КАФЕДРА: КИИ

ТЕМА: СТРУКТУРНАЯ, МАНИПУЛЯЦИОННАЯ И ЦЕЛОСТНАЯ СОСТАВЛЯЮЩИЕ РЕЛЯЦИОННОЙ МОДЕЛИ ДАННЫХ

ДИСЦИПЛИНА: ВВЕДЕНИЕ В БАЗЫ ДАННЫХ

ДЛЯ СТУДЕНТОВ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ
6B06105 «DATA SCIENCE»

СТАРШИЙ ПРЕПОДАВАТЕЛЬ КЛЮЕВА Е.Г.

ПЛАН ЛЕКЦИИ

1. Общая характеристика реляционной модели;
2. Структурная составляющая модели;
3. Фундаментальные свойства отношений реляционной модели;
4. Внутренняя организация реляционных БД;
5. Виды связей между отношениями БД;
6. Индексы;

ПЛАН ЛЕКЦИИ

7. Целостность базы данных;

8. Классификация ограничений целостности;

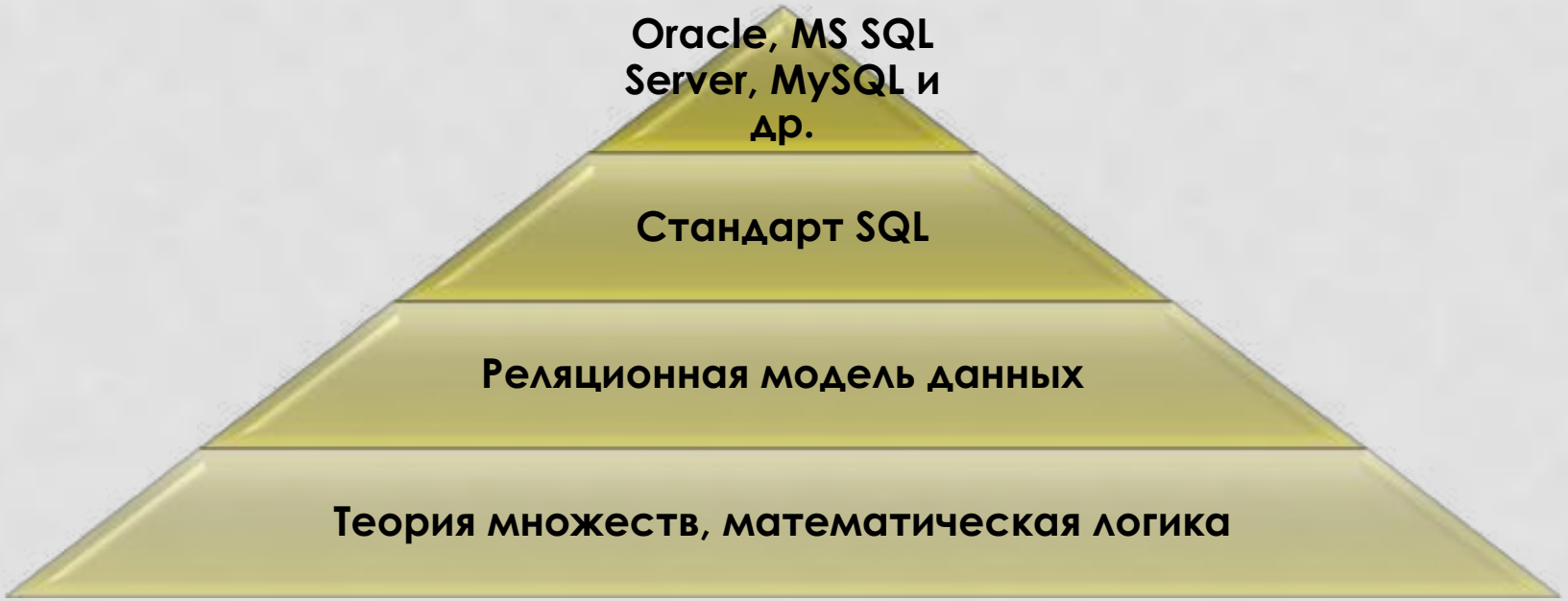
9. Классификация и характеристика средств манипулирования реляционными данными;

10. Общая интерпретация реляционных операций;

11. Реляционное исчисление.

СТРУКТУРА РЕЛЯЦИОННОЙ МОДЕЛИ

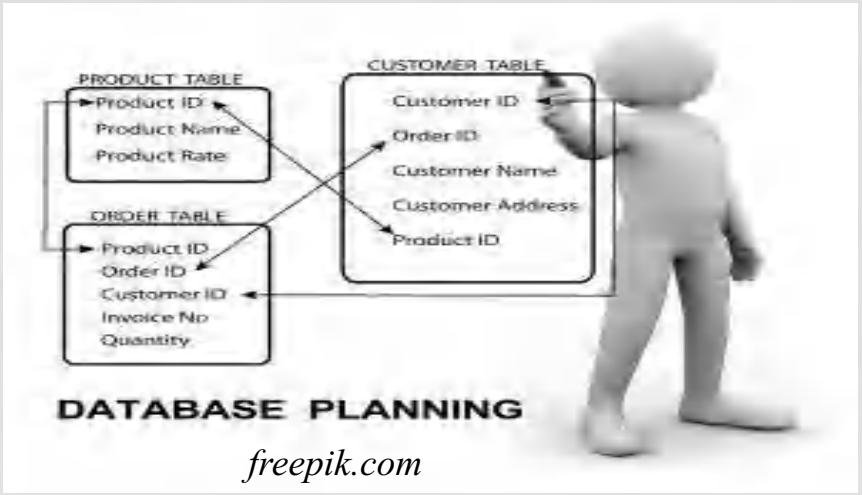
Взаимосвязь реляционной модели данных, стандарта языка SQL и различных его реализаций можно условно изобразить в виде следующей пирамиды:



СТРУКТУРА РЕЛЯЦИОННОЙ МОДЕЛИ

Реляционная модель данных содержит три компоненты:

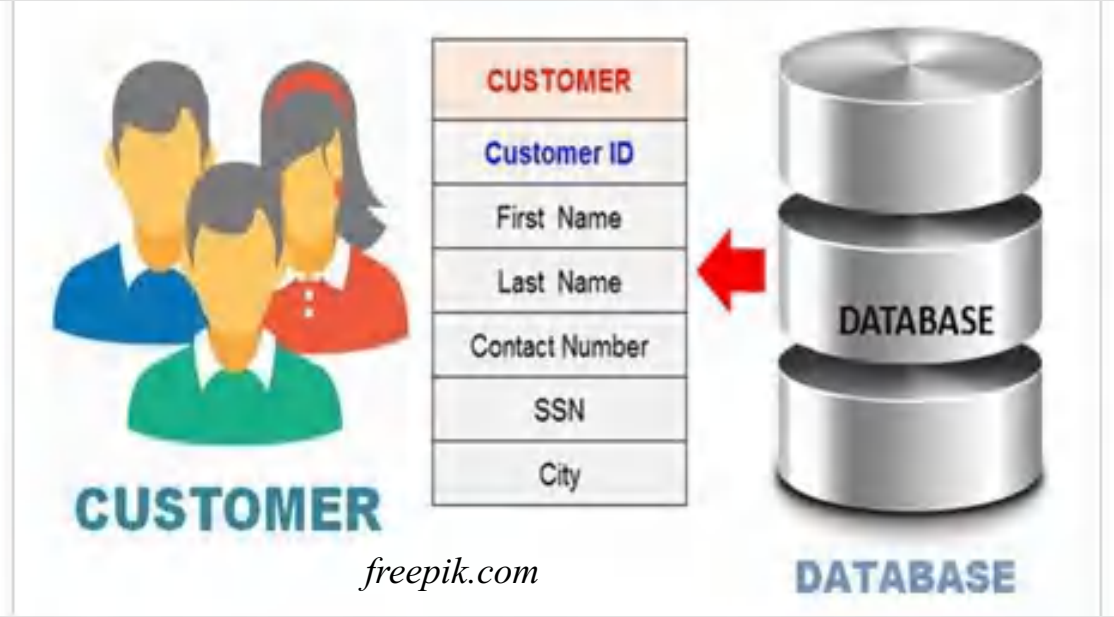
1. структурная составляющая;
2. манипуляционная составляющая;
3. целостная составляющая.



freepik.com

СТРУКТУРНАЯ СОСТАВЛЯЮЩАЯ МОДЕЛИ

Структурная часть постулирует, что единственной структурой данных, используемой в реляционной модели, являются нормализованные n-арные отношения.



МАНИПУЛЯЦИОННАЯ СОСТАВЛЯЮЩАЯ

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

ЦЕЛОСТНАЯ СОСТАВЛЯЮЩАЯ МОДЕЛИ

Целостная часть описывает ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных - это *целостность сущностей* и *целостность внешних ключей (по ссылкам)*.

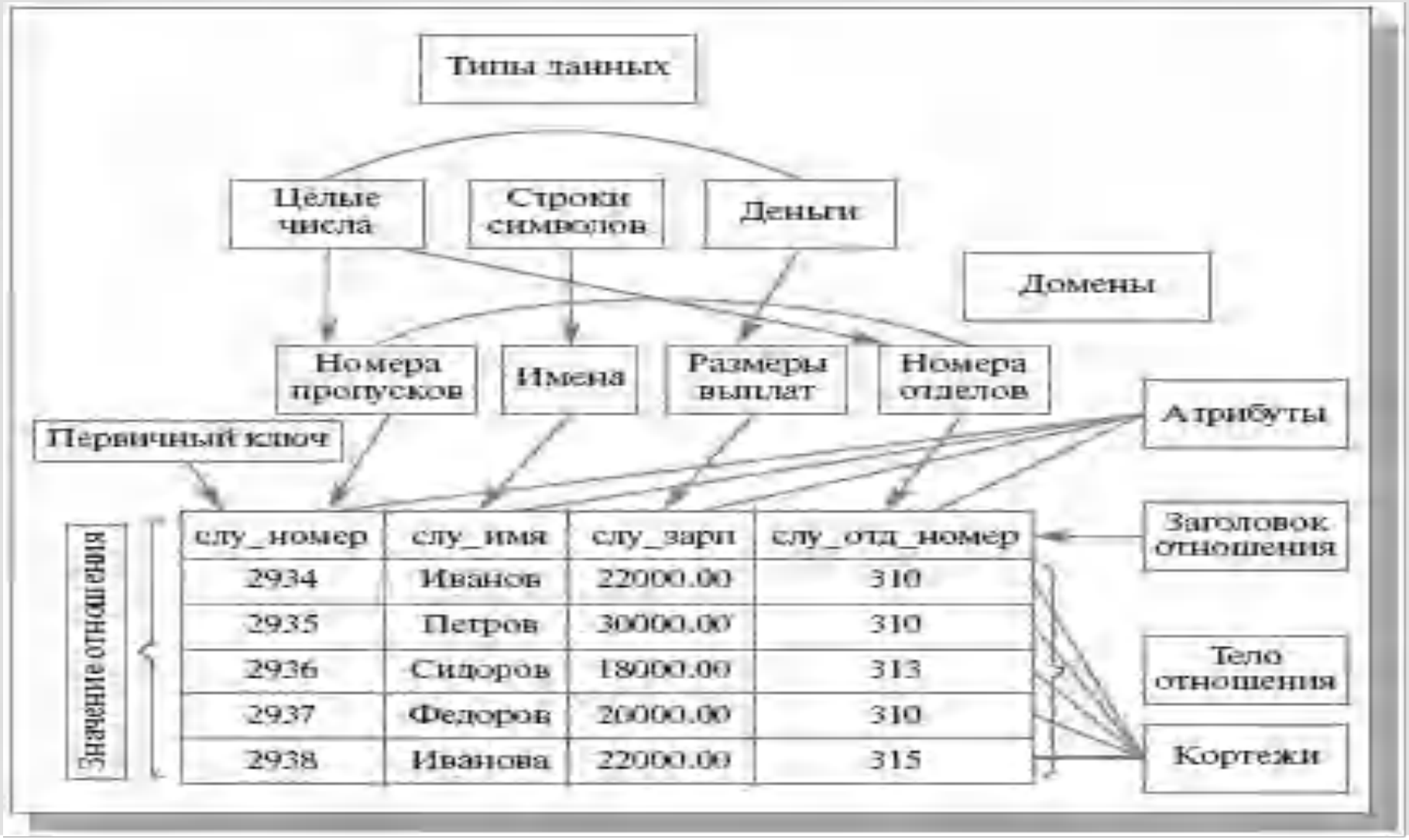
СТРУКТУРНАЯ ЧАСТЬ МОДЕЛИ РЕЛЯЦИОННОЙ БД

Основными понятиями реляционных баз данных являются:

- тип данных;
- домен;
- атрибут;
- кортеж;
- ключ;
- отношение;
- схема отношения.

freepik.com

СТРУКТУРНАЯ ЧАСТЬ МОДЕЛИ РЕЛЯЦИОННОЙ БД



ТИП ДАННЫХ В РЕЛЯЦИОННОЙ МОДЕЛИ

Понятие ***тип данных*** в реляционной модели данных полностью адекватно понятию типа данных в языках программирования.

Тип данных определяет:

1) множество значений;

2) набор операций, которые можно применять к таким значениям;

3) способ реализации хранения значений и выполнения операций.

ТИП ДАННЫХ В РЕЛЯЦИОННОЙ МОДЕЛИ

Реляционная модель требует, чтобы *типы* используемых *данных* были простыми.

Простые, или атомарные, типы данных не обладают внутренней структурой (или не должна учитываться внутренняя структура данных, но тогда должны быть описаны действия, которые можно производить с данными как с единым целым).

В некоторых реляционных СУБД можно создать свой тип данных, описать возможные действия с этим типом данных.

ТИП ДАННЫХ В РЕЛЯЦИОННОЙ МОДЕЛИ

В современных реляционных БД допускается хранение:

- символьных данных;
- числовых данных;
- битовых строк;
- специализированных числовых данных (таких как "деньги");
- специальных "темпоральных" данных (дата, время, временной интервал).

ДОМЕН В РЕЛЯЦИОННОЙ МОДЕЛИ

Домен - это семантическое понятие, которое можно рассматривать как подмножество значений некоторого базового типа данных, имеющих определенный смысл.

Домен определяется заданием некоторого базового типа данных и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

ДОМЕН В РЕЛЯЦИОННОЙ МОДЕЛИ

Домен характеризуется следующими свойствами:

- домен имеет уникальное имя (в пределах базы данных);
- домен определен на некотором простом типе данных или на другом домене;
- домен может иметь некоторое логическое условие, позволяющее описать подмножество данных, допустимых для данного домена;
- домен несет определенную смысловую нагрузку.

ДОМЕН В РЕЛЯЦИОННОЙ МОДЕЛИ

Описание домена

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

$$D = \{n \in N : n \geq 18 \; and \; n \leq 63\}$$

ДОМЕН В РЕЛЯЦИОННОЙ МОДЕЛИ

Основное значение доменов состоит в том, что домены ограничивают сравнения.

Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип

Например, домены "Вес детали" и "Имеющееся количество деталей" можно одинаково описать как множество неотрицательных целых чисел, но смысл этих доменов будет различным, и это будут различные домены.

ОТНОШЕНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ

Подмножество декартова произведения доменов называется ***отношением***.

Отношение содержит две части: заголовок и тело.

Заголовок отношения содержит фиксированное количество атрибутов отношения:

(<A1:D1>, <A2:D2>,... <An:Dn>)

Имена атрибутов должны быть уникальны в пределах отношения.

Тело отношения содержит множество кортежей отношения.

ОТНОШЕНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ

Отношение обычно записывается в виде:

R(<A1:D1>, <A2:D2>,... <An:Dn>),

или короче:

R(A1, A2, . . ., An),

или просто:

R

КОРТЕЖ РЕЛЯЦИОННОЙ МОДЕЛИ

Кортеж, соответствующий данной схеме отношения, - это множество пар вида {имя атрибута, значение атрибута}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения.

"Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается):

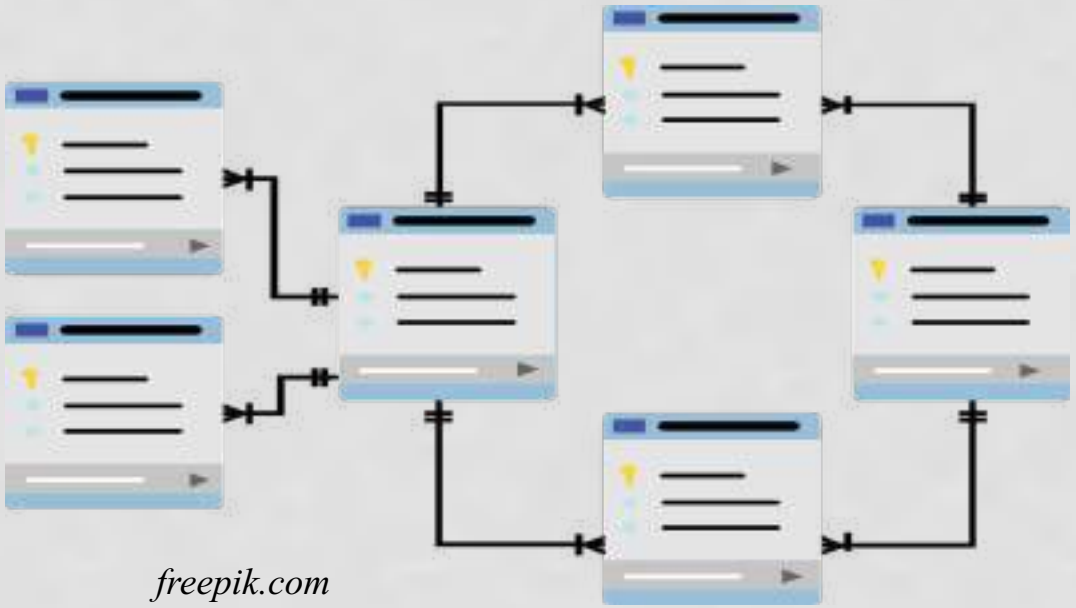
(<A1:Val1>, <A2:Val2>,... <An:Valn>)

таких что значение *Vali* атрибута *Ai* принадлежит домену *Di*

РЕЛЯЦИОННЫЕ ТЕРМИНЫ

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

Число кортежей отношения называют *мощностью* отношения.



freepik.com

РЕЛЯЦИОННЫЕ ТЕРМИНЫ

Схема отношения - это именованное множество пар {имя атрибута, имя домена} (или типа, если понятие домена не поддерживается).

Схема реляционной БД (в структурном смысле) - это набор именованных схем отношений.

Реляционной базой данных называется набор схем отношений.

РЕЛЯЦИОННАЯ МОДЕЛЬ

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)	Количество столбцов таблицы
Мощность отношения	Количество строк таблицы
Домены и типы данных	Типы данные в ячейках таблицы

ФУНДАМЕНТАЛЬНЫЕ СВОЙСТВА ОТНОШЕНИЙ

1. Отсутствие кортежей-дубликатов.
2. Отсутствие упорядоченности кортежей.
3. Отсутствие упорядоченности атрибутов.
4. Атомарность значений атрибутов.

ФУНДАМЕНТАЛЬНЫЕ СВОЙСТВА ОТНОШЕНИЙ

Отсутствие кортежей-дубликатов

В отношении нет одинаковых кортежей. Это свойство следует из определения отношения как множества кортежей и, как всякое множество, не может содержать неразличимые элементы.

Из этого свойства вытекает наличие у каждого отношения так называемого первичного ключа - набора атрибутов, значения которых однозначно определяют кортеж отношения. Таблицы в отличие от отношений могут содержать одинаковые строки.

ФУНДАМЕНТАЛЬНЫЕ СВОЙСТВА ОТНОШЕНИЙ

Отсутствие упорядоченности кортежей

Кортежи не упорядочены (сверху вниз). Это свойство является следствием определения отношения как множества кортежей, а множество не упорядочено.

Отсутствие требования к поддержанию порядка на множестве кортежей отношения дает дополнительную гибкость СУБД при хранении баз данных во внешней памяти и при выполнении запросов к базе данных.

ФУНДАМЕНТАЛЬНЫЕ СВОЙСТВА ОТНОШЕНИЙ

Отсутствие упорядоченности атрибутов

Атрибуты отношений не упорядочены (слева направо), поскольку каждый атрибут имеет уникальное имя в пределах отношения, то порядок атрибутов не имеет значения.

Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута. Это свойство теоретически позволяет, например, модифицировать схемы существующих отношений не только путем добавления новых атрибутов, но и путем удаления существующих атрибутов. Однако в большинстве существующих систем такая возможность не допускается.

ФУНДАМЕНТАЛЬНЫЕ СВОЙСТВА ОТНОШЕНИЙ

Атомарность значений атрибутов

Все значения атрибутов атомарны. Это следует из определения домена как потенциального множества значений простого типа данных, т.е. среди значений домена не могут содержаться множества значений.

Это четвертое отличие отношений от таблиц - в ячейки таблиц можно поместить что угодно - массивы, структуры и даже другие таблицы.

ФУНДАМЕНТАЛЬНЫЕ СВОЙСТВА ОТНОШЕНИЙ

Из свойств отношения следует, что не каждая таблица может задавать отношение.

Для того, чтобы некоторая таблица задавала отношение, необходимо, чтобы:

- таблица имела простую структуру (содержала бы только строки и столбцы, причем, в каждой строке было бы одинаковое количество полей);
- таблица не содержала одинаковых строк;
- любой столбец таблицы содержал данные только одного типа;
- все используемые типы данных были простыми.

РЕЛЯЦИОННАЯ МОДЕЛЬ

Достоинство модели заключается в простоте, понятности и удобстве физической реализации на ЭВМ.

Недостатком модели является отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

Примеры реляционных СУБД: IBM DB2 Universal Database, Oracle Database, Microsoft SQL Server, MariaDB, PostgreSQL, MySQL, SQLight.

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Реляционные СУБД обладают рядом особенностей, влияющих на организацию внешней памяти.

К наиболее важным особенностям можно отнести следующие:

1) регулярность структур данных. Поскольку основным объектом реляционной модели данных является плоская таблица, главный набор объектов внешней памяти имеет очень простую регулярную структуру. При этом необходимо обеспечить возможность эффективного выполнения операторов языкового уровня как над одним отношением, так и над несколькими отношениями. Для этого во внешней памяти должны поддерживаться дополнительные «управляющие» структуры – индексы;

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

К наиболее важным особенностям можно отнести следующие:

2) для корректной работы подсистемы управления данными во внешней памяти необходимо поддерживать служебную информацию в виде отношений-каталогов, содержащих следующие служебные данные:

- внутренние каталоги, описывающие физические свойства объектов базы данных;
- описатели свободной и занятой памяти в страницах отношения;
- сведения о связывание страниц отношений;

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

К наиболее важным особенностям можно отнести следующие:

3) для выполнения требования надежного хранения баз данных необходимо поддерживать избыточность хранения данных, что обычно реализуется в виде журнала изменений базы данных;

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

К наиболее важным особенностям можно отнести следующие:

4) наличие двух уровней системы: уровня непосредственного управления данными во внешней памяти (а также обычно управления буферами оперативной памяти, управления транзакциями и журнализацией изменений БД) и языкового уровня (например, уровня, реализующего язык SQL). При такой организации подсистема нижнего уровня должна поддерживать во внешней памяти набор базовых структур, конкретная интерпретация которых входит в число функций подсистемы верхнего уровня.

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Соответственно возникают следующие разновидности объектов во внешней памяти базы данных:

- строки отношений – основная часть базы данных, большей частью непосредственно видимая пользователям;
- управляющие структуры – индексы, создаваемые по инициативе пользователя или верхнего уровня системы и обычно автоматически поддерживаемые нижним уровнем системы;

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Соответственно возникают следующие разновидности объектов во внешней памяти базы данных:

- журнальная информация, поддерживаемая для удовлетворения потребности в надежном хранении данных;

- служебная информация, поддерживаемая для удовлетворения внутренних потребностей нижнего уровня системы (например, информация о свободной памяти).

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Существуют два принципиальных подхода к физическому хранению отношений:

- покортежное;
- поатрибутное.

freepik.com

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Наиболее распространенным является *покортежное хранение отношений* (единицей физического хранения является кортеж).

Это обеспечивает быстрый доступ к целому кортежу, но при этом во внешней памяти дублируются общие значения разных кортежей одного отношения и могут потребоваться лишние обмены с внешней памятью, если нужна только часть информации часть кортежа.

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Альтернативным и менее распространенным подходом является *хранение отношения по столбцам*, т.е. единицей хранения является столбец отношения с исключенными дубликатами.

При такой организации суммарно в среднем тратится меньше внешней памяти, поскольку дубликаты значений не хранятся; за один обмен с внешней памятью в общем случае считывается больше полезной информации.

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕЛЯЦИОННЫХ БД

Дополнительным преимуществом является возможность использования значений столбца отношения для оптимизации выполнения операций соединения.

Но при этом требуются существенные дополнительные действия для сборки целого кортежа (или его части).

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

Между отношениями БД могут быть установлены ***связи*** — ассоциации, показывающие, каким образом отношения соотносятся или взаимодействуют между собой.

freepik.com

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

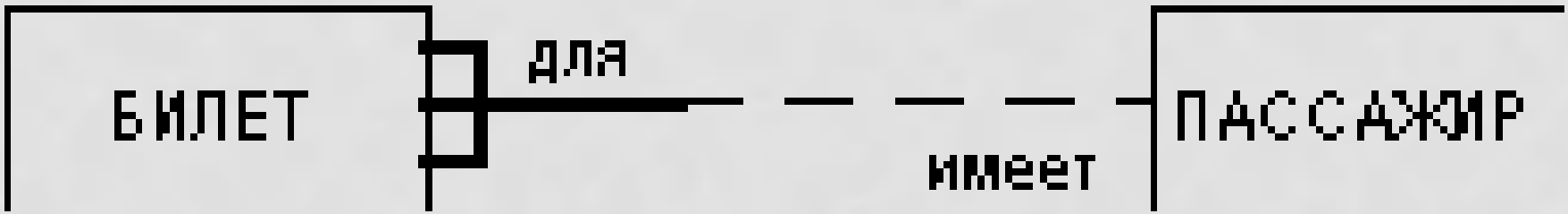
Степень связи – количество типов отношений, которые охвачены данной связью.

Между отношениями предметной области могут иметь место:

- бинарные связи (между двумя отношениями или между отношением и ем же самим - рекурсивная связь);
- тренарные связи (между тремя сущностями);
- и в общем случае - n-арные связи.

На практике чаще всего встречаются связи со степенью два, то есть бинарные связи.

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

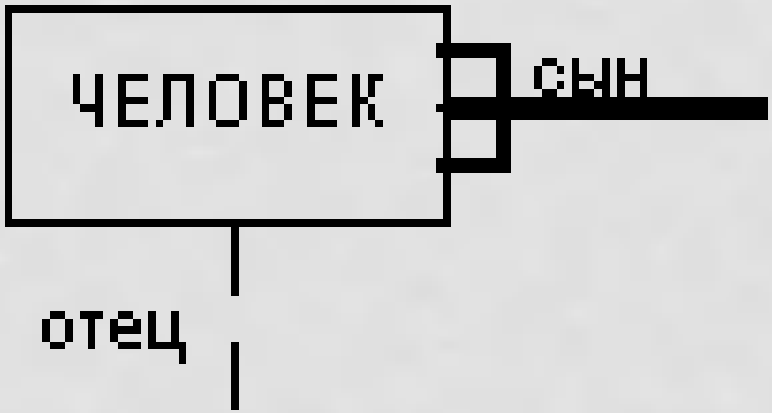


Лаконичной устной трактовкой изображенной диаграммы является следующая:

- каждый БИЛЕТ предназначен для одного и только одного ПАССАЖИРА;

- каждый ПАССАЖИР может иметь один или более БИЛЕТОВ.

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД



Рекурсивная связь – связь, в которой одни и те же отношения участвуют несколько раз в разных ролях.

Лаконичной устной трактовкой изображенной диаграммы является следующая:

- каждый ЧЕЛОВЕК является сыном одного и только одного ЧЕЛОВЕКА;

- каждый ЧЕЛОВЕК может являться отцом для одного или более ЛЮДЕЙ ("ЧЕЛОВЕКОВ").

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

Кратность связи – количество возможных экземпляров отношения некоторого типа, которые могут быть связаны с одним экземпляром отношения другого типа с помощью определенной связи.

Среди бинарных связей существуют три фундаментальных вида связи:

- один к одному (1:1);
- один ко многим (1:M);
- многие ко многим (M:N).

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

Связь ***один к одному (1:1)*** существует, когда один экземпляр одного отношения связан с единственным экземпляром другого отношения (студент может не "получать" стипендию, а может получать).

PATIENTS_REG

PT_REG_NNN	PT_REG_NUMBER
1	CS112/99-07
2	CS102/98-01
3	CS098/96-56
4	CS546/99-76
5	CS234/98-02

PATIENTS_KART

PT_NNN	PT_NAME	PT_BIRTHDATE
1	ИВАНОВ А.И.	01.02.1987
2	ПЕТРОВ А.А.	30.10.1945
3	СИДОРОВ В.Н.	03.12.1965
4	КРАСНОВ Б.Б.	10.11.1999
5	ВАСИН Н.Н.	15.05.1976

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

Связь ***один ко многим (1:M)*** существует, когда один экземпляр одного отношения связан с одним или более экземпляром другого отношения и каждый экземпляр второго отношения связан только с одним экземпляром первого отношения (квартира может пустовать, в ней может жить один или несколько жильцов).

CLINICS

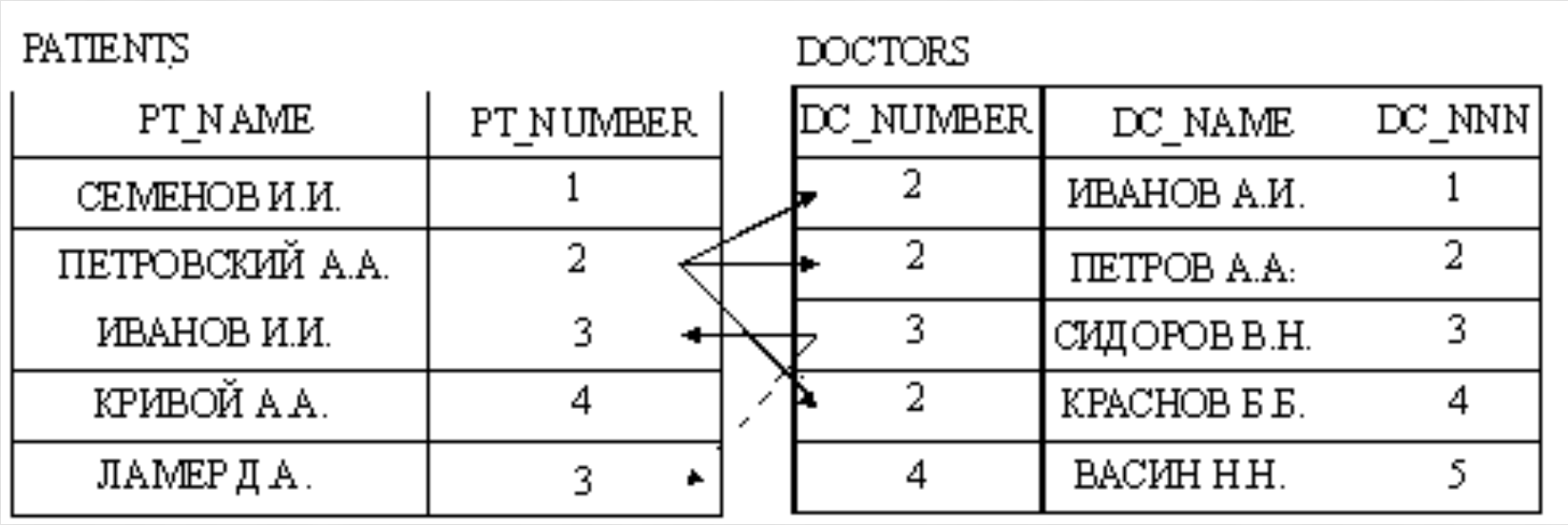
CS_NAME	CS_NNN
ПОЛИКЛИНИКА №2	1
ПОЛИКЛИНИКА №123	2
ХИРУРГИЧЕСКИЙ ЦЕНТР	3
ДИСПАНСЕР №12	4
ПОЛИКЛИНИКА №143	5

DOCTORS

DC_CS_NNN	DC_NAME	DC_NNN
2	ИВАНОВ А.И.	1
2	ПЕТРОВ А.А.	2
3	СИДОРОВ В.Н.	3
2	КРАСНОВ Б.Б.	4
3	ВАСИН Н.Н.	5

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

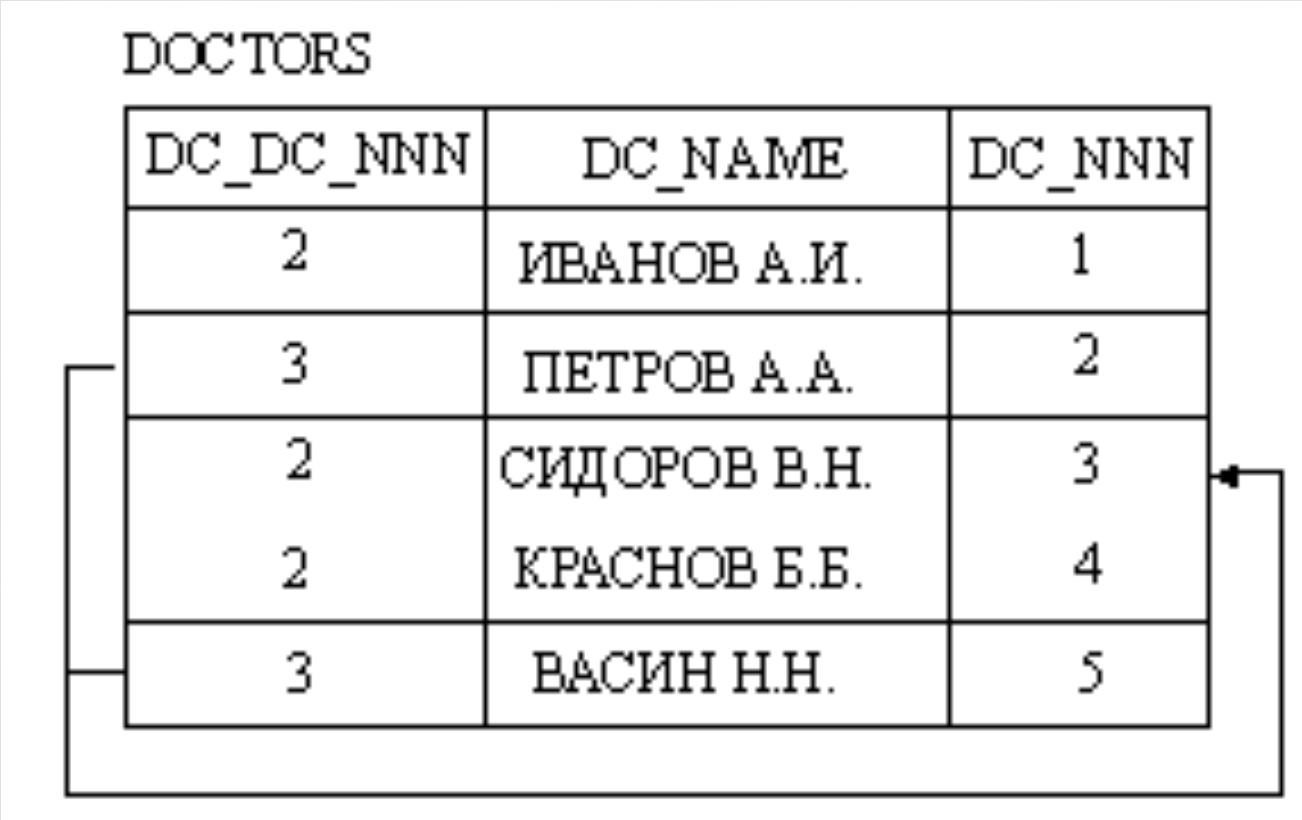
Связь ***многие ко многим (M:N)*** существует, когда один экземпляр одного отношения связан с одним или более экземпляром другого отношения и каждый экземпляр второго отношения связан с одним или более экземпляром первого отношения.



СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

Как можно реализовать связь «многие-ко-многим» в реляционной базе данных?

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД



DOCTORS

DC_DC_NNN	DC_NAME	DC_NNN
2	ИВАНОВ А.И.	1
3	ПЕТРОВ А.А.	2
2	СИДОРОВ В.Н.	3
2	КРАСНОВ Б.Б.	4
3	ВАСИН Н.Н.	5

Пример рекурсивного отношения

СВЯЗИ МЕЖДУ ОТНОШЕНИЯМИ РЕЛЯЦИОННЫХ БД

Связь любого из рассматриваемых выше типов может быть:

- *обязательной*, если в данной связи должен участвовать каждый экземпляр отношения;
- *необязательной* — если не каждый экземпляр отношения должен участвовать в данной связи.

При этом связь может быть обязательной с одной стороны и необязательной с другой стороны.

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Индексы - это специальные структуры в базах данных, которые позволяют ускорить поиск отдельных записей и их сортировку по определенному полю или набору полей в таблице, а также используются для обеспечения уникальности данных.

Основное назначение индексов состоит в обеспечении прямого доступа к кортежу отношения по ключу.

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Общей идеей организации индексов является хранение упорядоченного списка значений ключа с привязкой к каждому значению ключа списка идентификаторов кортежей отношения.

Индекс базы данных упорядочен, и каждый элемент индекса содержит название искомого объекта, а также один или несколько указателей (идентификаторов записей) на место его расположения.

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

ID кортежа	Отношение «Сотрудники»	
	Таб.номер	ФИО
1	100200	Иванов
2	100015	Петров
3	100307	Сидоров
4	100001	Комаров
5	100308	Зимин
6	100202	Шишкин

Индекс по «ФИО»	
Зимин	5
Иванов	1
Комаров	4
Петров	2
Сидоров	3
Шишкин	6
Индекс по «Таб.номер»	
100001	4
100015	2
100200	1
100202	6
100307	3
100308	5

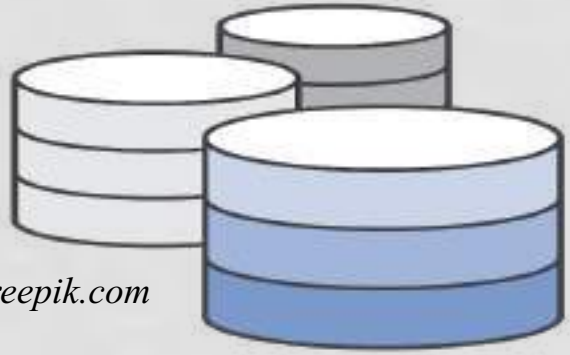
ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

ID кортежа	Отношение «Начисления»		
	Таб.номер	Месяц	Сумма
1	100	1	100
2	101	1	200
3	102	1	300
4	101	2	200
5	103	2	300
6	100	4	100
. . .	. . .	. . .	. . .
N	101	12	200

Индекс по «Таб.номер»	
100	1, 6
101	2,4,N
102	3
103	5

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

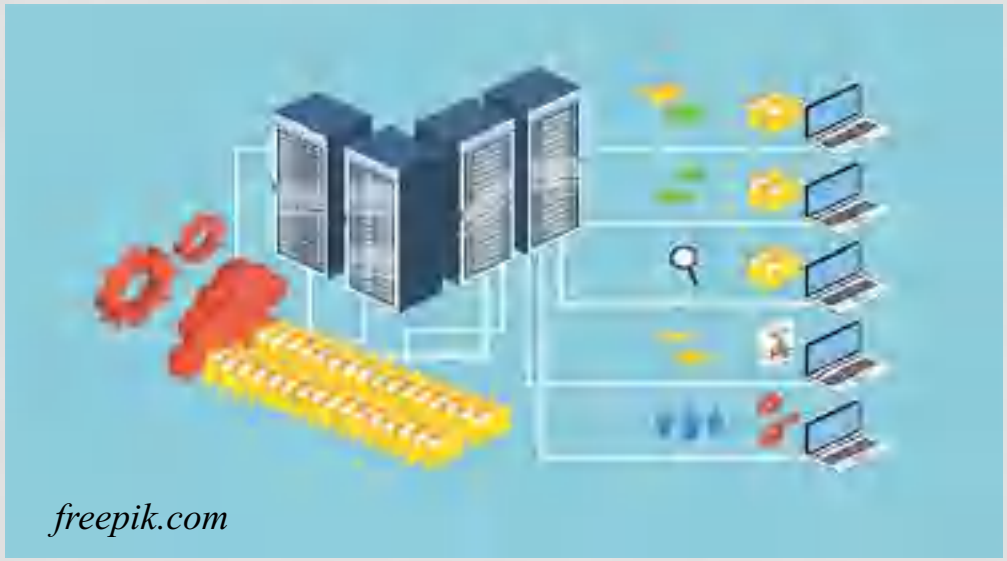
Поскольку при выполнении многих операций языкового уровня требуется сортировка отношений в соответствии со значениями некоторых атрибутов, то полезным свойством индекса является обеспечение последовательного просмотра кортежей отношения в диапазоне значений ключа в порядке возрастания или убывания ключей.



freepik.com

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Индексы поддерживаются *динамически*, т.е. после обновления БД – добавлении или удалении записей, а также при модификации полей записи, входящих в ключ, – индекс приводится в соответствие с обновленной версией БД.



ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Обращение к записи через индексы осуществляется в два этапа: сначала в индексной структуре находится требуемое значение атрибута и соответствующий адрес записи, затем по этому адресу происходит обращение к внешнему запоминающему устройству (ВЗУ).

Индекс загружается в ОП целиком (или хранится в ней постоянно во время работы с БД).

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Типы индексов:

- ***первичный индекс.*** Файл данных упорядочивается по полю ключа упорядочения, а на основе поля ключа упорядочения создается поле индексации, которое гарантированно имеет уникальное значение в каждой записи;
- ***вторичный индекс***. Файл данных последовательно упорядочивается по неключевому полю и на основе этого неключевого поля формируется поле индексации, поэтому в файле может быть несколько записей, соответствующих значению этого поля индексации;

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Типы индексов:

- если в файле индексации содержится только одна запись, соответствующих каждому значению индексируемого поля, то такой индекс называют ***уникальным***.

Для каждой таблицы БД можно одновременно поддерживать один первичный и несколько вторичных индексов, что также относится к достоинствам индексирования.

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Различают ***одиночные*** индексы и ***составные***. Составной индекс включает два или более столбца одной таблицы.

freepik.com

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Существует два типа индексов:
- кластеризованные индексы;
- некластеризованные индексы.

freepik.com

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Кластеризованный индекс хранит в своих узлах-листьях реальные строки данных. Поэтому данные становятся доступны, как только найден определенный узел-лист, что может сокращать количество операций ввода-вывода.

Еще одним преимуществом кластеризованных индексов является то, что считываемые данные получаются в отсортированном по индексу виде.

Недостатком использования кластеризованного индекса является то, что доступ к таблице всегда происходит через индекс, что может приводить к дополнительной нагрузке на систему.

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Некластеризованный индекс является вспомогательной структурой и не содержит реальных данных таблицы в своих узлах-листьях.

В некластеризованном индексе узел-лист содержит значение ключа, а также идентификатор строки (Row ID), указывающий нужную строку в таблице. Это значение обеспечивает быстрый доступ к реальным данным, указывая точное местоположение этих данных. На практике используется несколько некластеризованных индексов по различным колонкам таблицам.

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

Некоторые общие принципы, связанные с созданием индексов:

- индексы необходимо создавать для столбцов по которым часто производится поиск и операции сортировки;

- индекс обязательно в автоматическом режиме создается для столбцов, на которые наложено ограничение первичного ключа и уникальности;

- индексы можно создавать не только для таблиц, но и для представлений;

ИНДЕКСЫ В РЕЛЯЦИОННЫХ БД

- лучше всего индексы создавать для тех полей, в которых - минимальное число повторяющихся значений и данные распределены равномерно;

- если поиск постоянно производится по определенному набору столбцов (одновременно), то в этом случае, возможно, есть смысл создать композитный (составной) индекс;

- при внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Пример 1 нарушения целостности базы данных.

Таблица 1 DEPART		
Dept_Id	Dept_Name	Dept_Kol
1	Кафедра алгебры	3
2	Кафедра программирования	2

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Пример 1 нарушения целостности базы данных.

Таблица 2 PERSON		
Pers_Id	Pers_Name	Dept_Id
1	Иванов	1
2	Петров	2
3	Сидоров	1
4	Пушников	2
5	Шарипов	1

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Ограничение целостности этой базы данных состоит в том, что поле Dept_Kol не может заполняться произвольными значениями - это поле должно содержать количество сотрудников, реально числящихся в подразделении.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

С учетом этого ограничения можно заключить, что вставка нового сотрудника в таблицу не может быть выполнена одной операцией.

При вставке нового сотрудника необходимо одновременно увеличить значение поля Dept_Kol.

Шаг 1. Вставить сотрудника в таблицу PERSON: INSERT INTO PERSON (6, Мифтахов, 1) .

Шаг 2. Увеличить значение поля Dept_Kol: UPDATE DEPART SET Dept=Dept+1 WHERE Dept_Id=1 состоянии.

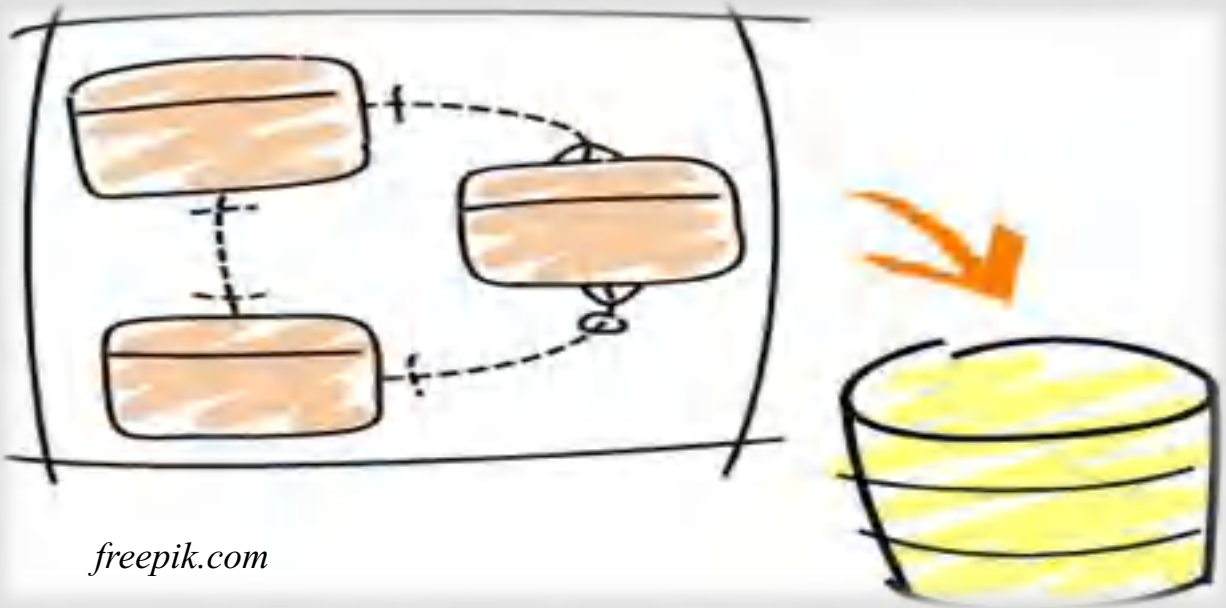
ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Если после выполнения первой операции и до выполнения второй произойдет сбой системы, то реально будет выполнена только первая операция и база данных остается в нецелостном состоянии.

freepik.com

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Ограничение целостности - это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.



ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Примерами ограничений целостности могут служить следующие утверждения.

Пример 2. Возраст сотрудника не может быть меньше 18 и больше 65 лет.

Пример 3. Каждый сотрудник имеет уникальный табельный номер.

Пример 4. Сотрудник обязан числиться в одном отделе.

Пример 5. Сумма накладной обязана равняться сумме произведений цен товаров на количество товаров для всех товаров, входящих в накладную.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

База данных находится в *согласованном* (целостном) состоянии, если выполнены (удовлетворены) все ограничения целостности, определенные для базы данных.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Важно подчеркнуть, что должны быть выполнены не все вообще ограничения предметной области, а только те, которые определены в базе данных. Для этого необходимо, чтобы СУБД обладала развитыми средствами поддержки ограничений целостности.

Если какая-либо СУБД не может отобразить все необходимые ограничения предметной области, то такая база данных хотя и будет находиться в целостном состоянии с точки зрения СУБД, но это состояние не будет правильным с точки зрения пользователя.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Вместе с понятием целостности базы данных возникает понятие *реакции системы на попытку нарушения целостности*. Система должна не только проверять, не нарушаются ли ограничения в ходе выполнения различных операций, но и должным образом реагировать, если операция приводит к нарушению целостности. Имеется два типа реакции на попытку нарушения целостности:

- отказ выполнить "незаконную" операцию;
- выполнение компенсирующих действий.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Например, если система знает, что в поле "Возраст_Сотрудника" должны быть целые числа в диапазоне от 18 до 68, то система отвергает попытку ввести значение возраста 70. При этом может генерироваться какое-нибудь сообщение для пользователя.

В противоположность этому, в примере 1 система допускает вставку записи о новом сотруднике (что приводит к нарушению целостности базы данных), но *автоматически* производит компенсирующие действия, изменяя значение поля Dept_Kol в таблице DEPART.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

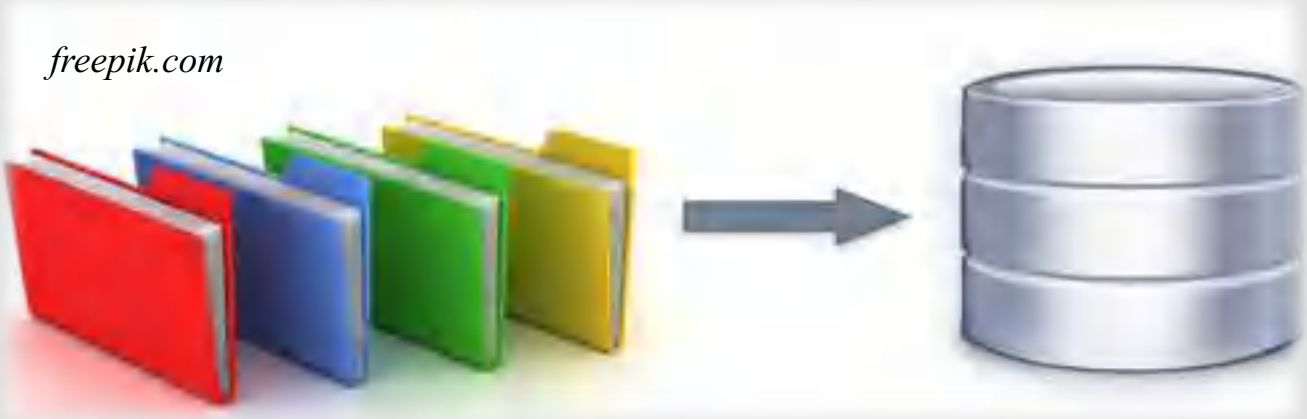
В некоторых случаях система может не выполнять проверку на нарушение ограничений, а сразу выполнять компенсирующие операции.

Действительно, в примере 1 при вставке или удалении сотрудника проверку производить *не нужно*, т.к. результаты ее известны заранее - ограничение обязательно будет нарушено. В этом случае необходимо сразу приступать к компенсированию возникшего нарушения.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Ограничения целостности можно классифицировать несколькими способами:

- по области действия;
- по времени проверки;
- по способам реализации.



КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

По области действия ограничения делятся на:
- ограничения домена;
- ограничения атрибута;
- ограничения кортежа;
- ограничения отношения (целостность сущностей);
- ограничения базы данных.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ ДОМЕНА

Ограничения целостности домена представляют собой ограничения, накладываемые только на допустимые значения домена. Фактически, ограничения домена обязаны являться частью определения домена.

Например, ограничением домена "Возраст сотрудника" может быть условие "Возраст сотрудника не менее 18 и не более 68".

Ограничения домена сами по себе не проверяются. Если на каком-либо домене основан атрибут, то ограничение соответствующего домена становится ограничением этого атрибута.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ АТРИБУТА

Ограничение целостности *атрибута* представляют собой ограничения, накладываемые на допустимые значения атрибута вследствие того, что атрибут основан на каком-либо домене.

Ограничение атрибута в точности совпадают с ограничениями соответствующего домена.

Отличие ограничений атрибута от ограничений домена в том, что ограничения атрибута проверяются

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ АТРИБУТА

Пример. Создадим таблицу Заказчиков таким способом, чтобы вводимые значения для Код_заказчика и Код_продавца ограничивались бы целыми положительными цифрами:

```
CREATE TABLE Заказчики
(Код_заказчика integer,
 Наим_заказчика  char (10),
 Город           char (10),
 Код_продавца integer);
```

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ АТРИБУТА

Пример. Создадим таблицу Заказчиков таким способом, чтобы вводимые значения для Код_заказчика и Код_продавца ограничивались бы целыми положительными цифрами:

```
CREATE TABLE Заказчики
(Код_заказчика integer CHECK (Код_заказчика >=0),
 Наим_заказчика  char (10) NOT NULL,
 Город           char (10),
 Код_продавца integer CHECK (Код_продавца >=0));
```

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ КОРТЕЖА

Ограничения целостности кортежа представляют собой ограничения, накладываемые на допустимые значения отдельного кортежа. Требование, что ограничение относится к отдельному кортежу отношения, означает, что для его проверки не требуется никакой информации о других кортежах отношения.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ КОРТЕЖА

Пример. Атрибут "Возраст сотрудника" в таблице "Спецподразделение", может иметь дополнительное ограничение "Возраст сотрудника не менее 25 и не более 45", помимо того, что этот атрибут уже имеет ограничение, определяемое доменом - "Возраст сотрудника не менее 18 и не более 68".

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ ОТНОШЕНИЯ

Ограничения целостности отношения представляют ограничения, накладываемые только на допустимые значения отдельного отношения. Требование, что ограничение относится к отдельному отношению, означает, что для его проверки не требуется информации о других отношениях.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ ОТНОШЕНИЯ

Пример. Предположим, что в отношении PERSON задано следующее ограничение - в каждом отделе должно быть не менее двух сотрудников. Это ограничение можно сформулировать так - количество строк с одинаковым значением Dept_Id должно быть не меньше 2.

Для того чтобы ввести в действие (объявить) это ограничение, необходимо, чтобы в отношение уже были вставлены некоторые кортежи.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ ОТНОШЕНИЯ

Еще одним примером ограничения *отношения* является *требование целостности сущностей*, так как для его проверки необходимо иметь информацию обо всех кортежах отношения.

Оно состоит в том, что любой кортеж любого отношения отличим от любого другого кортежа этого отношения, т.е. другими словами, любое отношение должно обладать первичным ключом.

Для соблюдения целостности сущности достаточно гарантировать отсутствие в любом отношении кортежей с одним и тем же значением первичного ключа.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ ОТНОШЕНИЯ

Пример. Определение первичного и уникального ключа в таблице Сотрудники:

```
CREATE TABLE Сотрудники
(Код          SMALLINT NOT NULL,
ФИО           CHAR (70) NOT NULL,
Адрес         CHAR  (15) NOT NULL,
Телефон CHAR (10),
Город         CHAR (10),
Дата рождения DATE NOT NULL,
ИИН           CHAR  (12) NOT NULL);
```

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ ОТНОШЕНИЯ

Пример. Определение первичного и уникального ключа в таблице Сотрудники:

```
CREATE TABLE Сотрудники
(Код          SMALLINT NOT NULL PRIMARY KEY,
ФИО           CHAR (70) NOT NULL,
Адрес         CHAR  (15) NOT NULL,
Телефон CHAR (10),
Город         CHAR (10),
Дата рождения DATE NOT NULL,
ИИН           CHAR  (12) NOT NULL UNIQUE);
```

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Ограничения целостности *базы данных* представляют ограничения, накладываемые на значения двух или более связанных между собой отношений (в том числе отношение может быть связано само с собой).

Ограничение целостности ссылок, задаваемое внешним ключом отношения, является ограничением базы данных.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Вернемся к примеру 1.

Атрибут Dept_Id появляется в отношении PERSON (Сотрудники) не потому, что номер отдела является собственным свойством сотрудника, а лишь для того, чтобы иметь возможность восстановить при необходимости полную сущность DEPART (Отделы).

Значение атрибута Dept_Id в любом кортеже отношения PERSON (Сотрудники) должно соответствовать значению атрибута Dept_Id в одном из кортежей отношения DEPART (Отделы).

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Атрибут такого рода называется *внешним ключом*, поскольку его значения однозначно характеризуют сущности, представленные кортежами некоторого другого отношения (т.е. задают значения их первичного ключа). Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Требование *целостности по ссылкам*, или требование внешнего ключа состоит в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать). Для нашего примера это означает, что если для сотрудника указан номер отдела, то этот отдел должен существовать.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

При обновлении ссылающегося отношения (вставке новых кортежей или модификации значения внешнего ключа в существующих кортежах) достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

При удалении кортежа из отношения, на которое ведет ссылка существуют три подхода, каждый из которых поддерживает целостность по ссылкам.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Первый подход заключается в том, что запрещается производить удаление кортежа, на который существуют ссылки (т.е. сначала нужно либо удалить ссылающиеся кортежи, либо соответствующим образом изменить значения их внешнего ключа).

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

При *втором подходе* при удалении кортежа, на который имеются ссылки, во всех ссылающихся кортежах значение внешнего ключа автоматически становится неопределенным или устанавливается в значение по умолчанию.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Третий подход (каскадное удаление) состоит в том, что при удалении кортежа из отношения, на которое ведет ссылка, из ссылающегося отношения автоматически удаляются все ссылающиеся кортежи.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Ограничения целостности по ссылкам должна поддерживаться СУБД.

В развитых реляционных СУБД обычно можно выбрать способ поддержания целостности по ссылкам для каждой отдельной ситуации определения внешнего ключа.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ. ОГРАНИЧЕНИЯ БАЗЫ ДАННЫХ

Пример объявления ограничений целостности по ссылкам:

CREATE TABLE Телефонный_справочник
(Номер_телефона integer NOT NULL PRIMARY KEY,
ФИО char(20),
Код_улицы integer REFERENCES Справ_улиц,
ON UPDATE Справочник_улиц CASCADES,
ON DELETE Справочник_улиц RESTRICTED);

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

По времени проверки ограничения делятся на:
- немедленно проверяемые ограничения;
- ограничения с отложенной проверкой.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Немедленно проверяемые ограничения проверяются непосредственно в момент выполнения операции, могущей нарушить ограничение. Если ограничение нарушается, то такая операция отвергается. Транзакция, внутри которой произошло нарушение немедленно проверяемого утверждения целостности, обычно откатывается.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Ограничение домена, атрибута и кортежа являются *немедленно проверяемыми* ограничениями, так как не зависит ни от каких других объектов базы данных, кроме домена, на котором основан атрибут. Поэтому никакие изменения в других объектах не могут повлиять на истинность ограничения.

Ограничение отношения может быть как *немедленно проверяемым* ограничением, так и ограничением *с отложенной проверкой*.

Требование целостности сущностей является *немедленно проверяемым* ограничением.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Ограничения с отложенной проверкой проверяется в момент фиксации всей транзакции оператором COMMIT WORK.

Внутри транзакции ограничение может не выполняться.

Если в момент фиксации транзакции обнаруживается нарушение, то транзакция откатывается.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Примером ограничения, которое не может быть проверено немедленно является ограничение из примера 1. Это происходит оттого, что транзакция, заключающаяся во вставке нового сотрудника в таблицу PERSON, состоит не менее чем из двух операций - вставки строки в таблицу PERSON и обновления строки в таблице DEPART. Ограничение, безусловно, неверно после первой операции и становится верным после второй операции.

Ограничение базы данных (внешнего ключа) может быть ограничением *с отложенной проверкой*.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

По способам реализации ограничений целостности различают:

- декларативную поддержку ограничений целостности;
- процедурную поддержку ограничений целостности.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Декларативная поддержка ограничений целостности заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности (если они имеются в СУБД) определяют ограничения на значения доменов и атрибутов, целостность сущностей и ссылочную целостность. Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Например, следующий оператор создает таблицу PERSON и определяет для нее некоторые ограничения целостности:

```
CREATE TABLE PERSON
(Pers_Id INTEGER PRIMARY KEY,
Pers_Name CHAR(30) NOT NULL,
Dept_Id REFERENCES DEPART(Dept_Id) ON
UPDATE CASCADE ON DELETE CASCADE);
```

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

После выполнения оператора для таблицы PERSON будут объявлены следующие ограничения целостности:

- поле Pers_Id образует потенциальный ключ отношения;
- поле Pers_Name не может содержать null-значений;
- поле Dept_Id является внешней ссылкой на родительскую таблицу DEPART, причем, при изменении или удалении строки в родительской таблице каскадно должны быть внесены соответствующие изменения в дочернюю таблицу.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Процедурная поддержка ограничений целостности заключается в использовании триггеров и хранимых процедур.

Примером такого ограничения может служить требование из примера 1, утверждающее, что поле Dept_Kol таблицы DEPART должно содержать количество сотрудников, реально числящихся в подразделении.

freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Для реализации этого ограничения необходимо создать триггер, запускающийся при вставке, модификации и удалении записей в таблице PERSON, который корректно изменяет значение поля Dept_Kol.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Например, при вставке в таблицу PERSON новой строки, триггер увеличивает на единицу значение поля Dept_Kol, а при удалении строки - уменьшает. Заметим, что при модификации записей в таблице PERSON могут потребоваться даже более сложные действия.

Действительно, модификация записи в таблице PERSON может заключаться в том, что мы переводим сотрудника из одного отдела в другой, меняя значение в поле Dept_Id. При этом необходимо в старом подразделении уменьшить количество сотрудников, а в новом - увеличить.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Кроме того, необходимо защититься от неправильной модификации строк таблицы DEPART. Действительно, пользователь может попытаться модифицировать запись об отделе, введя неверное значение поля Dept_Kol.

Для предотвращения подобных действий необходимо создать также триггеры, запускающиеся при вставке и модификации записей в таблице DEPART.

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Наличие ограничения целостности (как декларативного, так и процедурного характера) *всегда* приводит к созданию или использованию некоторого программного кода, реализующего это ограничение. Разница заключается в том, где такой код хранится и как он создается.

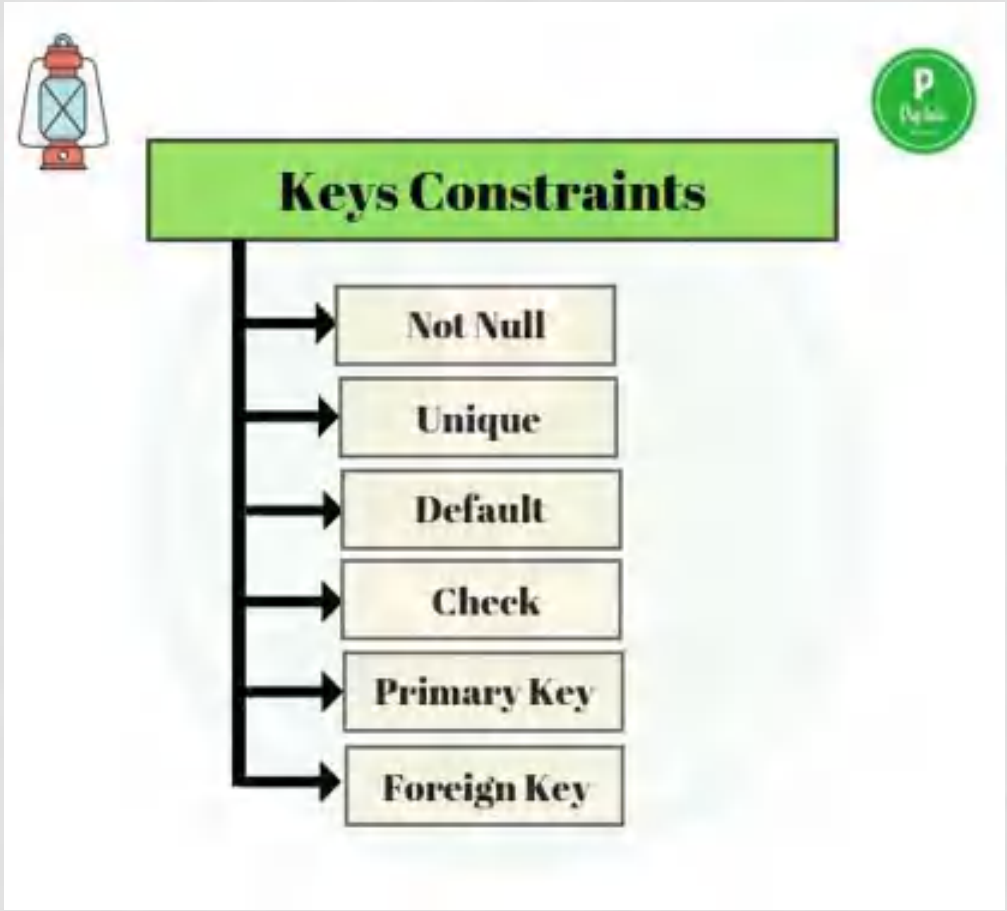
freepik.com

КЛАССИФИКАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Если ограничение целостности реализовано в виде триггеров, то этот программный код является просто телом триггера.

Если используется декларативное ограничение целостности, то текст ограничения хранится в виде некоторого объекта СУБД, а для реализации ограничения СУБД автоматически сама генерирует триггеры, выполняющие необходимые действия по проверке ограничений.

ПРИМЕР ЗАДАНИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ НА ЯЗЫКЕ SQL



https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcQAMILDz6W5KSBYi_PZIpGMooXaJKT6n_Ur7a7L4fZCJ5K6I3l7

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

В манипуляционной составляющей реляционной модели определяются два базовых механизма манипулирования реляционными данными - основанная на теории множеств ***реляционная алгебра*** и базирующееся на математической логике ***реляционное исчисление***.

Все эти механизмы обладают одним свойством: они замкнуты относительно понятия отношения. Это означает, что выражения реляционной алгебры и формулы реляционного исчисления определяются над отношениями реляционных БД и результатом вычисления также являются отношения.

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

Механизмы реляционной алгебры и реляционного исчисления ***эквивалентны***, т.е. для любого допустимого выражения реляционной алгебры можно построить эквивалентную (т.е. производящую такой же результат) формулу реляционного исчисления и наоборот.

freepik.com

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

Реляционная алгебра и исчисление *различаются уровнем процедурности*:

- запрос, представленный на языке реляционной алгебры, может быть вычислен на основе вычисления элементарных алгебраических операций с учетом их старшинства и возможных скобок;

- формула реляционного исчисления устанавливает условия, которым должны удовлетворять кортежи результирующего отношения. Поэтому языки реляционного исчисления являются более непроцедурными или декларативными.

РЕЛЯЦИОННАЯ АЛГЕБРА

Основная идея реляционной алгебры состоит в том, что коль скоро отношения БД являются множествами, то средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для баз данных.

РЕЛЯЦИОННАЯ АЛГЕБРА

В состав теоретико-множественных операций входят операции:

- объединения отношений;
- пересечения отношений;
- взятия разности отношений;
- прямого произведения отношений.

РЕЛЯЦИОННАЯ АЛГЕБРА

Специальные реляционные операции включают:

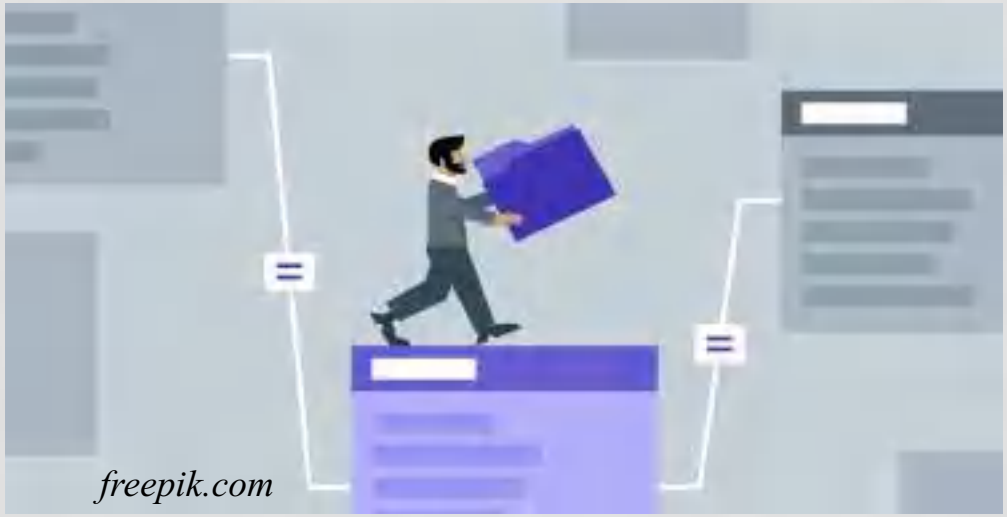
- ограничение отношения;
- проекцию отношения;
- соединение отношений;
- деление отношений.

РЕЛЯЦИОННАЯ АЛГЕБРА

Кроме того, в состав алгебры включается *операция присваивания*, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и *операция переименования* атрибутов, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

РЕЛЯЦИОННАЯ АЛГЕБРА

Каждое отношение обязано иметь *уникальное имя* в пределах базы данных. Имя отношения, полученного в результате выполнения реляционной операции, определяется в левой части равенства.



РЕЛЯЦИОННАЯ АЛГЕБРА

Можно не требовать наличия имен от отношений, полученных в результате реляционных выражений, если эти отношения подставляются в качестве аргументов в другие реляционные выражения. Такие отношения будем называть *неименованными отношениями*.

Неименованные отношения реально не существуют в базе данных, а только вычисляются в момент вычисления значения реляционного оператора.

РЕЛЯЦИОННАЯ АЛГЕБРА

Отношения будем называть *совместимыми по типу*, если они имеют идентичные заголовки, а именно:

- отношения имеют *одно и то же множество имен атрибутов*, т.е. для любого атрибута в одном отношении найдется атрибут с таким же наименованием в другом отношении;

- атрибуты с одинаковыми именами *определены на одних и тех же доменах*.

РЕЛЯЦИОННАЯ АЛГЕБРА

Некоторые отношения не являются совместимыми по типу, но становятся таковыми после некоторого переименования атрибутов. Для того чтобы такие отношения можно было использовать в реляционных операторах, вводится вспомогательный *оператор переименования атрибутов*.

Операция переименования производит отношение, тело которого совпадает с телом операнда, но имена атрибутов изменены.

Операция присваивания позволяет сохранить результат вычисления реляционного выражения в существующем отношении БД.

ОБЪЕДИНЕНИЕ ОТНОШЕНИЙ

Объединением двух совместимых по типу отношений *A* и *B* называется отношение с тем же заголовком, что и у отношений *A* и *B*, и телом, состоящим из строк (кортежей), принадлежащих или *A*, или *B*, или обоим отношениям.

Синтаксис операции объединения: ***A* UNION *B***.

Замечание. Объединение, как и любое отношение, не может содержать одинаковых кортежей. Поэтому если некоторый кортеж входит и в отношение *A*, и отношение *B*, то в объединение он входит только один раз.

ОБЪЕДИНЕНИЕ ОТНОШЕНИЙ

Таблица 1. Отношение A

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Таблица 2. Отношение B

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Пушников	2500
4	Сидоров	3000

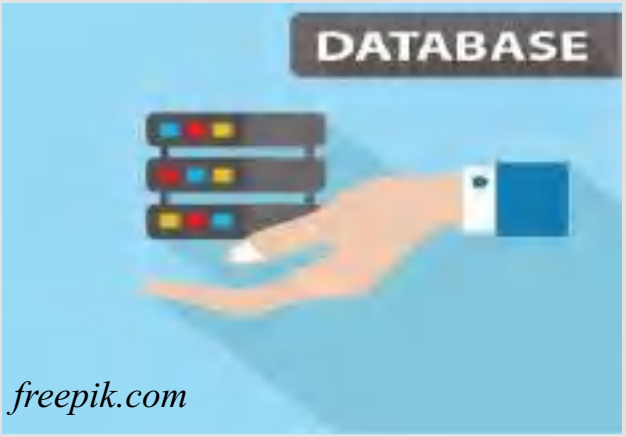
ОБЪЕДИНЕНИЕ ОТНОШЕНИЙ

Таблица 3. Отношение A UNION B		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000
2	Пушников	2500
4	Сидоров	3000

ОБЪЕДИНЕНИЕ ОТНОШЕНИЙ

Замечание

Никакие реляционные операторы не передают результирующему отношению никаких данных о потенциальных ключах. Причина заключается в том, что потенциальный ключ - семантическое понятие, отражающее различимость объектов предметной области.



ПЕРЕСЕЧЕНИЕ ОТНОШЕНИЙ

Пересечением двух совместимых по типу отношений *A* и *B* называется отношение с тем же заголовком, что и у отношений *A* и *B*, и телом, состоящим из строк (кортежей), принадлежащих одновременно обоим отношениям *A* и *B*.

Синтаксис операции пересечения:

A* INSERSECT *B

ПЕРЕСЕЧЕНИЕ ОТНОШЕНИЙ

Таблица 4. Отношение A INTERSECT B		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000

ВЫЧИТАНИЕ ОТНОШЕНИЙ

Вычитанием двух совместимых по типу отношений *A* и *B* называется отношение с тем же заголовком, что и у отношений *A* и *B*, и телом, состоящим из строк (кортежей), принадлежащих отношению *A* и не принадлежащих отношению *B*.

Синтаксис операции вычитания: ***A*** **MINUS** ***B***

freepik.com

ВЫЧИТАНИЕ ОТНОШЕНИЙ

Таблица 5. Отношение A MINUS B		
Табельный номер	**Фамилия**	**Зарплата**
2	Петров	2000
3	Сидоров	3000

ДЕКАРТОВО ПРОИЗВЕДЕНИЕ

Декартовым произведением двух отношений *A(A1, A2,...An)* и *B(B1, B2,...Bn)* называется отношение, заголовок которого является *сцеплением заголовков* отношений *A* и *B*: (*A1,A2,...,An,B1,B2,...Bn*) , а тело состоит из строк (кортежей), являющихся *сцеплением кортежей* отношений *A* и *B*: *(a1,a2,...,an,b1,b2,...,bn)*, таких, что $(a1,a2,...,an) \in A$, $(b1,b2,...,bn) \in B$.

Синтаксис операции декартового произведения: ***A* TIMES *B*.**

ДЕКАРТОВО ПРОИЗВЕДЕНИЕ

Мощность произведения *A* TIMES *B* равна произведению мощностей отношений *A* и *B*, т.к. каждая строка отношения *A* соединяется с каждой строкой отношения *B*.

Если в отношения *A* и *B* имеются атрибуты с одинаковыми наименованиями, то перед выполнением операции декартового произведения такие атрибуты необходимо переименовать.

Перемножать можно любые два отношения, совместимость по типу при этом не требуется.

ДЕКАРТОВО ПРОИЗВЕДЕНИЕ

Таблица 6. Отношение A (Поставщики)

Номер поставщика	**Наименование поставщика**
1	Завод 1
2	Завод 2
3	Завод 3

Таблица 7. Отношение B (Детали)

Номер детали	**Наименование детали**
1	Болт
2	Гайка
3	Винт

ДЕКАРТОВО ПРОИЗВЕДЕНИЕ

Таблица 8. Отношение A TIMES B

Номер поставщика	Наименование поставщика	Номер детали	Наименование детали
1	Завод 1	1	Болт
1	Завод 1	2	Гайка
1	Завод 1	3	Винт
2	Завод 2	1	Болт
2	Завод 2	2	Гайка
2	Завод 2	3	Винт
3	Завод 3	1	Болт
3	Завод 3	2	Гайка
3	Завод_3	3	Винт

ВЫБОРКА (ОГРАНИЧЕНИЕ, СЕЛЕКЦИЯ)

Выборкой (ограничением, селекцией) на отношении *A* с условием z называется отношение с тем же заголовком, что и у отношения *A*, и телом, состоящем из строк (кортежей), значения атрибутов которых при подстановке в условие *z* дают значение ИСТИНА.

«Z» представляет собой логическое выражение, в которое могут входить атрибуты отношения *A* и (или) скалярные выражения.

ВЫБОРКА (ОГРАНИЧЕНИЕ, СЕЛЕКЦИЯ)

В простейшем случае условие *z* имеет вид *XΘY*, где Θ - один из операторов сравнения (=, ≠, <, >, ≤, ≥), а *X* и *Y* - атрибуты отношения *A* или скалярные значения. Такие выборки называются *Θ-выборки* (*тэта-выборки*) или *Θ-ограничения*, *Θ-селекции*.

Синтаксис операции выборки:

A* WHERE z** или ***A* WHERE *XΘY

freepik.com

ВЫБОРКА (ОГРАНИЧЕНИЕ, СЕЛЕКЦИЯ)

Таблица 9. Отношение A

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Результат выборки *A* WHERE *Зарплата <3000* будет иметь вид:

Таблица 10. Отношение A WHERE Зарплата<3000

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000

ПРОЕКЦИЯ ОТНОШЕНИЯ

Проекцией отношения *A* по атрибутам *X,Y,...,Z*, где каждый из атрибутов принадлежит отношению *A*, называется отношение с заголовком *(X,Y,...,Z)* и телом, содержащим множество строк (кортежей) вида *(x,y,. . .,z)*, таких, для которых в отношении *A* найдутся строки (кортежи) со значением атрибута *X* равным *x*, значением атрибута *Y* равным *y*, …, значением атрибута *Z* равным *z*.

Синтаксис операции проекции: ***A[X,Y,...,Z]*.**

ПРОЕКЦИЯ ОТНОШЕНИЯ

Таблица 11. Отношение A (Поставщики)

Номер поставщика	Наименование поставщика	Город поставщика
1	Завод_1	Караганда
2	Завод_2	Алматы
3	Завод_3	Алматы
4	Завод_4	Нур-Султан

Таблица 12. Отношение A[Город поставщика]

Город поставщика
Караганда
Алматы
Нур-Султан

ПРОЕКЦИЯ ОТНОШЕНИЯ

Операция проекции дает "*вертикальный срез*" отношения, в котором удалены все возникшие при таком срезе дубликаты кортежей.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Соединением отношений *A* и *B* по условию z называется отношение ***(A* TIMES *B)* WHERE z**, *где z* представляет собой логическое выражение, в которое могут входить атрибуты отношений *A* и *B* и (или) скалярные выражения.

Таким образом, операция соединения есть результат последовательного применения операций декартового произведения и выборки. Если в отношениях *A* и *B* имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать, чтобы сделать их различными.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Пример. Пусть даны два отношения *A* и *B* с информацией о поставщиках и деталях:

Таблица 13 Отношение A (Поставщики)

Номер поставщика	**Наименование поставщика**
1	Иванов
2	Петров
3	Сидоров

Таблица 14. Отношение B (Детали)

Номер детали	**Наименование детали**
1	Болт
2	Гайка
3	Винт

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Декартово произведение отношений *A* и *B* будет иметь вид:

Таблица 15. Отношение A TIMES B

Номер поставщика	**Наименование поставщика**	**Номер детали**	**Наименование детали**
1	Иванов	1	Болт
1	Иванов	2	Гайка
1	Иванов	3	Винт
2	Петров	1	Болт
2	Петров	2	Гайка
2	Петров	3	Винт
3	Сидоров	1	Болт
3	Сидоров	2	Гайка
3	Сидоров	3	Винт

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Если к нему приложить условие «Какие поставщики поставляют болты», то есть «Какие поставщики поставляют детали с номером 1», то в результате получим отношение.

Таблица 16. Отношение A TIMES B WHERE Номер детали = 1

Номер поставщика	**Наименование поставщика**	**Номер детали**	**Наименование детали**
1	Иванов	1	Болт
2	Петров	1	Болт
3	Сидоров	1	Болт

СОЕДИНЕНИЕ ОТНОШЕНИЙ

На практике используются несколько разновидностей операции соединения, отличающиеся немного друг от друга:

- Θ-соединение (тэта-соединение);
- экви-соединение (эквивалентное соединение, частный случай тэта-соединения);
- естественное соединение.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Тэта-соединение

Пусть отношение *A* содержит атрибут *X*, отношение *B* содержит атрибут *Y*, а Θ - один из операторов сравнения (=, ≠, <, >, ≤, ≥). Тогда *Θ-соединением* отношения *A* по атрибуту *X* с отношением *B* по атрибуту *Y* называют отношение *(A TIMES B)* WHERE *XΘY*.

Иногда для операции Θ-соединения применяют следующий, короткий синтаксис:

A [XΘY] B.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Пример.

Рассмотрим некоторую БД, в которой хранятся данные о книгах и читателях. Книгам присвоена категория ценности, а каждому из читателей присвоен уровень доступа. Выдача книг в библиотеке организована таким образом, что читателя имеют право брать только те книги, если их уровень доступа совпадает или превышает категорию ценности книги.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Таблица 17. Отношение A (Читатели)

Код читателя	**ФИО**	**Уровень доступа читателя (X)**
1	Иванов	4
2	Петров	1
3	Сидоров	2

Таблица 18. Отношение B (Книги)

Номер книги	**Наименование книги**	**Ценность книги (Y)**
1	Книга 1	3
2	Книга 2	2
3	Книга 3	1

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Ответ на вопрос "какие читатели имеют право брать какие книги?" дает Θ-соединение *A [X>=Y]*.

Таблица 19. Отношение A [X>=Y] B

Код читателя	ФИО	X (Уровень доступа)	Номер книги	Наименование книги	Y (Ценность книги)
1	Иванов	4	1	Книга 1	3
1	Иванов	4	2	Книга 2	2
1	Иванов	4	3	Книга 3	1
2	Петров	1	3	Книга 3	1
3	Сидоров	2	2	Книга 2	2
3	Сидоров	2	3	Книга 3	1

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Экви-соединение

Наиболее важным частным случаем Θ-соединения является случай, когда Θ есть просто равенство. Синтаксис *экви-соединения*: ***A[X=Y]B***.

Недостатком экви-соединения является то, что если соединение происходит по атрибутам с одинаковыми наименованиями (а так чаще всего и происходит), то в результирующем отношении появляется два атрибута с одинаковыми значениями. Избавиться от этого недостатка можно, взяв проекцию по всем атрибутам, кроме одного из дублирующих. Именно так действует естественное соединение.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Таблица 20. Отношение A (Поставщики)

Номер поставщика	Наименование поставщика	Статус изделия (X)
1	Иванов	4
2	Петров	1
3	Сидоров	2

Таблица 21. Отношение B (Детали)

Номер детали	Наименование детали	Статус изделия (Y)
1	Болт	3
2	Гайка	2
3	Винт	1

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Таблица 22. Отношение A [X=Y] B					
Номер поставщика	**Наименование поставщика**	**X (Статус изделия)**	**Номер детали**	**Наименование детали**	**Y (Статус изделия_1**
3	Сидоров	2	2	Гайка	2
2	Петров	1	3	Винт	1

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Естественное соединение

Пусть даны отношения *A(A1,A2,...,An,X1,X2,...Xn)* и *B(X1,X2,...Xn, B1,B2,. .., Bn)*, имеющие одинаковые атрибуты *X1,X2,...Xn* (т.е. атрибуты с одинаковыми именами и определенные на одинаковых доменах).

Тогда *естественным соединением* отношений *A* и *B* называется отношение с заголовком *(A1,A2,...,An,X1,X2,...Xn, B1,B2,...,Bn)* и телом, содержащим множество строк (кортежей) (*a1,a2,...,an,x1,x2,...,xn,b1,b2,...,bn*), таких, что (*a1,a2,...,an,x1,x2,...,xn*) $\in$ *A* и (*x1,x2,...,xn,,b1. b2.. . . bn*) $\in$ *B*.

Естественное соединение настолько важно, что для него используют специальный синтаксис: ***A* JOIN *B***.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Естественное соединение

В синтаксисе естественного соединения не указываются, по каким атрибутам производится соединение. Естественное соединение производится по всем одинаковым атрибутам.

Можно выполнять последовательное естественное соединение нескольких отношений. Естественное соединение (как, впрочем, и соединение общего вида) обладает свойством *ассоциативности*, т.е.:

(A JOIN B) JOIN C = A JOIN (B JOIN C)

поэтому такие соединения можно записывать, опуская скобки:

A JOIN B JOIN C.

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Пример.

Пусть имеются отношения *P*, *D* и *PD*, хранящие информацию о поставщиках, деталях и поставках соответственно (для удобства введем краткие наименования атрибутов):

Таблица 23. Отношение P (Поставщики)

Номер поставщика (PNUM)	**Наименование поставщика (PNAME)**
1	Иванов
2	Петров
3	Сидоров

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Таблица 24. Отношение D (Детали)

Номер детали (DNUM)	Наименование детали (DNAME)
1	Болт
2	Гайка
3	Винт

Таблица 25. Отношение PD (Поставки)

Номер поставщика (PNUM)	Номер детали (DNUM)	Поставляемое количество VOLUME
1	1	100
1	2	200
1	3	300
2	1	150
2	2	250
3	1	1000

СОЕДИНЕНИЕ ОТНОШЕНИЙ

Ответ на вопрос "какие детали поставляются поставщиками", можно записать в виде естественного соединения трех отношений *P JOIN PD JOIN D*:

Таблица 26. Отношение P JOIN PD JOIN D

Номер поставщика PNUM	**Наименование поставщика PNAME**	**Номер детали DNUM**	**Наименование детали DNAME**	**Поставляемое количество VOLUME**
1	Иванов	1	Болт	100
1	Иванов	2	Гайка	200
1	Иванов	3	Винт	300
2	Петров	1	Болт	150
2	Петров	2	Гайка	250
3	Сидоров	1	Болт	1000

ДЕЛЕНИЕ ОТНОШЕНИЙ

Предположим, что отношение A определено на множестве атрибутов X, а отношение B - на множестве атрибутов Y, причем Y является подмножеством X.

Пусть Z= X- Y, то есть Z является множеством атрибутов отношения A, которые не являются атрибутами отношения *B*.

Результатом *операции деления* является набор строк (кортежей) отношения *A,* определенных на множестве атрибутов Z, которые соответствуют комбинации всех строк отношения *B*.

ДЕЛЕНИЕ ОТНОШЕНИЙ

Либо: Пусть даны отношения $A(X1,X2,\ldots,Xn,Y1,Y2,\ldots,Ym)$ и $B(Y1,Y2,\ldots,Ym)$, причем атрибуты $Y1,Y2,\ldots,Ym$ - общие для двух отношений. *Делением отношений* A на B называется отношение с заголовком $(X1,X2,\ldots,Xn)$ и телом, содержащим множество кортежей $(x1,x2,\ldots,xn)$, таких, что для *всех* кортежей $(y1,y2,\ldots,ym) \in B$ в отношении A найдется кортеж $(x1,x2,\ldots,xn,y1,y2,\ldots,ym)$.

Отношение A выступает в роли *делимого*, отношение B выступает в роли *делителя*. Деление отношений аналогично делению чисел с остатком.

Синтаксис операции деления: ***A* DIVIDBY *B*.**

ДЕЛЕНИЕ ОТНОШЕНИЙ

Пример.

В примере с поставщиками, деталями и поставками ответим на вопрос, "какие поставщики поставляют *все* детали?".

ДЕЛЕНИЕ ОТНОШЕНИЙ

В качестве делимого возьмем проекцию *X= PD [PNUM, DNUM]*, содержащую номера поставщиков и номера поставляемых ими деталей:

Таблица 27. Проекция X=PD[PNUM,DNUM]

Номер поставщика PNUM	**Номер детали DNUM**
1	1
1	2
1	3
2	1
2	2
3	1

ДЕЛЕНИЕ ОТНОШЕНИЙ

В качестве делителя возьмем проекцию *Y=D[DNUM]*, содержащую список номеров *всех* деталей (не обязательно поставляемых кем-либо):

Таблица 29. Проекция Y=D[DNUM]
Номер детали DNUM
1
2
3

ДЕЛЕНИЕ ОТНОШЕНИЙ

Деление *X DIVIDEBY Y* дает список номеров поставщиков, поставляющих *все* детали:

Таблица 29. Отношение X DIVIDEBY Y
Номер поставщика PNUM
1

Оказалось, что только поставщик с номером 1 поставляет все детали.

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

Пример:

Пусть даны два отношения:

СОТРУДНИКИ (СОТР_НОМЕР, СОТР_ИМЯ, СОТР_ЗАРПЛ, ОТД_НОМЕР)

ОТДЕЛЫ (ОТД_НОМЕР, ОТД_КОЛ, ОТД_НАЧ)

Мы хотим узнать имена и номера сотрудников, являющихся начальниками отделов с количеством работников более 10.

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

Выполнение этого запроса средствами реляционной алгебры распадается на четко определенную последовательность шагов:

1) выполнить соединение отношений СОТРУДНИКИ и ОТДЕЛЫ по условию СОТР_НОМ = ОТДЕЛ_НАЧ:

С1= СОТРУДНИКИ [СОТР_НОМ = ОТД_НАЧ] ОТДЕЛЫ

2) из полученного отношения произвести выборку по условию ОТД_КОЛ > 10:

С2 = С1 [ОТД_КОЛ > 10]

3) спроецировать результаты предыдущей операции на атрибуты СОТР_ИМЯ, СОТР_НОМЕР:

С3 = С2 [СОТР_ИМЯ, СОТР_НОМЕР]

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

Порядок выполнения шагов может повлиять на эффективность выполнения запроса.

Так, время выполнения приведенного выше запроса можно сократить, если поменять местами этапы (1) и (2). В этом случае сначала из отношения СОТРУДНИКИ будет сделана выборка всех кортежей со значением атрибута ОТДЕЛ_КОЛ > 10, а затем выполнено соединение результирующего отношения с отношением ОТДЕЛЫ.

Машинное время экономится за счет того, что в операции соединения участвуют меньшие отношения.

МЕХАНИЗМЫ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ

На языке реляционного исчисления данный запрос может быть записан как:

Выдать СОТР_ИМЯ и СОТР_НОМ для СОТРУДНИКИ таких, что существует ОТДЕЛ с таким же, что и СОТР_НОМ значением ОТД_НАЧ и значением ОТД_КОЛ большим 10.

Здесь мы указываем лишь характеристики результирующего отношения, но не говорим о способе его формирования. СУБД сама должна решить какие операции и в каком порядке надо выполнить над отношениями СОТРУДНИКИ и ОТДЕЛЫ. Задача оптимизации выполнения запроса в этом случае также ложится на СУБД.

РЕЛЯЦИОННОЕ ИСЧИСЛЕНИЕ

Реляционное исчисление является прикладной ветвью формального механизма исчисления предикатов первого порядка.

Базисными понятиями исчисления являются понятие *переменной* с определенной для нее областью допустимых значений и понятие *правильно построенной формулы*, опирающейся на переменные, предикаты и кванторы.

РЕЛЯЦИОННОЕ ИСЧИСЛЕНИЕ

В зависимости от того, что является областью определения переменной, различаются *исчисление кортежей и исчисление доменов.*

В исчислении кортежей областями определения переменных являются отношения базы данных, т.е. допустимым значением каждой переменной является кортеж некоторого отношения.

В исчислении доменов областями определения переменных являются домены, на которых определены атрибуты отношений базы данных, т.е. допустимым значением каждой переменной является значение некоторого домена.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Для определения кортежной переменной используется оператор RANGE.

Например, для того, чтобы определить переменную СОТРУДНИК, областью определения которой является отношение СОТРУДНИКИ, нужно употребить конструкцию

RANGE СОТРУДНИК IS СОТРУДНИКИ

Из этого определения следует, что в любой момент времени переменная СОТРУДНИК представляет некоторый кортеж отношения СОТРУДНИКИ.

При использовании кортежных переменных в формулах можно ссылаться на значение атрибута переменной.

Например, для того, чтобы сослаться на значение атрибута СОТР_ИМЯ переменной СОТРУДНИК, нужно употребить конструкцию СОТРУДНИК.СОТР_ИМЯ.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Правильно построенные формулы (WFF - Well-Formed Formula) служат для выражения условий, накладываемых на кортежные переменные.

Основой WFF являются *простые сравнения* (comparison), представляющие собой операции сравнения скалярных значений (значений атрибутов переменных или литерально заданных констант).

Например, конструкция:

«СОТРУДНИК.СОТР_НОМ = 140»

является простым сравнением.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Более сложные варианты WFF строятся с помощью логических связок NOT, AND, OR и IF ... THEN.

Так, если form - WFF, а comp - простое сравнение, то NOT form, comp AND form, comp OR form и IF comp THEN form являются WFF.

Наконец, допускается построение WFF с помощью кванторов.

Если form - это WFF, в которой участвует переменная var, то конструкции EXISTS var (form) и FORALL var (form) представляют wff.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Переменные, входящие в WFF, могут быть свободными или связанными.

Все переменные, входящие в WFF, при построении которой не использовались кванторы, являются *свободными.*

Фактически, это означает, что если для какого-то набора значений свободных кортежных переменных при вычислении WFF получено значение true, то эти значения кортежных переменных могут входить в результирующее отношение.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Если же имя переменной использовано сразу после квантора при построении WFF вида EXISTS var (form) или FORALL var (form), то в этой WFF и во всех WFF, построенных с ее участием, var - это *связанная переменная*.

Это означает, что такая переменная не видна за пределами минимальной WFF, связавшей эту переменную. При вычислении значения такой WFF используется не одно значение связанной переменной, а вся ее область определения.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Пусть СОТР1 и СОТР2 - две кортежные переменные, определенные на отношении СОТРУДНИКИ.

Тогда,

WFF EXISTS СОТР2 (СОТР1.СОТР_ЗАРП > СОТР2.СОТР_ЗАРП)

для текущего кортежа переменной СОТР1 принимает значение true в том и только в том случае, если во всем отношении СОТРУДНИКИ найдется кортеж (связанный с переменной СОТР2) такой, что значение его атрибута СОТР_ЗАРП удовлетворяет внутреннему условию сравнения.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Пусть СОТР1 и СОТР2 - две кортежные переменные, определенные на отношении СОТРУДНИКИ.

Тогда,

WFF FORALL СОТР2 (СОТР1.СОТР_ЗАРП > СОТР2.СОТР_ЗАРП)

для текущего кортежа переменной СОТР1 принимает значение true в том и только в том случае, если для всех кортежей отношения СОТРУДНИКИ (связанных с переменной СОТР2) значения атрибута СОТР_ЗАРП удовлетворяют условию сравнения.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

На самом деле, правильнее говорить не о свободных и связанных переменных, а о свободных и связанных вхождениях переменных.

Легко увидеть, что если переменная var является связанной в WFF form, то во всех WFF, включающих данную, может использоваться имя переменной var, которая может быть свободной или связанной, но в любом случае не имеет никакого отношения к вхождению переменной var в WFF form.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Пример:

EXISTS COTP2 (COTP1.COTP_ОТД_НОМ = COTP2.COTP_ОТД_НОМ)

AND

FORALL COTP2 (COTP1.COTP_ЗАРП > COTP2.COTP_ЗАРП)

Здесь мы имеем два связанных вхождения переменной COTP2 с совершенно разным смыслом.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Чтобы исчисление можно было использовать для реальной работы с БД, требуется еще один компонент, который определяет набор и имена столбцов результирующего отношения. Этот компонент называется *целевым списком* (target_list).

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Целевой список строится из целевых элементов, каждый из которых может иметь следующий вид:

- var.attr, где var - имя свободной переменной соответствующей WFF, а attr - имя атрибута отношения, на котором определена переменная var;
- var, что эквивалентно наличию подсписка var.attr1, var.attr2, ..., var.attrn, где attr1, attr2, ..., attrn включает имена всех атрибутов определяющего отношения;
- new_name = var.attr; new_name - новое имя соответствующего атрибута результирующего отношения.

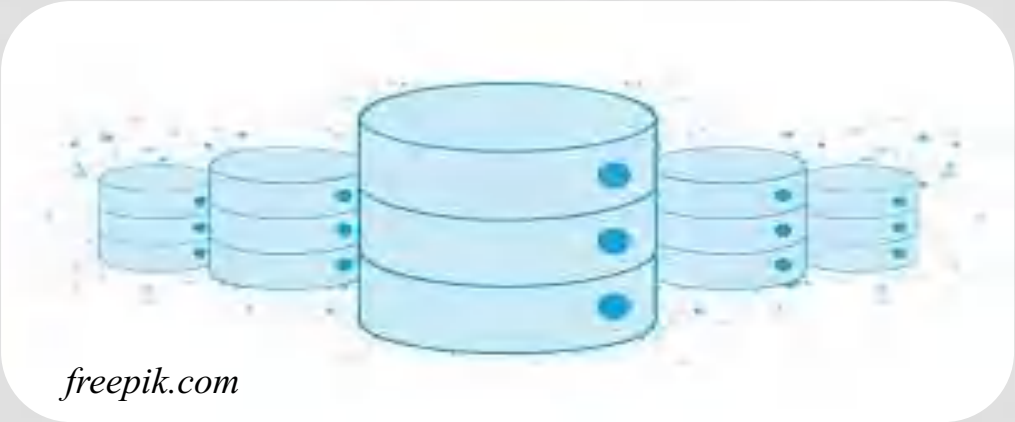
Последний вариант требуется в тех случаях, когда в WFF используются несколько свободных переменных с одинаковой областью определения.

ИСЧИСЛЕНИЕ КОРТЕЖЕЙ

Выражением реляционного исчисления кортежей называется конструкция вида

target_list WHERE wff.

Значением выражения является отношение, тело которого определяется WFF, а набор атрибутов и их имена - целевым списком.



ИСЧИСЛЕНИЕ ДОМЕНОВ

В исчислении доменов областью определения переменных являются не отношения, а домены.

Применительно к базе данных СОТРУДНИКИ-ОТДЕЛЫ можно говорить, например, о доменных переменных ИМЯ (значения - допустимые имена) или НОМ_СОТР (значения - допустимые номера сотрудников).

Основным формальным отличием исчисления доменов от исчисления кортежей является наличие дополнительного набора предикатов, позволяющих выражать так называемые условия членства.

ИСЧИСЛЕНИЕ ДОМЕНОВ

Если R - это n-арное отношение с атрибутами a1, a2, ..., an, то условие членства имеет вид:

R (ai1:vi1, ai2:vi2, ..., aim:vim) (m <= n),

где vij - это либо литерально задаваемая константа, либо имя доменной переменной.

Условие членства принимает значение true в том и только в том случае, если в отношении R существует кортеж, содержащий указанные значения указанных атрибутов.

Если vij - константа, то на атрибут aij задается жесткое условие, не зависящее от текущих значений доменных переменных.

Если же vij - имя доменной переменной, то условие членства может принимать разные значения при разных значениях этой переменной.

ЗАДАНИЯ ДЛЯ СРС

1. Назовите основные составляющие структурной составляющей реляционной базы данных.
2. Объясните, почему не каждая таблица является реляционным отношением?
3. Для чего предназначены связи между отношениями?
4. Как реализуется связь «многие-ко-многим» в реляционных СУБД?
5. Объясните назначение индексов в реляционных базах данных.

ЗАДАНИЯ ДЛЯ СРС

6. Какие виды связей существуют между данными отношениями?

- машина и ее части;
- люди и паспорта;
- дома и улицы;
- заказ билетов в отеле;
- товары и клиенты;
- студенты и банковские карточки;
- писатели и книги;
- кинотеатры и экраны;
- клиент и номер телефона.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Т. Конноли, К. Бегг. Базы данных: Проектирование, реализация и сопровождение. Теория и практика [Текст] : учебное пособие / 3-е изд. - М. ; СПб. ; Киев : Вильямс, 2018. - 1440 с.

2. К.Дж. Дейт. Введение в системы баз данных. М., 2018. - 1328 с.

freepik.com

ЛЕКЦИЯ ОКОНЧЕНА.
БЛАГОДАРЮ ЗА ВНИМАНИЕ!