

Содержание

| | |
|--|-----------|
| 1 Основные понятия БД. Архитектура СБД | 2 |
| 1.1 База данных (БД) | 2 |
| 1.2 Система БД (СБД) | 2 |
| 1.3 Основные составляющие СБД | 2 |
| 1.4 Реляционные БД | 2 |
| 1.5 Дореляционные и постреляционные БД | 3 |
| 1.6 Три уровня архитектуры (внутренний, внешний, концептуальный) | 5 |
| 2 Реляционная модель БД | 7 |
| 2.1 Таблица, столбец, строка, поле | 7 |
| 2.2 Домены, отношения, атрибуты, кортежи | 7 |
| 2.3 Степень, каринальное число отношения | 11 |
| 2.4 Свойства отношений | 11 |
| 3 Целостность данных | 12 |
| 3.1 Ограничения (правила) целостности | 12 |
| 3.2 Потенциальные ключи, первичный ключ | 12 |
| 3.3 Внешние ключи | 13 |
| 3.4 Ссылочная целостность | 14 |
| 3.5 NULL-значения | 16 |

1 Основные понятия БД. Архитектура СБД

1.1 База данных (БД)

Определение:

База данных (БД) — набор постоянных данных, которые используются прикладными системами для какого-либо предприятия.

1.2 Система БД (СБД)

Определение:

Система баз данных (СБД) — это, по сути, не что иное, как компьютеризированная система хранения записей. Саму же базу данных можно рассматривать как подобие электронной картотеки, то есть хранилище для некоторого набора занесённых в компьютер файлов данных (где файл — абстрактный набор данных).

1.3 Основные составляющие СБД

1. данные;
2. аппаратное обеспечение;
3. программное обеспечение;
4. пользователи.

или

1. СУБД;
2. прикладное программное обеспечение базы данных и операционной системы;
3. технические средства, обеспечивающие обслуживание пользователей.

1.4 Реляционные БД

Определение реляционной БД:

Реляционной базой данных называется набор отношений.

1.5 Дореляционные и постреляционные БД

Файлы и файловые системы:

Первые БД были созданы на основе файловых систем. Для каждой прикладной программы предоставлялся свой набор данных, оформленный в виде файла со своей структурой.

Такие системы имели ряд недостатков:

1. ФС не знает конкретной структуры файла. Поэтому при малейших изменениях в структуре файлов приходилось менять программы, работающие с ними;
2. В файловой системе для каждого файла имеется информация о пользователе — владельце файла и определённые доступные для других пользователей. Из-за этого было очень трудно выстраивать ФС, так как часто файл созданные одной программой не доступен для другой;
3. Файл в файловой системе был недоступен для нескольких пользователей одновременно. Его можно было модифицировать одновременно только 1 человеку, что делало совместную работу невозможной.

Иерархические БД:

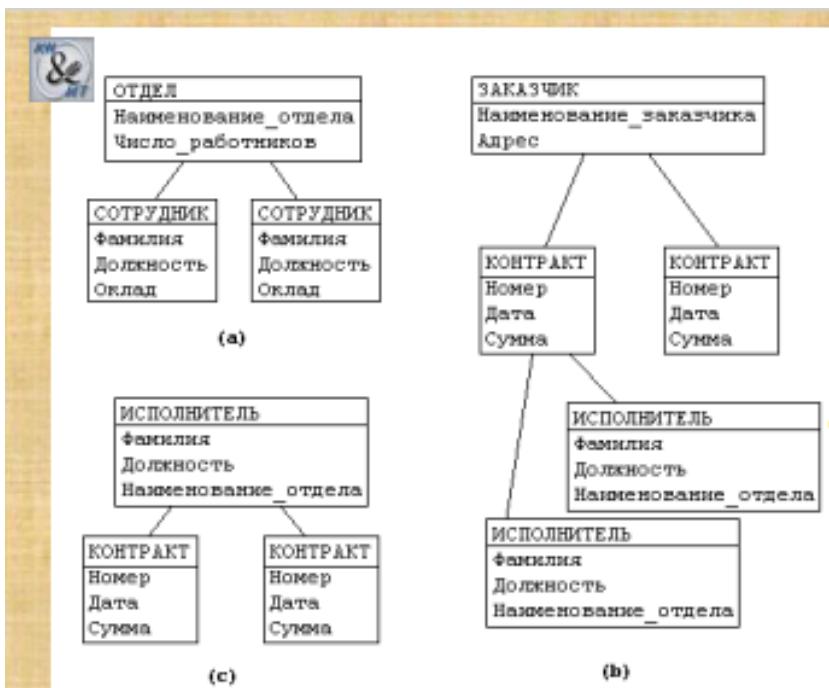


Иерархические БД

Рассмотрим следующую модель данных предприятия: предприятие состоит из отделов, в которых работают сотрудники. В каждом отделе может работать несколько сотрудников, но сотрудник не может работать более чем в одном отделе.

Поэтому, для информационной системы управления персоналом необходимо создать структуру, состоящую из родительской записи **ОТДЕЛ** (**НАИМЕНОВАНИЕ_ОТДЕЛА**, **ЧИСЛО_РАБОТНИКОВ**) и дочерней записи **СОТРУДНИК** (**ФАМИЛИЯ**, **ДОЛЖНОСТЬ**, **ОКЛАД**).

Для автоматизации учета контрактов с заказчиками необходимо создание еще одной иерархической структуры: заказчик - контракты с ним - сотрудники, задействованные в работе над контрактом. Это дерево будет включать записи
ЗАКАЗЧИК(**НАИМЕНОВАНИЕ_ЗАКАЗЧИКА**, **АДРЕС**),
КОНТРАКТ(**НОМЕР**, **ДАТА**, **СУММА**), **ИСПОЛНИТЕЛЬ** (**ФАМИЛИЯ**, **ДОЛЖНОСТЬ**, **НАИМЕНОВАНИЕ_ОТДЕЛА**).



Недостатки иерархических БД:

1. Частично дублируется информация между записями СОТРУДНИК и ИСПОЛНИТЕЛЬ, причем в иерархической модели данных не предусмотрена поддержка соответствия между парными записями.

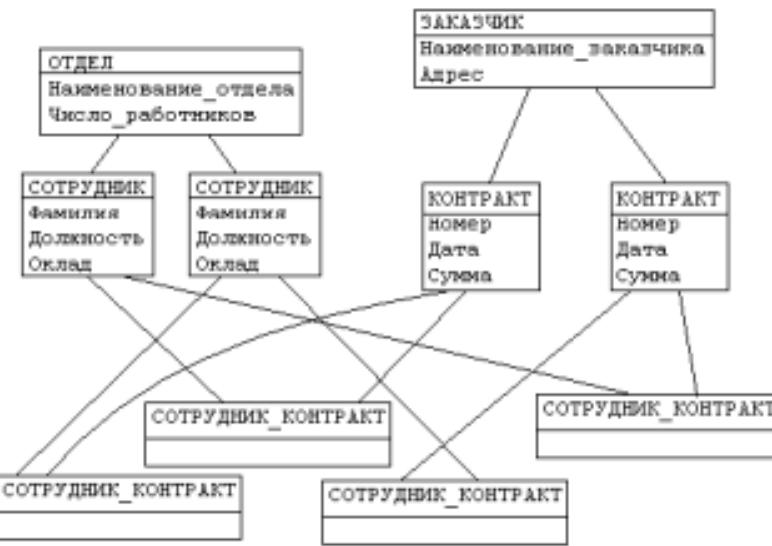
2. Иерархическая модель реализует отношение между исходной и дочерней записью по схеме **1:N**. Допустим, что исполнитель может принимать участие более чем в одном контракте (связь типа **M:N**). В этом случае в базу данных необходимо ввести еще одно групповое отношение, в котором ИСПОЛНИТЕЛЬ будет являться исходной записью, а КОНТРАКТ - дочерней (рис. (c)). Таким образом, опять дублируется информация.

7

Сетевые БД:



Сетевые БД



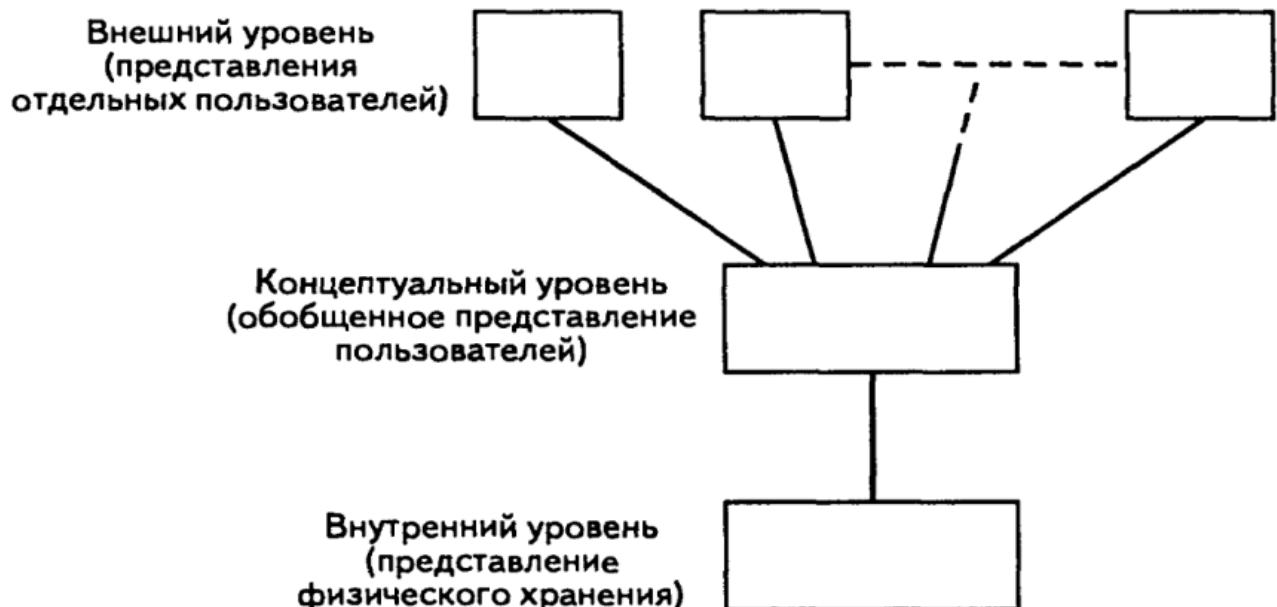
Иерархическая структура преобразовывается в сетевую следующим образом:

- 1) деревья (а) и (б) заменяются одной сетевой структурой, в которой запись СОТРУДНИК входит в обе группы;
- 2) для отображения типа M:N вводится запись СОТРУДНИК_КОНТРАКТ, которая не имеет полей и служит только для связи записей КОНТРАКТ и СОТРУДНИК.

«Если Вы хотите запутать базу данных – сделайте ее сетевой» - разработчик БД из IBM.

8

1.6 Три уровня архитектуры (внутренний, внешний, концептуальный)



Определение:

Внутренний уровень — уровень наиболее близкий к физическому хранению

данных, т.е. связанный с со способами сохранения информации на физических устройствах хранения. Внутренний уровень — собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации.

Определение:

Концептуальный уровень — уровень, на котором база данных представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной базой данных. Фактически концептуальный уровень отражает обобщенную модель предметной области (объектов реального мира), для которой создавалась база данных. Как любая модель, концептуальная модель отражает только существенные, с точки зрения обработки, особенности объектов реального мира.

Определение:

Внешний уровень — самый верхний уровень, где каждое представление имеет свое "видение" данных. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению. Например, система распределения работ использует сведения о квалификации сотрудника, но ее не интересуют сведения об окладе, домашнем адресе и телефоне сотрудника, и наоборот, именно эти сведения используются в подсистеме отдела кадров.

Размышления:



| Внешний (PL/I) | Внешний (COBOL) |
|---|---|
| DCL 1 EMPP, 2 EMP# CHAR(6), 3 SAL FIXED BIN(31); | 01 EMPC. 02 EMPNO PIC X(6). 02 DEPTNO PIC X(4). |
| Концептуальный | |
| EMPLOYEE EMPLOYEE_NUMBER | CHARACTER (6) CHARACTER (4) |
| DEPARTMENT_NUMBER | |
| SALARY | NUMERIC (5) |
| Внутренний | |
| STORED_EMP | BYTES=20 |
| PREFIX | TYPE=BYTE(6), OFFSET=0 |
| EMP# | TYPE=BYTE(6), INDEX=EMPK, OFFSET=6, |
| DEPT# | TYPE=BYTE(4), OFFSET=12 |
| PAY | TYPE=FULLWORD, OFFSET=16 |

Трехуровневая архитектура БД позволяет обеспечить логическую (между 1 и 2 ур.) и физическую (между 2 и 3 ур.) независимость при работе с данными.

Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же БД.

Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной БД.

Выделение концептуального уровня позволило разработать аппарат централизованного управления БД.

11

2 Реляционная модель БД

2.1 Таблица, столбец, строка, поле

2.2 Домены, отношения, атрибуты, кортежи

Определение домена:

Домен — допустимое потенциальное множество значений для отдельно взятого типа.

Дополнительная информация:

1.2. Домен

Понятие домена более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования.

Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа. Например, в число значений домена "Имена" могут входить только те строки, которые могут изображать имя.

Данные считаются сравнимыми только в том случае, когда они относятся к одному домену.

Домены более точно можно определить как именованное множество скалярных значений одного типа. Эти скалярные значения называют **скалярами**. По сути, это наименьшая семантическая (смысловая) единица данных. У скаляров нет внутренней структуры, т.е. они не разложимы в данной реляционной модели.

Например, если имеется атрибут (свойство объекта) «ФИО», он предусматривает скаляры, содержащие фамилию, имя и отчество. Конечно, эти скаляры можно еще разбить на буквы, но тогда будет утрачен нужный смысл. То есть для данной модели наименьшими семантическими единицами данных будут именно фамилия, имя и отчество.

Из доменов, как уже говорилось, берутся значения атрибутов. На практике домены часто не описывают, а задают типом, форматом и другими свойствами данных. Каждый атрибут должен быть определен на **единственном домене**.

Основное назначение доменов — **ограничение сравнения** различных по смыслу атрибутов.

Например: Если для атрибутов №ЗачетнойКнижки отношения Студенты и №Кабинета для отношения Кабинеты домены заданы следующим образом:

№ зачетной книжки = {100000, 100001, 100002, ... 999999}

№ кабинета = {1, 2, 3, ... 999},

то система выдаст ошибку на запрос типа: «Вывести всех студентов, № зачетной книжки которых совпадает с № кабинета». Если же домены не определены, а определен только целый тип данных для атрибутов №ЗачетнойКнижки и №Кабинета, то подобный запрос выполнится, хотя не будет иметь смысла.

Еще одно возможное применение доменов – использование их в специальных **запросах**. Например, «Какие отношения в БД включают атрибуты, определенные на домене «№ зачетной книжки»?». В системе, поддерживающей домены, такой запрос будет иметь смысл и результатом его будет список отношений, где используется № зачетной книжки (это могут быть отношения Студенты, Занятия, Успеваемость, ...). А в системе, где домены не определены, реализовать такого рода запрос гораздо сложнее – если через имена атрибутов, то они могут не совпадать (имена атрибутов, содержащих № зачетной книжки могут варьироваться: № зачетки, № зачетной книжки и т.п.), а если через тип – то получится много лишних отношений, т.к. немало атрибутов может иметь целый тип данных.

Определение отношения, атрибута и кортежа:

Еще одно возможное применение доменов – использование их в специальных **запросах**. Например, «Какие отношения в БД включают атрибуты, определенные на домене «№ зачетной книжки»?». В системе, поддерживающей домены, такой запрос будет иметь смысл и результатом его будет список отношений, где используется № зачетной книжки (это могут быть отношения Студенты, Занятия, Успеваемость, ...). А в системе, где домены не определены, реализовать такого рода запрос гораздо сложнее – если через имена атрибутов, то они могут не совпадать (имена атрибутов, содержащих № зачетной книжки могут варьироваться: № зачетки, № зачетной книжки и т.п.), а если через тип – то получится много лишних отношений, т.к. немало атрибутов может иметь целый тип данных.

Что такое отношение в реляционной модели

У отношения нет физического представления. Реляционная база данных — это попытка отразить реляционную модель, а не ее точная копия.

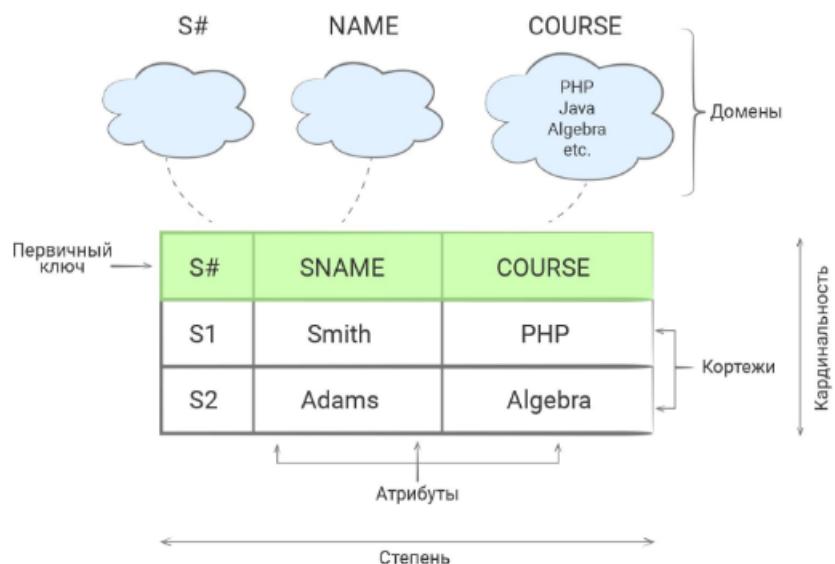
Реляционная модель опирается на раздел математики теорию множеств, которую нужно знать хотя бы немного. Множество — это совокупность произвольных элементов, которые объединены по некоторому признаку. Например, множество натуральных чисел или множество учеников одного класса. Изучить этот раздел подробнее вы можете в нашем курсе [«Теория множеств»](#).

Из любого множества можно выделить **подмножество** — множество элементов, все элементы которого входят в исходное множество. Подмножество — это часть множества. Например, множество натуральных чисел — это подмножество по отношению к множеству целых чисел, которое в свою очередь — подмножество рациональных чисел. Из этого следует, что натуральные числа — это подмножество рациональных чисел.

Еще одно важное понятие — **кортеж**. Это упорядоченный набор данных фиксированной длины. Элементами кортежа может быть все что угодно. Это математический способ представить некоторый набор связанных данных.

На основе этих понятий можно сформулировать более понятное определение отношения. Отношение — это множество кортежей, которые называются **телом отношения**. В нем каждый кортеж соответствует схеме.

Схема — это заголовок отношения. Она описывает общую структуру кортежей, количество элементов внутри них и их типы. Каждый такой элемент называется **атрибутом**:



Отношение визуально похоже на таблицу, но у него есть существенные отличия:

- Нет двух одинаковых элементов (кортежей)
- Порядок кортежей не определен
- Порядок атрибутов в заголовке не определен

По этой причине таблица непригодна для точного описания отношения. В любой таблице порядок столбцов (атрибуты схемы) и строк (кортеж) строго определен. При этом на практике мы оперируем таблицами и не можем игнорировать их.

По Батраевой:

1.3. Кортеж, отношение

Фундаментальным понятием реляционной модели данных является понятие **отношения**. В определении понятия отношения будем следовать книге К. Дейта [11].

Определение 1. Атрибут отношения есть пара вида
<Имя_атрибута : Имя_домена>.

Имена атрибутов должны быть уникальны в пределах отношения. Часто имена атрибутов отношения совпадают с именами соответствующих доменов.

Определение 2. Отношение R , определенное на множестве доменов D_1, \dots, D_n (не обязательно различных), содержит две части: заголовок и тело.

Заголовок отношения содержит фиксированное количество атрибутов отношения:

$(\langle A_1:D_1 \rangle, \dots, \langle A_n:D_n \rangle)$

Тело отношения содержит множество кортежей отношения. Каждый **кортеж отношения** представляет собой множество пар вида *<Имя_атрибута : Значение_атрибута>*:

$(\langle A_1:V_{11} \rangle, \dots, \langle A_n:V_{n1} \rangle)$

таких что значение V_{i1} атрибута A_i принадлежит домену D_i

Отношение обычно записывается в виде:

$R(\langle A_1:D_1 \rangle, \dots, \langle A_n:D_n \rangle)$, или короче $R(A_1, \dots, A_n)$, или просто R .

2.3 Степень, каринальное число отношения

Определение степени отношения:

Число атрибутов в отношении называют степенью отношения.

Каринальное число отношения:

Количество кортежей в отношении называется мощностью или каринальным числом отношения.

2.4 Свойства отношений

1. В отношении нет одинаковых кортежей. 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 (сверху вниз). Действительно, несмотря на то, что мы изобразили отношение "Сотрудники" в виде таблицы, нельзя сказать, что сотрудник Иванов "предшествует" сотруднику Петрову. Одно и то же отношение может быть изображено разными таблицами, в которых строки идут в различном порядке;
3. Атрибуты не упорядочены (слева направо). Т.к. каждый атрибут имеет уникальное имя в пределах отношения, то порядок атрибутов не имеет значения. Одно и то же отношение может быть изображено разными таблицами, в которых столбцы идут в различном порядке;
4. Все значения атрибутов атомарны. Это четвертое отличие отношений от таблиц — в ячейки таблиц можно поместить что угодно — массивы, структуры, и даже другие таблицы.

3 Целостность данных

3.1 Ограничения (правила) целостности

Правило целостности сущностей (NULL значения в потенциальных ключах):

Атрибуты, входящие в состав некоторого потенциального ключа не могут принимать null-значений.

Правило целостности внешних ключей:

Так как внешние ключи фактически служат ссылками на кортежи в другом (или в том же самом) отношении, то эти ссылки не должны указывать на несуществующие объекты.

Формально:

Внешние ключи не должны быть несогласованными, то есть для каждого значения внешнего ключа должно существовать значение первичного ключа в родительском отношении.

3.2 Потенциальные ключи, первичный ключ

Определение потенциального (и чуть-чуть первичного) ключа:

Определение 1. Пусть дано отношение R . Подмножество атрибутов K отношения R будем называть **потенциальным ключом**, если K обладает следующими свойствами:

- 1) *Свойством уникальности* - в отношении R не может быть двух различных кортежей, с одинаковым значением K .
- 2) *Свойством неизбыточности* - никакое подмножество в K не обладает свойством уникальности.

Любое отношение имеет, по крайней мере, один потенциальный ключ. Действительно, если никакой атрибут или группа атрибутов не являются потенциальным ключом, то, в силу уникальности кортежей, все атрибуты вместе образуют потенциальный ключ.

Потенциальный ключ, состоящий из одного атрибута, называется **простым**. Потенциальный ключ, состоящий из нескольких атрибутов, называется **составным**.

Отношение может иметь несколько потенциальных ключей. Традиционно, один из потенциальных ключей называется **первичным**, а остальные - **альтернативными**. Различия между первичным и альтернативными ключами могут быть важны в конкретной реализации реляционной СУБД, но с точки зрения реляционной модели данных, нет оснований выделять таким образом один из потенциальных ключей.

Замечание. Понятие потенциального ключа является **семантическим** понятием и отражает некоторый смысл (трактовку) понятий из конкретной предметной области.

3.3 Внешние ключи

Различные объекты предметной области, информация о которых хранится в базе данных, всегда взаимосвязаны друг с другом. Например, накладная на поставку товара *содержит* список товаров с количествами и ценами, сотрудник предприятия *имеет* детей, *числится* в подразделении и т.д. Термины "содержит", "имеет", "числится" отражают взаимосвязи между понятиями "накладная" и "список товаров", "сотрудник" и "дети", "сотрудник" и "подразделение". Такие взаимосвязи отражаются в реляционных базах данных при помощи **внешних ключей**, связывающих несколько отношений.

Определение внешнего ключа:

Определение 2.

Пусть дано отношение R . Подмножество атрибутов FK отношения R будем называть **внешним ключом**, если:

1) Существует отношение S (R и S не обязательно различны) с потенциальным ключом K .

2) Каждое значение FK в отношении R всегда совпадает со значением K для некоторого кортежа из S , либо является null-значением.

Отношение S называется **родительским отношением**, отношение R называется **дочерним отношением**.

Замечание. Внешний ключ, как правило, не обладает *свойством уникальности*. Это, собственно, и дает тип отношения 1:N.

Замечание. Хотя каждое значение внешнего ключа обязано совпадать со значениями потенциального ключа в некотором кортеже родительского отношения, то обратное, вообще говоря, неверно.

Например, могут существовать поставщики, не поставляющие никаких деталей.

Замечание. Для внешнего ключа не требуется, чтобы он был компонентом некоторого потенциального ключа.

3.4 Ссыльная целостность

Операции ведущие к нарушению ссылочной целостности при изменении родительского отношения:

1. Вставка кортежа в родительском отношении. При вставке кортежа в родительское отношение возникает новое значение потенциального ключа. Т.к. допустимо существование кортежей в родительском отношении, на которые нет ссылок из дочернего отношения, то вставка кортежей в родительское отношение не нарушает ссылочной целостности;
2. Обновление кортежа в родительском отношении. При обновлении кортежа в родительском отношении может измениться значение потенциального ключа. Если есть кортежи в дочернем отношении, ссылающиеся на обновляемый кортеж, то значения их внешних ключей станут некорректными. Обновление кортежа в родительском отношении может привести к нарушению ссылочной целостности, если это обновление затрагивает значение потенциального ключа;
3. Удаление кортежа в родительском отношении. При удалении кортежа в родительском отношении удаляется значение потенциального ключа. Если есть кортежи в дочернем отношении, ссылающиеся на удаляемый кортеж,

то значения их внешних ключей станут некорректными. Удаление кортежей в родительском отношении может привести к нарушению ссылочной целостности.

Операции ведущие к нарушению ссылочной целостности при изменении дочернего отношения:

1. Вставка кортежа в дочернее отношение. Нельзя вставить кортеж в дочернее отношение, если вставляемое значение внешнего ключа некорректно. Вставка кортежа в дочернее отношение привести к нарушению ссылочной целостности;
2. Обновление кортежа в дочернем отношении. При обновлении кортежа в дочернем отношении можно попытаться некорректно изменить значение внешнего ключа. Обновление кортежа в дочернем отношении может привести к нарушению ссылочной целостности;
3. Удаление кортежа в дочернем отношении. При удалении кортежа в дочернем отношении ссылочная целостность не нарушается.

Вывод:

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

- Обновление кортежа в родительском отношении.
- Удаление кортежа в родительском отношении.
- Вставка кортежа в дочернее отношение.
- Обновление кортежа в дочернем отношении.

Стратегии поддержания ссылочной целостности

RESTRICT (ОГРАНИЧИТЬ)- не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи в дочернем отношении, связанные с некоторым кортежем в родительском отношении.

CASCADE (КАСКАДИРОВАТЬ)- разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других отношениях так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи.

Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. Необходимо учитывать, что дочернее отношение само может быть родительским для некоторого третьего отношения. При этом может дополнительно потребоваться выполнение какой-либо стратегии и для этой связи и т.д. Если при этом какая-либо из каскадных операций (любого уровня) не может быть выполнена, то необходимо отказаться от первоначальной операции и вернуть базу данных в исходное состояние.

Стратегии поддержания ссылочной целостности реализуются в основном двумя способами:

- на уровне определения данных

```
alter table POSTAVKI add foreign key (fkidTovar) references  
TOVAR(idTovar) on delete cascade
```

- с использованием триггеров

```
create trigger trTovar for Tovar before update as  
declare variable n;  
begin  
select count(*) from Postavki where old.idTovar = Postavki.fkidTovar into n;  
if n>0 then delete from Postavki where old.idTovar=Postavki.fkidTovar ;  
end
```

3.5 NULL-значения

Достаточно часто встречается ситуация, когда данные неизвестны или не полны. Например, место жительства или дата рождения человека могут быть неизвестны (база данных разыскиваемых преступников). Если вместо неизвестного адреса уместно было бы вводить пустую строку, то что вводить вместо неизвестной даты? Ответ - пустую дату – не удовлетворителен, т.к. простейший запрос "выдать список людей в порядке возрастания дат рождения" даст заведомо неправильных ответ.

Для того чтобы обойти проблему неполных или неизвестных данных, в базах данных могут использоваться типы данных, пополненные так называемым **null-значением**. Null-значение - это, собственно, не значение, а некий маркер, показывающий, что значение неизвестно.

Дополнительная информация:

2.2. Трехзначная логика (3VL)

Т.к. null-значение обозначает на самом деле тот факт, что значение неизвестно, то любые алгебраические операции (сложение, умножение, конкатенация строк и т.д.) должны давать также неизвестное значение, т.е. null. Действительно, если, например, вес детали неизвестен, то неизвестно также, сколько весят 10 таких деталей.

При сравнении выражений, содержащих null-значения, результат также может быть неизвестен, например, значение истинности для выражения есть null, если один или оба аргумента есть null. Таким образом, определение истинности логических выражений базируется на **трехзначной логике (three-valued logic, 3VL)**, в которой кроме значений Т - ИСТИНА и F - ЛОЖЬ, введено значение U - НЕИЗВЕСТНО. Логическое значение U - это то же самое, что и null-значение.

| AND | F | T | U | OR | F | T | U | NOT | |
|-----|---|---|---|----|---|---|---|-----|---|
| F | F | F | F | F | F | T | U | F | T |
| T | F | T | U | T | T | T | T | T | F |
| U | F | U | U | U | U | T | U | U | U |

Имеется несколько парадоксальных следствий применения трехзначной логики.

Парадокс 1. Null-значение не равно самому себе. Действительно, выражение $null = null$ дает значение не ИСТИНА, а НЕИЗВЕСТНО. Значит выражение не обязательно ИСТИНА!

Парадокс 2. Неверно также, что null-значение не равно самому себе! Действительно, выражение $null \neq null$ также принимает значение не ИСТИНА, а НЕИЗВЕСТНО! Значит также, что и выражение тоже не обязательно ЛОЖЬ!

Парадокс 3. $a \text{ or } (\text{not } a)$ не обязательно ИСТИНА. Значит, в трехзначной логике не работает принцип исключенного третьего (любое высказывание либо истинно, либо ложно).