Язык sql

- Все запросы, которые мы рассматривали до сих пор, создавались либо с помощью мастера, либо с помощью Конструктора запросов. Конструктор запросов представляет собой графический инструмент для создания запросов по образцу (QBE — Query By Example). Однако на самом деле любой запрос хранится в базе данных в формате SQL (Structured Query Language — язык структурированных запросов). Основное достоинство этого языка состоит в том, что он является стандартом для большинства реляционных СУБД. SQL имеет унифицированный набор инструкций, которые можно использовать во всех СУБД, поддерживающих этот язык.

- На языке SQL описываются наборы данных, помогающие получить ответы на вопросы. При использовании SQL необходимо применять правильный синтаксис. Синтаксис — это набор правил, позволяющих правильно сочетать элементы языка. Синтаксис SQL основан на синтаксисе английского языка и включает много таких же элементов, как и синтаксис языка Visual Basic для приложений (VBA).

Типы команд SQL

1.команды определения данных

- CREATE TABLE - Создать таблицу
- ALTER TABLE Модифицировать таблицу
- DROP TABLE Удалить таблицу

2. **команды обработки данных**

- INSERT - Вставить данные в таблицу
- UPDATE - Обновить данные
- DELETE - Удалить данные

3. **команда запросов данных**

SELECT - Выполнить запрос из таблиц базы

- Ядром языка SQL является инструкция SELECT (Что выбрать?). Она используется для отбора полей из реляционных таблиц и содержит три основных предложения:
- FROM (Откуда выбирать?),
- WHERE (За каким условием?),
- ORDER BY (Как сортировать?).
- При формировании запроса на SQL обязательными в использовании являются SELECT и FROM. Программный модуль заканчивается знаком «;».

Синтаксис инструкции SELECT

SELECT [ALL | DISTINCT] <список данных>

FROM <список таблиц>

[WHERE <условие выборки>]

[GROUP BY <имя столбца> [,<имя столбца>]...] [HAVING <условие поиска>]

[ORDER BY <спецификация> [,<снецификация>] ...]

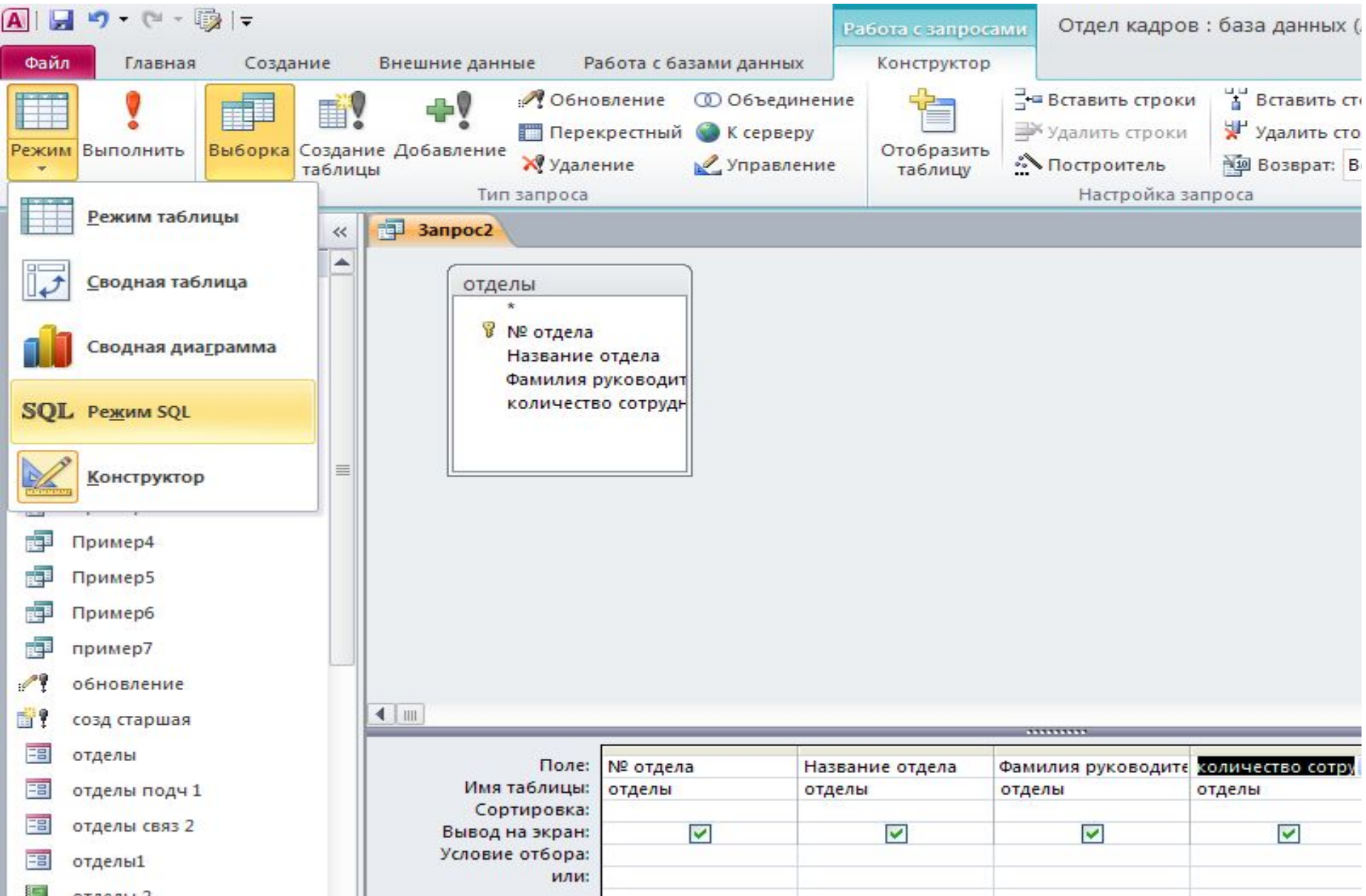
- Оператор SELECT позволяет производить выборку и вычисления над данными из одной или нескольких таблиц. Результатом выполнения оператора является ответная таблица, которая может иметь (ALL), или не иметь (DISTINCT) повторяющиеся строки. По умолчанию в ответную таблицу включаются все строки, в том числе и повторяющиеся.

- Список данных может содержать имена столбцов, участвующих в запросе, а также выражения над столбцами. В простейшем случае в выражениях можно записывать имена столбцов, знаки арифметических операций (+, -,*,/), константы и круглые скобки. Если в списке данных записано выражение, то наряду с выборкой данных выполняются вычисления, результаты которого попадают в новый (создаваемый) столбец ответной таблицы.
- При использовании в списках данных имен столбцов нескольких таблиц для указания принадлежности столбца некоторой таблице применяют конструкцию вида: <имя таблицы>.<имя столбца>.

- Операнд WHERE задает условия, которым должны удовлетворять записи в результирующей таблице. Выражение <условие выборки> является логическим. Его элементами могут быть имена столбцов, операции сравнения, арифметические операции, логические связки (И, ИЛИ, НЕТ), скобки, специальные функции LIKE, NULL, IN и т. д.
- Операнд GROUP BY позволяет выделять в результирующем множестве записей группы. **Группой** являются записи с совпадающими значениями в столбцах, перечисленных за ключевыми словами GROUP BY. Выделение групп требуется для использования в логических выражениях операндов WHERE и HAVING, а также для выполнения операций (вычислений) над группами.
- В логических и арифметических выражениях можно использовать следующие групповые операции (функции): AVG (среднее значение в группе), MAX (максимальное значение в группе), MIN (минимальное значение в группе), SUM (сумма значений в группе), COUNT (число значений в группе).

- Операнд HAVING действует совместно с операндом GROUP BY и используется для дополнительной селекции записей во время определения групп. Правила записи <условия поиска> аналогичны правилам формирования <условия выборки> операнда WHERE.
- Операнд ORDER BY задает порядок сортировки результирующего множества. Обычно каждая <спецификация> аналогична соответствующей конструкции оператора CREATE INDEX и представляет собой пару вида: <имя столбца> [ASC | DESC].

Просмотр SQL-инструкций



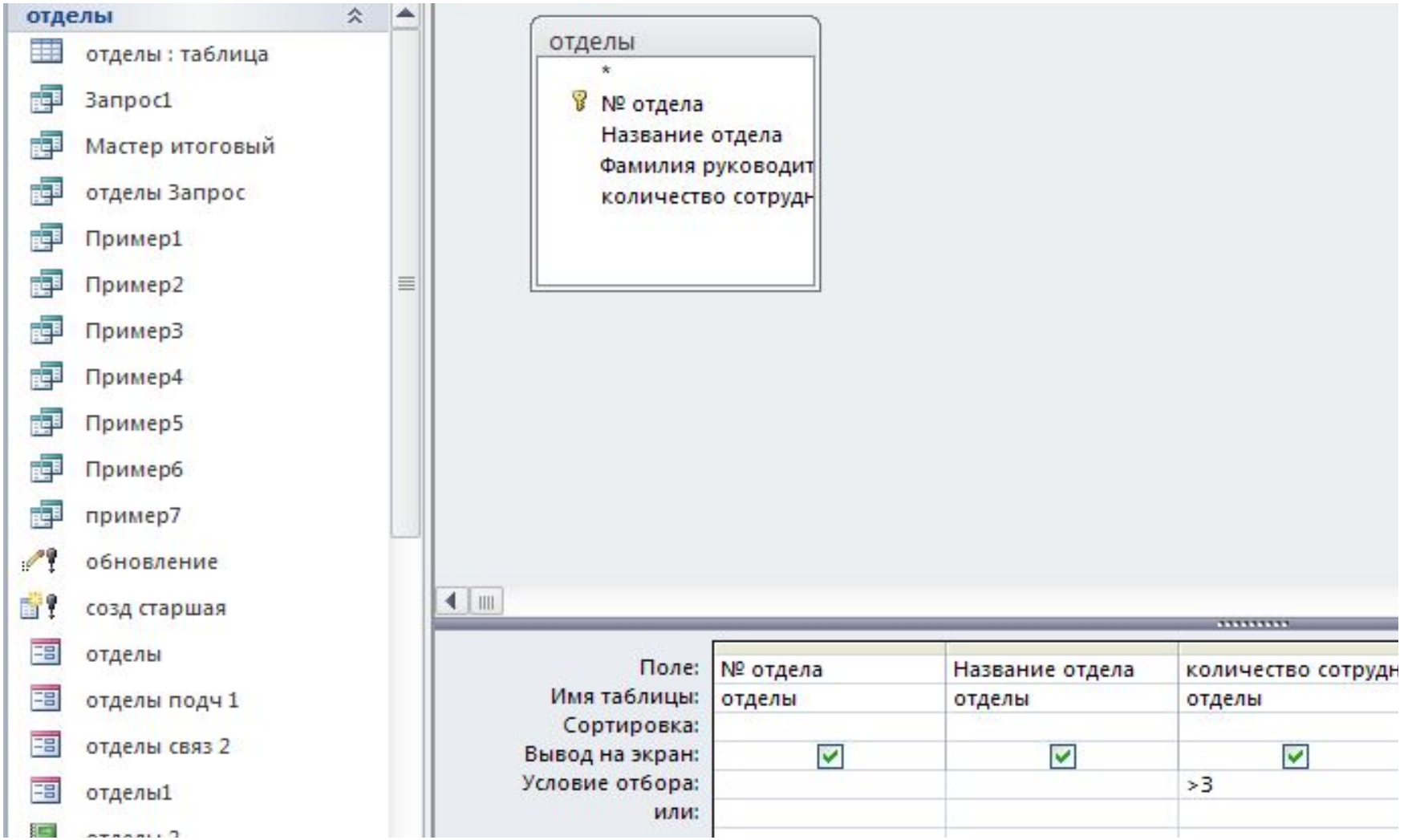
- SELECT отделы.[№ отдела], отделы.[Название отдела], отделы.[Фамилия руководителя], отделы.[количество сотрудников]

  FROM отделы;

или

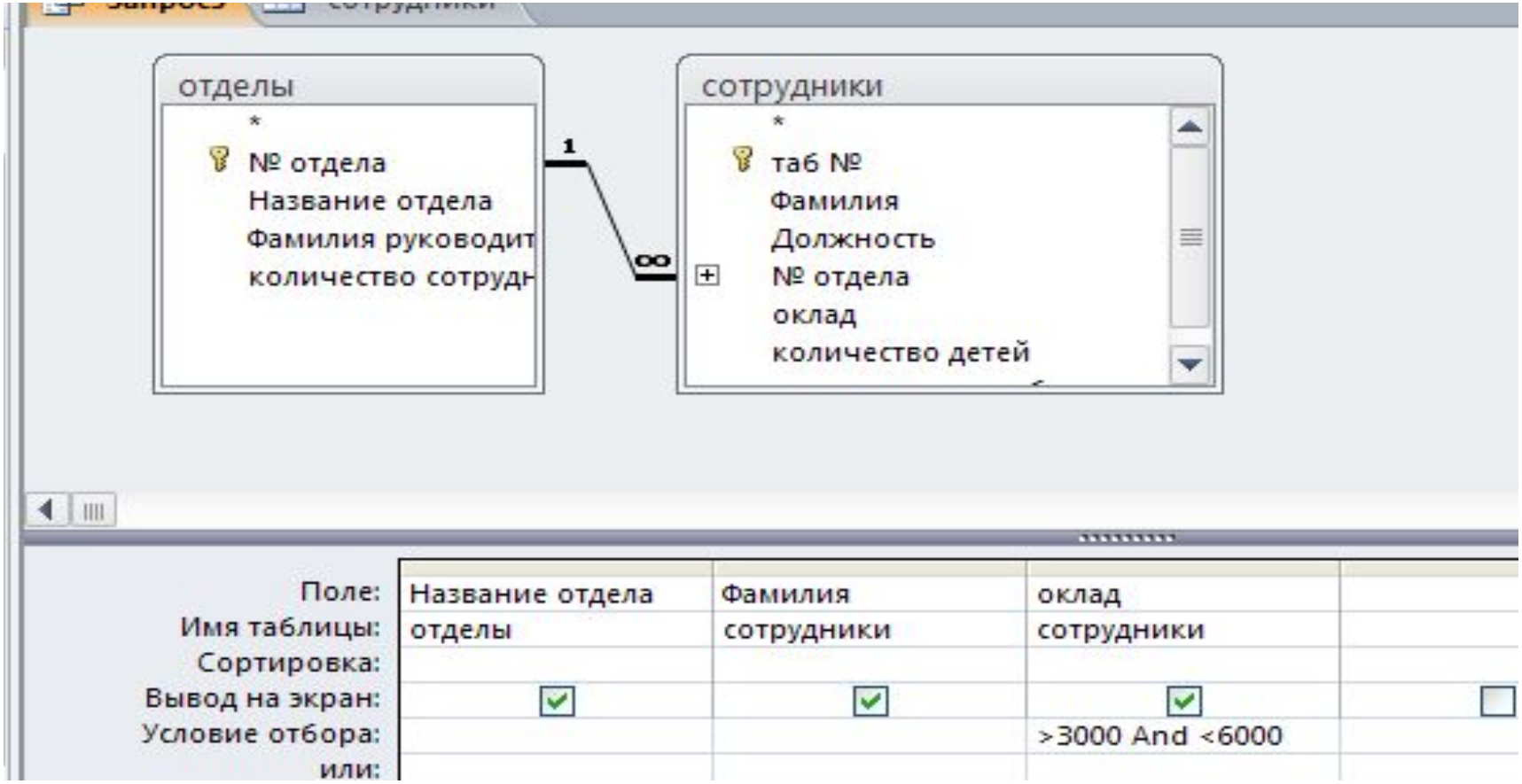
- SELECT * FROM отделы;

Квадратные скобки используются для задания имен полей, которые содержат недопустимые символы, включая пробелы и разделители, например [Название компании].



SELECT отделы.[№ отдела], отделы.[Название отдела],
отделы.[количество сотрудников]
FROM отделы
WHERE (((отделы.[количество сотрудников])>3));

Выборка из нескольких таблиц



SELECT отделы.[Название отдела], сотрудники.Фамилия, сотрудники.оклад
FROM отделы INNER JOIN сотрудники ON отделы.[№ отдела] = сотрудники.[№ отдела].Value
WHERE (((сотрудники.оклад)>3000 And (сотрудники.оклад)<6000));

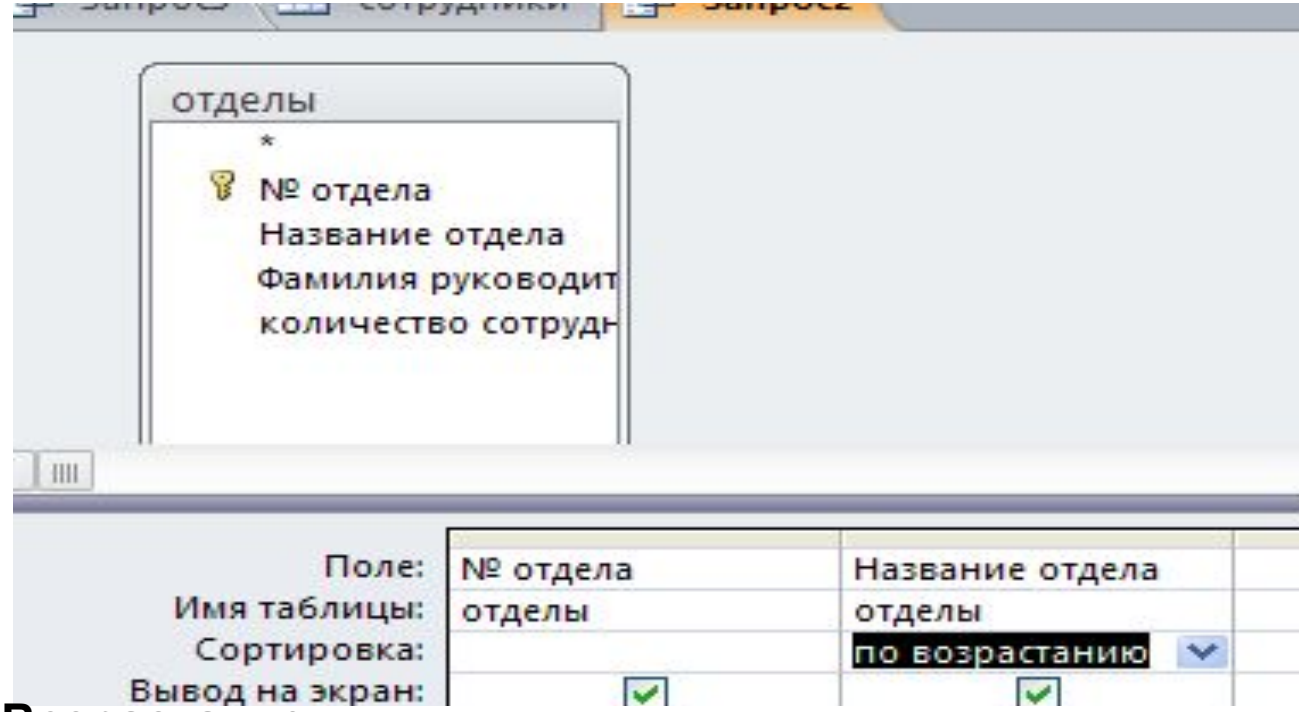
Связи между таблицами

- Таблица1 INNER JOIN Таблица2 (внутреннее объединение);
- Сразу после способа объединения необходимо поместить фразу
  ON Таблица1.Ключ = Таблица2.ВнешнийКлюч

- *Ключ* - имя ключевого поля со стороны 1.

  ВнешнийКлюч - имя связующего поля со стороны N.

Сортировка



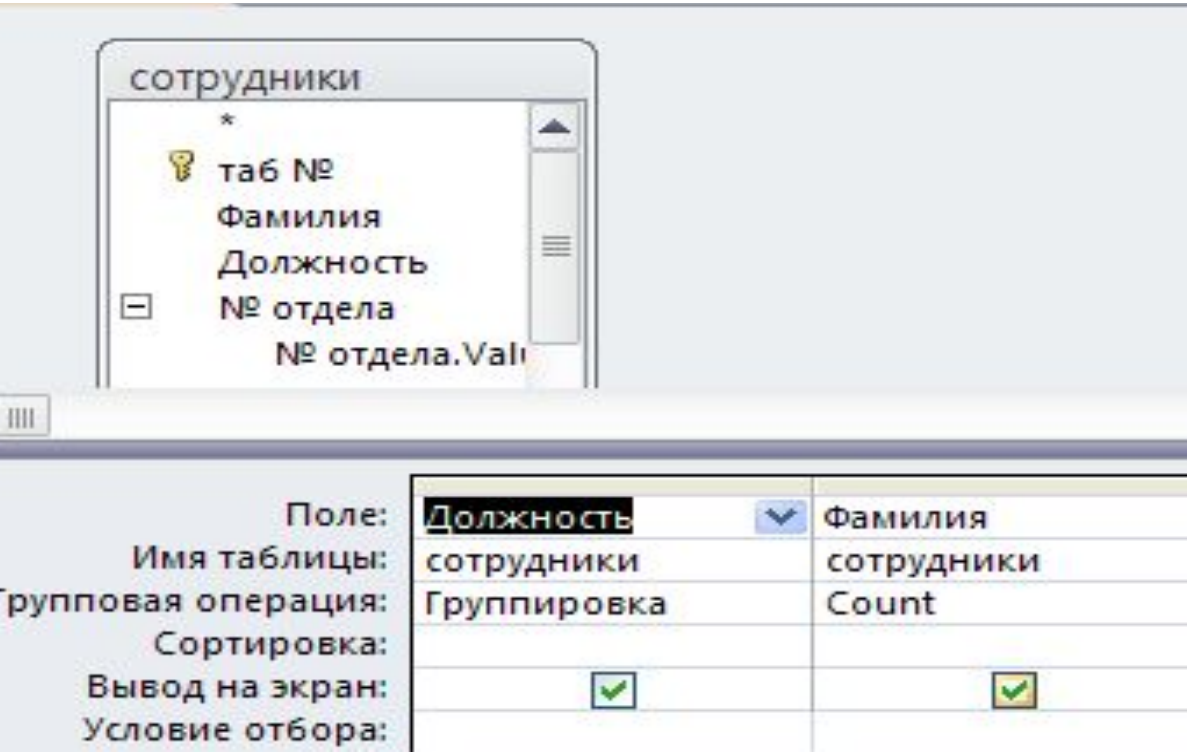
Возрастание

```
SELECT отделы.[№ отдела], отделы.[Название отдела]
FROM отделы
ORDER BY отделы.[Название отдела];
```

Убывание

```
SELECT отделы.[№ отдела], отделы.[Название отдела]
FROM отделы
ORDER BY отделы.[Название отдела] DESC;
```

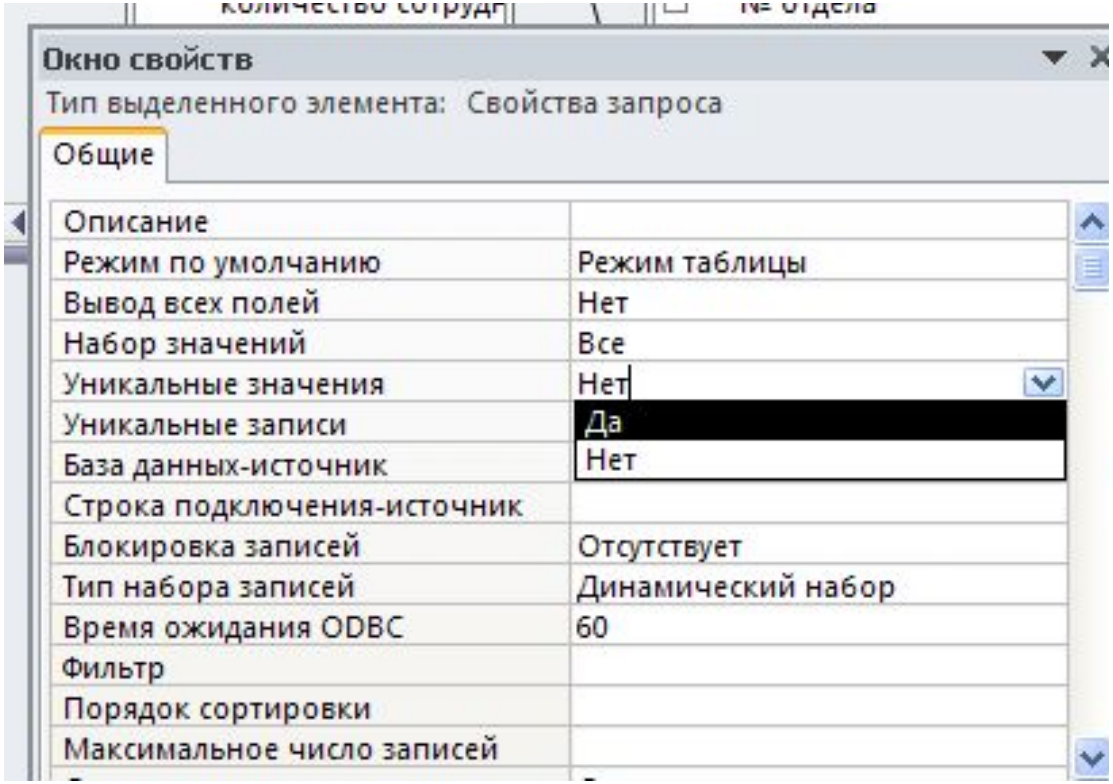
Группировка



Должность	Count-Фамилия
агент	1
бухгалтер	3
гл. бухгалтер	1
дворник	2
заведующий	5
инженер	3
конструктор	1
фотограф	1
художник	1
экономист	2

```
SELECT сотрудники.Должность, Count(сотрудники.Фамилия) AS
[Count-Фамилия]
FROM сотрудники
GROUP BY сотрудники.Должность;
```

```
SELECT DISTINCT отделы.[Название отдела], сотрудники.Фамилия,
    сотрудники.оклад
FROM отделы INNER JOIN сотрудники ON отделы.[№ отдела] =
    сотрудники.[№ отдела].Value
WHERE (((сотрудники.оклад)>3000 And (сотрудники.оклад)<6000));
```



НОРМАЛИЗАЦИЯ ОТНОШЕНИЙ

Реляционная модель данных

Теоретической основой этой модели является теория отношений и основной структурой данных – отношение. Именно поэтому модель получила название ***реляционной*** *(*от английского слова *relation* — отношение).

Отношение - абстракция описываемого объекта как совокупность его свойств. Подавляющее число создаваемых и используемых баз данных являются **реляционными**. Их создание и развитие связано с научными работами известного американского математика, специалиста в области систем баз данных Э. Кодда. Наглядной формой представления отношения является ***двумерная таблица***.

Обычное представление	База данных	Реляционная модель
Таблица	Таблица	Отношение
Строка	Запись	Кортеж
Название столбца	Поле	Атрибут
Множество значений столбца	Множество значений поля	Домен (множество значений атрибута)

Отношение представляет собой множество элементов, называемых *кортежами.*

Свойства реляционной таблицы

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая *реляционная таблица* обладает следующими свойствами:

- каждый элемент таблицы — один элемент данных;
- все столбцы (поля, атрибуты) в таблице однородные, т.е. все элементы в одном столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя;
- одинаковые строки (записи, кортежи) в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.
- Каждое поле содержит одну характеристику объекта предметной области. В записи собраны сведения об одном экземпляре этого объекта
- Поле, каждое значение которого однозначно определяет соответствующую запись, называется ***простым ключом*** (ключевым полем). Ключ, состоящий из нескольких полей называется ***составным ключом***

- Существуют **ключи двух типов: первичные и вторичные или внешние.**
  Первичный ключ – это одно или несколько полей (столбцов), комбинация значений которых однозначно определяет каждую запись в таблице. Первичный ключ не допускает значений **Null** и всегда должен иметь уникальный индекс. Первичный ключ используется для связывания таблицы с внешними ключами в других таблицах.
  Внешний (вторичный) ключ - это столбец или подмножество одной таблицы, который может служить в качестве первичного ключа для другой таблицы. *Внешний ключ таблицы является ссылкой на первичный ключ другой таблицы.* Внешний ключ определяет способ объединения таблиц.
- Если таблица связана с несколькими другими таблицами, она может иметь несколько внешних ключей.

- Наименьшая единица данных реляционной модели — это отдельное атомарное (неразложимое) для данной модели значение данных. Доменом называется множество атомарных значений одного и того же типа. Данные считаются сравнимыми только в том случае, когда они относятся к одному домену
- Основной элемент реляционной модели – это кортеж. Кортеж – это упорядоченный набор элементов, каждый из которых принадлежит определенному множеству или, иначе говоря, имеет свой тип. Совокупность однородных по структуре кортежей образует отношение.

- Отношение – это таблица с данными. Кортеж — строка таблицы. Какого типа кортежи содержатся в отношении, или, что то же самое, каков формат строк в таблице, определяется заголовком отношения или таблицы. Каждый из столбцов таблицы образует домен. Значения, которое могут принимать элементы домена, называются атрибутами. Строки таблицы – это совокупность атрибутов, соответствующих доменам.

Совокупность всех отношений определяет базу данных. Каждое отношение хранит свою логическую часть информации. Чтобы получить определенные сведения может потребоваться сопоставление информации из разных отношений. Кодд описал восемь основных операций реляционной алгебры, позволяющих манипулировать с кортежами:

- Объединение;
- Пересечение;
- Вычитание;
- Декартово произведение;
- Выборка;
- Проекция;
- Соединение;
- Деление.

- Совокупность всех операций над отношениями позволяет извлечь из базы данных любую интересующую информацию и сформировать ее в виде отношения (таблицы) с наперед заданными свойствами (заголовком).

- Основная идея реляционной алгебры состоит в том, что если отношения являются множествами, то средства манипулирования отношениями могут базироваться на традиционных операциях
- В реляционной алгебре отношения состоят из тела отношения и заголовка отношения. Отношения, имеющие одинаковые заголовки называют совместимыми по типу.

Отношение A

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

Отношение B

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

Объединение, пересечение

- **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.

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000
2	Пушников	2500
4	Сидоров	3000

- **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.

Табельный номер	Фамилия	Зарплата
1	Иванов	1000

Разность , Произведение

- **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 кортежей, принадлежащих отношению A и не принадлежащих отношению B.

Табельный номер	Фамилия	Зарплата
2	Петров	2000
3	Сидоров	3000

- Декартово произведение отношений можно применять к отношениям заголовки которых не содержат одноименных атрибутов, принадлежащих одному домену. Результатом декартова произведение отношения A с заголовком (A1, A2, …, Am) и от- ношения B с заголовком (B1, B2, …, Bn) является отношение заголовок которого (A1, A2, …, Am, B1, B2, …, Bn), является соединением (конкатенацией) заголовка (A1, A2, …, Am) отношения A и заголовка (B1, B2, …, Bn) отношения B, а тело состоит из кортежей (a1, a2, …, am, b1, b2, …, bn) ,где (a1, a2, …, am) $\in$ A, (b1, b2, …, bn) $\in$ B.

Номер поставщика	Наименование поставщика
1	Иванов
2	Петров
3	Сидоров

Отношение A (Поставщики)

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

Отношение B (Детали)

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

Декартово произведение отношений A и B

- Операция **проекции** позволяет получать отношения, состоящие из части элементов исходных отношений, ограничивая набор используемых доменов. **Выборка** или селекция позволяет получать отношения, содержащие только те кортежи, поля которых удовлетворяют условиям выборки.

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000

Отношение A WHERE Зарплата<3000

Город поставщика
Уфа
Москва
Челябинск

Отношение A[Город поставщика]

- Результатом применения операции **соединения** отношений A и B по условию, заданному логическим выражением p, является отношение R, заголовок которого совпадает с заголовком декартова произведения отношений A и B, а тело содержит множество корте- жей t, таких, что кортеж t принадлежит декартову произведению отношений A и B, и условие p для кортежа t истинно. Операция соединения есть результат последовательного применения операций декартового произведения и выборки.

Деление

- **Делением отношений** A на B называется отношение с заголовком ($X_1,X_2,\ldots,X_n$) и телом, содержащим множество кортежей ($x_1,x_2,\ldots,x_n$), таких, что для всех кортежей ($y_1,y_2,\ldots,y_m$) из B в отношении A найдется кортеж ($x_1,x_2,\ldots,x_n,y_1,y_2,\ldots,y_m$)

Делимое

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

Делитель

Номер детали DNUM
1
2
3

Частное

Номер поставщика PNUM
1

Проблемы проектирования баз данных

Проектирование базы данных – самый трудный и ответственный этап во всем процессе разработки БД. Если проект точен и выверен, работа с БД будет удобной и без конфликтов, необходимость внесение дополнений в БД не потребует кардинальных изменений в остальной программе.

Информационно-логическое (инфологическое) проектирование

- заключается в определении числа и структуры таблиц, определении запросов к БД, типов отчетных документов, разработке алгоритмов обработки информации, разработке вида форм для ввода и редактирования данных.
- Решение задач инфологического проектирования БД определяется спецификой задач предметной области и наиболее важной здесь является проблема структуризации данных.

Структурный анализ предметной области

это начало проектирования БД. В результате определяется вся совокупность данных разрабатываемой базы данных. Одни и те же данные могут группироваться в таблицы (отношения) различными способами. Группировка атрибутов в отношениях должна быть такой рациональной, чтобы в таблицах были сведены к минимуму или отсутствовали:

- избыточность данных;
- аномалии обновления;
- аномалии удаления;
- аномалии ввода.

Избыточность данных (дублирование)

проявляется в том, что в нескольких записях таблицы базы данных повторяется одна и та же информация. Различают простое (неизбыточное) дублирование, и избыточное дублирование.

Например, имеем отношение Сотрудник

Табельный_№	ФИО	ГодРождения	Пол	Отдел	Должность	ФИО_ребенка

Для сотрудников, у которых есть дети, их личные данные будут повторяться столько раз, сколько у него детей. Это пример избыточного дублирования. В качестве примеров простого (неизбыточного) дублирования могут быть значения реквизитов Отдел, Пол, Должность. Если в отделе работает 25 сотрудников, то в 25-ти записях номер отдела будет повторяться. Аналогичная ситуация с простым дублированием информации в полях Пол и Должность.

Аномалии

- **Аномалиями** называют противоречия в БД либо существенные сложности в обработке данных, вызванные состоянием структуры таблиц базы данных.
- **Аномалии обновления** проявляются в том, что изменение значения одних данных может повлечь за собой просмотр всей таблицы и соответствующее изменение некоторых других записей таблицы. Аномалии обновления тесно связаны с избыточностью данных. Предположим, что в отношении есть поля Номер рабочей комнаты и Номер рабочего телефона. Если в одной рабочей комнате изменился номер телефона, то необходимо внести изменения во все записи сотрудников, работающих в этой комнате. Если же исправление будет внесено не во все записи, то возникнет несоответствие информации, которое и

Аномалии удаления

состоят в том, что при удалении каких-либо данных из таблицы может пропасть и другая информация, которая не связана напрямую с удаляемым данным. Например, в фирме на одной из должностей работали только один или два сотрудника, которые увольняются. Тогда удаление записей об этих сотрудниках приведет к потере информации о должности, которую они занимали.

Аномалии добавления

- возникают в случаях, когда информацию в таблицу нельзя поместить до тех пор, пока она неполная, либо вставка новой записи требует дополнительного просмотра таблицы. В таблице, рассматриваемой в качестве примера, имеется поле «Рейтинг», в котором содержится информация об уровне квалификации сотрудника, устанавливаемом по результатам его работы. При приеме на работу нового сотрудника установить уровень его квалификации невозможно, так он еще не выполнял никаких работ в организации. Если для этого поля задать ограничение NOT NULL, то в таблицу нельзя будет ввести информацию о новом сотруднике. Это и есть пример аномалии ввода.
- Чтобы свести к минимуму возможность появления такого рода аномалий, и используется нормализация

- **Нормализация отношений** — формальный аппарат ограничений на формирование отношений (таблиц), который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудозатраты на ведение (ввод, корректировку) базы данных.
- ***Нормализация*** *— это разбиение таблицы на две или более, обладающих лучшими свойствами при добавлении, изменении и удалении данных.*

Американским математиком Е.Коддом выделены три *нормальные формы отношений. (1НФ, 2НФ, 3НФ)*

Основные свойства нормальных форм состоят в следующем:

1) каждая следующая нормальная форма в некотором смысле лучше предыдущей нормальной формы;

2) при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

В основе процесса проектирования лежит метод нормализации, т. е. декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, которые удовлетворяют требованиям следующей нормальной формы

Первая нормальная форма

- Отношение находится в **первой нормальной форме** тогда и только тогда, когда на пересечении каждого столба и каждой строки находятся только элементарные значения атрибутов.

Любое нормализованное отношение находится в 1НФ.

***Первая нормальная форма** (**1НФ**) - это обычное отношение. Свойства отношений (это и будут свойства 1НФ):*

- *В отношении нет одинаковых кортежей.*
- *Кортежи не упорядочены.*
- *Атрибуты не упорядочены и различаются по наименованию.*
- *Все значения атрибутов атомарны.*

Преобразование отношения к первой нормальной форме может привести к увеличению количества реквизитов (полей) отношения (таблицы) и изменению ключа.

Отношение Сотрудники.

№	Фамилия	Должность	Оклад	Дети
1.	Иванов	Директор	100	Оля 1990 Маша 1992
2.	Петров	Гл. инженер	100	Сережа 1989 Катя 1991 Коля 1995

Нормализованная база.

№	Фамилия	Должность	Оклад	Имя	Год рождения
1	Иванов	Директор	100	Оля	1990
1.	Иванов	Директор	100	Маша	1992
2.	Петров	Гл. инженер	100	Сережа	1989
2.	Петров	Гл. инженер	100	Катя	1991
2.	Петров	Гл. инженер	100	Коля	1995

Вторая нормальная форма

Эта форма применяется к таблицам с составными ключами. Таблица, у которой первичный ключ включает только одно поле, всегда находится во 2НФ.
Будем считать атрибут отношения ключевым, если он является элементом какого-либо ключа отношения.

Первичный ключ не должен иметь дополнительных атрибутов. Это значит, что если из первичного ключа исключить произвольный атрибут, оставшихся атрибутов будет недостаточно для однозначной идентификации отдельных кортежей. В противном случае атрибут будет считаться неключевым атрибутом.

Первичный ключ в отношении R1{код, товар}

Статус поставщика определяется его месторасположением.

Отношение R1:

Код	Статус	Город	**Товар**	Кол-во
1	20	Москва	1	300
1	20	Москва	2	200
1	20	Москва	3	400
1	20	Москва	4	200
1	20	Москва	5	100
1	20	Москва	6	100
2	10	Ростов	1	300
2	10	Ростов	2	400
3	10	Ростов	2	200
4	20	Москва	2	200

Данное отношение обладает избыточностью (для каждого поставщика указан город и статус). Избыточность приводит к различным аномалиям обновления:

- аномалия – вставки INSERT. Нельзя добавить информацию о поставщике, который не поставил ни одного товара.
- аномалия – удаления DELETE. Возможно, что с удалением некоторой строки таблица (удаление поставки) исчезнет информация о поставщике.
- аномалия UPDATE (переписать, обновить) – эта проблема возникает в том случае, если необходимо переместить поставщика из одного города в другой. Например, 1 из Москвы в Новгород. Необходимо откорректировать все записи о поставках от этого поставщика.

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы. Определены два вида таких зависимостей: функциональные и многозначные.

- *Функциональная зависимость.*

Поле В таблицы функционально зависит от поля А той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля А обязательно существует только одно из различных значений поля В. Допускается, что поля А и В могут быть составными.

- Другими словами, в отношении R атрибут Y функционально зависит от атрибута X в том и только в том случае, если каждому значению X соответствует одно значение Y.

  R.X -> R.Y

Функциональные зависимости в R1

{код, товар}⟶ {количество};

{код} ⟶ {город};

{код} ⟶ {статус};

{город} ⟶ {статус}

- В случае составного ключа вводится понятие *функционально полной* зависимости.
- **Функционально полная зависимость** неключевых атрибутов заключается в том, что каждый неключевой атрибут функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.
- Отношение будет находиться **во второй нормальной форме**, если оно находится в первой нормальной форме, и каждый неключевой атрибут функционально полно зависит от составного ключа.

Для решения проблемы аномалий заменим отношение R1 несколькими проекциями. В одно включим первичный ключ и все неключевые атрибуты, зависимые от первичного ключа. В остальные проекции включим неключевые атрибуты, зависимые от части первичного ключа и ту часть первичного ключа, от которой данные атрибуты зависят. Итак, получим 2 отношения: R2 и R3.

R2:

Код	**Товар**	Кол-во
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

R3:

Код	Статус	Город
1	20	Москва
2	10	Ростов
3	10	Ростов
4	20	Москва
5	30	Новгород

ФЗ для отношения R2:{код, товар}→{кол-во}

ФЗ для отношения R3:{код}→{город}, {код}→{статус}, {город}→{статус}

- Такие отношения позволяют преодолеть указанные противоречия:

INSERT: Можно вставить поставщика из Новгорода, который не поставлял товар;

DELETE: Можно удалить товар 2 от поставщика 3, а сведения о поставщике останутся;

UPDATE: Для того, чтобы переместить поставщика 1 из Москвы в Новгород, достаточно поменять запись в отношении R3.

- Физический смысл противоречия в отношении R1 в том, что это отношение описывает не один объект (поставку товара) а два: поставку и поставщика.

- Проблемы, возникающие в R3.
- INSERT: Нельзя включить город с некоторым статусом, из которого нет ни одного поставщика.
- DELETE: удалив поставщика 5, удалим информацию о том, что Новгороду был установлен статус 30.
- UPDATE: информация о статусе повторяется, т. о. изменив статус Москвы с 20 на 30 необходимо откорректировать несколько записей.
- Физический смысл противоречия тот же: информация о двух объектах предметной области (город и поставщик) находится в одном отношении

Для того чтобы привести отношение ко 2НФ, нужно:

1. построить его проекцию, исключив атрибуты, которые не находятся

в функционально полной зависимости от составного ключа;

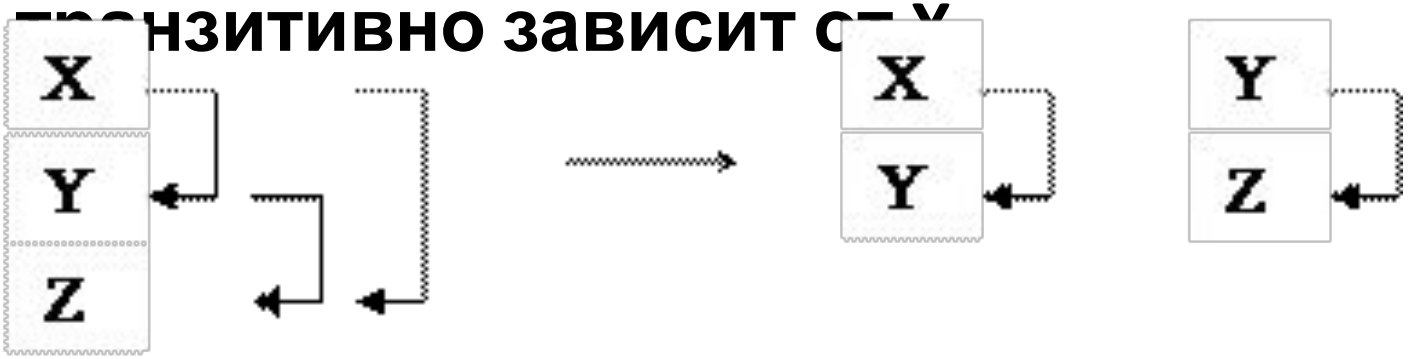
2. построить дополнительные проекции на часть составного ключа и

атрибуты, функционально зависящие от этой части ключа.

Третья нормальная форма

Понятие третьей нормальной формы основывается на понятии *нетранзитивной* зависимости.

- Пусть X, Y и Z – три атрибута. Если Z зависит от Y, а Y – от X, то Z зависит от X. Если при этом обратное соответствие неоднозначно, т.е. X не зависит от Y или Y не зависит от Z, то говорят, что **Z транзитивно зависит от X**



- Отношение будет находиться в **третьей нормальной форме**, если оно находится во второй нормальной форме, и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.
- Формальным признаком проблем в R3 является наличие транзитивной ФЗ. Для этого отношения ФЗ:
- {код} → {город} и {город} → {статус}.

Для решения проблемы найдем от R3 проекции, в которые включим первичный ключ и атрибут, через который осуществляется транзитивная зависимость. А во второе отношении, этот же атрибут и атрибут, транзитивно зависящий от первичного ключа исходного отношения.

R5:

Код	Город
1	Москва
2	Ростов
3	Ростов
4	Москва
5	Новгород

R6:

Город	Статус
Москва	20
Ростов	10
Новгород	30
Казань	40

ФЗ для отношения R5: {код} →{город}

ФЗ для отношения R6: {город}→ {статус}

Каждое отношение описывает только одну сущность (объект предметной области).

Для того чтобы привести отношение к 3НФ, нужно:

1. построить его проекцию, исключив атрибуты, которые находятся

в транзитивной зависимости от других атрибутов;

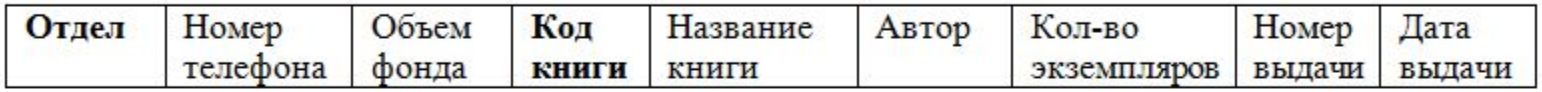
2. построить дополнительные проекции на атрибуты, находящиеся в транзитивной зависимости.

Не транзитивная зависимость означает, что все неключевые атрибуты взаимно независимы. Отношение, находящееся в 2НФ, можно получить из отношения находящегося в 3НФ.

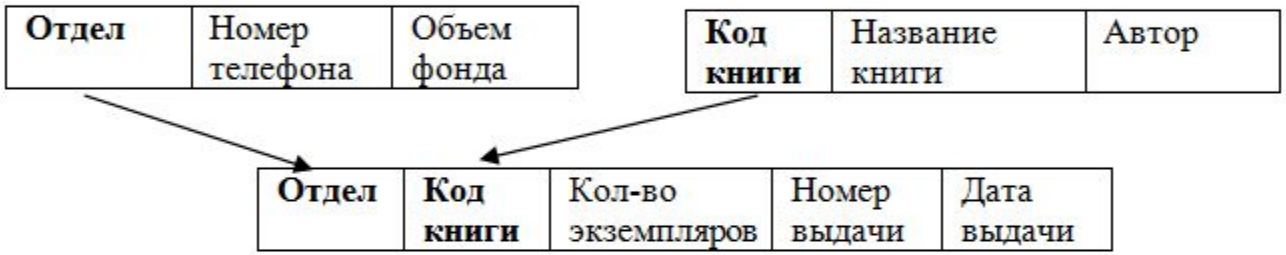
Уровень нормализации данного отношения определяется семантикой, а не конкретными значениями данных в некоторый момент времени. Нельзя с первого взгляда на таблицу с данными для некоторого отношения определить, находится ли оно, например в 3НФ. Для этого также необходимо проанализировать существующие ФЗ. Даже при наличии всех ФЗ можно только высказать предположение, что данные не противоречат гипотезе о принадлежности отношения к 3НФ. Однако этот факт гарантирует, что предложенная гипотеза верна.

Пример

1НФ



2НФ



3НФ