Тема № 1. Базы данных специального назначения

Лекция № 1: Введение в базы данных

Учебные цели занятия:

Изучить:

1) основные понятия теории баз данных,
2) основные принципы организации систем баз данных,
3) вопросы семантического моделирования (ER-моделирование)

Учебные вопросы:

4) Основные понятия теории баз данных
5) Архитектура систем баз данных
6) Семантическое моделирование

Литература:

- К. Дж. Дейт. - Введение в системы баз данных, 7-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1072 с., ил.
- Дж. Гроff, П. Вайнберг.- SQL: Полное руководство.- Пер. с англ.-2-е изд., перераб. и доп.-К.: Издательская группа BHV, 2001.- 816 с., ил.
- SQL в примерах и задачах; учеб. пособие / И.Ф. Астахова, А.П.Толстобров, В.М.Мельников.— Мн.: Новое знание, 2002. — 176 с.
- Теория и практика построения баз данных/Д. Кренке.- 8-е изд.- СПб.: Питер, 2003.- 800 с., ил.- (Серия «Классика computer science»).

1. Основные понятия теории баз данных

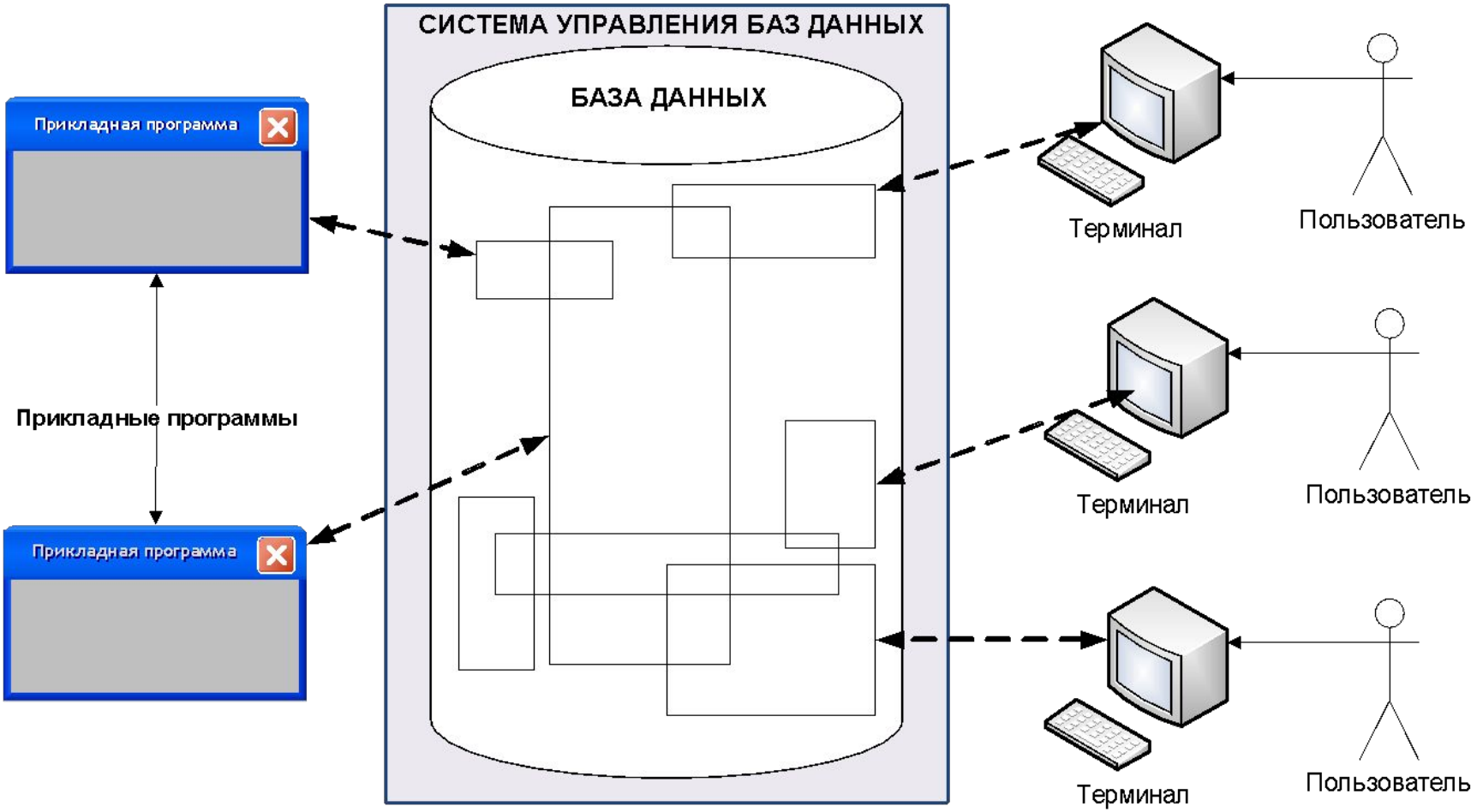
1.1 Понятие системы баз данных

Система баз данных (СБД) – компьютеризированная система хранения записей.

Основным назначением СБД является хранение информации и предоставление пользователям средства ее извлечения и модификации.

- ***Однопользовательская система*** (single-user system) – система, в которой одновременно к базе данных может получить доступ не более одного пользователя
- ***Многопользовательская система*** (multi-user system) – система в которой к базе данных может получить доступ одновременно несколько пользователей.

Упрощенная схема системы баз данных



ГЛАВНЫЕ КОМПОНЕНТЫ СБД

- *данные*
- *аппаратное обеспечение*
- *программное обеспечение*
- *пользователи*

ДАННЫЕ в БАЗЕ ДАННЫХ являются:

- ***интегрированными;***
- ***разделяемыми.***

- ***Интегрированность данных – возможность представления базы данных как объединение нескольких отдельных файлов данных, полностью или частично исключающее избыточность хранения информации.***

- ***Разделяемость данных – возможность использования отдельных элементов, хранимых в базе данных несколькими различными пользователями. Имеется в виду, что каждый их пользователей сможет получить доступ к одному и тому же элементу данных в одно и то же время, возможно, для достижения различных целей.***

Аппаратное обеспечение СБД:

- Тома вторичной (внешней) памяти (обычно это магнитные диски), используемые для хранения информации, а также соответствующие устройства ввода-вывода (дисководы и т.п.), контроллеры устройств, каналы ввода-вывода и т.д.
- Аппаратный процессор (или процессоры) вместе с основной (первичной) памятью, предназначенные для поддержки работы программного обеспечения СБД.

Программное обеспечение СБД:

- ***система управления базами данных,*** СУБД – это наиболее важный программный компонент системы, называемый также: ***менеджер базы данных*** (database manager), ***сервер базы данных*** (database server);
- утилиты
- средства разработки приложений;
- средства проектирования;
- генераторы отчетов;
- ***менеджер транзакций*** (transaction manager) или ***диспетчер выполнения транзакций*** (TP monitor).

Пользователи :

- ***Прикладные программисты***
- ***Конечные пользователи***
- ***Администраторы базы данных*** (АБД).

АДМИНИСТРАТОР базы данных (АБД)

- (АБД) – человек, обеспечивающий необходимую техническую поддержку с целью реализации принятых решений. АБД отвечает за управление системой на техническом уровне.

Функции АБД:

- ***Определение концептуальной схемы.***
- ***Определение внутренней схемы.***
- ***Взаимодействие с пользователями.***
- ***Определение требований защиты и обеспечение целостности данных.***
- ***Определение процедур резервного копирования и восстановления.***
- ***Управление производительностью и реагирование на изменяющиеся требования.***

1.2 Базы данных и их назначение

- ***База данных*** – это некоторый набор перманентных (постоянных) данных, используемых прикладными системами какого-либо предприятия.

Преимущества использования однопользовательских СБД :

- ***Компактность***.
- ***Скорость***.
- ***Низкие трудозатраты***.
- ***Актуальность***.

Многопользовательская среда имеет дополнительное преимущество:

: СБД *предоставляет предприятию средства централизованного управления его данными*

Преимущества централизованного подхода к управлению данными:

- ***Возможность совместного доступа к данным***
- ***Сокращение избыточности данных***
- ***Устранение противоречивости данных (до некоторой степени)***
- ***Возможность поддержки транзакций***
- ***Обеспечение целостности данных***
- ***Организация защиты данных***
- ***Возможность балансировки противоречивых требований***
- ***Возможность введения стандартизации***
- ***Независимость данных***.

1.3 Данные и модели данных

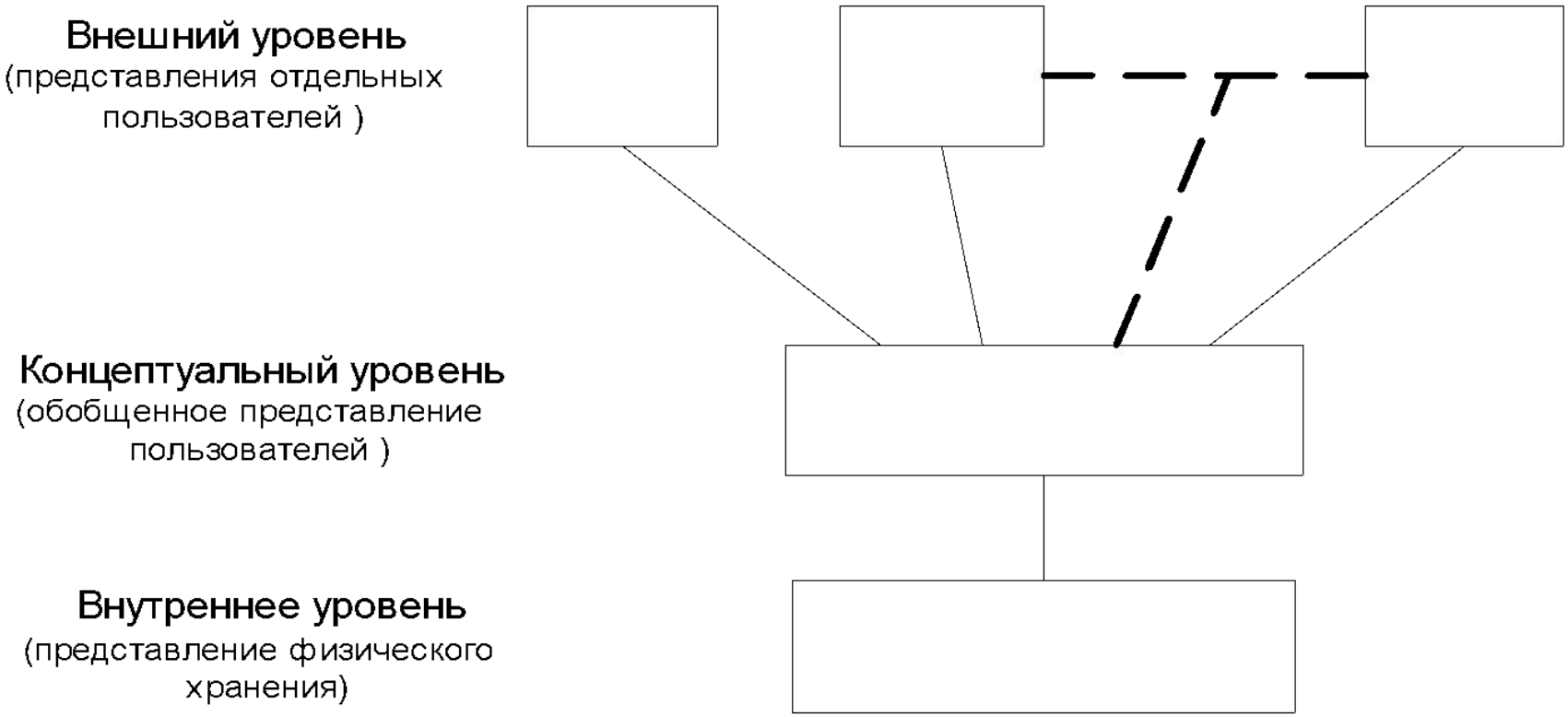
- ***Модель данных*** **– это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактно машину, с которой взаимодействует пользователь. Упомянутые объекты позволяют моделировать структуру данных, а операторы – поведение данных.**
- ***Реализация*** **(implementation) – заданной модели данных – это фактическое воплощение на реальной машине компонентов абстрактной машины, которые в совокупности составляют эту модель.**

1.4 Типы систем баз данных

Категории системы баз данных:

- системы ***инвертированных списков***
- ***иерархические***
- ***сетевые***
- ***объектно-ориентированные*** и ***объектно-реляционные***

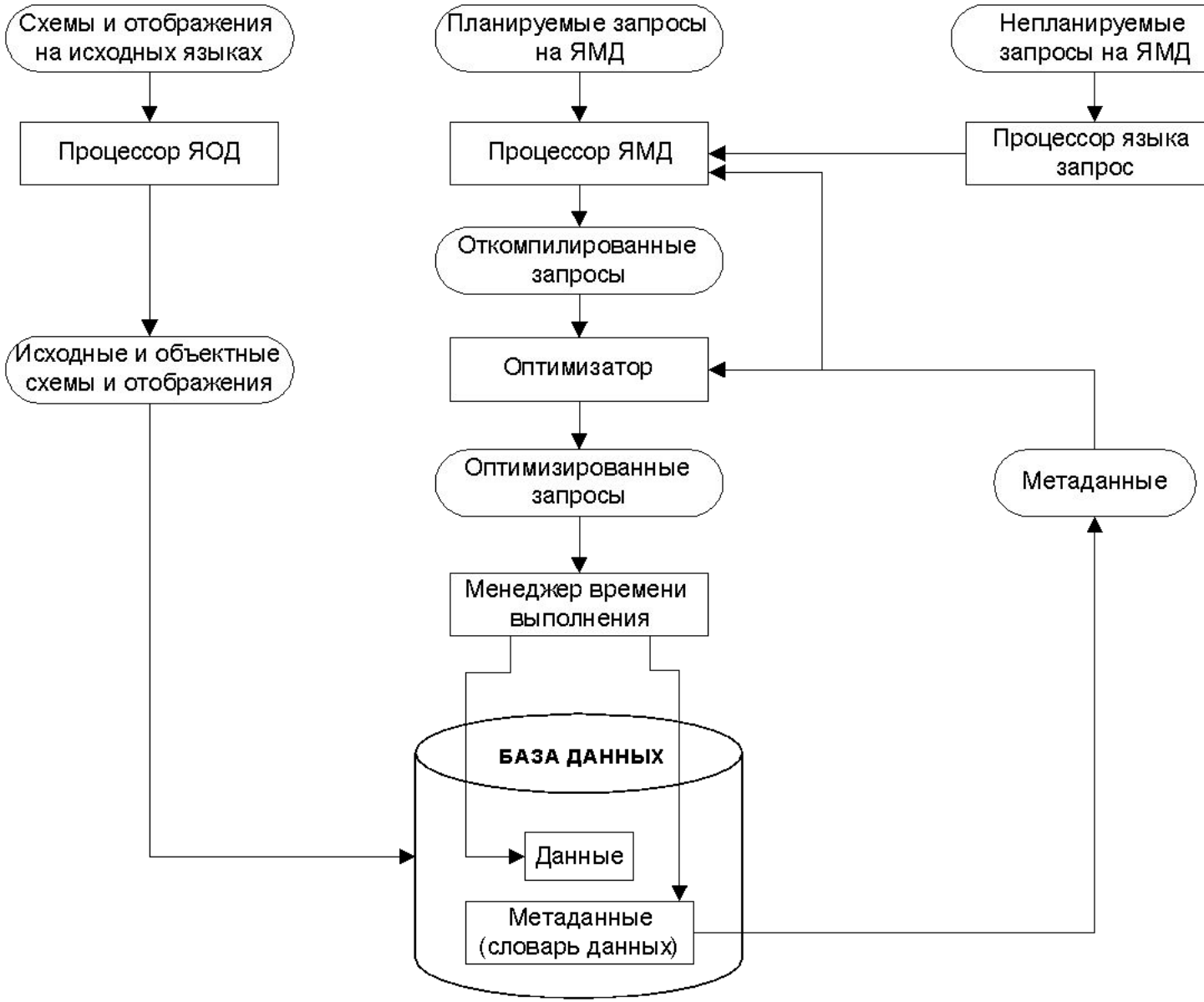
2. Архитектура системы баз данных
Три уровня архитектуры ANSI/SPARC



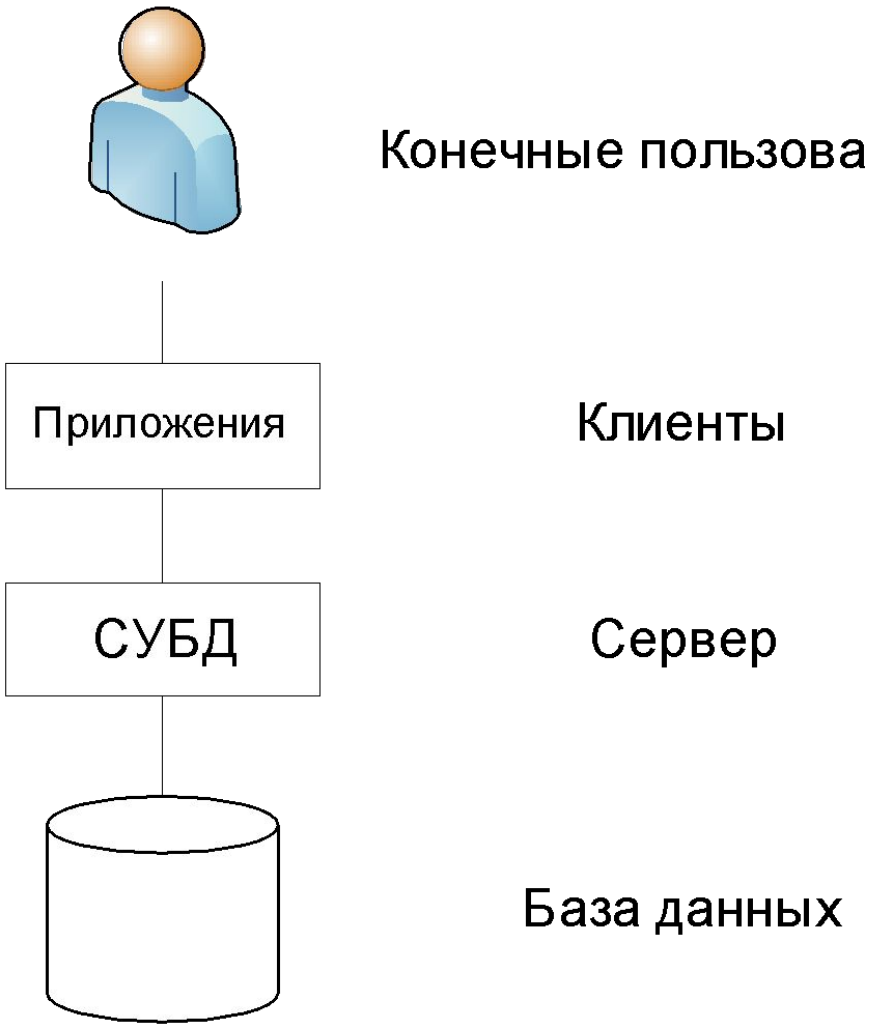
Три уровня архитектуры ANSI/SPARC

- **Внешний уровень - индивидуальный уровень пользователя**
- **Концептуальный уровень**. ***Концептуальное представление*** – это представление всей информации базы данных в несколько более абстрактной форме по сравнению с физическим способом хранения данных
- **Внутренний уровень**. ***Внутреннее представление*** – это низкоуровневое представление всей базы данных как базы, состоящей из некоторого множества экземпляров каждого из существующих типов внутренних записей.

Основные функции и компоненты типичной СУБД

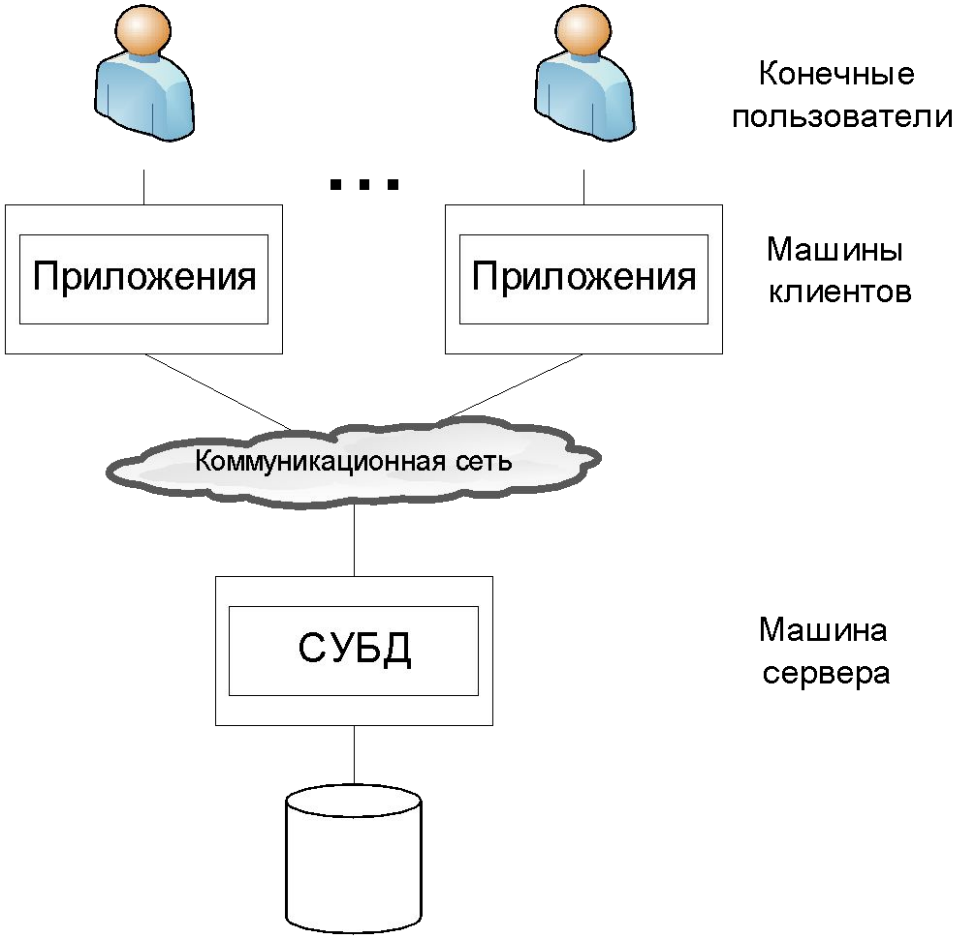


Схематическое представление архитектуры «клиент/сервер»

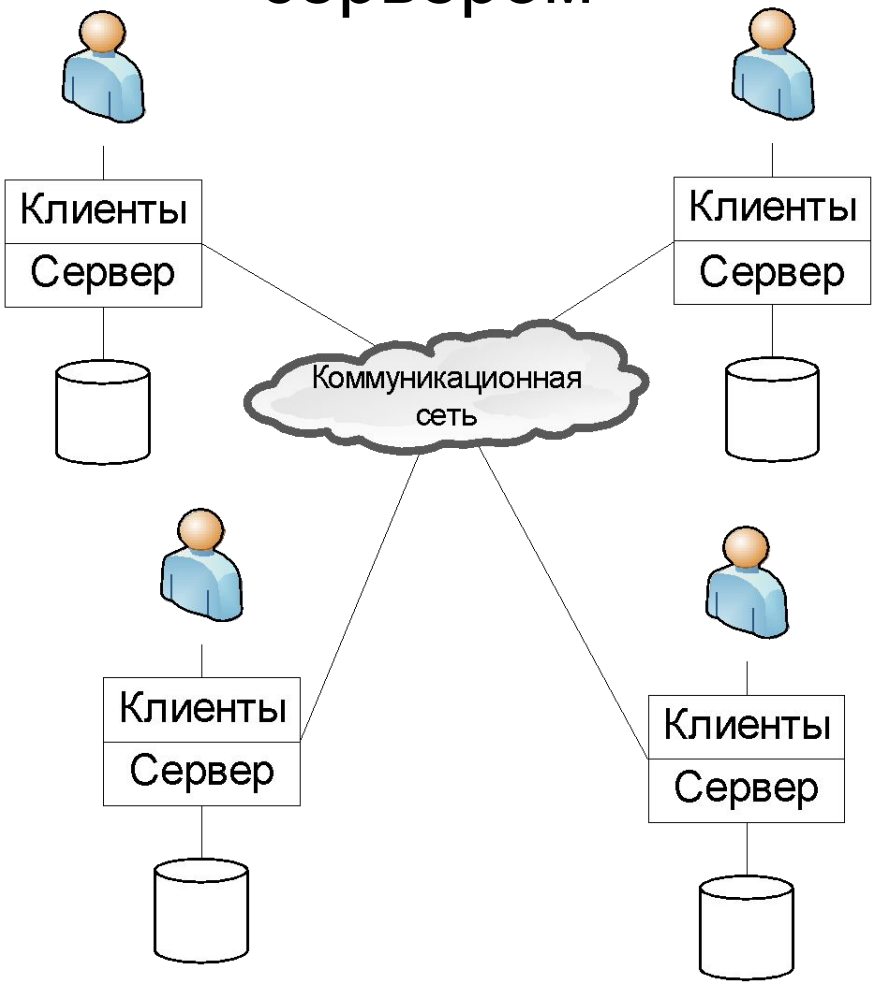


Варианты распределенной обработки: (а) клиент и сервер запускаются на разных машинах

Варианты распределенной обработки: (б) один сервер и несколько клиентов



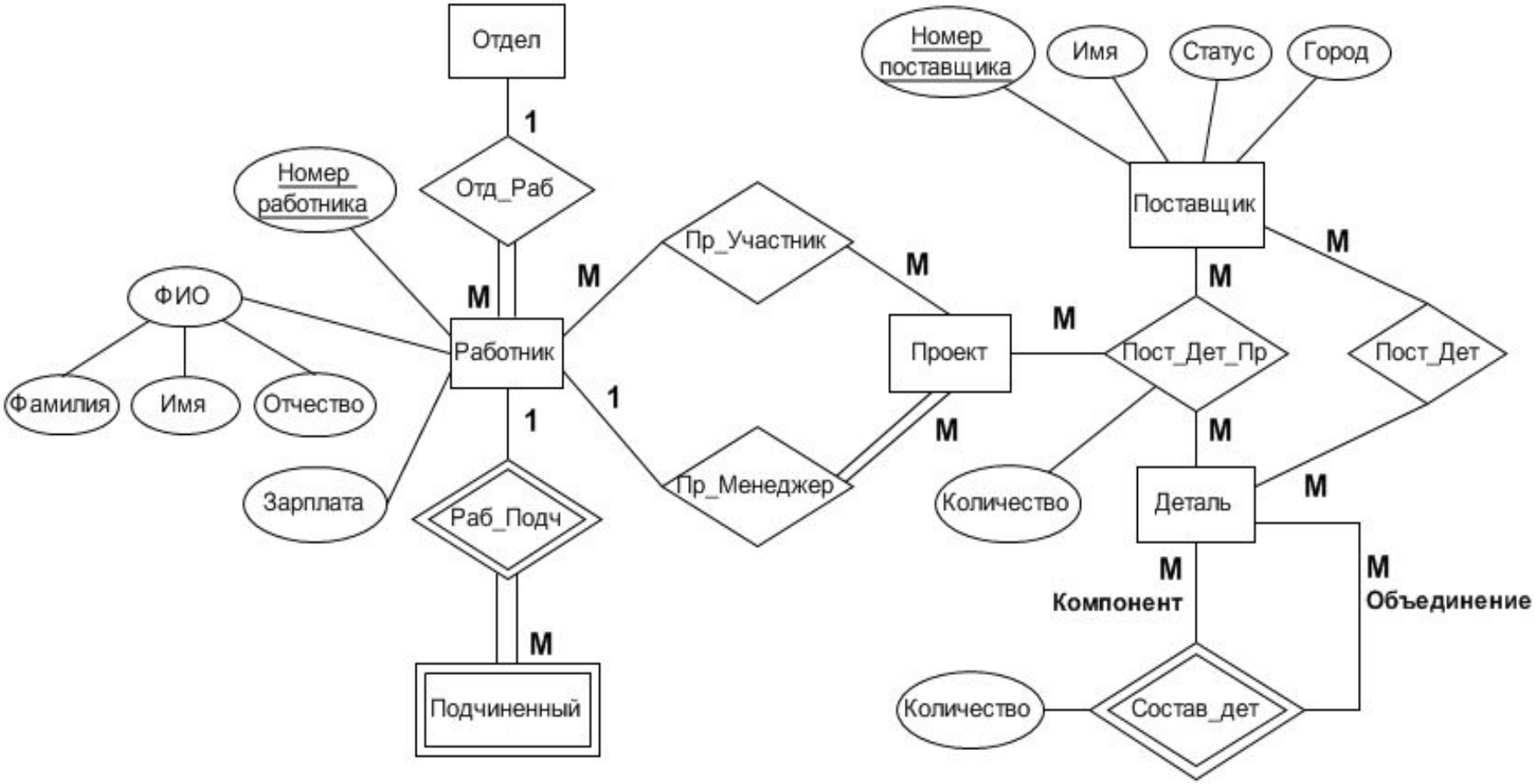
Варианты распределенной обработки: (в) каждая машина является и клиентом, и сервером



Определения семантических концепций

Понятие	Неформальное определение	Примеры
СУЩНОСТЬ (Entity)	Некоторый отличимый объект	Работник, подразделение Поставщик, деталь, поставка
СВОЙСТВО (Property)	Элемент информации, описывающий сущность	Номер поставщика, год рождения работника Вес детали, юридический адрес поставщика
СВЯЗЬ (Relationship)	Сущность, которая служит для обеспечения взаимодействия между двумя или более другими сущностями	Поставка (поставщик – деталь) Должность (работник – подразделение)
ПОДТИП (Subtype)	Сущность типа Y является подтипом сущности типа X тогда и только тогда, когда каждый экземпляр сущности типа Y обязательно является экземпляром сущности типа X	«Работник» является подтипом сущности «Человек» Поставщик является подтипом сущности «Юридическое лицо»

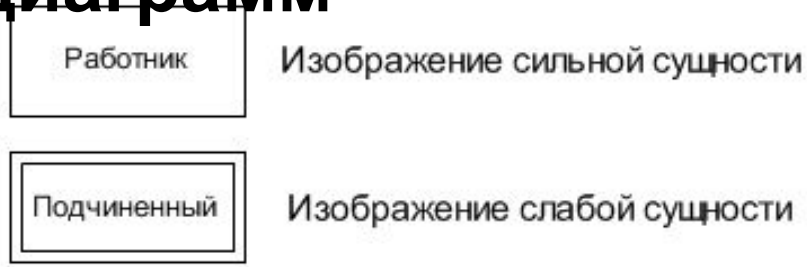
Пример диаграммы модели «сущность/связь»



Пример иерархии типов сущностей

Спецификация ER-диаграмм

Сущности



Свойства

Спецификация ER-диаграмм (окончание)

Изображение связи

Изображение связи между слабой сущностью и сущностью, от которой она зависит

Вопросы на самоподготовку:

1. Понятие системы базы данных (СБД). Схема СБД. Характеристики данных. Типы пользователей СБД и их характеристики.
2. Понятие базы данных (БД). Преимущества использования СБД для реализации БД.
3. Модели данных и их реализация. Основные типы СБД.
4. Архитектура ANSI/SPARC организации СБД. Понятие СУБД, ее основные функции и компоненты.
5. Система управления передачей данных. Архитектура «клиент/сервер» и ее адаптация для систем распределенной обработки данных.
6. Семантическое моделирование: назначение и суть. Модель «сущность/связь». ER-диаграммы: назначение и правила построения. Примеры.

Тема № 1. Базы данных специального назначения

Лекция № 2: **Реляционная модель. Введение в реляционные БД. Основы языка SQL.**

Учебные цели занятия:

Сформировать представление о:

1) Понятии реляционной модели данных ,
2) Языке SQL и его возможностях,
3) Средствах определения структуры данных и типов данных реляционной модели и языка SQL.

Учебные вопросы:

4) Введение в реляционные базы данных
5) Введение в язык SQL
6) Домены, отношения и базовые переменные-отношения

Литература:

- К. Дж. Дейт. - Введение в системы баз данных, 7-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1072 с., ил.
- Дж. Гроff, П. Вайнберг.- SQL: Полное руководство.- Пер. с англ.-2-е изд., перераб. и доп.-К.: Издательская группа BHV, 2001.- 816 с., ил.
- SQL в примерах и задачах; учеб. пособие / И.Ф. Астахова, А.П.Толстобров, В.М.Мельников.— Мн.: Новое знание, 2002. — 176 с.
- Теория и практика построения баз данных/Д. Кренке.- 8-е изд.- СПб.: Питер, 2003.- 800 с., ил.- (Серия «Классика computer science»).

1.Введение в реляционные базы данных

1.1. Реляционная модель

- **реляционная модель данных** - формальная основа или теория, на которой базируются реляционные системы. Для таких систем характерно выполнение трех условий:
- *Структурный аспект*.
- *Аспект целостности*.
- *Аспект обработки*.

База данных отделов и служащих. Операции выборки, извлечения столбцов и соединения

Отделы

НомОтдела	НазваниеОтд	Бюджет
Отд1	Экономический	10М
Отд2	Конструкторский	12М
Отд3	Исследовательский	5М

Служащие

НомСлужащего	ИмяСлужащего	НомОтдела	Зарплата
С1	Иванов	Отд1	21К
С2	Петров	Отд1	18К
С3	Никитин	Отд2	20К
С4	Буденко	Отд2	19К

SELECT (RESTRICT)
Выборка строк из **Отделы,** где БЮДЖЕТ > 8М

НомОтдела	НазваниеОтд	Бюджет
Отд1	Экономический	10М
Отд2	Конструкторский	12М

PROJECT
Извлечение столбцов НомОтдела, Бюджет из **Отделы**

НомОтдела	Бюджет
Отд1	10М
Отд2	12М
Отд3	5М

JOIN:
Соединение **Отделы** и **Служащие** на основе столбца НомОтдела

НомОтдела	НазваниеОтд	Бюджет	НомСлужащего	ИмяСлужащего	Зарплата
Отд1	Экономический	10М	С1	Иванов	21К
Отд1	Экономический	10М	С2	Петров	18К
Отд2	Конструкторский	12М	С3	Никитин	20К
Отд2	Конструкторский	5М	С4	Буденко	19К

Назначение некоторых операций базы данных

- Операция SELECT предназначена для извлечения определенных строк из таблицы.
- Операция PROJECT предназначена для извлечения определенных столбцов из таблицы.
- Операция JOIN предназначена для соединения двух таблиц на основе общих значений в общих столбцах. При выполнении операции JOIN строки таблиц соединяются на основе значения некоторого столбца и только тогда, когда две строки соединяемых таблиц содержат в этих полях одинаковое значение.
- Следует отметить, что в результате выполнения любой из приведенных операций получается также таблица. Это реляционное свойство называется *замкнутостью*.

- Таблицы в реляционной системе являются ***логическими***, а не физическими структурами.
- ***каждая строка в таблице должна иметь уникальное значение столбца,*** указанные столбцы в соответствующих таблицах являются ***потенциальными ключами***. Среди всех потенциальных ключей выбирается один, который называется ***первичным ключом***.

Реляционная модель

Реляционная модель состоит из следующих пяти компонентов:

- Неограниченный набор ***скалярных типов*** (включая, в частности, логический тип или значение истины).
- Генератор ***типов отношений*** и соответствующая интерпретация для таких сгенерированных типов отношений.
- Возможность определения ***переменных отношений*** для таких сгенерированных типов отношений.
- Операции ***реляционного присвоения*** для присвоения реляционных значений таким переменным отношений.
- Неограниченный набор общих ***реляционных операторов*** для получения значений отношений из других значений отношений.

Иллюстрация к реляционному присвоению

Служащие

НомСлужащего	ИмяСлужащего	НомОтдела	Зарплата
С1	Иванов	Отд1	21К
С2	Петров	Отд1	18К
С3	Никитин	Отд2	20К
С4	Буденко	Отд2	19К

Служащие’

НомСлужащего	ИмяСлужащего	НомОтдела	Зарплата
С1	Иванов	Отд1	21К
С2	Петров	Отд1	18К
С3	Никитин	Отд2	20К

- мы удалили строку о сотруднике с фамилией «Буденко» (его номер «С4»):
- **DELETE Служащие WHERE НомСлужащего=«С4».**
- Концептуально это можно описать следующим образом: Старое значение отношения Служащие было заменено в целом совершенно другим, новым значением отношения. Т.е. приведенная операция удаления строки, по сути, - это просто другой, упрощенный способ записи операции ***реляционного присвоения***:
- **Служащие := Служащие MINUS (Служащие WHERE НомСлужащего=«**С4»)
- Ключевое слово **MINUS** используется для описания оператора реляционной разности.
- Соответственно INSERT и UPDATE являются также иными формами записи соответствующих операций реляционного присвоения.

Предикат и истинное высказывание

- Рассмотрим важный, способ представления смысла отношений:
- Во-первых, данное отношение r и заголовок отношения r представляют определенный ***предикат*** или логическую функцию.
- Во-вторых, каждая строка в теле отношения r представляет собой определенное ***истинное высказывание***, полученное из предиката путем подстановки определенных значений аргументов соответствующего типа вместо местодержателей или параметров этого предиката.
- Например, в случае, показанном на слайде, предикат будет следующим:
- *Служащий с номером НомСлужащего по фамилии ИмяСлужащего работает в отделе с номером НомОтдела и получает зарплату Зарплата.*
- Здесь параметрами являются НомСлужащего, ИмяСлужащего, НомОтдела и Зарплата, которые соответствуют 4-м столбцам переменной-отношения Служащие. Примером соответствующего истинного высказывания может быть:
- *Служащий с номером ‘С1’ по фамилии ‘Иванов’ работает в отделе с номером ‘Отд1’ и получает зарплату ’21 тыс. ден. ед.’.*

Можно сказать, что:

- типы- объекты (множества объектов), которые можно обсуждать;
- отношения – факты (множества фактов), касающиеся объектов, которые можно обсуждать.
- Важно понимать, что каждое отношение имеет связанный с ним предикат, включая отношения, полученные с помощью реляционных операторов, например оператора соединения.
- **Пример 2.3.** Отношение «Отделы», представленное на рис. 2.1 и три результирующих отношения, представленные на рис. 2.2, имеют следующие предикаты:
- «Отделы»: «Отдел с номером *НомОтдела* называется *НазваниеОтд* и имеет бюджет *Бюджет*».
- Выборка строк из «Отделы», где *Бюджет* > 8М: «Отдел с номером *НомОтдела* называется *НазваниеОтд* и имеет бюджет *Бюджет,* который больше 8 миллионов денежных единиц».
- Извлечение столбцов НомОтдела и Бюджет из «Отделы»: «Отдел с номером *НомОтдела* имеет какое-то название и бюджет *Бюджет*».
- Соединение переменных-отношений «Отделы» и «Служащие» на основе столбца *НомОтдела*: «Отдел с номером *НомОтдела* называется *НазваниеОтд* и имеет бюджет *Бюджет*, а служащий с номером *НомСлужащего* по фамилии *ИмяСлужащего* работает в отделе с номером *НомОтдела* и получает зарплату *Зарплата*».

- ***Реляционная БД*** – это такая БД, которая воспринимается ее пользователями как множество переменных, значениями которых являются отношения.
- Поэтому реляционные системы иногда называют системами ***автоматической навигации***.
- Ответственность за то, как именно выполняется автоматическая навигация, несет компонент СУБД – ***оптимизатор***.

Базовые переменные-отношения и представления

Исходные (заданные) переменные-отношения называются ***базовыми переменными-отношениями***, а присвоенные им значения называются ***базовыми отношениями*** (или значениями базовых отношений).

Отношение, которое получено или может быть получено из базового отношения в результате выполнения каких-либо реляционных выражений, называется ***производным отношением***.

```
CREATE VIEW TOPEMP AS
    (EMP WHERE SALARY >= 19K) { EMP#, ENAME, SALARY }
```

EMP#	ENAME	DEPT#	SALARY
С1	Иванов	Отд1	21К
С2	Петров	Отд1	18К
С3	Никитин	Отд2	20К
С4	Буденко	Отд2	19К

- **(TOPEMP WHERE SALARY < 21K) { EMP#, SALARY }**
- **((EMP WHERE SALARY > = 19K) { EMP#, ENAME, SALARY}**
- **WHERE SALARY < 21K) { EMP#, SALARY }**
- **После определенного количества перегруппировок это выражение может быть упрощено и может принять следующий вид:**
- **(EMP WHERE SALARY > = 19K AND SALARY < 21K) { EMP#, SALARY }**
- **первоначальная операция над представлением конвертируется в эквивалентную операцию над соответствующей базовой переменной-отношением (при этом она *оптимизируется*).**

Транзакции

Транзакция – логическая единица работы, обычно включающая несколько операций над базой данных. Для пользователя должна иметься возможность указать системе, что отдельные операции являются частью одной транзакции. Для этого используются операции BEGIN TRANSACTION, COMMIT и ROLLBACK. Как правило, транзакция начинается при выполнении операции BEGIN TRANSACTION и прекращается при выполнении операции COMMIT или ROLLBACK.

```
BEGIN TRANSACTION /* Перевести деньги со счета A на счет B */
    UPDATE account A     /* Снятие денег со счета A */
    UPDATE account B     /* Помещение денег на счет B */
    IF <проверка условия> THEN
    COMMIT;              /* Нормальное завершение */
    ELSE ROLLBACK    /* Аварийное завершение (откат) */
ENDIF
```

Транзакции

Свойства транзакций:

- *Атомарность* – гарантия (с логической точки зрения), что операции будут выполнены полностью или не выполнены вовсе, даже если в системе до окончания процесса выполнения произойдет сбой.
- *Продолжительность* – гарантия того, что если транзакция успешно выполнила оператор COMMIT, то все выполненные ею изменения будут реализованы в БД, даже если в системе в какой-то момент произойдет сбой.
- *Изолированность.* Если в системе выполняется одновременно несколько транзакций, то все изменения, сделанные одной из них, не будут видны остальным, пока она не выполнит оператор COMMIT.
- *Упорядоченность*. При параллельном выполнении нескольких транзакций, операции которых чередуются между собой, гарантируется, что осуществление этих операций будет упорядоченным (serializable), т.е. результат будет таким же, как при строго последовательном выполнении этих же транзакций в произвольном порядке.

2.1 Обзор языка SQL
Определение БД поставщиков и деталей: типы данных

```
CREATE TABLE S
   (SN       CHAR(5),
   SNAME          CHAR(20),
   STATUS    NUMERIC(5),
   CITY      CHAR(15),
   PRIMARY KEY ( SN ) );
CREATE TABLE P
   (PN       CHAR(6),
   PNAME          CHAR(20),
   COLOR          CHAR(6),
   WEIGHT    NUMERIC(5,1),
   CITY      CHAR(15),
   PRIMARY KEY ( PN ) );
CREATE TABLE SP
   (SN       CHAR(5),
   PN        CHAR(6),
   QTY       NUMERIC(9),
   PRIMARY KEY (SN, PN),
   FOREIGN KEY (SN ) REFERENCES S,
   FOREIGN KEY (PN ) REFERENCES P);
```

CHARACTER (n) INTEGER DATE INTERVAL
BIT (n) SMALLINT TIME
NUMERIC (r, q) FLOAT (p) TIMESTAMP DECIMAL(p,q)

2.2 Каталог в языке SQL: Основные компоненты информационной схемы

SCHEMA
(схемы)
VIEWS
(представления)
TABLE_CONSTRAINTS
(ограничения для таблицы)

DOMAINS
(домены)
COLUMNS
(столбцы)
REFERENTIAL_CONSTRAINTS
(ссылочные ограничения)

TABLES
(таблицы)
DOMAIN_CONSTRAINTS
(ограничения для домена)
ASSERTIONS
(утверждения)

2.3 **Представления.**

Пример представления в языке SQL

Определение представления:

```
CREATE VIEW GOOD_SUPPLIER
   AS   SELECT SN, STATUS, CITY
        FROM S
        WHERE STATUS > 15.
```

Определение запроса к представлению:

```
SELECT SN, STATUS
FROM GOOD_SUPPLIER
WHERE CITY=’Москва’
```

Запрос после подстановки представления:

```
SELECT GOOD_SUPPLIER.SN, GOOD_SUPPLIER .STATUS
FROM (  SELECT SN, STATUS, CITY
   FROM S
   WHERE STATUS > 15) AS GOOD_SUPPLIER
WHERE GOOD_SUPPLIER.CITY = ’Москва’
```

Запрос после упрощения:

```
SELECT SN, STATUS
FROM S
WHERE STATUS > 15 AND CITY = ’
Москва’
```

2.4 **Транзакции**

- Для операторов COMMIT и ROLLBACK в языке SQL есть прямые аналоги. Это операторы COMMIT WORK и ROLLBACK WORK соответственно (в обоих случаях слово WORK – необязательное). Но в языке SQL нет явного оператора, соответствующего оператору BEGIN TRANSACTION. Неявно транзакция начинается всякий раз, когда выполняется оператор, способный «инициализировать транзакцию» (transaction-initiating), но только в том случае, когда никакая транзакция не выполняется. Таковыми операторами являются практически все операторы, которые мы обсуждаем в этом разделе.
- **Замечание.** Некоторые реализации языка SQL имеют в своем составе аналоги оператора BEGIN TRANSACTION, которые в явном виде позволяют задавать начало выполнения транзакции. Примером такой реализации может послужить СУБД Microsoft SQL Server 2000.

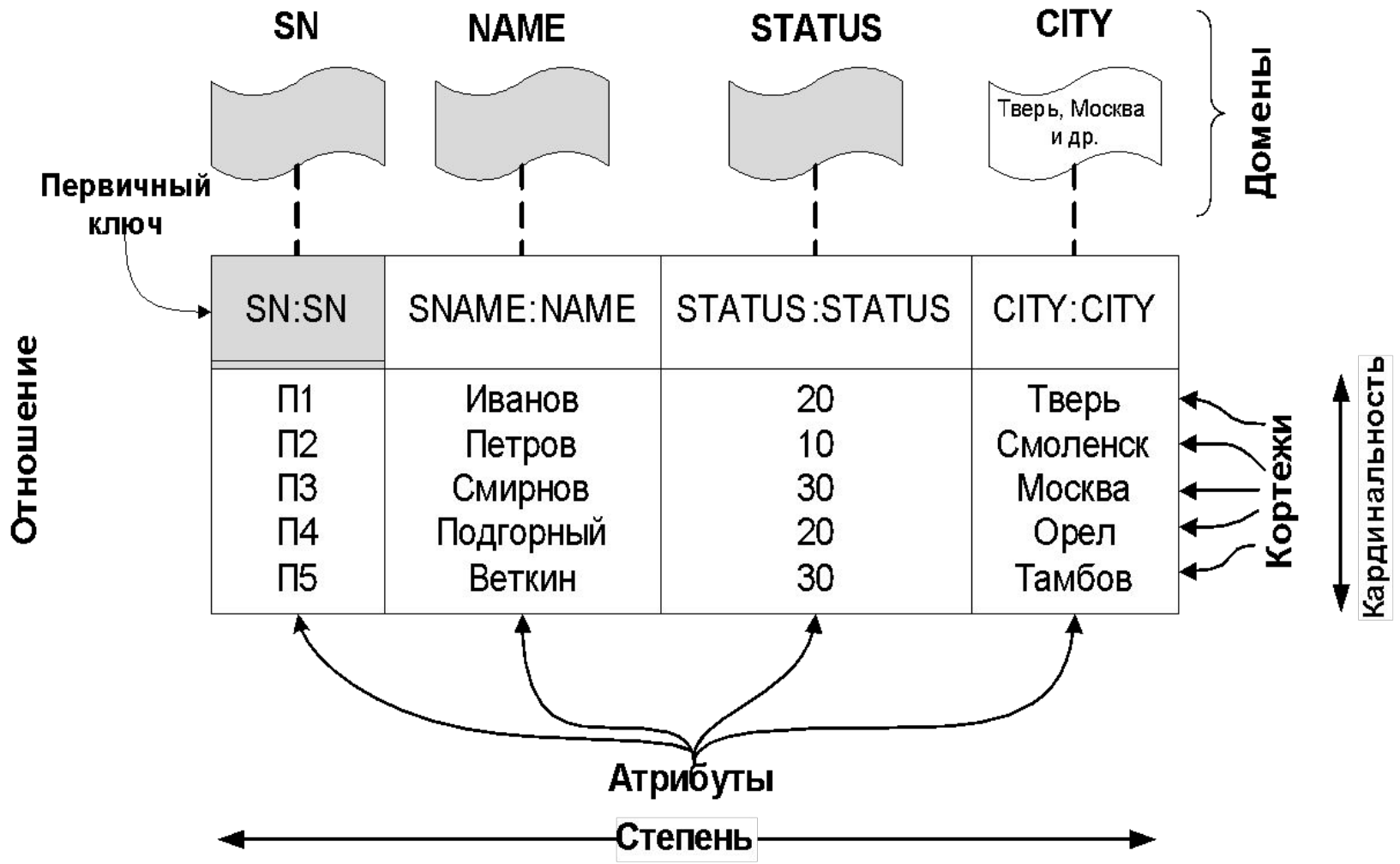
2.5 **Взаимодействие приложений и СУБД**

- Для реализации возможности взаимодействия прикладных приложений, написанных прикладными программистами, с реляционной СУБД, использующей язык SQL, возможны несколько вариантов:
- Первый вариант состоит во внедрении SQL-операторов в синтаксические конструкции того или иного языка и поддержка их на уровне компилятора, который осуществляет достаточно прозрачно взаимодействие с СУБД.
- Второй вариант представляет форму взаимодействия, когда с языком программирования поставляется несколько библиотек, которые реализуют возможность взаимодействия с СУБД

3. Домены, отношения и базовые переменные-отношения

- Основой современной технологии БД является реляционная модель.
- в реляционной модели рассматриваются три принципиальных аспекта данных –
- *структура* данных,
- *манипулирование* данными
- поддержание *целостности данных*.

Термины, используемые для описания структур данных



3.1 **Домены**

- Домен – это не что иное, как ***тип данных***. В частности, возможно, простой, ***определяемый системой***, подобно типам INTEGER и CHAR. В общем случае этот тип ***определяется пользователем***.
- Прежде всего, домен, или тип данных, это ***множество значений*** – всех возможных значений рассматриваемого типа. Например, тип INTEGER – это множество всех целых чисел. Говоря о каком-либо типе данных необходимо помнить об ***операторах***, которые могут корректно применяться к значениям этого типа. Другими словами, значениями заданного типа можно манипулировать только с помощью операторов, определенных для этого типа.
- Типы данных можно разделить на ***скалярные*** и ***нескалярные***.

3.2 **Значения отношений**

- Как уже отмечалось, следует различать собственно *отношения* и *переменные-отношения* (т.е. переменные, значениями которых являются отношения). В данном подразделе рассматриваются значения отношений. Прежде всего дадим точное определение термина *отношение*:

Отношение, заголовок, тело, атрибут, кортеж (определения)

Пусть задано множество из n типов или доменов T_i (i=1,2,…,n), причем все они необязательно должны быть различными. Тогда r будет ***отношением***, определенным на этих типах, если оно состоит из двух частей: *заголовка* и *тела*, где:

а) ***заголовок*** – это множество из n ***атрибутов*** вида $A_i:T_i$; здесь A_i – имена *атрибутов* (которые должны отличаться одно от другого) отношения r, а T_i – соответствующие *имена типов* (i=1,2,…,n).

б) ***тело*** – это множество из m ***кортежей*** t; здесь t в свою очередь, является множеством компонентов вида $A_i:V_i$, в которых V_i – значение типа T_i , т.е. *значение атрибута* для атрибута A_i в кортеже t (i=1,2,…,n).

Замечание: Следует отметить, что заголовок отношения также называется ***схемой*** отношения.
Значения m и n называются соответственно ***кардинальностью*** и ***степенью*** отношения r.

- отношение и таблица – это в действительности не одно и то же. ***Отношение*** – это некоторый абстрактный вид объекта, соответствующий данному ранее определению, а *таблица* – это конкретное изображение данного абстрактного объекта.
- **Свойства отношений:**
- **Отсутствие одинаковых кортежей**.
- **Отсутствие упорядочения кортежей (сверху вниз)**.
- **Отсутствие упорядочения атрибутов (слева направо)**.
- **Каждый кортеж содержит ровно одно значений для каждого атрибута**.

3.3 Средства SQL определения типов и структур данных

Используемые операторы:

CREATE DOMAIN	CREATE TABLE
ALTER DOMAIN	ALTER TABLE
DROP DOMAIN	DROP TABLE

Домены

```
CREATE DOMAIN <имя домена> <имя встроенного типа>
    [ <определение значения по умолчанию> ]
    [ <ограничения> ] ;

DROP DOMAIN <имя домена> <режим>;
```

Базовые таблицы

```
CREATE TABLE <имя базовой таблицы>
      ( <список элементов базовой таблицы>);


DROP TABLE <имя базовой таблицы> <режим>;
```

- Отличия между настоящими доменами и конструкциями языка SQL:
- Домены языка SQL – это просто синтаксические сокращения. Они не относятся к истинному типу данных, определяемых пользователем.
- Значения доменов языка SQL не могут быть «произвольной внутренней сложности». Их сложность ограничена сложностью встроенных типов.
- Домены языка SQL определяются в терминах одного из встроенных типов, а не в терминах другого домена, определенного пользователем.
- На практике каждый домен SQL должен определяться в терминах только одного из существующих встроенных типов.
- Домены языка SQL не могут иметь больше одного «допустимого» представления, которое определяется их физическим представлением.
- В языке SQL нет строго контроля типов и выполняемая проверка правильности типов совершенно незначительна.
- Язык SQL не содержит возможностей по определению пользователем операций, применяемых к данному домену. Допустимыми являются лишь встроенные операции, применимые к соответствующим представлениям этого типа.

- Форма записи:
- CREATE DOMAIN <имя домена> <имя встроенного типа>
- [<определение значения по умолчанию>]
- [<ограничения>] ;

Необязательный параметр *определение значения по умолчанию* задает значение по умолчанию, которое будет применяться к каждому столбцу, определенному на этом домене и не имеющему собственного явно заданного значения по умолчанию. Значение этого параметра должно иметь вид DEFAULT <*значение по умолчанию*>, где значение по умолчанию может быть как литералом, именем 0-адического оператора (без операндов, CURRENT_DATE) или значением NULL. В параметре *ограничения* указываются все ограничения, накладываемые на домен.

- оператор DROP DOMAIN, имеющего следующий синтаксис:
- DROP DOMAIN <имя домена> <режим>;
- Здесь параметр режим может принимать значения RESTICT или CASCADE.

- Оператор CREATE TABLE (обращаем внимание, что слово TABLE подразумевает только базовую таблицу, также как и в операторах ALTER TABLE, DROP TABLE).
- Синтаксис выражения следующий:
- CREATE TABLE <имя базовой таблицы>
- (<список элементов базовой таблицы>);

- Здесь каждый элемент базовой таблицы является либо *определением столбца*, либо *определением ограничения базовой таблицы*. В последнем случае элемент задает ограничение поддержки целостности данных, которое будет применяться к создаваемой таблице. Этот вопрос мы рассмотрим в последующих лекциях. Определение столбца, в свою очередь, выглядит следующим образом:
- <имя столбца> <тип или имя домена> [<значение по умолчанию>]
- Параметр <имя столбца> указывает название столбца, параметр <тип или имя домена> задает используемый тип данных иди домен, а необязательный параметр <значение по умолчанию> определяет значение по умолчанию для соответствующего столбца, которое подавляет значение по умолчанию, указанное для домена.

- Существующее определение базовой таблицы можно изменить в любое время с помощью оператора ALTER TABLE, который позволяет производить следующие изменения:

- Добавить столбец;
- Задать для существующего столбца новое значение по умолчанию;
- Удалить значение по умолчанию для существующего столбца;
- Удалить существующий столбец;
- Задать новые ограничения целостности;
- Удалить существующие ограничения целостности.

Пример для первого случая приведен ниже:

- ALTER TABLE S ADD COLUMN DISCOUNT INTEGER DEFAULT -1;

Этот оператор добавляет в базовую таблицу S столбец с именем DISCOUNT типа INTEGER со значением по умолчанию -1.

- существующую базовую таблицу можно уничтожить с помощью оператора DROP TABLE:
- DROP TABLE <имя базовой таблицы> <режим>;

  Параметр *режим* принимает те же значения RESTRICT и CASCADE, смысл которых аналогичен.

Вопросы на самоподготовку:

1. Реляционная модель данных. Основные черты. Строгое определение.
2. Отношения и переменные-отношения. Определение и смысл отношений. Примеры.
3. Оптимизация: цели основы для ее выполнения. Каталог: понятие и назначение. Транзакции: определение, назначение и способ организации.
4. Базовые переменные-отношения и представления. Производные отношения. Примеры.
5. Язык SQL: история, возможности, соотношение с реляционной моделью. Каталог в SQL: структура и состав. Представления и транзакции в SQL. Взаимодействие приложений с реляционными БД, динамический SQL.
6. Язык SQL: средства описания/изменения структуры данных и типов данных. Встроенные типы данных и домены. Примеры.

Тема № 1. Базы данных специального назначения

Лекция № 3: Реляционная алгебра. Реляционное исчисление. Средства языка SQL.

Учебные цели занятия:

Сформировать представление о:

1) Положениях реляционной алгебры и ее назначении,
2) Положениях реляционного исчисления и его назначении,
3) Средствах языка SQL манипулирования данными.
4) Ограничениях целостности используемых реляционной моделью

Учебные вопросы:

5) Реляционная алгебра
6) Реляционное исчисление
7) Целостность данных

1. Реляционная алгебра
1.1 Введение в реляционную алгебру

- **Выборка** - Возвращает отношение, содержащее все кортежи заданного отношения, которые удовлетворяют указанным условиям. Операцию выборки также иногда называют операцией ограничения.

- **Проекция** - Возвращает отношение, содержащее все кортежи (подкортежи) заданного отношения, которые остались в этом отношении после исключения из него нескольких атрибутов.

- **Произведение** - Возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям.

- **Объединение** - Возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим.

1. Реляционная алгебра
1.1 Введение в реляционную алгебру

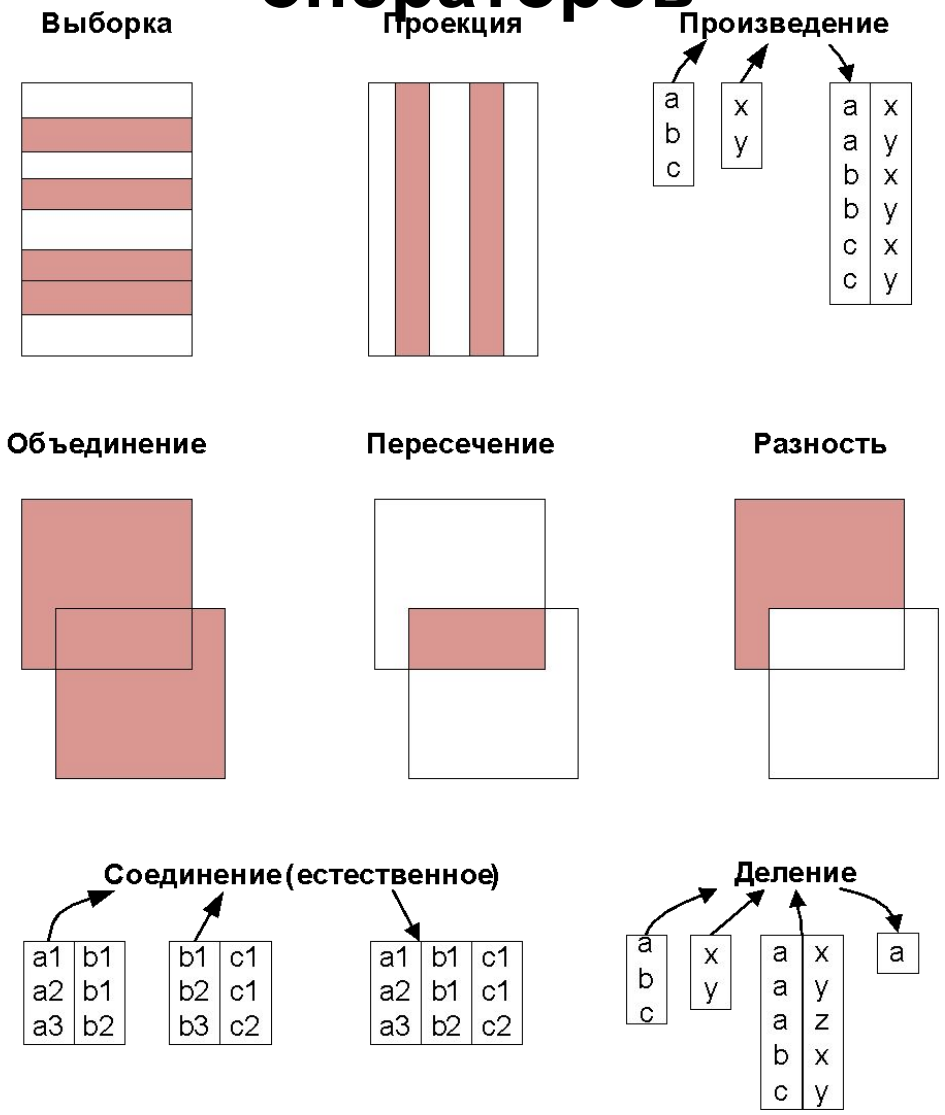
- **Пересечение** - Возвращает отношение, содержащее кортежи, которые принадлежат одновременно двум заданным отношениям.

- **Разность** - Возвращает отношение, содержащее кортежи, которые принадлежат первому отношению, но не принадлежат второму.

- **Соединение** - Возвращает отношение, содержащее все возможные кортежи, которые представляют собой комбинацию атрибутов двух кортежей, принадлежащих двум заданным отношениям, при условии, что в этих двух комбинируемых кортежах присутствуют одинаковые значения в одном или нескольких общих для исходных отношений атрибутах.

- **Деление** - Для заданных двух унарных отношений и одного бинарного возвращает отношение, содержащее все кортежи из первого унарного отношения, которые содержатся также в бинарном отношении и соответствуют всем кортежам во втором унарном отношении.

Графическая интерпретация восьми операторов



1.2 Реляционная замкнутость

- Результат выполнения любой операции над отношением также является отношением. Эта особенность является свойством ***реляционной замкнутости***.
- Благодаря этому свойству можно записывать ***вложенные реляционные выражения***, т.е. выражения, в которых операнды сами представлены реляционными выражениями, причем произвольной сложности.
- результат обязательно должен иметь определенный ***тип отношения***.

1.2 Реляционная замкнутость

- Необходим встроенный в реляционную алгебру набор ***правил вывода типов*** (отношений), чтобы выводить тип (отношения) на выходе произвольной реляционной операции, зная типы (отношения) на ее входе.
- Полезным в этом направлении является введение оператора переименования RENAME, который позволяет вернуть новое отношение, только указанные атрибуты которого имеют новые имена, а его значение остается прежним.
- P RENAME PNAME AS PN, WEIGTH AS WT
- Данный оператор позволяет устраниться от необходимости использования механизма уточнения имен атрибутов (P.WEIGHT, как в SQL).

1.3 Реляционная алгебра. Синтаксис (начало)

```
<реляционное выражение> ::= RELATION { <список выражений кортежей> }
| <имя переменной-отношения>
| <реляционная операция>
| ( <реляционное выражение> )

<реляционная операция> ::= <проекция> | <не проекция>
<проекция> ::= <реляционное выражение>
              { [ ALL BUT } <список имен атрибутов> }
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>.

```
<не проекция> ::= <переименование> | <объединение> | <пересечение>
| <вычитание> | <произведение> | <выборка>
| <соединение> | <деление>

<переименование> ::= <реляционное выражение>
RENAME <список переименовываемых элементов>
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>.

```
<объединение>::=<реляционное выражение> UNION <реляционное выражение>
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>, если только оба не объединения.

```
<пересечение> ::= <реляционное выражение> INTERSECT
<реляционное выражение>
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>, если только оба не пересечения.

Реляционная алгебра. Синтаксис (конец)

<вычитание>::= <реляционное выражение> MINUS <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>.

<произведение>::=<реляционное выражение> TIMES <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>,
если только оба не являются произведениями.

<выборка> ::= <реляционное выражение> WHERE <логическое выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>.
Логическое выражение может иметь ссылки на атрибуты отношения,
обозначенного как реляционное выражение.

<соединение>::= <реляционное выражение> JOIN <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>,
если только хотя бы одно не является соединением.

<деление> ::= <реляционное выражение> DIVIDEBY
<реляционное выражение> PER <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>.

Объединение

Для заданных отношений A и B одного и того же типа ***объединением*** этих двух отношений (A UNION B) называется новое отношение того же типа с телом, состоящим из множества всех кортежей t, которые принадлежат или отношению A, или отношению B, или обоим отношениям одновременно.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва

B

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь

Объединение (A UNION B)

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва
П2	Ильин	10	Тверь

Пересечение

Пересечением двух совместимых по типу отношений A и B (A INTERSECT B) называется отношение того же типа с телом, состоящим из множества всех кортежей t, которые принадлежат одновременно обоим исходным отношениям A и B.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва

B

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь

Пересечение (A INTERSECT B)

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва

Вычитание

Вычитанием двух совместимых по типу отношений A и B (A MINUS B) называется отношение того же типа с телом, состоящим из множества всех кортежей t, которые принадлежат отношению A, но не принадлежат отношению B.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва

B

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь

Вычитание (A MINUS B)

S#	SNAME	STATUS	CITY
П4	Иванов	20	Москва

Вычитание (B MINUS A)

S#	SNAME	STATUS	CITY
П2	Ильин	10	Тверь

Декартово произведение

Декартовым произведением двух отношений A и B (A TIMES B), где отношения A и B не имеют общих имен атрибутов, называется новое отношение с заголовком, представляющим объединение заголовков двух исходных отношений A и B, и с телом, состоящим из множества всех кортежей t, таких, что каждый кортеж t представляет собой объединение двух кортежей, один из которых принадлежит отношению A, а другой – отношению B.

Выборка

Пусть задано отношение A с атрибутами X и Y (и, возможно, с другими атрибутами), а символ Θ обозначает любой скалярный оператор сравнения, такой, что условие XΘY корректно определено при заданных значениях этих атрибутов и дает значение *истина* или *ложь*.

Тогда **Θ-выборкой** из отношения A по атрибутам X и Y называется отношение, имеющее тот же заголовок, что и отношение A, и тело, содержащее множество всех кортежей t отношения A, для которых проверка условия XΘY дает значение *истина*.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь
П3	Коробов	15	Смоленск
П4	Иванов	20	Москва

A WHERE CITY = ‘Москва’ OR STATUS > 14

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П3	Коробов	15	Смоленск
П4	Иванов	20	Москва

Проекция

Пусть задано отношение A с атрибутами X, Y, …, Z (и, возможно, другими). Тогда проекцией отношения A по атрибутам X, Y, …, Z (A {X, Y, …, Z}) называется отношение, удовлетворяющее следующим требованиям:

1) Его заголовок получается из заголовка отношения A посредством удаления из него всех атрибутов, не входящих в множество {X, Y, …, Z}.

2) Его тело содержит множество всех кортежей вида {X:x, Y:y, …, Z:z}, таких для которых в отношении A значение атрибута X равно x, значение атрибута Y равно y, …, значение атрибута Z равно z.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь
П3	Коробов	15	Смоленск
П4	Иванов	20	Москва

A {STATUS, CITY } = A { ALL BUT S#, SNAME }

STATUS	CITY
20	Москва
10	Тверь
15	Смоленск

Соединение

Пусть даны два отношения A и B имеют соответственно заголовки

{X1, X2, …, Xm, Y1, Y2,…, Yn}

и

{ Y1, Y2,…, Yn, Z1, Z2, …, Zp}.

Пусть X, Y и Z являются соответствующими составными атрибутами {X1, X2, …, Xm}, {Y1,Y2,…,Yn} и { Z1, Z2, …, Zp}. Тогда ***естественным соединением*** отношений A и B (A JOIN B) называется отношение с заголовком {X, Y, Z} и телом, содержащим множество всех кортежей вида {X:x,Y:y, Z:z}, таких, для которых в отношении A значение атрибута X равно x, а значение атрибута Y равно y, и в отношении B значение атрибута Y равно y, а значение атрибута Z равно z.

Деление

Пусть отношения A и B имеют заголовки {X1, X2, …, Xm} и {Y1,Y2,…,Yn} соответственно. Пусть также имеется отношение C с заголовком {X1, X2, …, Xm,Y1,Y2,…,Yn}. Пусть X, Y являются соответствующими составными атрибутами {X1, X2, …, Xm} и {Y1,Y2,…,Yn}.

Тогда результатом ***деления*** отношения A на отношение B по соотношению C (A DIVIDEBY B PER C) называется отношение с заголовком {X} и телом, содержащим множество всех кортежей вида {X:x}, таких, что кортеж вида {X:x, Y:y} принадлежит отношению C для всех кортежей вида {Y:y}, принадлежащих отношению B.

A

S#
П1
П2
П3
П4

B

D#
Д1
Д2
Д3
Д4

C

S#	D#
П1	Д1
П1	Д2
П1	Д3
П1	Д4
П2	Д1
П2	Д2
П3	Д2

A DIVIDEBY B PER C

S#
П1

1.5 Реляционная алгебра. Примеры

Получить имена поставщиков детали с номером ‘P2’:

```
( (SP JOIN S) WHERE P# = ‘P2’ ) {SNAME}
```

Получить имена поставщиков по крайней мере одной красной детали:

```
( ( (P WHERE COLOR = ‘Красный’) JOIN SP ){S#}
  JOIN S) {SNAME}
```

Получить имена поставщиков всех типов деталей:

```
( (S{S#} DIVIDEBY P{P#} PER SP{S#,P#})
  JOIN S) {SNAME}
```

Получить имена поставщиков, которые не поставляют деталь с номером ‘P2’:

```
( (S{S#} MINUS (SP WHERE P#=’P2’){S#} )
  JOIN S) {SNAME}
```

1.6 Назначение реляционной алгебры

- Основная цель реляционной алгебры – ***обеспечить запись реляционных выражений***.

Некоторые из возможных применений подобных выражений:

- Определение области выборки
- Определение области обновления
- Определение правил поддержки целостности данных
- Определение производных переменных-отношений
- Определение требований устойчивости, т.е. данных, которые должны быть включены в контролируемую область для некоторых операций управления параллельным доступом к информации.
- Определение ограничений защиты, т.е. данных, для которых осуществляется тот или иной тип контроля доступа.

Выражения реляционной алгебры служат для *символического высокоуровневого представления намерений пользователя*.

- Данными выражениями можно манипулировать в соответствии с различными символическими высокоуровневыми ***правилами преобразования***.
- Запрос ((SP JOIN S) WHERE P#=’P2’) {SNAME}
- может быть преобразован в более рациональное выражение вида:
- ((SP WHERE P#=’P2’) JOIN S) {SNAME}
- Таким образом, реляционная алгебра может быть хорошим основанием для выполнения ***оптимизации*** (что должно производиться оптимизатором автоматически).
- В общем случае язык называют ***реляционно полным***, если его возможности, по крайней мере, соответствуют возможностям, обеспечиваемым алгебраическими операциями, т.е. выражения этого языка позволяют определить каждое отношение, которое может быть определено с помощью алгебраических выражений

2. Реляционное исчисление

2.1 Введение в реляционное исчисление

- часть реляционной модели, которая связана с операторами манипулирования данными, основывается на использовании реляционной алгебры
- Однако можно сказать, что она построена на базе *реляционного исчисления*.
- реляционная алгебра и реляционное исчисление представляют два альтернативных подхода.
- в реляционной алгебре предоставляется в явном виде набор операторов для формирования требуемого отношения
- в реляционном исчислении имеется система обозначений для определения требуемого отношения в терминах данных отношений.

Пример

- В качестве примера рассмотрим следующий запрос: «Выбрать номера поставщиков и названия городов, в которых находятся поставщики детали с номером ‘P2’».
- Алгебраическая версия запроса выглядит следующим образом: 1) сначала выполнить соединение отношения поставщиков S и отношения поставок по атрибуту S#; 2) Выбрать из результата соединения кортежи с номером детали ‘P2’; 3)Выполнить проекцию для результата этой выборки по атрибутам S# и CITY.
- В терминах реляционного исчисления запрос формулируется следующим образом:

  Получить атрибуты S# и CITY для таких поставщиков S, для которых в отношении SP существует запись о поставке с тем же значением атрибута S# и со значением атрибута P#, равным ‘P2’.

  Т.е. указываются лишь некоторые характеристики требуемого результата, оставляя системе решать, что именно и в какой последовательности соединять, проецировать и т.д., чтобы получить необходимый результат.

Реляционное исчисление носит описательный характер, а реляционная алгебра – предписывающий, т.е. не описывается, в чем заключается проблема, а задается процедура решения этой проблемы.

- Реляционное исчисление основано на разделе математической логики, которое называется исчислением ***предикатов***.
- Основным понятием реляционного исчисления является понятие ***переменной кортежа*** – переменная, «изменяющаяся на» некотором заданном отношении, т.е. переменная, допустимыми значениями для которой являются кортежи заданного отношения.
- Другими словами, если переменная кортежа V изменяется в пределах отношения r, то в любой заданный момент времени переменная V представляет некоторый кортеж t отношения r.
- В связи с тем, что реляционное исчисление основано на переменных кортежа, его первоначальную версию называют также ***исчислением кортежей***.

2.Реляционная исчисление. 2.2 Исчисление кортежей. Синтаксис (начало)

```
<реляционное выражение> ::= RELATION { <список выражений кортежей> }
| <имя переменной-отношения>
| <реляционная операция>
| ( <реляционное выражение> )
```

Определение реляционного выражения осталось прежним, но <реляционная операция> имеет иное определение.

```
<определение переменной кортежа> ::=
   RANGEVAR <имя переменной кортежа>
   RANGES OVER <список реляционных выражений>;
```

<Имя переменной кортежа> может использоваться в следующих случаях:

- Перед точкой и последующим уточнением в параметре <ссылка на атрибут кортежа>;
- Сразу после квантора в параметре <логическое выражение с квантором>;
- Как операнд в параметре <логическое выражение>;
- Как параметр <прототип кортежа> или как подпараметр <выражение> в параметре <прототип кортежа>.

```
<ссылка на атрибут кортежа> ::=
<имя переменной кортежа>.<ссылка на атрибут> [AS <имя атрибута>]
```

2. Реляционная исчисление. Синтаксис (конец)

Параметр <ссылка на атрибут кортежа> может использоваться как параметр <выражение>, но только в определенном контексте:

- Как операнд параметра <логическое выражение>;
- Как параметр <прототип кортежа> или как подпараметр <выражение> в параметре <прототип кортежа>.

```
<логическое выражение> ::=… все обычные возможности |
<логическое выражение с квантором>

<логическое выражение с квантором> ::=
   EXISTS <имя переменной кортежа> ( <логическое выражение> )
   | FORALL <имя переменной кортежа> (< логическое выражение > )

<реляционная операция> ::= <прототип кортежа>
              [WHERE <логическое выражение>]

<прототип кортежа> ::= <выражение кортежа>
```

Переменные кортежей

- Приведем примеры определения переменных кортежей для БД поставщиков и деталей:
- RANGEVAR SX RANGES OVER S
- RANGEVAR SY RANGES OVER S
- RANGEVAR SPX RANGES OVER SP
- RANGEVAR SPY RANGES OVER SP
- RANGEVAR PX RANGES OVER P

  RANGEVAR SU RANGES OVER
  (SX WHERE SX.CITY = ‘Москва’),
  (SX WHERE EXISTS SPX (SPX.S# = SX.S# AND
  SPX.P# = ‘P1’))
- Переменная кортежа SU определенная на объединении множества поставщиков, находящихся в Москве, и множества кортежей поставщиков детали с номером ‘P1’. Конечно, отношения при их объединении должны быть совместимы по типу.
- **Замечание**. Переменные кортежей не являются переменными в обычном смысле, а скорее представляют некоторый аналог местодержателям, или параметрам, предикатов, а, следовательно, являются переменными в логическом смысле.

Свободные и связанные переменные кортежей

- Каждая ссылка на переменную кортежа является либо ***свободной***, либо ***связанной***.
- Пусть V – переменная кортежа, тогда:
- Ссылки на переменную V в логических выражениях типа NOT p свободны или связаны в пределах этого выражения в зависимости от того, свободны они или нет в формуле p. Ссылки на переменную V в логических выражениях типа (p AND q) и (p OR q) свободны или связаны в зависимости от того, свободны ли они в выражениях p и q.
- Ссылки на переменную V, которые свободны в логическом выражении p, связаны в логических выражениях типа EXISTS V(p) и FORALL V(p) в соответствии с тем, свободны ли они в формуле p

Пример

Приведем некоторые примеры свободных и связанных переменных кортежей:

- Примеры свободных переменных кортежей:

  SX.S# = ’П1’

  SX.S# = SPX.S#

  NOT (SX.CITY = ’Москва’)

  SX.S#=SPX.S# AND SPX.P# <> PX.P#

  PX.COLOR = ‘Красный’ OR PX.CITY = ’Москва’
- Примеры связанных переменных кортежей:

  EXISTS SPX (SPX.S#=SX.S# AND SPX.P#=’P2’)

  FORALL PX (PX.COLOR=’Красный’)

Кванторы

- Существует два квантора: EXISTS и FORALL.
- Квантор EXISTS является квантором существования, а FORALL – квантором всеобщности.
- Если выражение p – логическое выражение, в которой переменная V свободна, то выражения EXISTS V(p) и FORALL V(p) также являются допустимыми логическими выражениями, но переменная V в них обеих будет связанная.
- Первая формула означает: «Существует, по крайней мере, одно значение переменной V, такое, что вычисление выражения p дает для него значение *истина*». Второе выражение означает: «Для всех значений переменной V вычисление выражения p дает для него значение *истина*».

Пример

- Рассмотрим следующий квантор существования:

  EXISTS SPX (SPX.S#=SX.S# AND SPX.P#=’P2’)

- Данное выражение может быть прочитано следующим образом:

  В текущем значении переменной-отношения SP существует, по крайней мере, один кортеж (скажем, SPX), такой, для которого значение атрибута S# в этом кортеже равно значению атрибута SX.S# (какое бы оно ни было), а значение атрибута P# в кортеже SPX равно ‘P2’.

2.3 Примеры использования исчисления кортежей

1. Определить номера поставщиков из Твери со статусом, большим 20.
(SX.S#, SX.STATUS) WHERE SX.CITY = ‘Тверь’ AND SX.STATUS > 20

2. Определить имена поставщиков детали с номером ‘P2’.
SX.SNAME WHERE EXISTS SPX(SPX.S#=SX.S# AND SPX.P#=’P2’)

3. Определить имена поставщиков по крайней мере одной красной детали.
SX.SNAME WHERE EXISTS SPX (SX.S#=SPX.S# AND
EXISTS PX (PX.P# = SPX.P# AND PX.COLOR = ‘Красный’))

4. Найти имена поставщиков по крайней мере одной детали, поставляемой поставщиком с номером ‘П2’.
SX.SNAME WHERE
EXISTS SPX (EXISTS SPY(SX.S# = SPX.S# AND
SPX.P# = SPY.P# AND SPY.S# = ‘П2’))

5. Выбрать имена поставщиков всех типов деталей.
SX.SNAME WHERE FORALL PX (EXISTS SPX (SPX.S# = SX.S# AND SPX.P# = PX.P#))

6. Определить имена поставщиков, которые не поставляют деталь с номером ‘P2’.
SX.SNAME WHERE NOT EXISTS SPX
(SPX.S# = SX.S# AND SPX.P# = ‘P2’)

2.4 Средства языка SQL (начало)

1. Указать цвета и названия городов, в которых находятся детали «не из Твери» с весом, превышающим 10 кг.

```
SELECT PX.COLOR, PX.CITY
FROM P AS PX
WHERE PX.CITY <> ‘Тверь’ AND PX.WEIGHT > 10
```

2. Для всех деталей указать номер и вес в фунтах

```
SELECT P.P#, P.WEIGHT / 0.454 AS WF
FROM P
```

3. Выбрать информацию обо всех парах поставщиков и деталей, находящихся в одном городе

```
SELECT S.*, P.P#, P.PNAME, P.COLOR, P.WEIGHT
FROM S, P
WHERE S.CITY = P.CITY
```

4. Определить общее количество поставщиков

```
SELECT COUNT(*) AS N
FROM S
```

Средства языка SQL (продолжение)

5. Для каждой поставляемой детали указать номер и общий объем поставки в штуках

```
SELECT SP.P#, SUM(SP.QTY) AS TOTQTY
FROM SP
GROUP BY SP.P#
```

6. Указать номера всех типов деталей, поставляемых более чем одним поставщиком

```
SELECT SP.P#
FROM SP
GROUP BY SP.P#
HAVING COUNT(SP.S#) > 1;
```

7. Определить имена поставщиков детали с номером ‘P2’

```
SELECT DISTINCT S.SNAME
FROM S
WHERE S.S# IN
      (SELECT SP.S#
      FROM SP
      WHERE SP.P# = ‘P2’);
```

Средства языка SQL (продолжение)

8. Определить имена поставщиков, по крайней мере, одной красной детали

```
SELECT DISTINCT S.SNAME
FROM S
WHERE S.S# IN     (SELECT SP.S#
            FROM SP
            WHERE SP.P# IN    (SELECT P.P#
                        FROM P
                        WHERE P.COLOR=’Красный’));
```

9. Указать имена поставщиков, статус которых меньше текущего максимального статуса в таблице S

```
SELECT S.S#
FROM S
WHERE S.STATUS < (SELECT MAX(S.STATUS) FROM S)
```

10. Выбрать имена поставщиков, которые не поставляют деталь с номером ‘P2’

```
SELECT DISTINCT S.SNAME
FROM S
WHERE NOT EXISTS
      (SELECT *
      FROM SP
      WHERE SP.S#=S.S# AND SP.P#=’P2’)
```

Средства языка SQL (конец)

11. Определить имена поставщиков все типов деталей

```
SELECT DISTINCT S.SNAME
FROM S
WHERE NOT EXISTS
      (SELECT *
      FROM P
      WHERE NOT EXISTS
            (SELECT *
            FROM SP
            WHERE SP.S#=S.S# AND SP.P#=P.P#)
```

Типы (категории) ограничений целостности данных

Ограничения целостности можно классифицировать по четырем основным категориям:

- *Ограничения целостности типа*, в которых задаются допустимые значения для данного типа.
- *Ограничения целостности атрибута*, в которых задаются допустимые значения для данного атрибута.
- *Ограничения целостности переменной-отношения*, в которых задаются допустимые значения для переменной-отношения.
- *Ограничения целостности БД*, в которых задаются допустимые значения для БД.

Ограничения переменной-отношения и БД. Примеры

Примеры ограничений переменной-отношения:

«Поставщики в Твери должны обладать статусом, равным 20»:

```
CONSTRAINT SC5
    IS_EMPTY ( S WHERE CITY = 'Тверь' AND STATUS <> 20 ).
```

«Номера поставщиков должны быть уникальны» или «Ключ {S#} – это потенциальный ключ отношения поставщиков»:

```
CONSTRAINT SCK
    COUNT ( S ) = COUNT ( S { S#} )
```

«Если детали вообще имеются, то одна из них должна быть красной»:

```
CONSTRAINT PC1
    IF NOT ( IS_EMPTY( P ) ) THEN
        COUNT ( P WHERE COLOR = 'Красный') > 0
    END IF
```

Примеры ограничений БД:

«Поставщики со статусом, меньшим 20, не могут поставлять детали в количестве свыше 500 штук»:

```
CONSTRAINT DBC1
    IS_EMPTY( (S JOIN SP)
            WHERE STATUS < 20 AND QTY > 500)
```

«Каждая деталь должна быть поставлена хотя бы один раз»:

```
CONSTRAINT DBC2 SP {P#} = P{P#}
```

«Золотое правило»

Вариант 1:
Ни одна из операций изменения не имеет права переводить переменную-отношение в состояние, нарушающее ее собственный предикат.

Вариант 2 (уточненный):
Ни одна из операций изменения не имеет права переводить переменную-отношение в состояние, нарушающее ее собственный предикат. Аналогично ни одна из транзакций изменения не имеет права переводить БД в состояние, нарушающее ее собственный предикат.

Потенциальные ключи

Пусть K – множество атрибутов переменной-отношения R. В этом случае множество K будет ***потенциальным ключом*** переменной-отношения R тогда и только тогда, когда оно обладает следующими свойствами:

а) ***Уникальность***. Никакие допустимые значения переменной-отношения R не содержат двух различных кортежей с одинаковыми значениями атрибутов множества K.

б) ***Неизбыточность***. Никакое из собственных подмножеств множества K не обладает свойством уникальности.

Суперключом называется некоторое надмножество потенциального ключа. Суперключ обладает свойством уникальности, но не обладает свойством неизбыточности.

Пример:

```
VAR MARRIAGE BASE RELATION {
    HUSBAND      /* Муж */    NAME,
    WIFE     /* Жена */   NAME,
    DATE     /* Дата бракосочетания */  DATE }
/* Подразумевается, что муж может иметь одну жену, а жена одного мужа,
причем не допускается повторного брака между одними и теми людьми */
        KEY { HUSBAND, DATE }
        KEY { DATE, WIFE }
        KEY { WIFE, HUSBAND }
```

Внешние ключи

Пусть R2 – некоторая переменная-отношение. Тогда ***внешний ключ*** (скажем, FK) в переменной-отношении R2 представляет собой множество атрибутов этой переменной-отношения, такое, что:

а) существует переменная-отношение R1 (причем переменные-отношения R1 и R2 необязательно различны) с потенциальным ключом CK;

б) каждое значение внешнего ключа FK в текущем значении переменной-отношения R2 обязательно совпадает со значением ключа CK некоторого кортежа в текущем значении переменной-отношения R1.

Ссылочная целостность – ограничение целостности на то, что БД не должна содержать внешних ключей, не имеющих соответствия.

Ограничения целостности в SQL (начало)

Ограничения домена

CREATE DOMAIN COLOR CHAR(6) DEFAULT ‘???’
    CONSTRAINT VALID_COLORS CHECK (VALUE IN
        (‘Красный’, ‘Желтый’, ‘Синий’,’Зеленый’,’???’)

Ограничения базовой таблицы

Потенциальные ключи

UNIQUE (<список имен столбцов>) или для первичного ключа:
PRIMARY KEY (<список имен столбцов>)

Внешние ключи

FOREIGN KEY (<список имен столбцов>)
    REFERENCES <имя базовой таблицы> [<список имен столбцов>]
    [ON DELETE <ссылочная операция>]
    [ON UPDATE <ссылочная операция>]

Проверочные условия

CHECK (<условное выражение>)

Пример.
```
CREATE TABLE SP (     S# S# NOT NULL, P# P# NOT NULL, QTY QTY NOT NULL,
        PRIMARY KEY (S#, P#),
        FOREIGN KEY (S#) REFERENCES (S)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (P#) REFERENCES (P)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        CHECK (QTY > 0 AND QTY < 5001) );
```

Ограничения целостности в SQL (конец)

Утверждения

CREATE ASSERTION <имя ограничения>
CHECK (<условное выражение>);

Для отмены общего ограничения используется оператор DROP ASSERTION:
DROP ASSERTION <имя ограничения>;

Примеры:

1. Каждый поставщик должен иметь статус не менее 5.

```
CREATE ASSERTION AS1 CHECK
    ( (SELECT MIN (S.STATUS) FROM S) > 4 );
```

2. Значение веса любой детали должно быть положительным:

```
CREATE ASSERTION AS2 CHECK
    ( NOT EXISTS ( SELECT * FROM P
            WHERE NOT (P.WEIGHT > 0) ) );
```

3. Поставщики со статусом меньшим 20, не имеют права поставлять любую деталь в количестве более 500 штук:

```
CREATE ASSERTION AS3 CHECK
    ( NOT EXISTS ( SELECT * FROM S, SP
            WHERE S.STATUS < 20 AND S.S# = SP.S#
            AND SP.QTY > 500) );
```

Вопросы на самоподготовку:

1. Реляционная алгебра. Операторы. Реляционная замкнутость. Примеры.
2. Реляционная алгебра. Семантика операторов. Назначение реляционной алгебры. Примеры.
3. Реляционное исчисление. Исчисление кортежей. Переменные кортежей. Свободные и связанные переменные. Кванторы. Примеры.
4. Средства языка SQL манипулирования данными: Запросы SQL. Структура запроса. Вложенные подзапросы. Обобщающие функции. Примеры.
5. Средства языка SQL манипулирования данными: Запросы SQL. Структура запроса. IN-условия. Кванторы. Примеры.
6. Ограничения целостности данных. Типы ограничений целостности. Ограничения целостности типа и атрибута. «Золотое правило». Триггеры.
7. Ограничения целостности данных. Типы ограничений целостности. Ограничения целостности переменной-отношения и БД. Ключи.
8. Средства языка SQL поддержания ограничений целостности данных: Ограничения домена, базовой таблицы и утверждения. Операторы языка SQL. Примеры.

Тема № 1. Базы данных специального назначения

Лекция № 4: Нормализация баз данных

Учебные цели занятия:

Сформировать представление о:

1) Функциональных зависимостях и их назначении,
2) Процессе нормализации и его назначении,
3) Нормальных формах 1НФ, 2НФ, 3НФб НФБК
4) Процедурах нормализации (до 3НФ, НФБК)

Учебные вопросы:

5) Функциональные зависимости
6) Нормализация: формы 1НФ, 2НФ, 3НФ и НФБК
7) Нормализация: более высокие нормальные формы

1. Функциональная зависимость

Пусть R является переменной-отношением, а X и Y – произвольными подмножествами множества атрибутов переменной-отношения R. Тогда Y ***функционально зависимо*** от X, что в символическом виде записывается как:

$X \rightarrow Y$

(читается либо как «X ***функционально определяет*** Y», либо «X стрелка Y») тогда и только тогда, когда для любого допустимого значения переменной-отношения R каждое значение множества X отношения R связано в точности с одним значением множества Y отношения R. Иначе говоря, для любого допустимого значения переменной-отношения R, если два кортежа переменной-отношения R совпадают по значению X, они также совпадают и по значению Y.

1.2 Основные определения

- Левая и правая части функциональной зависимости (X и Y соответственно), называют ***детерминантом*** и ***зависимой частью соответственно***
- Следует отметить, что если X является потенциальным ключом переменной-отношения R, то все атрибуты Y переменной-отношения R должны обязательно быть функционально зависимы от X.
- Если переменная-отношение R удовлетворяет функциональной зависимости A → B и A не является потенциальным ключом, то R будет характеризоваться некоторой ***избыточностью***. Например, в переменной-отношении SCP наличие ФЗ S# → CITY приведет к тому, что сведения о месте расположения поставщика в определенном городе повторятся много раз

Функциональные зависимости (примеры)

SCP

S#	CITY	P#	QTY
S1	Москва	P1	100
S1	Москва	P2	100
S2	Тверь	P1	200
S2	Тверь	P2	200
S3	Тверь	P2	300
S4	Москва	P2	400
S4	Москва	P4	400
S4	Москва	P5	400

Примеры функциональных зависимостей для SCP:

{S#,P#} → QTY
{S#,P#} → CITY
{S#,P#} → {CITY, QTY}
{S#,P#} → S#
{S#,P#} → {S#, P#, CITY, QTY}
{S#} → CITY

1.3 Тривиальные и нетривиальные зависимости

- Зависимость называется тривиальной, если она не может не выполняться. В качестве примера приведем тривиальную ФЗ, существующую в переменной-отношении SCP, которая обсуждалась в предыдущем разделе:
- {S#, P#} → S#
- Функциональная зависимость называется ***тривиальной*** тогда и только тогда, когда первая часть ее символической записи является подмножеством (не обязательно собственным) левой части. Формальное определение выглядит следующим образом:
- Функциональная зависимость называется ***тривиальной*** тогда и только тогда, когда ее зависимая часть является подмножеством (не обязательно собственным) детерминанта.
- С практической точки зрения такие зависимости не представляют никакого интереса – в отличие от ***нетривиальных*** зависимостей, которые действительно являются реальными ограничениями целостности. Однако в формальной теории зависимостей необходимо учитывать все зависимости, как тривиальные, так и нетривиальные

1.4 Замыкание множества зависимостей

- Как уже упоминалось, одни ФЗ могут подразумевать другие ФЗ. Например, рассмотрим приведенную ниже ФЗ.
- {S#, P#} → {CITY, QTY}
- Она подразумевает следующие ФЗ:
- {S#, P#} → CITY
- {S#, P#} → QTY
- В качестве более сложного примера рассмотрим переменную-отношение R с атрибутами A, B, C, для которых выполняются ФЗ A → B и B → C. Нетрудно заметить, что в этом случае выполняется и ФЗ A → C, которая называется ***транзитивной*** ФЗ, т.е. C зависит от A транзитивно, через B.
- Множество всех ФЗ, которые подразумеваются заданным множеством ФЗ S, называется замыканием множества S и обозначается символом S+.
- Из этого определения следует, что необходим способ вычисления S+ на основе S.

Правила вывода (*аксиомы Армстронга)*

1. Правило ***рефлексивности***: если множество B является подмножеством множества A, то A → B.
2. Правило ***дополнения***: если A → B, то AC → BC.
3. Правило ***транзитивности***: если A → B и B → C, то A → C.
4. Правило ***самоопределения***: A → A.
5. Правило ***декомпозиции***: если A → BC, то A → B и A → C.
6. Правило ***объединения***: если A → B и A → C, то A → BC.
7. Правило ***композиции***: если A → B и C → D, то AC → BD.
8. Если A → B и C → D, то A ∪ (C - B)→ BD. (*Общая теорема объединения*)

- **Пример 4.2.** Пусть дана переменная-отношение R с атрибутами A, B, C, D, E, F и следующими ФЗ:
- A → BC
- B → E
- CD → EF
- Можно показать, что для переменной-отношения R выполняется ФЗ AD → F, которая вследствие этого принадлежит замыканию множества ФЗ.
- A → BC (дано)
- A → C (декомпозиция из п.1)
- AD → CD (дополнение из п.2)
- CD → EF (дано)
- AD → EF (транзитивность из пп.3 и 4)
- AD → F (декомпозиция из п.5)

1.5 Замыкание множества атрибутов

```
CLOSURE[Z,S] := Z;
do <бесконечно>
     for each FD X → Y in S          /* для каждой ФЗ X → Y в S */
     do
          if X ≤ CLOSURE[Z,S]   /* ≤ = <является подмножеством> */
          then CLOSURE[Z,S] := CLOSURE[Z,S] ∪ Y;
     end;
     if CLOSURE[Z,S] <не изменилось в этой итерации>
          then leave loop; /* вычисления завершаются */
end;
```

для заданной переменной-отношения R, заданного множества ФЗ S, выполняющихся для переменной-отношения R, можно найти множество всех атрибутов переменной-отношения R, которые функционально зависимы от Z, т.е. так называемое *замыкание Z+ множества Z в пределах S+.*

1.5 Замыкание множества атрибутов (пример)

- **Пример 4.3.** Пусть дана переменная-отношение R с атрибутами A, B, C, D, E, F и следующими ФЗ:
- $A \rightarrow BC$
- $E \rightarrow CF$
- $B \rightarrow E$
- $CD \rightarrow EF$
- Вычислим замыкание {A,B}+, исходя из указанного множества ФЗ:
- Присвоим замыканию CLOSURE[Z,S] начальное значение – множество {A,B}.

1.5 Замыкание множества атрибутов (пример)

- Выполним внутренний цикл четыре раза – по одному для каждой ФЗ. На первой итерации будет обнаружено, что левая часть действительно является подмножеством замыкания CLOSURE[Z,S]. Таким образом, к результату добавляются атрибуты B и C. В результате замыкание CLOSURE[Z,S] представляет собой множество {A,B,C}.
- На второй итерации (E → CF) к замыканию не добавляется атрибутов.
- На третьей итерации (B → E) к замыканию CLOSURE[Z,S] добавляется атрибут E. Теперь замыкание CLOSURE[Z,S] представляет собой множество {A,B,C,E}.
- На четвертой итерации (CD → EF) к замыканию не добавляется атрибутов.
- Далее внутренний цикл выполняется еще четыре раза, в ходе чего на второй итерации (E → CF) к замыканию добавится атрибут F.
- После еще одного четырехкратного прохождения внутреннего цикла замыкание не изменится, и процесс завершится с результатом {A,B}+={A,B,C,E,F}.
- Данный алгоритм представляет простой способ определения, будет ли данная ФЗ X→Y в замыкание S+ множества S.

1.6 Неприводимые множества зависимостей

Множество ФЗ S называется ***неприводимым*** тогда и только тогда, когда оно обладает всеми перечисленными ниже свойствами:

Зависимая часть каждой ФЗ из множества S содержит только один атрибут (т.е. является одноэлементным множеством).

Детерминант каждой ФЗ из множества S является неприводимым, т.е. ни один атрибут из детерминанта не может быть удален без изменения замыкания S^+. В этом случае ФЗ называется ***неприводимой слева***.

Ни одна ФЗ из множества S не может быть удалена без изменения его замыкания S^+.

- Пусть S1 и S2 – два множества ФЗ. Если любая ФЗ, которая выводится из множества ФЗ S1, также выводится из множества ФЗ S2 (т.е. S1+ - подмножество S2+), то множество S2 называется ***покрытием*** множества S1. Это означает, что если СУБД поддерживает все ограничения целостности, представленных ФЗ S2, то она автоматически поддерживает и все ограничения целостности, представленные ФЗ S1.
- Если множество S1 является покрытием для множества S2, а множество S2 одновременно является покрытием для множества S1 (S1+=S2+), то множества S1 и S2 называются ***эквивалентными***.

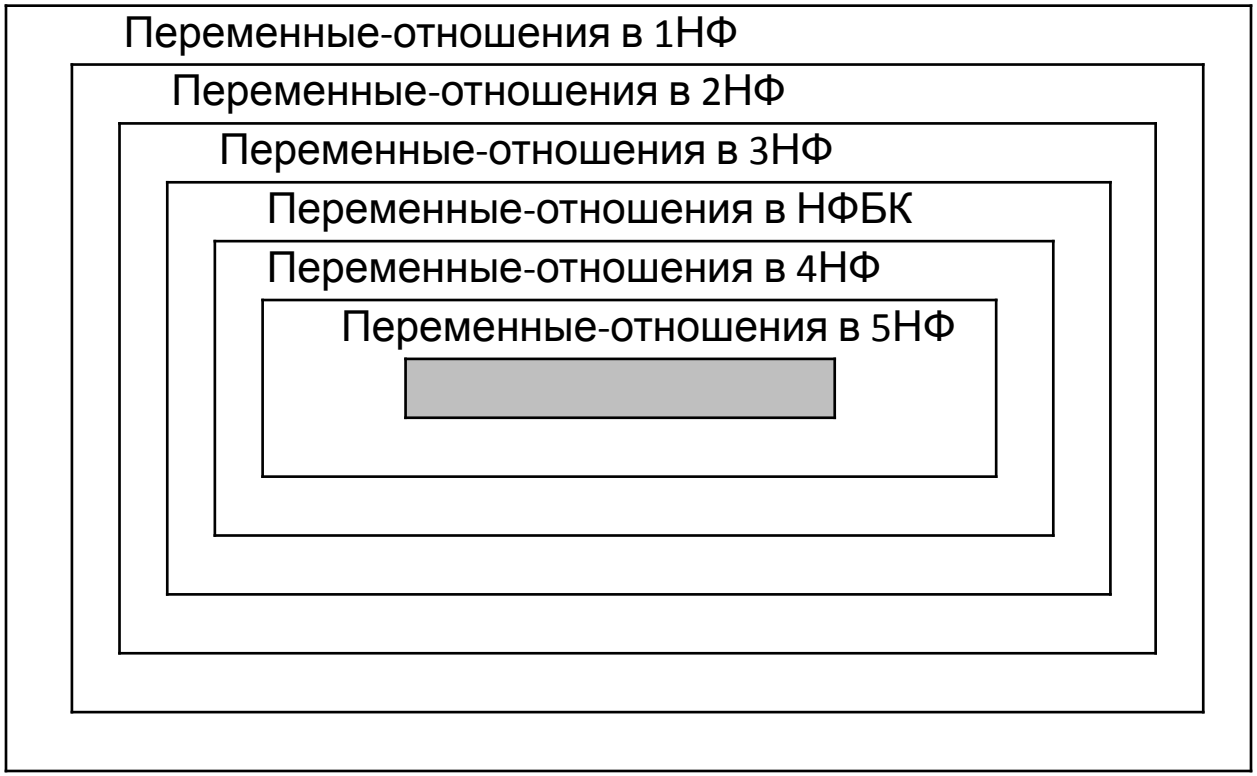
- **Пример 4.4.** Пусть дана переменная-отношение R с атрибутами A, B, C, D и следующими ФЗ:
- $A \rightarrow BC$
- $B \rightarrow C$
- $A \rightarrow B$
- $AB \rightarrow C$
- $AC \rightarrow D$
- Найдем неприводимое множество ФЗ, эквивалентное данному множеству.
- Прежде всего, перепишем заданные ФЗ, чтобы каждая из них имела одноэлементную правую часть, используя правило декомпозиции, при этом удалив одну из ФЗ $A \rightarrow B$:
- $A \rightarrow B$
- $A \rightarrow C$

- B → C
- AB → C
- AC → D
- Затем в детерминанте ФЗ AC → D может быть опущен атрибут C, поскольку имеется зависимость A → C, из которой по правилу дополнения можно получить зависимость A→AC. Кроме того, дана ФЗ AC → D, из которой по правилу транзитивности можно получить ФЗ A → D. Таким образом, атрибут C в детерминанте ФЗ AC → D является избыточным.
- Далее, ФЗ AB → C может быть исключена, поскольку дана зависимость A → C, из которой по правилу дополнения можно получить ФЗ AB → CB, а затем по правилу декомпозиции – ФЗ AB → C.
- Наконец ФЗ A → C подразумевается зависимостями A → B и B → C, а следовательно может быть отброшена. В результате получаем неприводимое множество ФЗ:
- A → B
- B → C
- A → D
- Множество ФЗ I, которое неприводимо и эквивалентно другому множеству ФЗ S, называется ***неприводимым покрытием*** множества S. Таким образом, в системе вместо исходно множества ФЗ S может использоваться его неприводимое покрытие I. Однако следует отметить, что для заданного множества ФЗ не всегда существует *уникальное* неприводимое покрытие.

2. Нормализация: формы 1НФ, 2НФ, 3НФ и НФБК

- Процесс дальнейшей нормализации, который ниже будет упоминаться просто как нормализация, основывается на концепции ***нормальных форм***. Говорят, что переменная-отношение находится в определенной нормальной форме, если она удовлетворяет некоторому набору условий. Например, переменная-отношение находится во второй нормальной форме (2НФ), если она находится в первой нормальной форме и удовлетворяет некоторым дополнительным условиям.

Уровни нормализации



2.2 Декомпозиция без потерь и функциональные зависимости

S

S#	STATUS	CITY
S3	30	Москва
S5	30	Смоленск

а) SST

S#	STATUS
S3	30
S5	30

SC

S#	CITY
S3	Москва
S5	Смоленск

б) SST

S#	STATUS
S3	30
S5	30

STC

STATUS	CITY
30	Москва
30	Смоленск

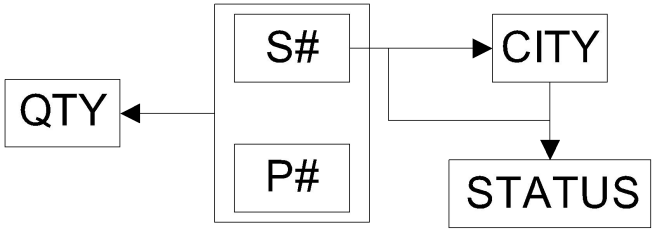
Теорема Хита. Пусть R{A,B,C} является переменной-отношением, где A, B и C – множества атрибутов этой переменной-отношения. Если R удовлетворяет ФЗ A → B, то R равна соединению проекций {A, B} и {A, C}.

Первая нормальная форма (1НФ)

Первая нормальная форма. Переменная-отношение находится в 1НФ тогда и только тогда, когда в любом допустимом значении этой переменной-отношения каждый ее кортеж содержит только одно значение для каждого из атрибутов.

Нормализация (пример 1НФ)

Диаграмма ФЗ:



FIRST

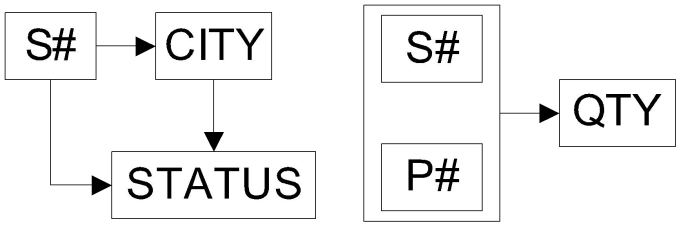
S#	STATUS	CITY	P#	QTY
S1	20	Москва	P1	300
S1	20	Москва	P2	200
S1	20	Москва	P3	400
S1	20	Москва	P4	200
S1	20	Москва	P5	100
S1	20	Москва	P6	100
S2	10	Смоленск	P1	300
S2	10	Смоленск	P2	400
S3	10	Смоленск	P2	200
S4	20	Москва	P2	200
S4	20	Москва	P4	300
S4	20	Москва	P5	400

Вторая нормальная форма (2НФ)

Вторая нормальная форма. Переменная-отношение находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме, и каждый неключевой атрибут неприводимо зависит от ее первичного ключа.

Нормализация (пример 2НФ)

Диаграмма ФЗ:



SECOND

S#	STATUS	CITY
S1	20	Москва
S2	10	Смоленск
S3	10	Смоленск
S4	20	Москва

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Третья нормальная форма (3НФ)

Третья нормальная форма. Переменная-отношение находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и каждый неключевой атрибут нетранзитивно зависит от ее первичного ключа.

Диаграмма ФЗ:

SC

S#	CITY
S1	Москва
S2	Смоленск
S3	Смоленск
S4	Москва

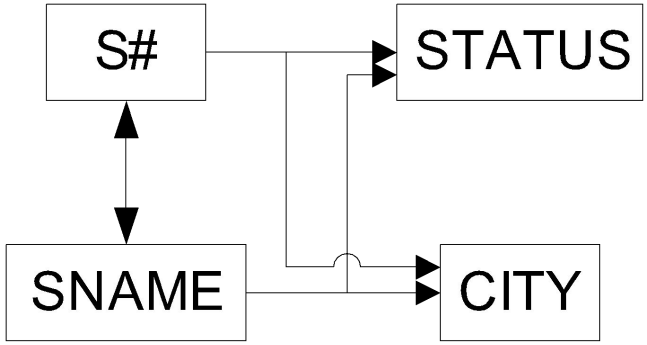
CS

CITY	STATUS
Москва	20
Смоленск	10

Нормальная форма Бойса-Кодда (НФБК)

Переменная-отношение находится в ***нормальной форме Бойса-Кодда*** тогда и только тогда, когда каждая ее нетривиальная и неприводимая слева ФЗ имеет в качестве детерминанта некоторый потенциальный ключ.

Диаграмма ФЗ в переменной-отношении S для случая, когда атрибут SNAME является ее потенциальным ключом



Четвертая нормальная форма (4НФ)

НСТХ

COURSE	TEACHERS	TEXTS
Физика	TEACHERS Профессор Иванов Доцент Колосов	TEXT Механика Основы оптики
Математика	TEACHERS Профессор Иванов	TEXT Векторный анализ Тригонометрия Дифференцирование

СТХ

COURSE	TEACHERS	TEXTS
Физика	Профессор Иванов	Механика
Физика	Профессор Иванов	Основы оптики
Физика	Доцент Колосов	Механика
Физика	Доцент Колосов	Основы оптики
Математика	Профессор Иванов	Векторный анализ
Математика	Профессор Иванов	Тригонометрия

Четвертая нормальная форма (4НФ) Продолжение

CT

COURSE	TEACHERS
Физика	Профессор Иванов
Физика	Доцент Колосов
Математика	Профессор Иванов

CX

COURSE	TEXTS
Физика	Механика
Физика	Основы оптики
Математика	Векторный анализ
Математика	Тригонометрия

Пусть A, B и C являются произвольными подмножествами множества атрибутов переменной-отношения R. Тогда подмножество B ***многозначно зависит*** от подмножества A, что символически выражается записью:

$A \rightarrow\!\!\!\rightarrow B$

(читается как «A многозначно определяет B» или «A двойная стрелка B»), тогда и только тогда, когда множество значений B, соответствующее заданной паре (значение A, значение C) переменной-отношения R, зависит от A, но не зависит от C.

Четвертая нормальная форма (4НФ) Окончание

Теорема Фейгина. Пусть A, B и C являются множествами атрибутов переменной-отношения R{A,B,C}. Переменная-отношение R будет равна соединению ее проекций {A,B} и {A,C} тогда и только тогда, когда для переменной-отношения R выполняется многозначная зависимость A →→ B | C.

Переменная-отношение R находится в ***четвертой нормальной форме*** (4НФ) тогда и только тогда, когда в случае существования таких подмножеств A и B атрибутов этой переменной-отношения R, для которых выполняется *нетривиальная* многозначная зависимость A →→ B, все атрибуты переменной-отношения R также функционально зависят от атрибута A.

МЗЗ A →→ B называется ***тривиальной***, если либо A является супермножеством B, либо объединение A и B образует весь заголовок отношения.

Пятая нормальная форма (5НФ)

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

SP

S#	P#
S1	P1
S1	P2
S2	P1

PJ

P#	J#
P1	J2
P2	J1
P1	J1

JS

S#	J#
S1	J2
S1	J1
S2	J1

Соединение по атрибуту P#

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S2	P1	J2
S1	P1	J1

Соединение по комбинации атрибутов S#,J#

Исходное состояние SPJ

Пятая нормальная форма (5НФ) Продолжение

ЕСЛИ пара (s1,p1) присутствует в SP
И пара (p1,j1) присутствует в PJ
И пара (j1,s1) присутствует в JS
ТО тройка (s1,p1,j1) присутствует в SPJ

ЕСЛИ кортежи (s1,p1,j2), (s2,p1,j1), (s1,p2,j1) присутствуют в SPJ
ТО кортеж (s1,p1,j1) также присутствует в SPJ.

Пусть R является переменной-отношением, а A,B,…,Z – произвольными подмножествами множества ее атрибутов. Переменная-отношение R удовлетворяет ***зависимости соединения:***
* {A,B,…,Z} (читается «звездочка A,B,…,Z»)
тогда и только тогда, когда любое допустимое значение переменной-отношения R эквивалентно соединению ее проекций по подмножествам атрибутов A,B,…,Z.

Пятая нормальная форма (5НФ) Продолжение

Теорема Фейгина. Переменная-отношение R{A,B,C} удовлетворяет зависимости соединения *{AB,AC} тогда и только тогда, когда она многозначной зависимости A →→ B | C.

Примеры аномалий обновления:

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

Если вставляется кортеж (S2,P1,J1), то также должен быть вставлен кортеж (S1,P1,J1)

Обратное утверждение неверно.

Кортеж (S2,P1,J1) может быть удален без побочных эффектов.

Если удаляется кортеж (S1,P1,J1), то также должен быть удален еще один кортеж (но какой?)

Пятая нормальная форма (5НФ) Окончание

Переменная-отношение R находится в ***пятой нормальной форме*** (5НФ), которую иногда иначе называют проекционно-соединительной нормальной формой (ПСНФ), тогда и только тогда, когда каждая нетривиальная зависимость соединения в переменной-отношении R подразумевается ее потенциальными ключами.

Зависимость соединения * {A,B,…,Z} называется ***тривиальной*** тогда и только тогда, когда одна из проекций A,B,…,Z является проекцией идентичной R.

Вопросы на самоподготовку:

1. Функциональные зависимости. Замыкание множества зависимостей (правила вывода). Примеры.
2. Функциональные зависимости. Замыкание множества атрибутов. Неприводимые множества зависимостей. Примеры.
3. Концепция нормальных форм. Декомпозиция без потерь (теорема Хита). Диаграммы ФЗ. Примеры.
4. Нормализация. Первая, вторая и третья нормальные формы. Аномалии обновления. 1-ый и 2-ой этапы нормализации. Пример. Нормальная форма Бойса-Кодда.
5. Нормализация. Четвертая и пятая нормальные формы. Общая процедура проектирования БД.

Общая схема процедуры нормализации

- Весь процесс нормализации можно неформально определить с помощью перечисленных ниже правил.
- Переменную-отношение в 1НФ следует разбить на такие проекции, которые позволят исключить все функциональные зависимости, не являющиеся неприводимыми. В результате будет получен набор переменных-отношений в 2НФ.
- Полученные переменные-отношения в 2НФ следует разбить на такие проекции, которые позволят исключить все существующие транзитивные ФЗ. В результате будет получен набор переменных-отношений в 3НФ.
- Полученные переменные-отношения в 3НФ следует разбить на проекции, позволяющие исключить любые оставшиеся ФЗ, в которых детерминанты не являются потенциальными ключами. В результате такого разбиения будет получен набор переменных-отношений в НФБК.
- Полученные переменные-отношения в НФБК следует разбить на проекции, позволяющие исключить любые многозначные зависимости, которые не являются функциональными. В результате будет получен набор переменных-отношений в 4НФ.
- Полученные переменные-отношения в 4НФ следует разбить на проекции, позволяющие исключить любые зависимости соединения, которые не подразумеваются потенциальными ключами. В результате получаем набор переменных-отношений в 5НФ.

Общая схема процедуры нормализации

- Процесс разбиения на проекции на каждом этапе должен быть выполнен без потерь и с сохранением зависимостей. Общее назначение процесса нормализации заключается в следующем:
- исключение некоторых типов избыточности;
- устранение некоторых аномалий обновления;
- разработка проекта БД, который является достаточно «хорошим» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
- упрощение процедуры описания необходимых ограничений целостности.
- Зависимости и нормализация являются чисто семантическими понятиями, т.е. связаны со смыслом данных, тогда как реляционная алгебра и реляционное исчисление, а также построенные на их основе языки наподобие SQL, наоборот, имеют дело со значениями данных и не требуют выполнения нормализации выше первого уровня. Рекомендации по выполнению дальнейшей нормализации должны рассматриваться, прежде всего, как некая упорядоченность, позволяющая разработчику БД (и, следовательно, ее пользователю) зафиксировать некую часть, пусть даже небольшую, семантики реального мира в простой и понятной форме.