Кемеровский институт (филиал) РЭУ им. Г.В. Плеханова
Экономический факультет
Кафедра вычислительной техники и информационных технологий

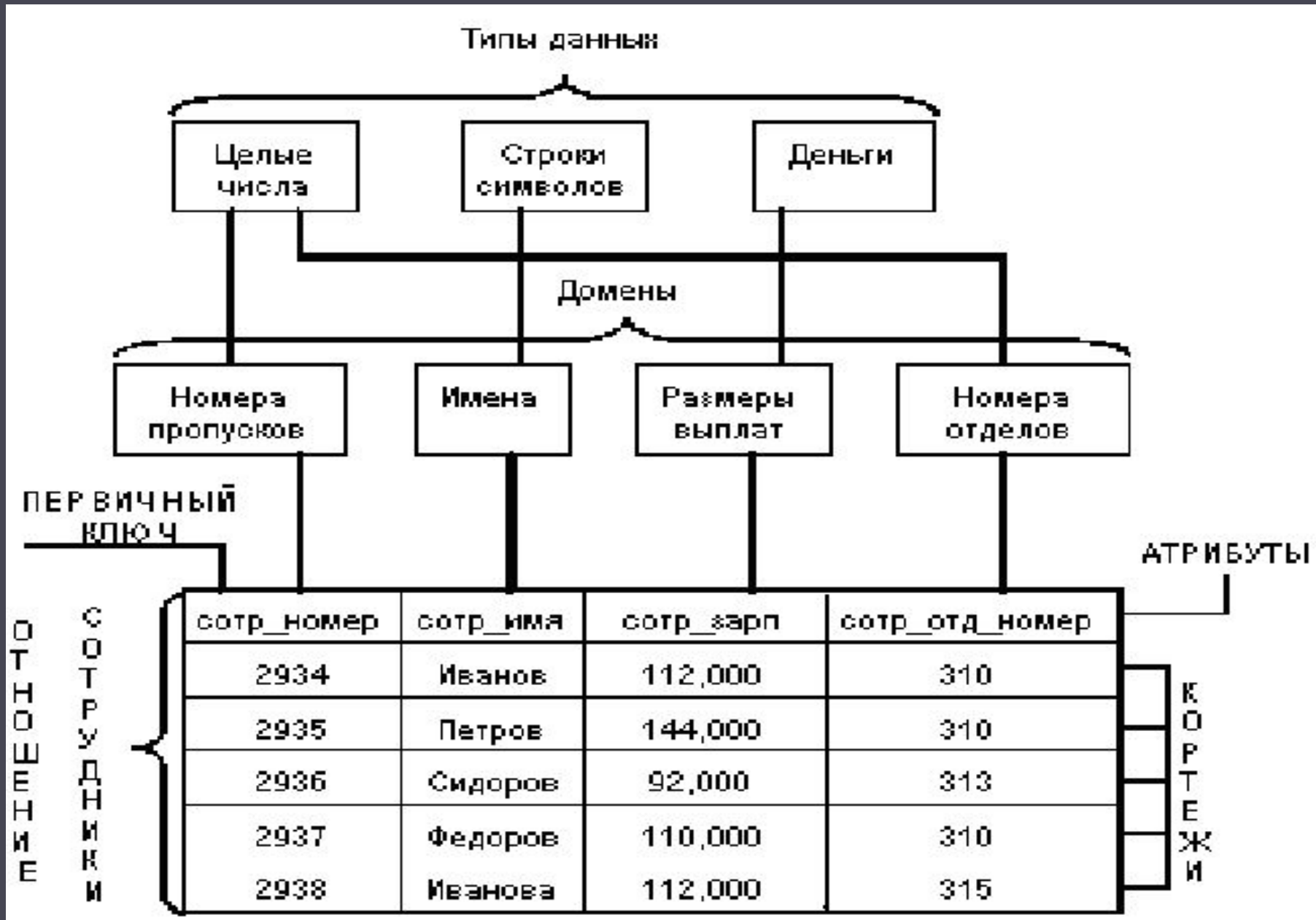
БАЗЫ ДАННЫХ

Лебедева Т.Ф.

2013 г.

Вопросы

1. Чем даталогические документальные модели отличаются от фактографических?
2. Приведите примеры даталогических документальных моделей.
3. Какие компоненты входят в структуру логической (даталогической) модели?
4. Назовите структуры данных иерархических моделей.
5. Что включает в себя физическая модель данных?
6. Чем характеризуется последовательный доступ к данным?
7. Чем характеризуется прямой (произвольный) доступ к данным?
8. Какие методы адресации используются для ускорения доступа к данным?
9. Дайте характеристику методу хеширования.
0. Опишите алгоритм адресации с использованием индексно-последовательного файла?
1. Что такое страница данных? Опишите ее структуру.
2. Укажите последовательность действий доступа к данным.
3. Как связаны страницы данных в наборе?
4. Укажите последовательность действий по добавлению записи о поставках РД7 (пример 1).
5. Укажите последовательность действий по удалению записи о детали Д1 (пример


Типы данных
Целые числа
Строки символов
Деньги
Домены
Номера пропусков
Имена
Размеры выплат
Номера отделов
ПЕРВИЧНЫЙ КЛЮЧ
АТРИБУТЫ
ОТНОШЕНИЕ СОТРУДНИКИ
сотр_номер
сотр_имя
сотр_зарп
сотр_отд_номер
2934
Иванов
112,000
310
2935
Петров
144,000
310
2936
Сидоров
92,000
313
2937
Федоров
110,000
310
2938
Иванова
112,000
315
КОРТЕЖИ

Реляционная модель данных.
Базовые понятия реляционных баз данных

Реляционные (от английского слова relation – отношение) модели были разработаны Э. Коддом в начале 70-х годов. Основными понятиями реляционных баз данных являются *тип данных, домен, атрибут, кортеж, ключи, отношение, схема отношения.*

- *Атрибут* – это наименьшая поименованная единица данных, к которой СУБД может адресоваться непосредственно, и с помощью которой выполняется построение всех остальных структур. Атрибут имеет имя и значение.
- Понятие *тип данных* в реляционной модели данных полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных БД допускается хранение данных следующих типов: символьных, числовых, битовых, специализированных числовых данных (таких как «деньги», «темпоральных» данных (дата, время, временной интервал). Достаточно активно развивается подход к расширению возможностей реляционных систем абстрактными типами данных (соответствующими возможностями обладают, например, системы семейства Ingres/Postgres). В нашем примере мы имеем дело с данными трех типов: строки символов, целые числа и «деньги».

Реляционная модель данных.
Базовые понятия реляционных баз данных

- *Домен* – допустимое потенциальное множество значений простого типа данных. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат «истина», то элемент данных является элементом домена. Например, домен «Имена» в нашем примере определен на базовом типе строк символов, но в число его значений могут входить только те строки, которые могут представлять имя (в частности, такие строки не могут начинаться с мягкого знака).

Следует отметить также семантическую нагрузку понятия домена: *данные считаются сравнимыми только в том случае, когда они относятся к одному домену*. В нашем примере значения доменов «Номера пропусков» и «Номера отделов» относятся к типу целых чисел, но не являются сравнимыми. Заметим, что в большинстве реляционных СУБД понятие домена не используется, хотя в Oracle V.7 оно поддерживается.

- *Кортеж*, соответствующий данной схеме отношения, - это множество пар {имя атрибута - значение}, 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). Степень или «арность» кортежа, т.е. число элементов в кортеже, совпадает с «арностью» соответствующей схемы отношения.

- *Кортеж* - это набор именованных значений заданного типа.

Реляционная модель данных.

Базовые понятия реляционных баз данных

- *Схема отношения* - это именованное множество пар {имя атрибута – имя домена (или типа, если понятие домена не поддерживается)}. Степень или "арность" схемы отношения - мощность этого множества. Степень отношения СОТРУДНИКИ равна четырем, то есть оно является 4-арным.
- *Схема БД* (в структурном смысле) - это набор именованных схем отношений
- *Отношение* - это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят «отношение-схема» и «отношение-экземпляр», иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой. Однако в реляционных базах данных это не принято. Имя схемы отношения в таких базах данных всегда совпадает с именем соответствующего отношения-экземпляра. В классических реляционных базах данных после определения схемы базы данных изменяются только отношения-экземпляры. В них могут появляться новые и удаляться или модифицироваться существующие кортежи. Однако во многих

Реляционная модель данных.
Базовые понятия реляционных баз данных

- *Реляционная база данных* - это набор отношений, имена которых совпадают с именами схем отношений в схеме БД.
- *Ключ* – набор атрибутов, значение которых однозначно идентифицирует кортежи. Отношение может иметь несколько ключей, но всегда один из ключей объявляется *первичным* и его значения не могут обновляться.
- Все остальные ключи называются возможными ключами.
- Атрибуты, представляющие собой копии ключей других отношений называются *внешними ключами.*

Связи в реляционных базах данных

В реляционных БД связи позволяют избежать избыточности данных. В большинстве случаев связь сопоставляет первичный ключ одной таблицы с внешним ключом другой таблицы. Связи между таблицами могут быть трех видов:

- *Один-к-одному*. Одной записи таблицы **А** соответствует одна запись таблицы **Б**, и наоборот. Этот тип связи применяется достаточно редко. Единственный случай, когда применение этого типа связи оправданно – разбиение таблицы, содержащей очень большое количество полей, на несколько частей.

- *Один-ко-многим.* Одной записи таблицы **А** (главной) соответствует несколько записей таблицы **Б** (или ни одной). В свою очередь каждой записи таблицы **Б** (подчиненной) может соответствовать только одна запись таблицы **А**. Наиболее употребительный вид связи.
- *Многие-ко-многим.* При этом типе связи многим записям из таблицы **А** может соответствовать много записей из таблицы **Б** (и наоборот). Такую связь в реляционных БД можно организовать только при помощи третьей вспомогательной таблицы. По сути связь «*многие-ко-многим*» представляет собой две связи типа «*один-ко-многим*». При этом таблицы **А** и **Б** расположены со стороны *«один»*, а вспомогательная таблица со стороны – «*многие*».

Как видно, основные структурные понятия реляционной модели данных (если не считать понятия домена) имеют очень простую интуитивную интерпретацию, хотя в теории реляционных БД все они определяются абсолютно формально и точно.

Таблица 1 – Соответствие терминов

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	Столбец (колонка) таблицы
Кортеж отношения	Строка таблицы
Степень (-арность) отношения	Количество столбцов таблицы
Мощность отношения	Количество строк таблицы
Домены и типы данных	Типы данных в ячейках таблицы

Реляционная модель данных. Фундаментальные свойства отношений

1 Отсутствие кортежей-дубликатов

То свойство, что отношения не содержат кортежей-дубликатов, следует из определения отношения как множества кортежей. В классической теории множеств по определению каждое множество состоит из различных элементов.

Из этого свойства вытекает наличие у каждого отношения так называемого первичного ключа - набора атрибутов, значения которых однозначно определяют кортеж отношения. Для каждого отношения по крайней мере полный набор его атрибутов обладает этим свойством. Однако при формальном определении первичного ключа требуется обеспечение его «минимальности», т.е. в набор атрибутов первичного ключа не должны входить такие атрибуты, которые можно отбросить без ущерба для основного свойства, - однозначно определять кортеж. Понятие *первичного ключа* является исключительно важным в связи с понятием целостности баз данных.

2 Отсутствие упорядоченности кортежей

Свойство отсутствия упорядоченности кортежей отношения также является следствием определения отношения-экземпляра как множества кортежей. 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.

3 Отсутствие упорядоченности атрибутов

Атрибуты отношений не упорядочены, поскольку по определению схема отношения есть множество пар {имя атрибута – имя домена}. 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, и хотя упорядоченность набора атрибутов отношения явно не требуется, часто в качестве неявного порядка атрибутов используется их порядок в линейной форме определения схемы отношения.

4 Атомарность значений атрибутов

Значения всех атрибутов являются атомарными. Это следует из определения домена как потенциального множества значений *простого типа данных*, т.е. среди значений домена не могут содержаться множества значений (отношения). Принято говорить, что в реляционных базах данных допускаются только нормализованные отношения или отношения, представленные в *первой нормальной форме*.

Реляционная модель данных. Характеристика реляционной модели данных

Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода: структурной части, манипуляционной части и целостной части.

- *В структурной части реляционной модели данных* фиксируется, что единственной структурой данных, используемой в реляционных БД, является нормализованное n-арное отношение. По сути дела, ранее мы рассматривали именно понятия и свойства структурной составляющей реляционной модели.
- *В манипуляционной части реляционной модели данных* утверждаются два фундаментальных механизма манипулирования реляционными БД - *реляционная алгебра* и *реляционное исчисление*. Первый механизм базируется в основном на классической теории множеств (с некоторыми уточнениями), а второй - на классическом логическом аппарате исчисления предикатов первого порядка.
- *В целостной части реляционной модели данных* фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД.

Реляционная модель данных. Характеристика реляционной модели данных

- Первое требование называется *требованием целостности сущностей*. Объектам или сущностям реального мира в реляционных БД соответствуют кортежи отношений. Конкретно требование 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.* Как видно из предыдущего раздела, это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.
- Второе требование называется *требованием целостности по ссылкам* и является более сложным. Очевидно, что при соблюдении нормализованности отношений сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений. Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом.

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.

Реляционная модель данных.
Характеристика реляционной модели данных

Ограничения целостности сущности и по ссылкам должны поддерживаться СУБД.

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

С целостностью по ссылкам дела обстоят несколько более сложно.

Существуют три подхода, каждый из которых поддерживает целостность по ссылкам:

- 1. запрещается производить удаление кортежа, на который существуют ссылки (т.е. сначала нужно либо удалить ссылающиеся кортежи, либо соответствующим образом изменить значения их внешнего ключа);
- 2. при удалении кортежа, на который имеются ссылки, во всех ссылающихся кортежах значение внешнего ключа автоматически становится неопределенным;
- 3. *каскадное удаление,* состоящее в том, что при удалении кортежа из отношения, на которое ведет ссылка, из ссылающегося отношения автоматически удаляются все ссылающиеся кортежи.

В развитых реляционных СУБД обычно можно выбрать способ поддержания

В подходе 2

использовано понятие «неопределенного» значения ключа. В реальном мире часто встречается ситуация, когда данные неизвестны или неполны. Для того чтобы обойти проблему неполных или неизвестных данных, в базах данных могут использоваться так называемые ***null-значения***.

Null-значение - это, собственно, не значение, а некий маркер, показывающий, что значение неизвестно.

- Таким образом, в ситуации, когда возможно появление неопределенных, неизвестных или неполных данных, разработчик имеет на выбор два варианта.
- *Первый вариант* состоит в том, чтобы не использовать null-значения, а вместо неизвестных данных вводить либо нулевые значения, либо значения специального вида - например, договориться, что строка "КЛЮЧ НЕИЗВЕСТЕН" и есть те данные, которые нужно вводить вместо неизвестного ключа. В любом случае пользователь (или разработчик) ответственен за правильную трактовку таких данных.

Трехзначная логика (3VL)

Второй вариант состоит в использовании null-значений вместо неизвестных данных. Возникает необходимость использования *трехзначной логики* при оперировании с данными, которые могут содержать null-значения. В этом случае при неаккуратном формулировании запросов, даже самые естественные запросы могут давать неправильные ответы. Практически все реализации современных реляционных СУБД позволяют использовать null-значения, несмотря на их недостаточную теоретическую обоснованность.

Приведем здесь описание трехзначной логики, необходимой для работы с null-значениями.

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 **T - ИСТИНА** и **F - ЛОЖЬ**, введено значение **U - НЕИЗВЕСТНО**. Логическое значение U - это то же самое, что и null-значение. Трехзначная логика базируется на следующих таблицах истинности:

Таблицы истинности логических операций AND, OR, NOT

AND	F	T	U
F	F	F	F
T	F	T	U
U	F	U	U
OR	**F**	**T**	**U**
F	F	T	U
T	T	T	T
U	U	T	U
NOT			
F		T	
T		F	
U		U	

Реляционная модель данных. Трехзначная логика (3VL)

- 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 ≠ null** также принимает значение не ИСТИНА, а НЕИЗВЕСТНО! Значит также, что и выражение **x ≠ x** тоже не обязательно ЛОЖЬ!
- Парадокс 3. **a ∧ not(a)** не обязательно ИСТИНА. Значит, в трехзначной логике не работает принцип исключенного третьего (любое высказывание либо истинно, либо ложно).
- Таких парадоксов можно построить сколько угодно. Конечно, это на самом деле не парадоксы, а просто следствия из аксиом трехзначной логики.

Реляционная алгебра

Вторая часть реляционной модели, манипуляционная часть, утверждает, что доступ к реляционным данным осуществляется при помощи *реляционной алгебры* или эквивалентного ему *реляционного исчисления*.

- В реализациях конкретных реляционных СУБД сейчас не используется в чистом виде ни реляционная алгебра, ни реляционное исчисление.
- Фактическим стандартом доступа к реляционным данным стал язык SQL (Structured Query Language). Язык SQL представляет собой смесь операторов реляционной алгебры и выражений реляционного исчисления, использующий синтаксис, близкий к фразам английского языка и расширенный дополнительными возможностями, отсутствующими в реляционной алгебре и реляционном исчислении.
- Вообще, язык доступа к данным называется ***реляционно полным***, если он по выразительной силе не уступает реляционной алгебре (или, что то же самое, реляционному исчислению), т.е. любой оператор реляционной алгебры может быть выражен средствами

Реляционная алгебра

Реляционная алгебра представляет собой набор операторов, использующих отношения в качестве аргументов, и возвращающие отношения в качестве результата. Таким образом, реляционный оператор выглядит как функция с отношениями в качестве аргум $R = f(R_1, R_2, \ldots, R_n)$

Реляционная алгебра является *замкнутой*, т.к. в качестве аргументов в реляционные операторы можно подставлять другие реляционные операторы, подходящие по типу.

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

Реляционная алгебра

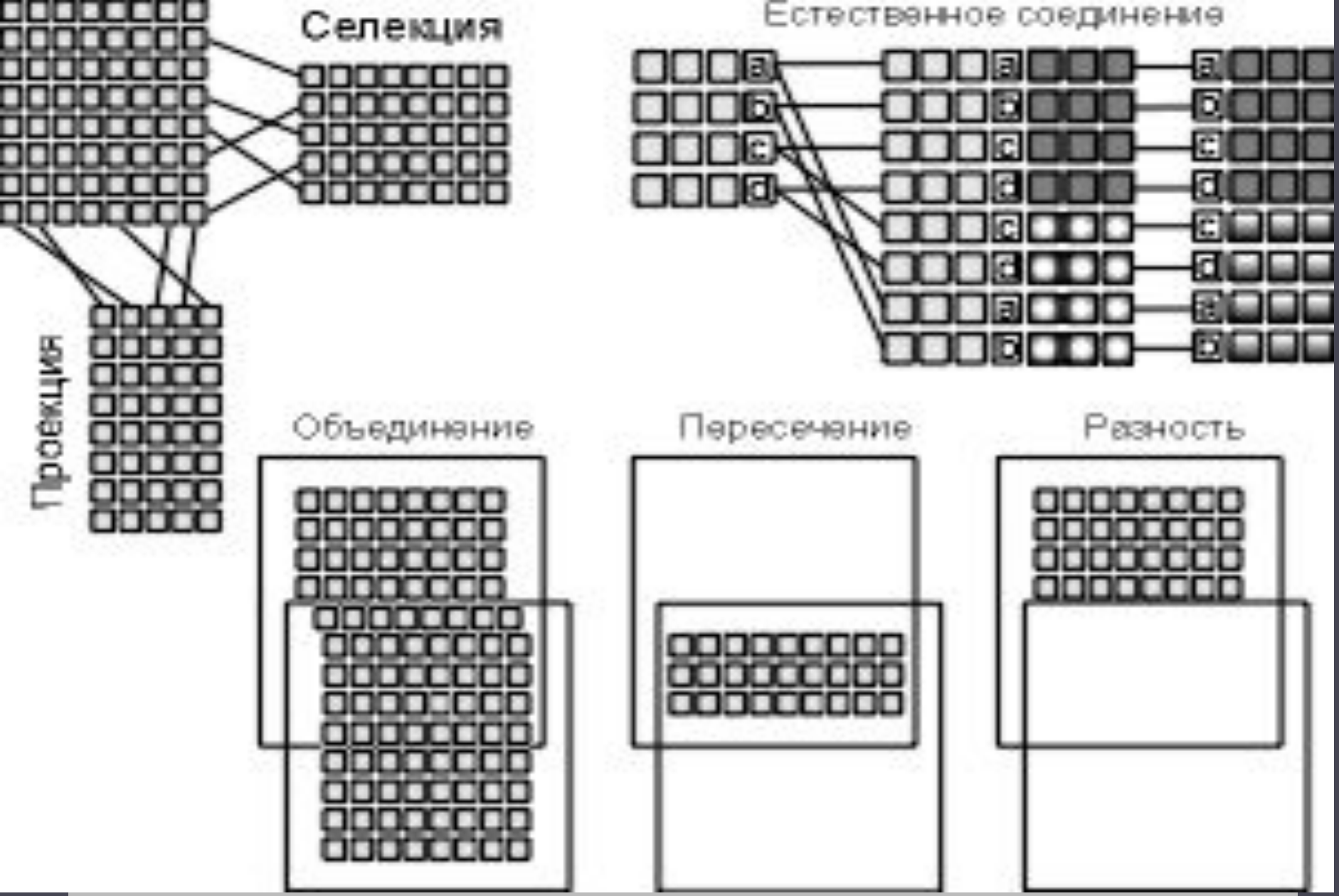
Традиционно, вслед за Э. Коддом, определяют восемь реляционных операций реляционной алгебры, объединенных в две группы.

1 *Теоретико-множественные операции:*

- объединение;
- пересечение;
- вычитание;
- декартово произведение.

2 *Специальные реляционные операции:*

- выборка (селекция, ограничение);
- проекция;
- соединение;
- деление.



Графическая интерпретация некоторых операций реляционной алгебры

Реляционная алгебра

- Не все они являются независимыми, т.е. некоторые из этих операций могут быть выражены через другие реляционные операции. Каждая операция реляционной алгебры использует одну или несколько таблиц (отношений) в качестве своих операндов и продуцирует в результате новую таблицу, т.е. позволяет «разрезать» или «склеивать» таблицы (рис.)
- 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. Если не вдаваться в некоторые тонкости, которые мы рассмотрим в следующих подразделах, то почти все операции предложенного выше набора обладают очевидной и простой интерпретацией:

Реляционная алгебра

1. *Объединение.* При выполнении операции объединения двух отношений производится отношение, включающее все кортежи, входящие хотя бы в одно из отношений-операндов. *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 кортежей, принадлежащих или A , или B, или обоим отношениям.

Синтаксис операции объединения: A UNION B

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 один раз.

Пример 1 Пусть даны два отношения A и B с информацией о сотрудниках:

Отношение A		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	6000
2	Петров	7800
3	Сидоров	9000
Отношение B		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	6000
4	Пушников	8500
3	Сидоров	9000
Отношение	**A UNION B**	
Табельный номер	**Фамилия**	**Зарплата**
I	Иванов	6000
2	Петров	7800
3	Сидоров	9000
4	Пушников	8500

Реляционная алгебра

2. *Пересечение.* Операция пересечения двух отношений производит отношение, включающее все кортежи, входящие в оба отношения-операнда. *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 кортежей, принадлежащих одновременно обоим отношениям A и B .

Синтаксис операции пересечения: A INTERSECT B

Пример 2 Для тех же отношений A и B , что и в предыдущем примере пересечение

Отношение A		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	6000
2	Петров	7800
3	Сидоров	9000
Отношение B		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	6000
4	Пушников	8500
3	Сидоров	9000
Отношение	**A INTERSECT B**	
Табельный номер	**Фамилия**	**Зарплата**
I	Иванов	6000
3	Сидоров	9000

Реляционная алгебра

3. Разность. Отношение, являющееся разностью двух отношений включает все кортежи, входящие в отношение - первый операнд, такие, что ни один из них не входит в отношение, являющееся вторым операндом. *Вычитанием (разностью)* двух совместимых по типу отношений и называется отношение с тем же заголовком, что и у отношений A и B , и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B .

Синтаксис операции вычитания: A MINUS B

Пример 3 Для тех же отношений и , что и в предыдущем примере вычитание Отношение A MINUS B имеет вид:

Отношение A		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	6000
2	Петров	7800
3	Сидоров	9000
Отношение B		
Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	6000
4	Пушников	8500
3	Сидоров	9000
Отношение	**A MINUS B**	
Табельный номер	**Фамилия**	**Зарплата**
2	Петров	7800