Системы баз данных

Иванюкович
Владимир Александрович

Содержание курса

- Концепция баз данных и СУБД
- Модели данных (иерархическая, сетевая, реляционная)
- Типы связей
- Ключи и целостность
- Реляционная алгебра
- Основы SQL
- Проектирование реляционных баз данных

Рекомендуемая литература

- К.Дж. Дейт. Введение в системы баз данных. Восьмое издание. – М.: Вильямс, 2005. – 1328 С.)
- В. Иванюкович. Системы баз данных. Вводный курс. Учебное пособие для студентов специальности 1-40 01 02. – Мн., МГЭУ им. А.Д. Сахарова, 2010. – 193 С.

Концепция баз данных

- Файлы, содержащие описание структур хранения данных и сведения о данных и находящиеся под управлением СУБД, называются базами данных.
- СУБД – это программный продукт, предназначенный для создания структур хранения данных, а также для ввода, хранения и обработки самих данных.

Создание структуры хранения данных

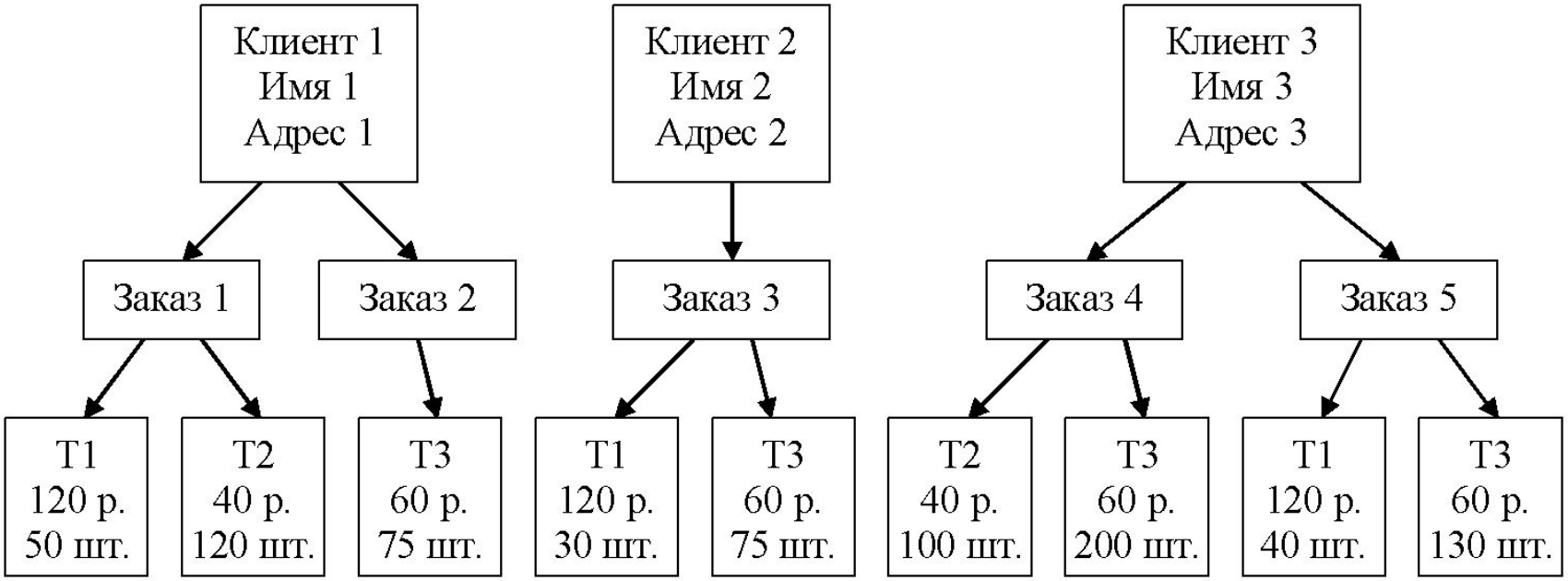
- СОЗДАТЬ ТАБЛИЦУ Расписание
  (Номер_Рейса Целое,
  Дни_Недели Текст (8),
  Пункт_Отправления Текст (24),
  Время_Вылета Время,
  Пункт_Назначения Текст (24),
  Время_Прибытия Время,
  Тип_Самолета Текст (8),
  Стоимость_Билета Денежный);

Поиск и обработка данных

- ВЫБРАТЬ Номер_Рейса, Дни_Недели,
  Время_Вылета
  ИЗ ТАБЛИЦЫ Расписание
  ГДЕ Пункт_Отправления = 'Москва'
  И Пункт_Назначения = 'Минск'
  И Время_Вылета > ‘17’;

- ВЫБРАТЬ КОЛИЧЕСТВО (Номер_Рейса)
  ИЗ ТАБЛИЦЫ Расписание
  ГДЕ Пункт_Отправления = 'Москва'
  И Пункт_Назначения = 'Минск';

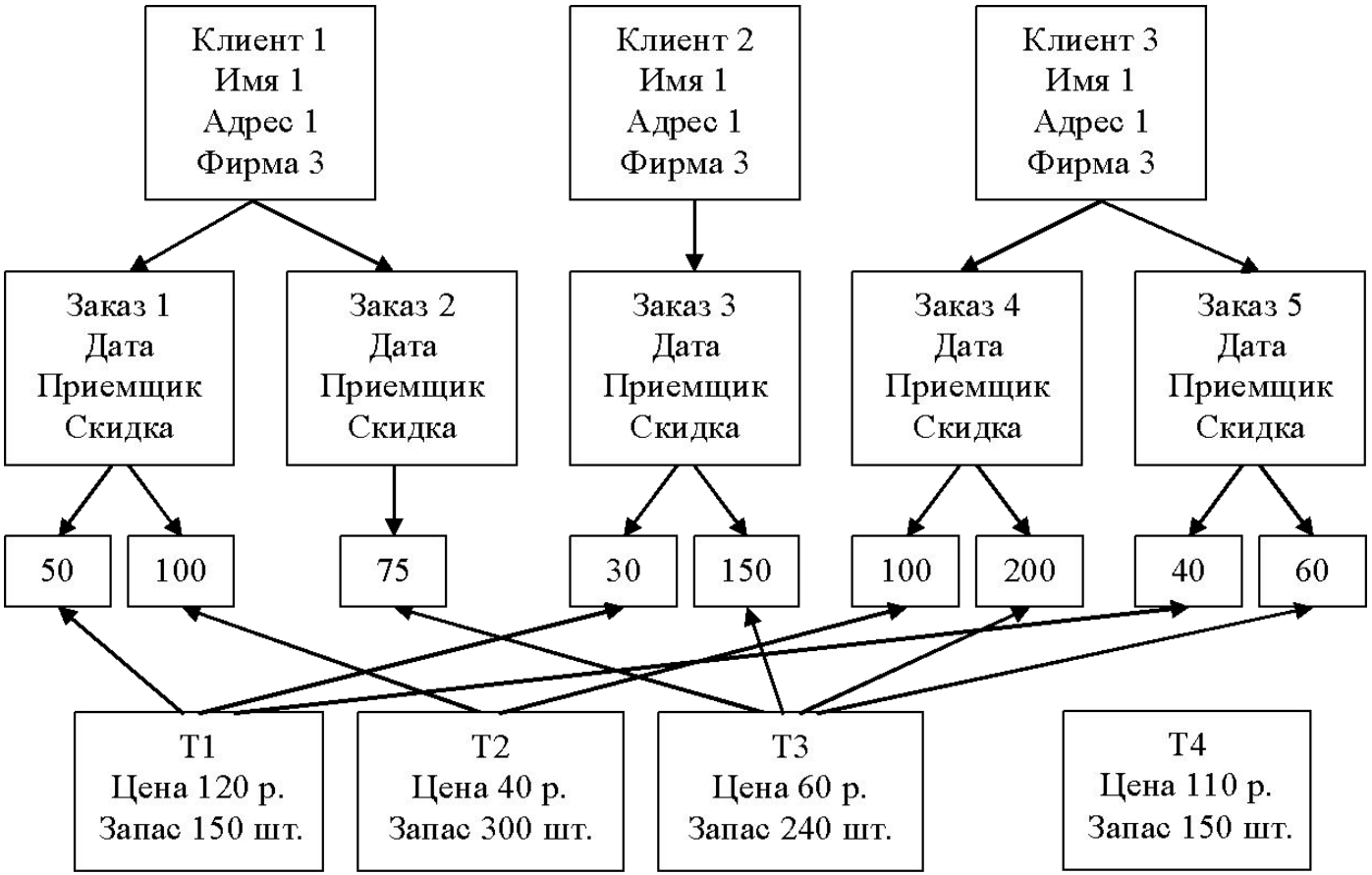
Иерархическая модель данных



Недостатки:

1. невозможно избежать дублирования информации о товарах;
2. нельзя ввести информацию о товарах, на которые нет заказов;
3. нельзя вести учет запаса товаров на складе.

Сетевая модель данных



Недостатки:

1. базы данных сложны и их сложность возрастает при увеличении количества сущностей или атрибутов;
2. новые манипуляции с данными потребуют создания новых связей.

Реляционная модель

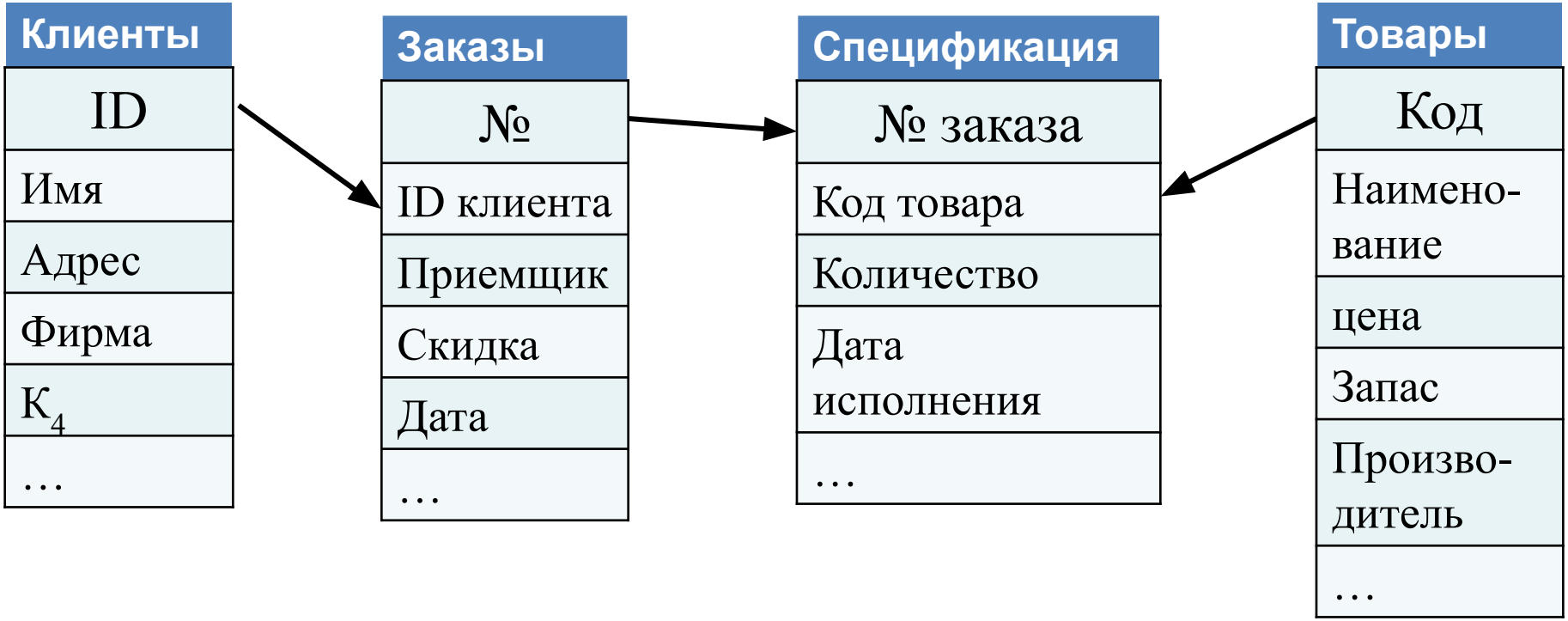
Клиенты			
ID	Имя	Адрес	Фирма
$К_1$	Андрей	Минск	ИНКО
$К_2$	Анна	Гродно	Тролль
$К_3$	Дмитрий	Брест	Нейрон
$К_4$	Петр	Пинск	Антей
…			

Товары					
Код	Наименование	Цена	Запас	Производитель	…
$Т_1$	Шнур	120	150		
$Т_2$	Розетка	40	300		
$Т_3$	Пила	60	240		
$Т_4$	Щепцы	110	150		
…					

Заказы					
№	ID клиента	Приемщик	Скидка	Дата	…
1	$К_1$	Маша	5%	21.01.2007	…
2	$К_1$	Илья	0	22.01.2007	
3	$К_2$	Надежда	7%	22.01.2007	
4	$К_3$	Анна	10%	25.01.2007	
5	$К_3$	Виктор	5%	26.01.2007	
…					

Спецификация заказа				
№ заказа	Код товара	Количество	Дата исполн.	…
1	$Т_1$	50		
1	$Т_2$	100		
2	$Т_3$	75		
3	$Т_1$	30		
3	$Т_3$	150		
4	$Т_2$	100		
4	$Т_3$	200		
5	$Т_1$	40		
5	$Т_3$	130		

Реляционная модель



Реляционная модель (relation – отношения)

- данные на концептуальном уровне представляются в виде двумерных таблиц, в которых хранится информация о сущностях;
- строки называются кортежами или записями и содержат информацию об экземплярах сущности;
- столбцы называются полями и содержат информацию об атрибутах сущности.
- реляционные системы представляют более простую среду для разработки баз данных, чем иерархические или сетевые;
- программное обеспечение, основанное на реляционной алгебре, позволяет организовать практически любое манипулирование данными.

Идея реляционной модели данных

Строки таблицы с n колонками, состоящими из элементов множеств A1, A2, …, An, можно представить как подмножество в прямом произведении A1×A2×… ×An. Строки образуют список из n элементов, по одному из каждого множества Ai, а вся таблица представляет собой n-арное отношение. Например, таблицу КЛИЕНТЫ можно рассматривать как подмножество множества A1×A2×A3×A4, где A1 – множество кодов клиентов, A2 – множество имен клиентов, A3 – множество их адресов, An – множество названий организаций. Один из элементов этого отношения – строка К1, Андрей, Минск, ИНКО.

Представленные таким образом таблицы можно обрабатывать, используя алгебру отношений на множествах.

Связь «один-к-одному» (1:1): В каждый момент времени каждому представителю сущности А соответствует 1 или 0 представителей сущности В, а каждому представителю сущности В соответствует 1 или 0 представителей сущности А.

Код студента (номер зачетной книжки)	Фамилия
0125	Ильин
0134	Петров

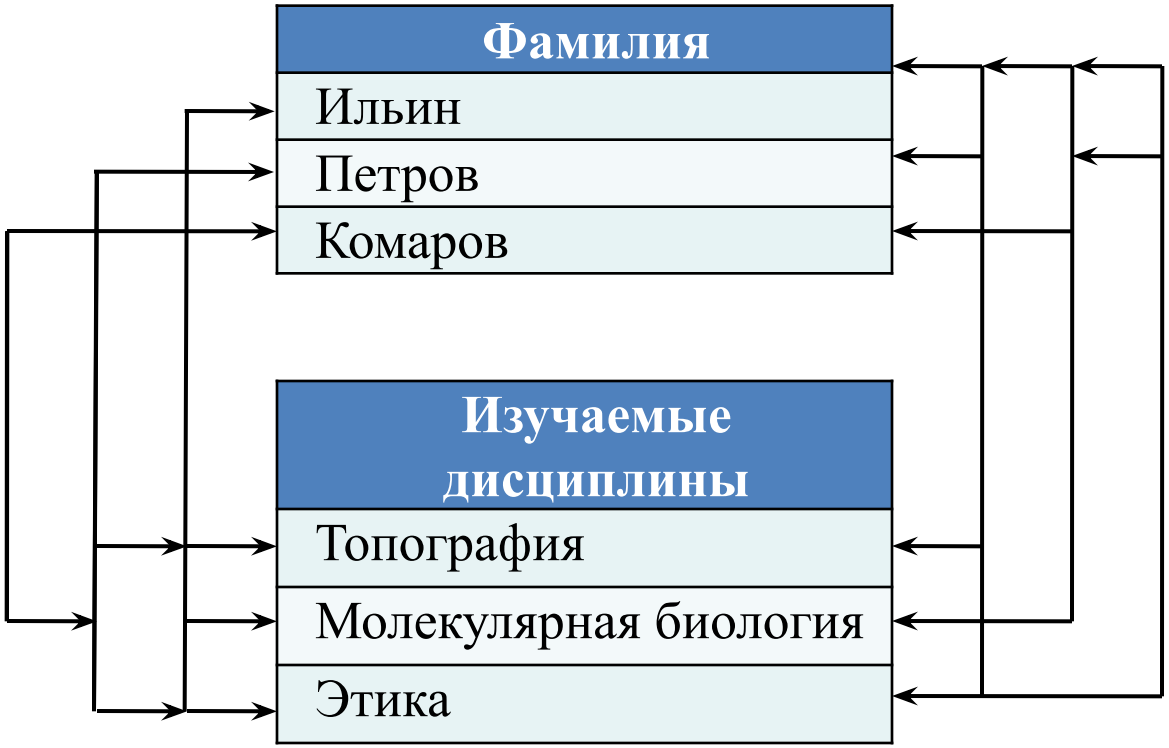
Код студента	Личное дело
0134	№1
0125	№2

Связь «один-ко-многим» (1:M): одному представителю сущности A соответствуют 0, 1 или несколько представителей сущности B, а любому представителю сущности B соответствует 1 или 0 представителей сущности A.

Код студента	Фамилия	Номер группы
0125	Ильин	1
0134	Петров	1
0086	Комаров	2

Номер группы	Кафедра
1	Экологических информационных систем
2	Ядерной и радиационной безопасности

Связь «многие-ко-многим» (N:M): каждому представителю сущности А может соответствовать множество представителей сущности В, а каждому представителю сущности В может соответствовать множество представителей сущности А.



Ключи и целостность реляционных данных (1)

Целостность (integrity – неприкосновенность, сохранность, целостность) – правильность данных в любой момент времени.

Поддержание целостности базы данных – защита данных от неверных изменений или разрушений.

Основные механизмы обеспечения целостности данных связаны с понятием первичных и внешних ключей.

Ключи и целостность данных (2)

Потенциальные ключи

Потенциальный ключ K для некоторого отношения R – это подмножество множества атрибутов R, обладающее следующими свойствами:

1. Свойством уникальности (нет двух различных кортежей в отношении R с одинаковым значением K).
2. Свойством неизбыточности (никакое из подмножеств K не обладает свойством уникальности).

Ключи и целостность данных (3)
Потенциальные ключи

Первичный и альтернативные ключи являются частным случаем потенциального ключа.

Надмножество потенциального ключа называется суперключом (например, суперключом для отношения Проекты является множество атрибутов {Пр№, Имя_проекта}).

Суперключ обладает свойством уникальности, но не обязательно обладает свойством несократимости.

Потенциальный ключ – это частный случай суперключа .

Ключи и целостность данных (4)

Целостность объектов

Правило *целостности объектов:*

Ни один элемент первичного ключа базового отношения не может быть Null-значением.

Если кортеж имеет Null-значение некоторого атрибута, то это означает, что в таком кортеже значение атрибута по какой-то причине отсутствует.

Ключи и целостность данных (5)

Внешние ключи

Основное назначение внешних ключей – организация связей между отношениями. Связь создается по данным, хранящимся в поле первичного ключа одной таблицы и в поле внешнего ключа другой таблицы с данными, такими же по смыслу и типу.

Условия необходимости выбора внешних ключей:

1. Если сущность С связывает сущности А и В, то она должна включать внешние ключи, соответствующие первичным ключам сущностей А и В.
2. Если сущность В обозначает сущность А, то она должна включать внешний ключ, соответствующий первичному ключу сущности А.

Ключи и целостность данных (6)

Целостность по ссылкам

Целостность по ссылкам:

База данных не должна содержать несогласованных значений внешних ключей.

Т.е., если В ссылается на А, то А должно существовать.

Правило целостности по ссылкам позволяет поддерживать базу данных в корректном состоянии.

Ключи и целостность данных (8)

Внешние ключи

Правило внешних ключей предполагает принятие решения:

1. Что должно случиться при попытке удалить объект ссылки внешнего ключа?
2. Что должно случиться при попытке обновить потенциальный ключ, на который ссылается внешний ключ?

Существует две возможности:

- ограничение – «ограничить» операции до момента появления первой ссылки;
- каскадирование – «каскадировать» операции, удаляя или обновляя все соответствующие атрибуты.

Ключи и целостность данных (9)
Внешние ключи

Реляционная модель допускает появление Null-значений среди атрибутов внешних ключей!

Определение внешнего ключа:

Внешний ключ FK в отношении R2 – это подмножество множества атрибутов R2 такое, что существует базовое отношение R1 с потенциальным ключом CK, для которого каждое значение FK в текущем значении R2 или является Null-значением, или совпадает со значением CK некоторого кортежа в текущем значении R1.

Ключи и целостность данных (10)

Целостность атрибута

Значение каждого атрибута берется из соответствующего домена.

Типы данных
Категории

- Character string – Строки символов;
- Bit string – Строки битов;
- Exact numeric – Рациональные (целые и действительные) числа с плавающей десятичной точкой;
- Approximate numeric – Вещественные числа (с плавающей точкой);
- Date time – значения даты и времени;
- Interval – интервалы даты и времени.

Строковые типы данных

- Character (n) – строка фиксированной длины n. Если символов меньше чем n, то добавляются пробелы. Синонимы – Char(n).
- Character varying (n) – строка переменной длины, длинной менее n. Синонимы: Char varying, Charvar.
- National Character (National Char, NChar) – совпадает с типом Char, только хранит лишь стандартизованные многобайтовые или двухбайтовые знаки (Unicode). National Character Varying – то же для строк переменной длины.
- Unicode – единое множество 16-разрядных чисел, которое представляет знаки почти всех мировых языков. Содержит $65536 = 2^{16}$ знаков.

В СУБД Access к строковым типам данных относятся: text и memo.

Битовые типы данных

- BIT (n) – строка фиксированной длины (фиксированные числа битов). Max длина определяется СУБД. Если длина строки меньше n, то получите сообщение об ошибке. В стоке BIT перед первой кавычкой должна стоять латинская B, например, B'01001' – это строка типа BIT(5). Bit varying – аналогично, как Charvar.
- Тип данных BIT используется для хранения так называемых больших бинарных объектов (Binary Large Object – BLOB) – например, звук, изображение.

В СУБД Access к BIT типу данных относятся: YES, NO, BINARY, OLE OBJECT.

Точные числовые типы данных

Точность – число значащих цифр в записи числа;

Масштаб – число цифр справа от десятичной точки (масштаб ≤ точности).

Типы:

- Numeric (точность [,масштаб]) – представляет произвольное рациональное число.
- Decimal – аналогичен NUMERIC, но только задает нижнюю границу точности, т.е. СУБД может выбрать большую точность, чем заказано пользователем.
- Integer (или INT) – представляет произвольное целое число.
- SMALLINT – повторяет INT, только интервал допустимых значений уже.

В СУБД Access: DECIMAL, INTEGER, BYTE, LONG INTEGER.

Пример:
Хранение числа 123,55

Спецификация столбца	**Хранится значение**
Numeric (5)	124
Numeric (5.0)	124
Numeric (5.1)	123,6
Numeric (5.2)	123,55
Numeric (4.0)	124
Numeric (4.1)	123,6
Numeric (4.2)	Выходит за пределы точности

Вещественные числовые типы данных

Числа с плавающей точкой применяются для хранения приближенных числовых значений.

- FLOAT (точность) – представляет произвольное рациональное приближение действительного числа с плавающей точкой. Значение точности представляется не в количестве значащих десятичных цифр, а в количестве битов. Точность не должна быть меньше 1.
  Для преобразования десятеричной точности в бинарную надо умножить десятеричную точность на 3.32193.
  Например, 7 знаков точности дают 24 бита.
- REAL – совпадает с FLOAT, но точность вводить не надо, ее автоматически определяет СУБД. Числа типа REAL называют числами одинарной точности с плавающей точкой.
- DOUBLE PRECISION –числа двойной точности с плавающей точкой.

В СУБД Access – SINGLE, DOUBLE.

Календарные типы данных

- DATE – имеет формат YYYY-MM-DD.
- TIME – имеет формат HH:MM:SS. Можно добавить аргумент “точность” для долей секунд.
- TIMESTAMP – имеет формат YYYY-MM-DD_ HH:MM:SS.
- Интервальные типы данных.

В СУБД ACCESS: date/time.

Реляционная алгебра

- Замкнутость;
- Правила наследования имен атрибутов;
- Правила наследования потенциальных ключей;
- Совместимость по типу: Два отношения совместимы по типу, если каждое из них имеет одно и то же множество имен атрибутов и соответствующие атрибуты определены на одном и том же домене.

Традиционные реляционные операции (1) Объединение

- *Объединением* двух совместимых по типу отношений A и B (A UNION B) называется отношение с тем же заголовком, как и в отношениях A и B, и с телом, состоящим из множества всех кортежей t, принадлежащих A или B или обоим отношениям. При этом совпадающие кортежи записываются один раз.

Детали1

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов
Д3	Радиометр	Черный	1400	Гродно

Детали2

Д№	Имя_Д	Цв	Вес	Гор
Д3	Радиометр	Черный	1400	Гродно
Д4	Часы	Желтый	140	Минск
Д5	Рулетка	Красный	200	Брест
Д6	Лом	Черный	5000	Варшава

Детали1 UNION Детали2

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов
Д3	Радиометр	Черный	1400	Гродно
Д4	Часы	Желтый	140	Минск
Д5	Рулетка	Красный	200	Брест
Д6	Лом	Черный	5000	Варшава

Традиционные реляционные операции (2) Пересечение

- *Пересечением* дух совместимых по типу отношений A и B (A INTERSECT B) называется отношение с тем же заголовком, как и в отношениях A и B, и с телом, состоящим из множества всех кортежей t, которые принадлежат одновременно обоим отношениям A и B.

Детали1

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов
Д3	Радиометр	Черный	1400	Гродно

Детали2

Д№	Имя_Д	Цв	Вес	Гор
Д3	Радиометр	Черный	1400	Гродно
Д4	Часы	Желтый	140	Минск
Д5	Рулетка	Красный	200	Брест
Д6	Лом	Черный	5000	Варшава

Детали1 INTERSECT Детали2

Д№	Имя_Д	Цв	Вес	Гор
Д3	Радиометр	Черный	1400	Гродно

Традиционные реляционные операции (3) Вычитание

- *Вычитанием* двух совместимых по типу отношений A и B (A MINUS B) называется отношение с тем же заголовком, как и в отношениях A и B, и с телом, состоящим из множества всех кортежей t, принадлежащих отношению A и не принадлежащих отношению B.

Детали1

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов
Д3	Радиометр	Черный	1400	Гродно

Детали2

Д№	Имя_Д	Цв	Вес	Гор
Д3	Радиометр	Черный	1400	Гродно
Д4	Часы	Желтый	140	Минск
Д5	Рулетка	Красный	200	Брест
Д6	Лом	Черный	5000	Варшава

Детали1 MINUS Детали2

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов

Традиционные реляционные операции (4) Произведение

- Декартово произведение двух отношений A и B (A TIMES B), где A и B не имеют общих имен атрибутов, определяется как отношение с заголовком, который представляет собой сцепление (конкатенацию) двух заголовков исходных отношений A и B, и телом, состоящим из множества всех кортежей t таких, что t представляет собой сцепление кортежа a, принадлежащего отношению A, и кортежа b, принадлежащего отношению B. Кардинальное число результата равняется произведению кардинальных чисел исходных отношений, а степень равняется сумме их степеней.

Детали

Д№	Имя_Д
Д1	Тестер
Д2	Дозиметр
Д3	Радиометр
Д4	Часы

Проекты

Пр№	Имя_П
Пр1	Спутник
Пр2	Корунд
Пр3	Лес

Детали TIMES Проекты

Д№	Имя_Д	Пр№	Имя_П
Д1	Тестер	Пр1	Спутник
Д1	Тестер	Пр2	Корунд
Д1	Тестер	Пр3	Лес
Д2	Дозиметр	Пр1	Спутник
Д2	Дозиметр	Пр2	Корунд
Д2	Дозиметр	Пр3	Лес
Д3	Радиометр	Пр1	Спутник
Д3	Радиометр	Пр2	Корунд
Д3	Радиометр	Пр3	Лес
Д3	Радиометр	Пр1	Спутник
Д3	Радиометр	Пр2	Корунд
Д3	Радиометр	Пр3	Лес

Специальные реляционные операции (1)
Выборка

- *Выборка* (RESTRICT или SELECT) – это сокращенное название θ-выборки, где θ обозначает любой скалярный оператор сравнения (=, ≠, ≥, > и т. д.). θ-выборкой из отношения A по атрибутам X и Y (A WHERE X θ Y) (порядок учитывается!) называется отношение, имеющее тот же заголовок, что и отношение A, и тело, содержащее множество всех кортежей t отношения A, для которых проверка условия «X θ Y» дает значение истина. Атрибуты X и Y должны быть определены на одном и том же домене, а оператор сравнения θ должен иметь смысл для данного домена.

Детали				
Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов
Д3	Радиометр	Черный	1400	Гродно
Д4	Часы	Желтый	140	Минск
Д5	Рулетка	Красный	200	Брест
Д6	Лом	Черный	5000	Варшава

Детали WHERE Гор='Минск'

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д4	Часы	Желтый	140	Минск

Специальные реляционные операции (2) Проекции

Проекцией (PROJECT) 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 кортежей с атрибутами, совпадающими с соответствующими атрибутами отношения A.

Т.е., с помощью операции проекции получается вертикальное подмножество исходного отношения

ДеталиX

Д№	Имя_Д	Цв	Вес	Гор
Д1	Тестер	Черный	250	Минск
Д2	Дозиметр	Серый	700	Борисов
Д3	Радиометр	Черный	1400	Гродно
Д4	Часы	Желтый	140	Минск
Д5	Рулетка	Красный	200	Брест
Д6	Тестер	Черный	5000	Варшава

ДеталиX [Имя_Д, Цв]

Имя_Д	Цв
Тестер	Черный
Дозиметр	Серый
Радиометр	Черный
Часы	Желтый
Рулетка	Красный

Специальные реляционные операции (3)

- *Соединение* (JOIN) – это разновидность операции произведения, в которой сцепление кортежей основывается на задаваемом атрибуте или наборе атрибутов каждого из двух отношений. Значения указанных атрибутов сравниваются с целью наложения определенных ограничений на результат. Наиболее часто используется естественное или внутреннее соединение, когда отношения имеют общий атрибут и результат содержит только строки, в которых значения общего атрибута совпадают.

Естественное (или внутреннее) соединение (4)

Естественным (или внутренним) соединением отношений A и B (A JOIN B) с заголовками X,Y и Y,Z соответственно и с атрибутами Y, определенными на одном и том же домене, называется отношение с заголовком {X,Y,Z} и телом, содержащим множество кортежей с атрибутами, совпадающими с соответствующими атрибутами отношений A и B.

ПоставщикиX

П№	Имя_П	Статус	Гор
П1	Волк	20	Брест
П2	Заяц	10	Минск
П3	Лев	30	Гродно
П4	Лиса	20	Минск
П5	Бык	30	Брест

ПоставкиX

П№	Д№	Пр№	Кол
П1	Д1	Пр1	200
П1	Д1	Пр4	700
П2	Д3	Пр1	400
П2	Д3	Пр2	200
П2	Д3	ПР3	200

ПоставщикиX JOIN ПоставкиX

П№	Имя_П	Статус	Гор	Д№	Пр№	Кол
П1	Волк	20	Брест	Д1	Пр1	200
П1	Волк	20	Брест	Д1	Пр4	700
П2	Заяц	10	Минск	Д3	Пр1	400
П2	Заяц	10	Минск	Д3	Пр2	200
П2	Заяц	10	Минск	Д3	ПР3	200

Специальные реляционные операции (5)
Внешнее соединение

- При внешнем соединении кортеж, который невозможно соединить с кортежем соответствующей таблицы из-за отсутствия совпадающих значений, будет помещен в результирующую таблицу, а для присоединенных атрибутов значения определены не будут, т.е., им присвоят Null-значения.

ПоставщикиX

ПN№	Имя_П	Статус	Гор
П1	Волк	20	Брест
П2	Заяц	10	Минск
П3	Лев	30	Гродно
П4	Лиса	20	Минск
П5	Бык	30	Брест

ПоставкиX

ПN№	ДN№	ПрN№	Кол
П1	Д1	Пр1	200
П1	Д1	Пр4	700
П2	Д3	Пр1	400
П2	Д3	Пр2	200
П2	Д3	ПР3	200

ПоставщикиX LEFT JOIN ПоставкиX

ПN№	Имя_П	Статус	Гор	ДN№	ПрN№	Кол
П1	Волк	20	Брест	Д1	Пр1	200
П1	Волк	20	Брест	Д1	Пр4	700
П2	Заяц	10	Минск	Д3	Пр1	400
П2	Заяц	10	Минск	Д3	Пр2	200
П2	Заяц	10	Минск	Д3	ПР3	200
П3	Лев	30	Гродно			
П4	Лиса	20	Минск			
П5	Бык	30	Брест			

Специальные реляционные операции (6) Соединения

- Отношения можно соединять по атрибутам, имеющим либо общие домены, либо сопоставимые домены, когда значения данных из одного домена можно сопоставить со значениями данных из другого домена.

- Соединение обладает свойствами ассоциативности и коммутативности.

- Если отношения A и B не имеют общих имен атрибутов, то естественное соединение превращается в декартово произведение.

Специальные реляционные операции (7)

Делением (DIVIDED BY) двух отношений, бинарного и унарного, является отношение, содержащее все значения одного атрибута бинарного отношения, которые соответствуют (в другом атрибуте) всем значениям в унарном отношении.

ПN	ДN		ДN		ПN
П1	Д1	DIVIDED BY	Д2	=	П1
П1	Д2		Д4		П4
П1	Д4				
П2	Д1				
П2	Д2		Второй пример:		
П3	Д2		ДN		ПN
П4	Д1	DIVIDED BY	Д1	=	П1
П4	Д2				П2
П4	Д4				П4

Дополнительные реляционные операции

Операция расширения

EXTEND A ADD expr AS Z;

Результат: Отношение с заголовком, эквивалентным заголовку отношения A, расширенному новым атрибутом Z, который рассчитывается скалярным выражением expr для кортежа отношения A.

Операция расширения обеспечивает возможность горизонтального или построчного вычисления.

EXTEND Детали ADD Вес/1000 AS Вес(кг)					
Д№	Имя_Д	Цв	Вес	Гор	Вес(кг)
Д1	Тестер	Черный	250	Минск	0,25
Д2	Дозиметр	Серый	700	Борисов	0,7
Д3	Радиометр	Черный	1400	Гродно	1,4
Д4	Часы	Желтый	140	Минск	0,14
Д5	Рулетка	Красный	200	Брест	0,2
Д6	Лом	Черный	5000	Варшава	5

Пример: Подсчитать количество поставок, сделанных каждым поставщиком.

EXTEND Поставщики ADD COUNT ((Поставки RENAME П№ AS X) WHERE X= №) AS Кол_П;

П№	Имя_П	Статус	Гор	Кол_П
П1	Волк	20	Брест	6
П2	Заяц	10	Минск	2
П3	Лев	30	Гродно	1
П4	Лиса	20	Минск	3
П5	Бык	30	Брест	0

Операция расширения обеспечивает возможность горизонтального или построчного вычисления.

Дополнительные реляционные операции.

Операция подведения итогов

SUMMARIZE A BY (A1,A2,…,An) ADD expr AS Z;

Результат: Отношение с заголовком {A1,A2,…,An,Z} и с телом, содержащим все такие кортежи t, которые являются кортежами проекции отношения A по атрибутам A1,A2,…,An, расширенного значением для нового атрибута Z. Значение Z подсчитывается вычислением итогового значения expr по всем кортежам отношения A.

Пример:

SUMMARIZE Поставки BY(Д№) ADD SUM(Кол) AS Общ_кол

Д№	Общ_кол
Д1	900
Д2	300
Д3	3500
Д4	1300
Д5	1100
Д6	1300

Пример: Подсчитать количество поставок, сделанных каждым поставщиком.

SUMMARIZE Поставки BY (П№) ADD COUNT AS Кол_П;

П№	Кол_П
П1	6
П2	2
П3	1
П4	3

Дополнительные реляционные операции.

Возможные операции:

Переименование имени поля:

Детали RENAME Гор AS Гор_Д

Присвоение:

Поставки := Поставки MINUS (Поставки WHERE Кол = 0);

Обновление:

INSERT (Поставщики WHERE Гор_П=Минск) INTO Temp;
UPDATE (Поставщики WHERE Гор_П=Брест) СТАТУС<40;
DELETE Поставщики WHERE Статус < 20;

Примеры использования реляционной алгебры для выражения словесных запросов в виде формул (1)

Получить имена поставщиков, которые поставляют деталь Д2.

((Поставки JOIN Поставщики) WHERE Д№=’Д2’) [Имя_П];

Примеры использования реляционной алгебры для выражения словесных запросов в виде формул (2)

Получить имена поставщиков, которые поставляют по крайней мере одну черную деталь.

(((Детали WHERE Цв = ‘Черный’) JOIN Поставки) [П№] JOIN Поставщики) [Имя_П];

или

(((Детали WHERE Цв = ‘Черный’) [Д№] JOIN Поставки) JOIN Поставщики) [Имя_П];

Примеры использования реляционной алгебры для выражения словесных запросов в виде формул (3)

Получить имена поставщиков, которые поставляют все детали.

((Поставки [П№,Д№] DIVIDED BY Детали [Д№] JOIN Поставщики) [Имя_П];

Примеры использования реляционной алгебры для выражения словесных запросов в виде формул (4)

Получить номера поставщиков, которые поставляют по крайней мере все те детали, которые поставляет поставщик П2.

Поставки [П№,Д№] DIVIDED BY (Поставки WHERE Имя_П=’П2’) [Д№]

Примеры использования реляционной алгебры и SQL для выражения словесных запросов в виде формул (5)

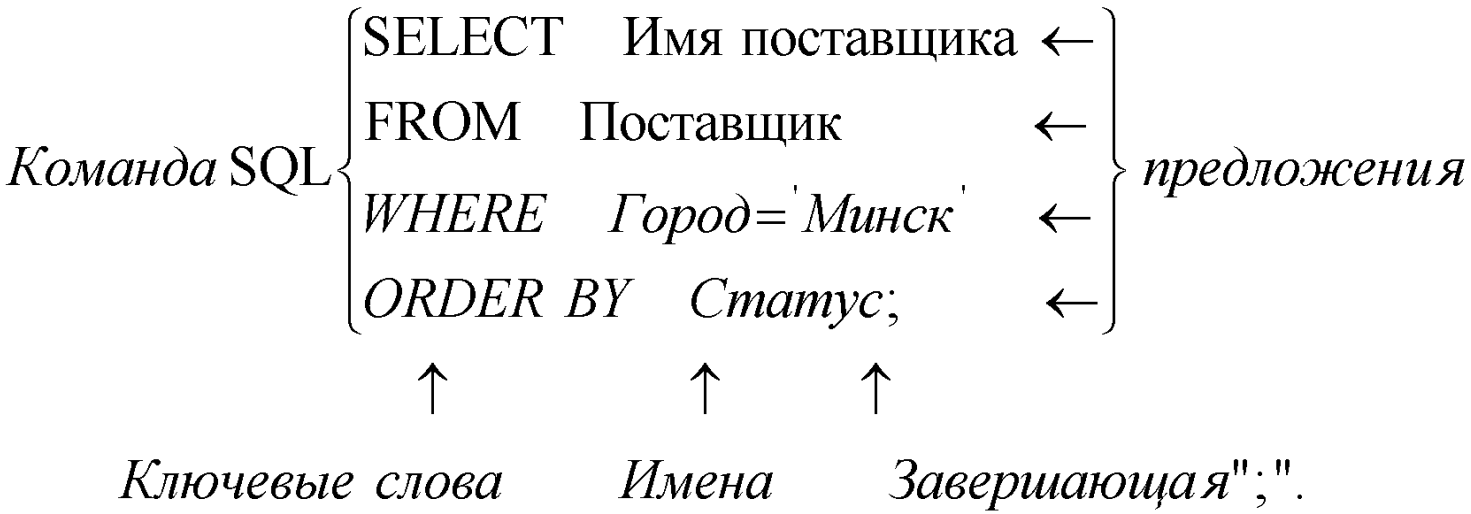
Получить имена поставщиков, которые не поставляют деталь Д2.

((Поставщики [П№] MINUS (Поставки WHERE Д№=’Д2’) [П№])
JOIN (Поставщики) [Имя_П];

SQL:

```
SELECT DISTINCT Поставщики.[Имя_П] FROM Поставщики
    WHERE Поставщики.[П.№] NOT IN
    (SELECT Поставки.[П№] FROM Поставки
    WHERE Поставки.[Д№]=’Д2’);
```

Основы SQL
(Structured Query Language)



Синтаксис SQL

- запятые используются для разделения компонентов списка параметров;
- точки используются для отделения имен таблиц от имен полей;
- точка с запятой ставится в конце инструкции Jet SQL;
- квадратные скобки используются для выделения имен полей только тогда, когда в именах используются пробелы или другие знаки пунктуации, не разрешенные в SQL;
- одинарная кавычка применяется для описания строчных переменных;
- символы * и ? используются для маскирования окончания или одного символа соответственно;
- символ # применяется для представления одной цифры в операторе LIKE.

Классификация операторов SQL

- Операторы определения данных – определяют содержимое реляционной базы данных в виде таблиц и представлений;
- Операторы манипулирования данных – используются для извлечения, вставки, обновления и удаления данных, содержащихся в таблицах и представлениях;
- Операторы управления данными – ограничивают доступ к данным.

Типы данных
Категории

- Character string – Строки символов;
- Bit string – Строки битов;
- Exact numeric – Рациональные (целые и действительные) числа с плавающей десятичной точкой;
- Approximate numeric – Вещественные числа (с плавающей точкой);
- Date time – значения даты и времени;
- Interval – интервалы даты и времени.

Создание и обслуживание таблиц

- CREATE TABLE Проекты
  (Пр№ CHAR(3) NOT NULL PRIMARY KEY,
  ИмяПр CHAR(15) UNUQUE,
  Гор CHAR(20));

Ограничения на атрибуты:

- NOT NULL – не разрешает присваивать значения NULL;
- DEFAULT – задает значения по умолчанию;
- PRIMARY KEY – задает первичный ключ для таблицы;
- FOREIGN KEY (или REFERENCES) – задает внешний ключ;
- UNIQUE – не позволяет вводить в столбец повторяющиеся значения;
- CHECK – ограничивает с помощью логических выражений значения, которые могут добавляться в столбец.

Создание внешних ключей

```
CREATE TABLE Поставки
      (П№      CHAR(3) NOT NULL REFERENCES Поставщики,
      Пр№      CHAR(5) NOT NULL REFERENCES Проекты,
      Д№       CHAR(3) NOT NULL REFERENCES Детали,
      Кол      INTEGER       DEFAULT '???'
CONSTRAINT ключ PRIMARY KEY (П№ , Пр№, Д№));
```

Обеспечение целостности данных по ссылкам

```
CREATE TABLE Поставки
    (П№      CHAR(3)       REFERENCES Поставщики
     ON UPDATE CASCADE
     ON DELETE SET NULL,
    Пр№      CHAR(5) NOT NULL REFERENCES Проекты   RESTRICT,
    Д№       CHAR(3) NOT NULL REFERENCES Детали,
    Кол      INTEGER       DEFAULT '???'
CONSTRAINT ключ PRIMARY KEY (П№ , Пр№, Д№));
```

Обеспечение целостности атрибута

```
CREATE TABLE Детали
    (Д№       CHAR(3) NOT NULL PRIMARY KEY,
    Имя_Д     CHAR(15)      UNUQUE,
    Цвет      CHAR(10)      CHECK (Цвет='Черный' OR Цвет  ='Красный' OR
    Цвет ='Желтый' OR Цвет ='???'),
    Вес       INTEGER,
    Гор       CHAR(20));
```

Редактирование таблицы

ALTER TABLE Детали ADD [Дата изготовления] DATE;

Характер изменения:
ADD, MODIFY, DELETE

Удаление таблицы:
DROP TABLE Детали;

Управление данными

- **Доступ к данным**

  Виды полномочий: SELECT, UPDATE, ALL

  GRANT UPDATE ON Поставки TO USER1;

  Полномочия для всех пользователей:

  GRANT UPDATE ON Поставки TO PUBLIC;

- **Удаление полномочий:**

  REVOKE UPDATE ON Поставки FROM USER1;

Запрос на выборку

SELECT [ALL/DISTINCT] [TOP n [PERCENT]] список полей
FROM имена таблиц
[WHERE условие отбора]
[ORDER BY столбцы сортировки [ASC/DESC]];

- ALL – включает все строки, соответствующие указанным далее условиям отбора;
- DISTINCT (ключевое слово из ANSI SQL-92) – исключает строки с повторяющимися данными на основе только данных результирующего набора записей;
- TOP n [PERCENT] ограничивает количество записей в результирующей таблице первыми n или n% набора.

Запрос на выборку
Пример

```
SELECT Имя_Д, Вес
    FROM Детали
    WHERE Вес>500
    ORDER BY [Вес] DESC;
```

Статистические функции

SELECT статистическая функция (имя поля) AS заголовок поля [, список полей]
FROM имена таблиц
[WHERE условие отбора]
GROUP BY условие группировки
[HAVING условие для результата]
[ORDER BY столбцы сортировки];

Статистические функции.
Пример 1

Подсчитать общее количество деталей:

```
SELECT SUM(Поставки.Кол)
    FROM Поставки;
```

Можно рассчитать несколько статистических выражений:

```
SELECT MIN(Кол), MAX(Кол), SUM(Кол), AVG(Кол)
    FROM Поставки;
```

Статистические функции.
Пример 2

Количество кортежей в отношении:

```
SELECT COUNT (*) AS Кол_кортежей
    FROM Поставки;
```

Статистические функции.
Пример 3

Применение статистических функций к отдельным группам кортежей:

```
SELECT Пк.ПN, SUM(Пк.Кол)
    FROM Поставки AS Пк
    GROUP BY Пк.ПN;
```

В предложении SELECT необходимо указывать атрибут, по которому производится группировка и нельзя указывать имена атрибутов, не входящих в предложение GROUP BY.

Статистические функции.

Пример 3

Ограничения на результат:

```
SELECT Пк.ПN, SUM (Пк.Кол)
   FROM Поставки AS Пк
    GROUP BY Пк.ПN
    HAVING COUNT(*)>2;
```

Создание соединений (1)

Произведение двух отношений:

```
SELECT *
    FROM Проекты, Поставки;
```

Соединение:

```
SELECT *
    FROM Проекты, Поставки
    WHERE Проекты.ПрN=Поставки.ПрN;
```

Создание соединений (2)

Можно соединить произвольное число отношений:

```
SELECT DISTINCT П.Имя_П, Д.Имя_Д, Пр.
Имя_Пр, Пк.Кол
    FROM Поставщики  AS П, Детали AS Д,
    Проекты AS Пр, Поставки AS Пк
        WHERE Д.ДN=Пк.ДN
        AND П.ПN=Пк.ПN
        AND Пр.ПрN=Пк.ПрN
        AND Пк.Кол>500;
```

Создание соединений (3)

SELECT список полей
FROM имя таблицы {INNER/LEFT/RIGHT} JOIN связанная таблица
ON условие связи
[WHERE условие отбора]
[ORDER BY столбцы сортировки];

Тип соединения:

- INNER – соединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения;
- LEFT (RIGHT) –левое внешнее соединение включает все записи из первой (левой) таблицы и присоединяет к ним записи из второй таблицы, если связующие поля содержат одинаковые значения. Правое внешнее соединение включает все записи из второй (правой) таблицы и присоединяет к ним записи из первой таблицы, если связующие поля содержат одинаковые значения.

Конструкция **ON условие связи** описывает связь между полями соединений.

Создание соединений

Пример 1

Соединение отношений **Проекты** и **Поставки** базы данных **Проекты-Поставщики-Детали**

```
SELECT DISTINCT Пр.Имя_Пр, Пр.Гор, Пк.ДN, Пк.Кол
    FROM (Проекты Пр INNER JOIN Поставки Пк)
        ON Пр.ПрN=Пк.ПрN;
```

Создание соединений

Пример 2

Какие детали поставляются несколькими поставщиками?

```
SELECT F.ПN, S.ПN, F.ДN
    FROM Поставки AS F,
    Поставки AS S
     WHERE F.ДN=S.ДN;
```

Добавим:

AND F.ПN<>S.ПN

и

DISTINCT

F.ПN	S.ПN	F.ДN
П1	П1	Д1
П1	П1	Д1
П1	П5	Д1
П1	П1	Д1
П1	П1	Д1
П1	П5	Д1
П5	П1	Д1
П5	П1	Д1
П5	П5	Д1
…	…	…

Вложенные запросы

```
SELECT список полей
FROM список таблиц
WHERE [имя таблицы.] имя поля
    IN (SELECT оператор выборки
        [GROUP BY условие группировки]
            [HAVING условие отбора])
[ORDER BY столбцы сортировки];
```

Вложенные запросы

Пример 1

Найти номера поставщиков, поставляющих хотя бы одну черную деталь.

```
SELECT Пк.ПN FROM Поставки Пк
    WHERE Пк.ДN IN
        (SELECT Д.ДN FROM Детали Д
            WHERE Д.Цвет='Черный');
```

Вложенные запросы

Пример 2

Можно добавить имена поставщиков:

```
SELECT П.Имя_П FROM Поставщики П
    WHERE П.ПN IN
        (SELECT Пк.ПN FROM Поставки Пк
        WHERE Пк.ДN IN
            (SELECT Д.ДN FROM Детали Д
            WHERE Д.Цв='Черный'));
```

Вложенные запросы

Пример 3

Такой же результат можно получить соединением:

```
SELECT DISTINCT П.Имя_П
   FROM Поставщики П, Поставки Пк, Детали Д
   WHERE П.ПN=Пк.ПN
         AND Пк.ДN=Д.ДN
         AND Д.Цв='Черный';
```

Как лучше?

Вложенные запросы

Пример 4

Проверка на существование :

```
SELECT * FROM Поставщики П
    WHERE П.ПN NOT IN
        (SELECT Пк.ПN FROM Поставки Пк);
```

Запрос на объединение

```
SELECT оператор выборки
   UNION
   SELECT оператор выборки
         [GROUP BY условие группировки]
         [HAVING итоговое условие]
   [UNION
   SELECT оператор выборки
         [GROUP BY условие группировки]
         [HAVING итоговое условие]]
   [UNION
   …]
   [ORDER BY столбцы сортировки];
```

Запрос на объединение

Пример

```
SELECT Имя_П AS Наименование
    FROM Поставщики
    WHERE Гор=’Минск’
  UNION SELECT Имя_Пр AS Наименование
    FROM Проекты
    WHERE Гор=’Минск’
    ORDER BY Наименование;
```

Оператор EXISTS

Оператор EXISTS в предложении WHERE выполняет проверку на существование данных, которые удовлетворяют критериям соответствующего вложенного запроса, и возвращает булево значение «истина» или «ложь».

Пример. Найти имена поставщиков, которые поставляют деталь Д1:

```
SELECT DISTINCT П.Имя_П FROM Поставщики AS П
    WHERE EXISTS
    (SELECT * FROM Поставки AS Пк
          WHERE Пк.ПN=П.ПN
              AND Пк.ДN='Д1');
```

Два решения задачи:
Найти номера деталей, поставляемых поставщиком из города, название которого начинается с буквы М

- SELECT DISTINCT Пк.ДN FROM Поставки AS Пк
  WHERE Пк.ПN IN
    (SELECT П.ПN FROM Поставщики AS П
      WHERE П.Гор LIKE 'М*');

- SELECT DISTINCT Пк.Д№ FROM Поставки Пк
  WHERE EXISTS
    (SELECT * FROM Поставщики П
      WHERE П.П№ = Пк.П№
        AND Гор LIKE 'М*');

Можно добавить сведения из третьей таблицы:

```
SELECT DISTINCT Д.Имя_Д
    FROM Детали AS Д
     WHERE EXISTS
         (SELECT DISTINCT Пк.ДN
             FROM Поставки AS Пк
                 WHERE EXISTS
         (SELECT *
             FROM Поставщики AS П
             WHERE П.ПN=Пк.ПN
                 AND Гор LIKE 'М*')
                 AND Д.ДN=П.ДN);
```

Реализация операции пересечения

Пересечение таблиц Детали и Поставщики по полю Гор

```
SELECT DISTINCT Д.Гор
   FROM Детали AS Д
    WHERE EXISTS
       (SELECT *
          FROM Поставщики П
             WHERE Д.Гор=П.Гор);
```

Реализация операции вычитания (1)

Разность таблиц Детали и Поставщики по полю Гор

```
SELECT DISTINCT Д.Гор
   FROM Детали Д
    WHERE NOT EXISTS
       (SELECT *
          FROM Поставщики П
             WHERE Д.Гор=П.Гор);
```

Реализация операции пересечения (2)

Разность таблиц Детали и Поставщики по полю Гор

```
SELECT DISTINCT Д.Гор
   FROM Детали Д
    WHERE
       (SELECT COUNT (*)
          FROM Поставщики П
             WHERE Д.Гор = П.Гор) >0;
```

Реализация операции деления

Получить номера поставщиков, поставляющих все детали

```
SELECT DISTINCT Пк.ПN
    FROM Поставки AS Пк
    WHERE NOT EXISTS
    (SELECT Д.ДN FROM Детали AS Д
          WHERE NOT EXISTS
          (SELECT Пк1.ДN
               FROM Поставки AS Пк1
                    WHERE Пк1.ПN=Пк.ПN
                         AND Пк1.ДN=Д.ДN));
```

Запросы на изменение записей

- *Добавление записей:*

```
INSERT INTO таблица-получатель
    SELECT список полей FROM таблица-источник;
     [WHERE условие удаления];
```

- *Удаление записей:*

```
DELETE FROM имя таблицы
    [WHERE условие удаления];
```

- *Создание таблицы:*

```
SELECT список полей
    INTO новая таблица
    FROM исходная таблица
    [WHERE условие выбора];
```

- *Обновление:*

```
UPDATE имя таблицы
    SET имя_поля_1=значение [,имя_поля_2=значение[,…]]
    [WHERE условие обновления];
```

Перекрестные запросы

TRANSFORM статистическая функция (имя поля) [AS наименование]
SELECT список полей
FROM имя таблицы
PIVOT *поле* [IN (*значение_1*[, *значение_2*[, ...]])];

Перекрестные запросы

Пример

Представить данные о количествах деталей, поставленных каждым поставщиком.

```
TRANSFORM Sum(Пк.Кол)
    SELECT Пк.ПN, Sum(Пк.Кол) AS [ВСЕГО:]
     FROM Поставки AS Пк
     GROUP BY Пк.ПN
    PIVOT Пк.ДN;
```

ПN	ВСЕГО:	Д1	Д2	Д3	Д4	Д5	Д6
П1	900	900					
П2	3200			3100		100	
П3	700			200	500		
П4	600						600
П5	3110	110	300	200	800	1000	700

Проектирование баз данных.
Проблемы, которые необходимо избегать

- Аномалии обновления – из-за избыточности данных при их обновлении необходимо просматривать все данные, тем не менее, может возникнуть ситуация, когда не все данные будут обновлены (потенциальная противоречивость данных).
- Аномалии включения – возможна ситуация, когда в базу нельзя ввести данные, прежде чем не будут получены и введены некоторые дополнительные сведения.
- Аномалии удаления – обратная проблема может возникнуть при удалении некоторых данных (возможна потеря полезной информации).
- Не минимизировано количество Null-значений. Так же как избыточность, неопределенные значения являются источниками потенциальных проблем в реляционных базах данных, так как невозможно определить, что они означают. Поэтому их использование желательно свести к минимуму.

Нормализация отношений

- *Нормализация* – это разбиение (или декомпозиция) таблицы на две или более, обладающих лучшими свойствами при добавлении, изменении и удалении данных;
- Нормализованное отношение;
- Нормальные формы.

Функциональные зависимости (1)

Пусть X и Y – произвольные подмножества множества атрибутов отношения R.

Y функционально зависит от X тогда и только тогда, когда каждое значение множества X связано в точности с одним значением множества Y.

Обозначение: X→Y

П№	Гор	Д№	Кол
П1	Брест	Д1	100
П1	Брест	Д2	100
П2	Минск	Д1	200
П2	Минск	Д2	200
П3	Гродно	Д2	300
П4	Минск	Д2	400
П4	Минск	Д4	400
П4	Минск	Д5	400

Функциональные зависимости (2)

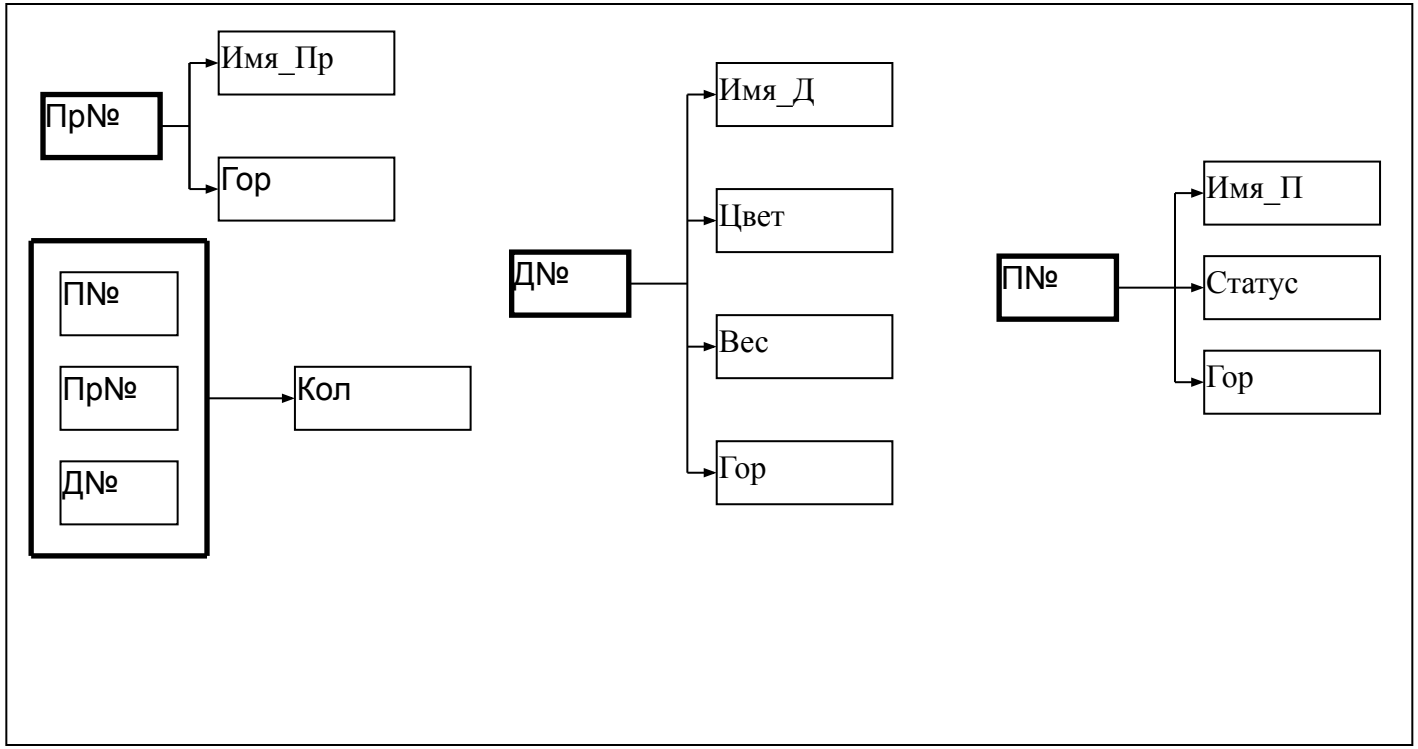
- Тривиальные зависимости – те, которые не могут не выполняться:

  $\{П№,Д№\} \rightarrow П№$

- Неприводимые зависимости:
  Атрибут B неприводимо зависим от составного атрибута A, если он функционально зависит от A и не зависит функционально от любого подмножества атрибута A.

Функциональные зависимости (3)

Диаграмма функциональных зависимостей для учебной базы данных «Проекты, Поставщики, Детали»



Первая нормальная форма (1)

Таблица находится в первой нормальной форме (1НФ) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто.

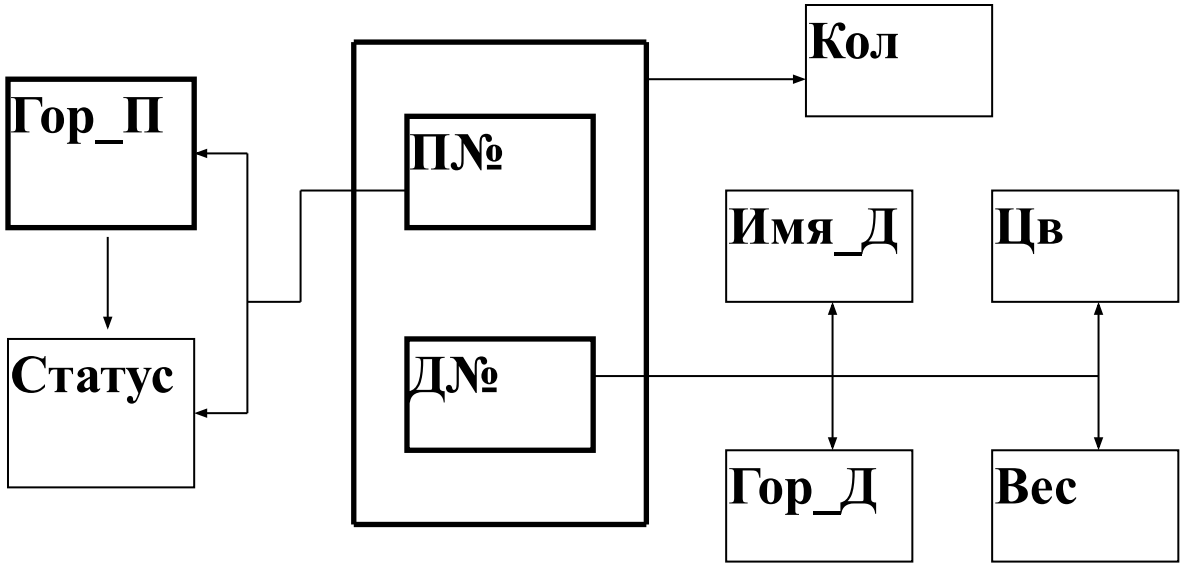
П№	Гор	Д№	Кол
П4	Минск	Д2,Д4,Д5	400

Первая нормальная форма (2)

Универсальное отношение

П№	Статус	Гор_П	**Д№**	Имя_Д	Цв	Вес	Гор_Д	Кол
П1	20	Брест	Д1	Тестер	Черный	250	Борисов	300
П1	20	Брест	Д2	Дозиметр	Серый	700	Минск	200
П1	20	Брест	Д3	Радиометр	Черный	1400	Минск	400
П1	20	Брест	Д4	Часы	Желтый	140	Брест	200
П1	20	Брест	Д5	Рулетка	Красный	200	Пинск	100
П1	20	Брест	Д6	Лом	Черный	5000	Могилев	100
П2	10	Минск	Д1	Тестер	Черный	250	Борисов	300
П2	10	Минск	Д2	Дозиметр	Серый	700	Минск	400
П3	30	Гродно	Д2	Дозиметр	Серый	700	Минск	200
П4	10	Минск	Д2	Дозиметр	Серый	700	Минск	200
П4	10	Минск	Д4	Часы	Желтый	140	Брест	300

Первая нормальная форма (3)
Диаграмма функциональных зависимостей



Первая нормальная форма (4)
Аномалии

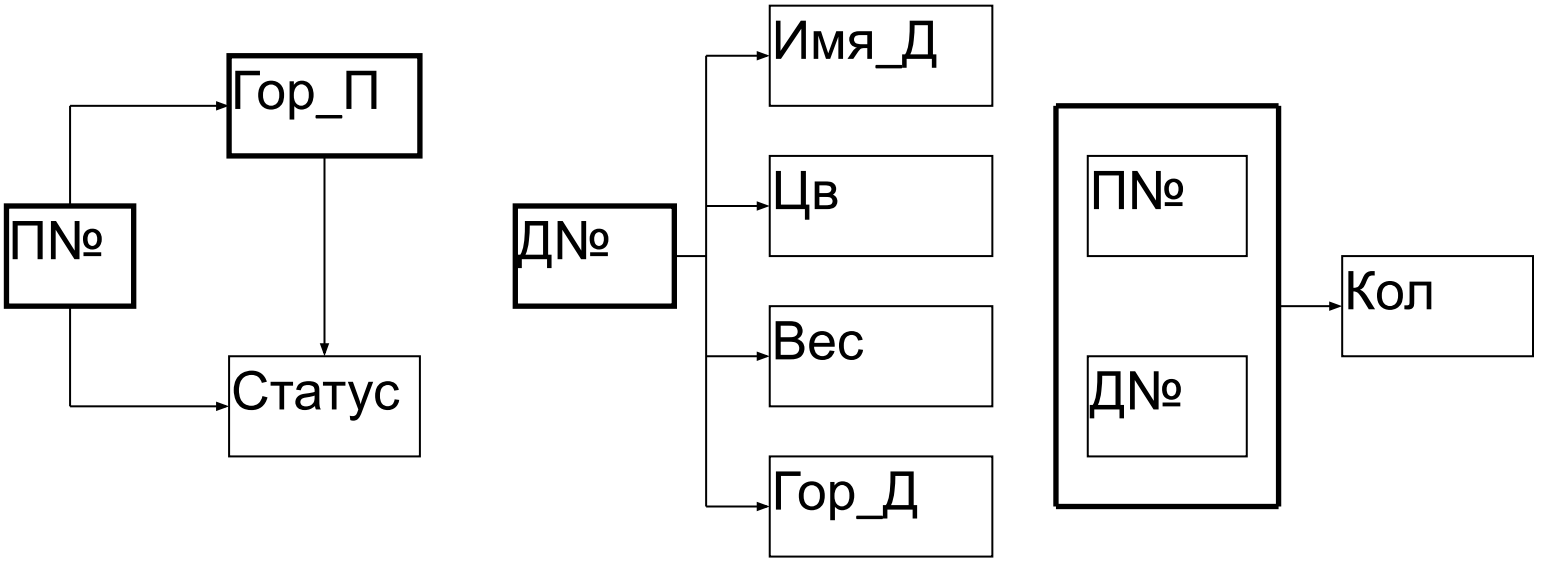
- Вставка (Insert). Нельзя вставить данные о поставщике (П5), не указав деталь (Null-значение в ключевом поле недопустимо).
- Удаление (Delete). При удалении некоторого кортежа приходится удалять слишком много другой информации (удаление информации о поставке удаляет информацию о поставщике).
- Обновление (Update). Избыточная информация может привести к несовместимым результатам. Если поставщик П1 переехал в другой город, а обновление сделано не во всех кортежах, то база данных будет содержать противоречивую информацию.

Вторая нормальная форма (1)

Таблица находится во второй нормальной форме (2НФ), если она удовлетворяет определению 1НФ и все ее поля, не входящие в первичный ключ, связаны неприводимой зависимостью с первичным ключом.

Вторая нормальная форма (2)

Диаграмма функциональных зависимостей отношения, приведенного к 2НФ



Вторая нормальная форма (3)

Преобразованные отношения

П№	Статус	Гор_П
П1	20	Брест
П2	10	Минск
П3	30	Гродно
П4	10	Минск
П5	20	Брест

Д№	Имя_Д	Цв	Вес	Гор_Д
Д1	Тестер	Черный	250	Борисов
Д2	Дозиметр	Серый	700	Минск
Д3	Радиометр	Черный	1400	Минск
Д4	Часы	Желтый	140	Брест
Д5	Рулетка	Красный	200	Пинск
Д6	Лом	Черный	5000	Могилев

П№	**Д№**	Кол
П1	Д1	300
П1	Д2	200
П1	Д3	400
П1	Д4	200
П1	Д5	100
П1	Д6	100
П2	Д1	300
П2	Д2	400
П3	Д2	200
П4	Д2	200
П4	Д4	300

Вторая нормальная форма (4)
Аномалии

- Вставка – нельзя включить данные о некотором городе и его статусе, пока в нем нет поставщика.

- Удаление – при удалении поставщика теряется информация о статусе города.

- Обновление – статус городов повторяется несколько раз. При изменении статуса города приходится просматривать множество строк, чтобы исключить получение противоречивого результата, но вероятность ошибки остается.

Транзитивные зависимости

Если выполняются функциональные зависимости A→B и B→C, то выполняется также и функциональная зависимость A→C.

Возможная декомпозиция:

Третья нормальная форма (1)

- Отношение находится в третьей нормальной форме (3НФ) тогда и только тогда, когда оно находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Другими словами:

- Таблица находится в третьей нормальной форме (3НФ), если она находится в 2НФ и ни одно из ее неключевых полей не зависит функционально от любого другого неключевого поля.

Третья нормальная форма (2)
Варианты декомпозиции

П№	Статус	Гор_П
П1	20	Брест
П2	10	Минск
П3	30	Гродно
П4	10	Минск
П5	20	Брест

Функциональные зависимости:

П№→Город

П№→ Статус

Город→Статус

Варианты декомпозиции:

A: (П№, Город) и (Город, Статус)

B: (П№, Город) и (П№, Статус)

C: (П№, Статус) и (Город, Статус)

Третья нормальная форма (3)

Декомпозиция с сохранением зависимости (Риссанен (Rissanen)).

Проекции R1 и R2 отношения R независимы тогда и только тогда, когда

- каждая функциональная зависимость в отношении R является логическим следствием функциональных зависимостей в проекциях R1 и R2;
- общие атрибуты проекций R1 и R2 образуют потенциальный ключ, по крайней мере, для одной из них.

Т.е., отношение R{A,B,C}, удовлетворяющее функциональным зависимостям A→B и B→C, следует разбивать на проекции {A,B} и {B,C}, а не {A,B} и {A,C}

Нормальная форма Бойса-Кодда (1)

(Bouce-Codd)

Определение 3НФ не корректно, если

- отношение имеет два или более потенциальных ключа;
- два потенциальных ключа являются сложными и они перекрываются

Нормальная форма Бойса-Кодда (2)

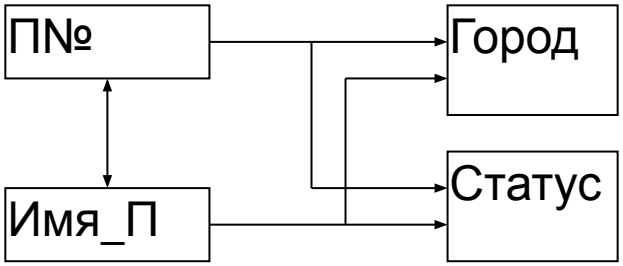
Отношение находится в нормальной форме Бойса-Кодда (НФБК) тогда и только тогда, когда детерминанты являются потенциальными ключами.

Или

Таблица находится в нормальной форме Бойса-Кодда (НФБК), тогда и только тогда, когда любая функциональная зависимость между ее полями сводится к неприводимой функциональной зависимости от потенциального ключа.

Нормальная форма Бойса-Кодда (3)
Пример отношения в НФБК

Отношение Поставщик (П№, Имя_П, Статус, Город) с неперекрывающимися ключами



Нормальная форма Бойса-Кодда (4)
Пример

Отношение СДП с атрибутами (С, Д,П).

Ограничения:

- Каждый студент изучает данный предмет у одного преподавателя ({С, Д}→П);
- Каждый преподаватель ведет только один предмет (но каждый предмет может преподаваться несколькими преподавателями) (П→Д).

Есть два перекрывающихся ключа – {С, Д} и {С,П}

С	Д	П
Олег	Математ	Ильин
Олег	Физика	Петров
Петр	Математ	Ильин
Петр	Физика	Иванов

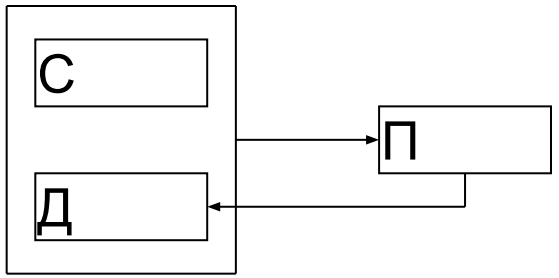


Схема учебной базы данных