заботиться, и информация о них остается в базе данных. Опишем эту ситуацию на SQL

```
CREATE TABLE  Employees
```

```
     (tab_number integer NOT NULL UNIQUE,
     name varchar (50) NOT NULL,
     position varchar (20) NOT NULL)
     PRIMARY KEY tab_number;
CREATE TABLE Childrens
     (tab_number integer NOT NULL,
     child_name varchar (10) NOT NULL,
     age integer NOT NULL)
PRIMARY KEY (tab_number, child_name)
FOREIGN KEY (tab_number) REFERENCES Employees
     ON DELETE SET NULL
     ON UPDATE CASCADE;
```