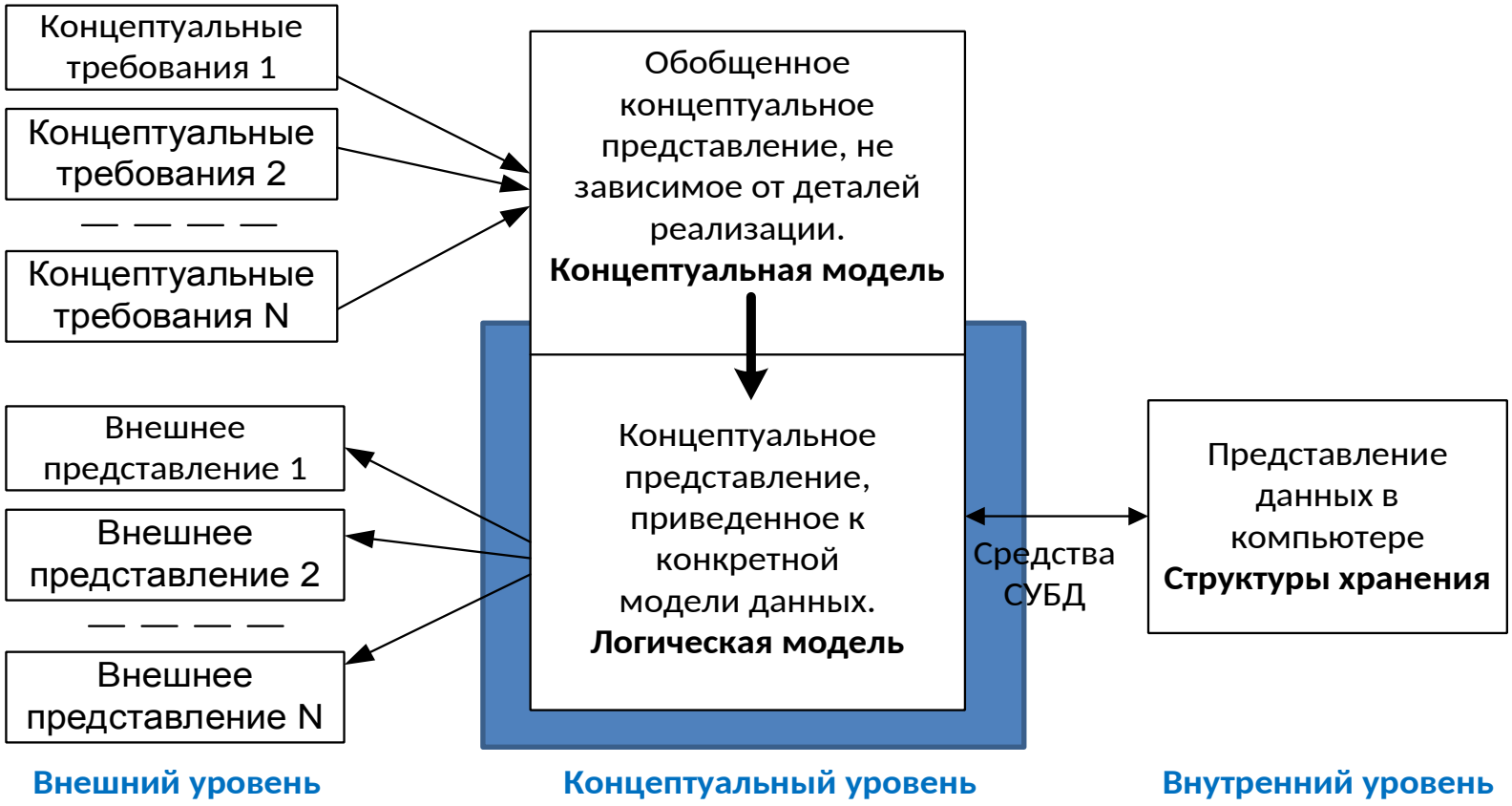
Лекция 4.

Реляционная модель данных.

Трехуровневая архитектура данных



Студенты проходят предметы по плану и сдают экзамены

Списки групп (Word)

Учебные планы (Excel)

Итоги сессии (PDF)

Набор файлов и база данных – в чем отличие?

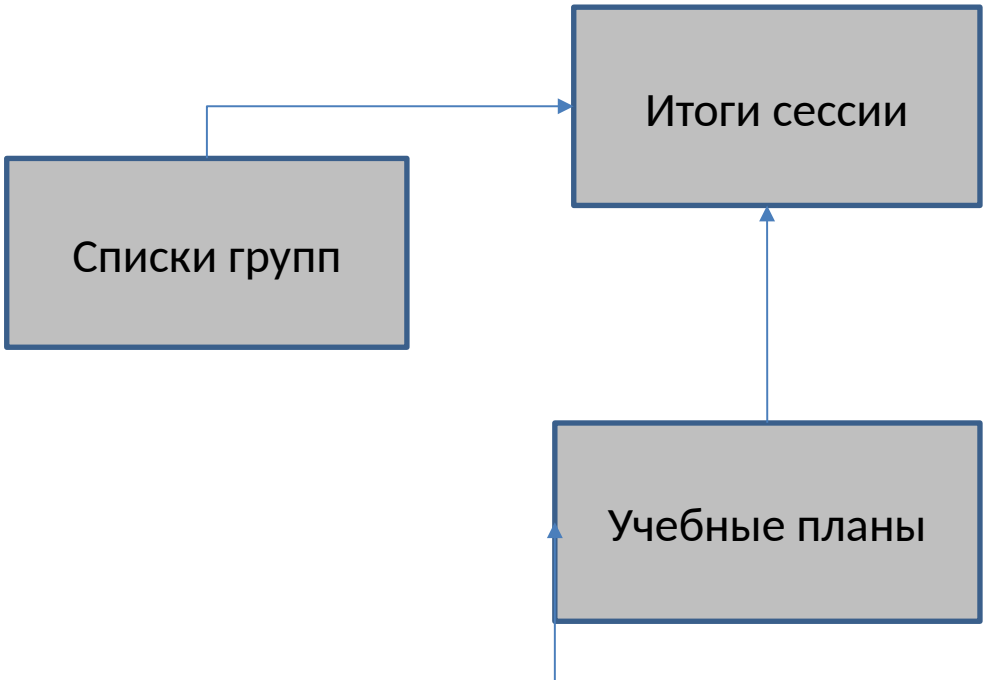
Студенты проходят предметы по плану и сдают экзамены

Списки групп (Word)

Учебные планы (Excel)

Итоги сессии (PDF)

Набор файлов и база данных – в чем отличие?



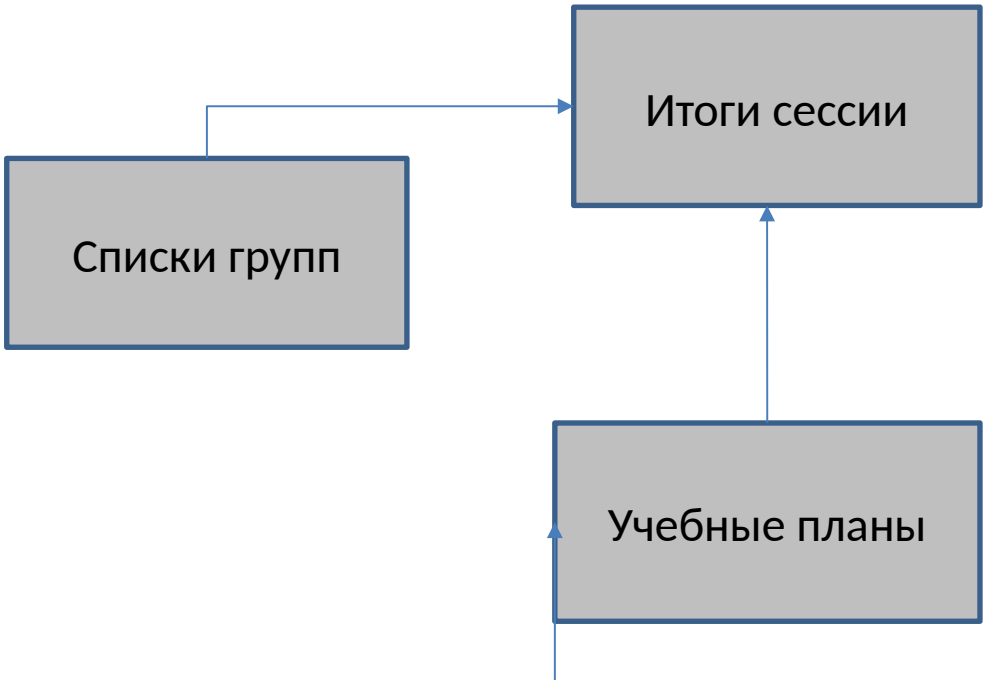
Студенты проходят предметы по плану и сдают экзамены

Списки групп (Word)

Учебные планы (Excel)

Итоги сессии (PDF)

Набор файлов и база данных – в чем отличие?



1. **Структура данных**
2. **Связи между данными**

Эволюция моделей данных

Связь - ключевое слово, отличающее базу данных от простого файла или набора файлов. Как представить эту связь в базе данных?
В разное время применялись различные подходы (**модели данных**).

Годы	Используемые модели данных
1960-е	Сетевая и иерархическая (*навигационный подход, императивное программирование*)
1970-е	1. Сетевая и иерархическая 2. **Реляционная** (*математическая база, декларативное программирование*)
1980-е	1. Реляционная 2. Сетевая и иерархическая
1990-е	Реляционная
2000-е	1. Реляционная 2. NoSQL

Навигационная (иерархическая) модель: пример программы

Есть ассортимент деталей, каждая из которых принадлежит определенному классу. Некоторые детали могут собираться из других деталей.

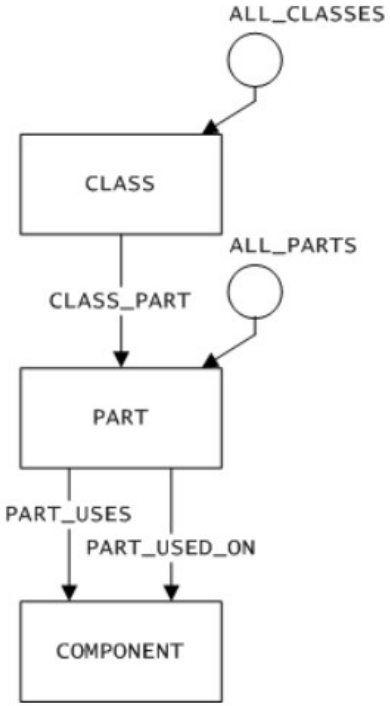
Задача: по заданному классу вывести все детали этого класса

Навигационная (иерархическая) модель: пример программы

Есть ассортимент деталей, каждая из которых принадлежит определенному классу. Некоторые детали могут собираться из других деталей.

Задача: **по заданному классу вывести все детали этого класса**

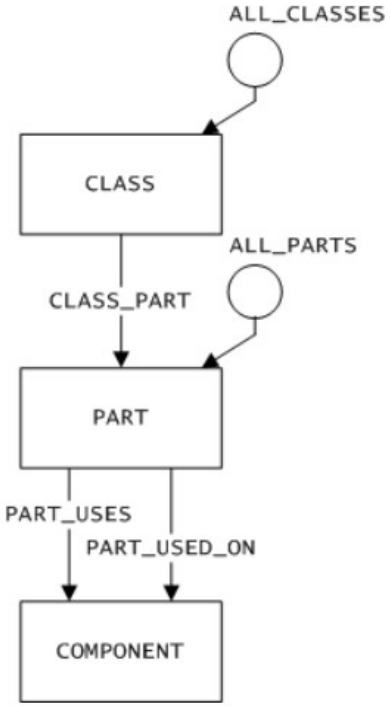
Диаграмма Бахмана



Навигационная (иерархическая) модель: пример программы

Есть ассортимент деталей, каждая из которых принадлежит определенному классу. Некоторые детали могут собираться из других деталей.

Диаграмма Бахмана



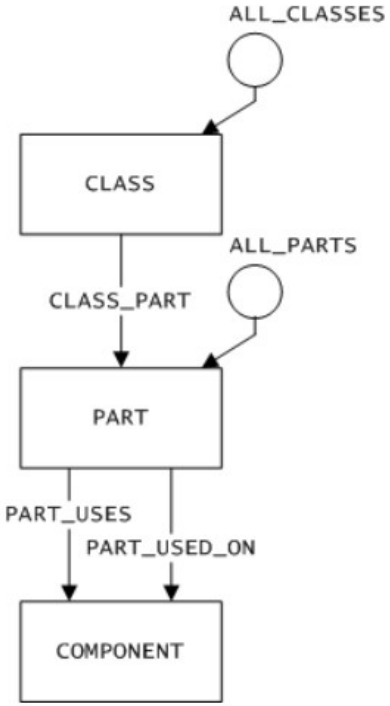
Задача: по заданному классу вывести все детали этого класса (COBOL)

```
...
SET DONE TO FAILURE.
DISPLAY "Class number: >" NO ADVANCING.
ACCEPT CLASS_ID.
FETCH FIRST CLASS WITHIN ALL_CLASSES USING CLASS_ID
AT END
    DISPLAY "Class not found"
    SET DONE TO SUCCESS.
IF DONE IS FAILURE
    DISPLAY "Parts for class number ", CLASS_ID, ":".
PERFORM UNTIL DONE IS SUCCESS
    FETCH NEXT PART WITHIN CLASS_PART
    AT END
        SET DONE TO SUCCESS
    END-FIND
    IF DONE IS FAILURE
        DISPLAY PART_ID
    END-IF
END-PERFORM.
 ...
```

Навигационная (иерархическая) модель: пример программы

Есть ассортимент деталей, каждая из которых принадлежит определенному классу. Некоторые детали могут собираться из других деталей.

Диаграмма Бахмана



Задача: по заданному классу вывести все детали этого класса (COBOL)

```
...
SET DONE TO FAILURE.
DISPLAY "Class number: >" NO ADVANCING.
ACCEPT CLASS_ID.
FETCH FIRST CLASS WITHIN ALL_CLASSES USING CLASS_ID
AT END
    DISPLAY "Class not found"
    SET DONE TO SUCCESS.
IF DONE IS FAILURE
    DISPLAY "Parts for class number ", CLASS_ID, ":".
PERFORM UNTIL DONE IS SUCCESS
    FETCH NEXT PART WITHIN CLASS_PART
    AT END
        SET DONE TO SUCCESS
    END-FIND
    IF DONE IS FAILURE
        DISPLAY PART_ID
    END-IF
END-PERFORM.
 ...
```

IBM IMS
http://www-01.ibm.com/software/data/ims/

Навигационная модель vs реляционной

Проблемы **навигационной** модели:

- БД трудно использовать (требуются навыки программирования).
- Нет теоретического (математического) фундамента.
- БД смешивают логическую и физическую реализацию данных.

Навигационная модель vs реляционной

Проблемы **навигационной** модели:

- БД трудно использовать (требуются навыки программирования).
- Нет теоретического (математического) фундамента.
- БД смешивают логическую и физическую реализацию данных.

Основные идеи **реляционной** модели:

- Представлять и сущности, и связи единообразно, с помощью простых таблиц (строки и столбцы).
- Разработать простой, похожий на естественный, декларативный язык обращения к данным.

Реляционная модель: начало

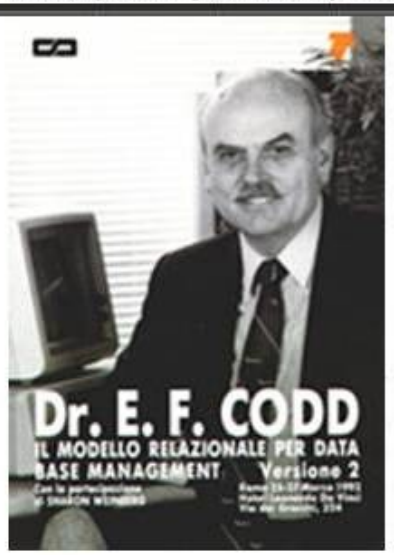
A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").



Relational Model

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Классическая статья Эдгара Кодда.

Association of Computer Machinery journal (1970 год)

Теоретические (математические) основы реляционной модели

тип данных, таблица, строка таблицы

домен, отношение, кортеж

множество, декартово произведение, кортеж

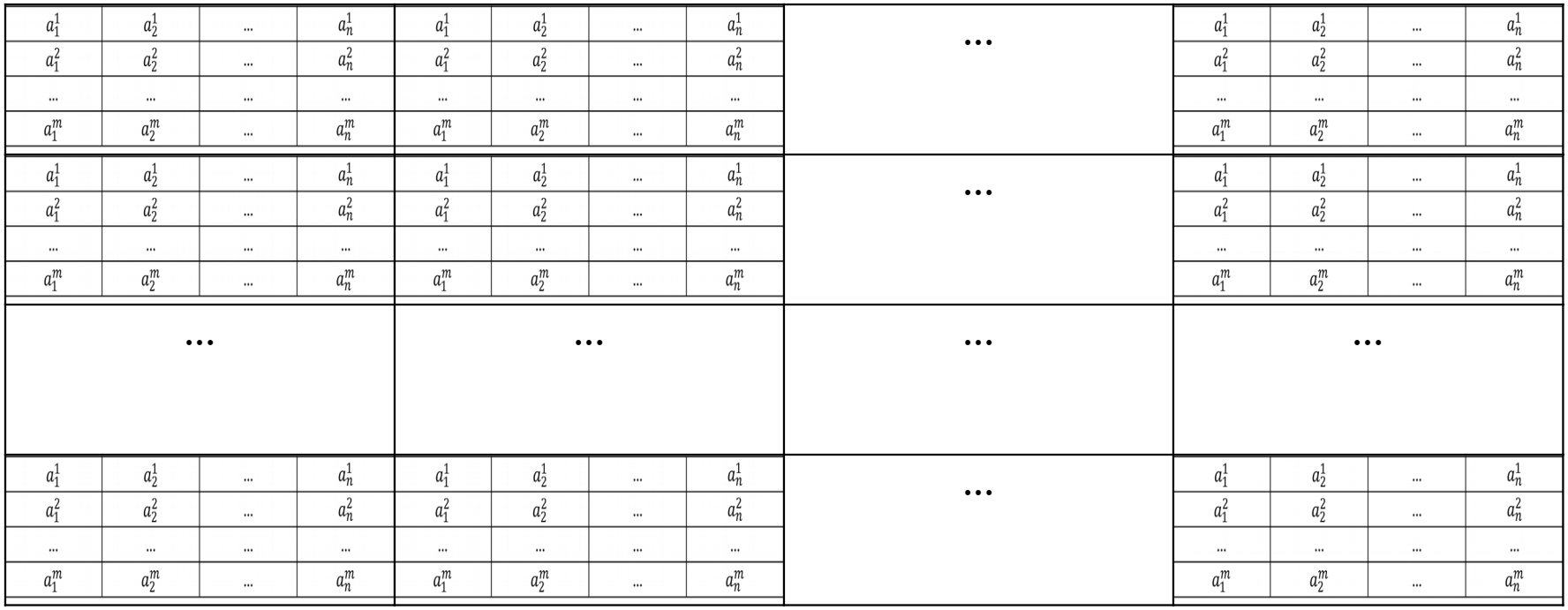
Математические основы реляционной модели

В теории множеств:

- **Отношение степени *n*** – подмножество R декартового произведения множеств $A_1 x A_2 x \ldots x A_n$.

- **Мощность *m* отношения *R*** – мощность множества кортежей, входящих в отношение R.

Математические основы реляционной модели



Отношения – математический аналог таблиц.

- Все элементы отношения есть однотипные кортежи. Поэтому кортежи можно считать аналогами строк в простой таблице.

- Отношение включает в себя не все возможные кортежи (есть критерий, задающий семантику отношения).

Пример бинарного отношения

Множество A={Вова, Петя, Маша, Лена}. Известно:

1. Вова любит только себя (эгоист).
2. Петя любит Машу (взаимно).
3. Маша любит Петю (взаимно).
4. Маша любит себя.
5. Лена любит Петю (несчастная любовь).

Пример бинарного отношения

Множество A={Вова, Петя, Маша, Лена}. Известно:

1. Вова любит только себя (эгоист).
2. Петя любит Машу (взаимно).
3. Маша любит Петю (взаимно).
4. Маша любит себя.
5. Лена любит Петю (несчастная любовь).

Это бинарное отношение **"любить"**, заданное на множестве A^2.

Пример бинарного отношения

Множество A={Вова, Петя, Маша, Лена}. Известно:

1. Вова любит только себя (эгоист).
2. Петя любит Машу (взаимно).
3. Маша любит Петю (взаимно).
4. Маша любит себя.
5. Лена любит Петю (несчастная любовь).

Это бинарное отношение **"любить"**, заданное на множестве A^2.

Как представить это отношение, чтобы с ним удобно было работать на компьютере?

Способ 1. Произвольный текст.

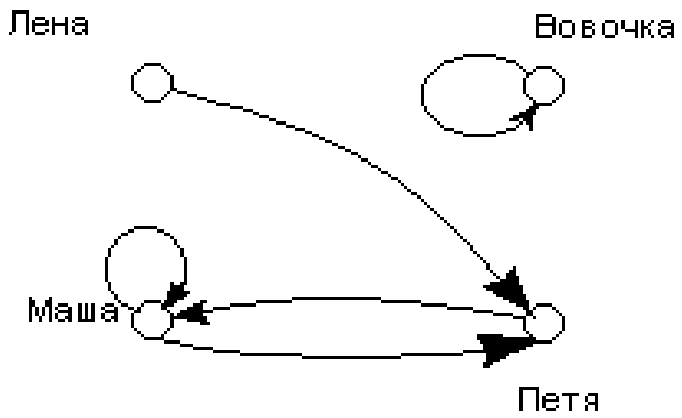
1. Вова любит только себя (эгоист).
2. Петя любит Машу (взаимно).
3. Маша любит Петю (взаимно).
4. Маша любит себя.
5. Лена любит Петю (несчастная любовь).

Способ 1. Произвольный текст. *Тяжело обрабатывать алгоритмами.*

1. Вова любит только себя (эгоист).
2. Петя любит Машу (взаимно).
3. Маша любит Петю (взаимно).
4. Маша любит себя.
5. Лена любит Петю (несчастная любовь).

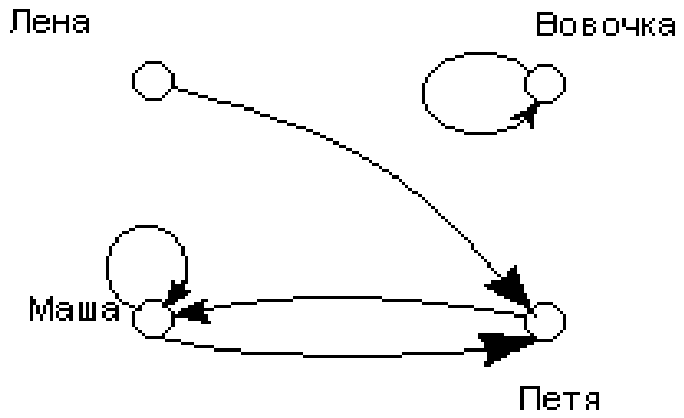
Способ 1. Произвольный текст. *Тяжело обрабатывать алгоритмами.*

Способ 2. Граф взаимоотношений.



Способ 1. Произвольный текст. *Тяжело обрабатывать алгоритмами.*

Способ 2. Граф взаимоотношений. *Наглядно, но хранить данные в компьютере в графическом виде неудобно.*



Пример бинарного отношения

Способ 3. Матрица взаимоотношений.

Кого Кто	Вова	Петя	Маша	Лена
Вова	Любит			
Петя			Любит	
Маша		Любит	Любит	
Лена		Любит		

Пример бинарного отношения

Способ 3. Матрица взаимоотношений.

Кого Кто	Вова	Петя	Маша	Лена
Вова	Любит			
Петя			Любит	
Маша		Любит	Любит	
Лена		Любит		

Негибко – при появлении нового человека придется изменять и с троки, и столбцы.

Пример бинарного отношения

Способ 3. Матрица взаимоотношений.

Кого Кто	Вова	Петя	Маша	Лена
Вова	Любит			
Петя			Любит	
Маша		Любит	Любит	
Лена		Любит		

Негибко – при появлении нового человека придется изменять и с троки, и столбцы.

Способ 4. Таблица фактов.

Кто любит	Кого любят
Вовочка	Вовочка
Петя	Маша
Маша	Петя
Маша	Маша
Лена	Петя

Пример бинарного отношения

Способ 3. Матрица взаимоотношений.

Кого Кто	Вова	Петя	Маша	Лена
Вова	Любит			
Петя			Любит	
Маша		Любит	Любит	
Лена		Любит		

Негибко – при появлении нового человека придется изменять и с троки, и столбцы.

Способ 4. Таблица фактов.

Кто любит	Кого любят
Вовочка	Вовочка
Петя	Маша
Маша	Петя
Маша	Маша
Лена	Петя

При появлении новых лиц в таблицу добавляются только строки.

Основные понятия реляционной модели

Рис. 2.1. Наглядное представление основных понятий реляционной модели

- Данные хранятся в **отношениях**, воспринимаемых как таблицы. Отношением R, определенном на множествах D_1, D_2 , …, D_n, называется подмножество из D_1 x D_2 x … x D_n.
- Множества D_1, D_2 , …, D_n – **домены** отношения. Каждый атрибут отношения определяется на некотором домене. Домен имеет уникальное имя, определен на простом типе данных или на другом домене.
- Элементы декартова произведения $\{d_1, d_2, \ldots, d_n\}$ – **кортежи**.
- Число n – **степень** отношения.

Основные понятия реляционной модели

Рис. 2.1. Наглядное представление основных понятий реляционной модели

- Количество кортежей – **мощность** (**кардинальность**) отношения.
- **Атрибут** отношения – пара вида <*Имя_атрибута*:*Имя_домена*>
- **Заголовок** отношения – набор {<A_1:D_1>, <A_2:D_2>, …, <A_n:D_n>}. Заголовок статичен.
- **Тело** отношения – множество кортежей отношения. Каждый кортеж – множество пар вида <*Имя_атрибута*:*Значение*>. Тело отношения изменяется.
- **Первичный ключ** отношения – однозначно идентифицирует кортеж в теле отношения.

Реляционная алгебра – способ работы с отношениями

Язык SQL – создан специально для представлений операций реляционной алгебры в простом и доступном для всех виде.

- Традиционные теоретико-множественные операции (**объединение**, **пересечение**, **разность**). Для этих операций отношения должны иметь одинаковый набор атрибутов.

Реляционная алгебра – способ работы с отношениями

Язык SQL – создан специально для представлений операций реляционной алгебры в простом и доступном для всех виде.

- Традиционные теоретико-множественные операции (**объединение**, **пересечение**, **разность**). Для этих операций отношения должны иметь одинаковый набор атрибутов.
- **Проекция** – оставляет из всего набора атрибутов некоторое их подмножество. Пример: A[2,5] – проекция отношения A на 2 и 5 атрибуты. В SQL: SELECT DISTINCT

Реляционная алгебра – способ работы с отношениями

Язык SQL – создан специально для представлений операций реляционной алгебры в простом и доступном для всех виде.

- Традиционные теоретико-множественные операции (**объединение**, **пересечение**, **разность**). Для этих операций отношения должны иметь одинаковый набор атрибутов.
- **Проекция** – оставляет из всего набора атрибутов некоторое их подмножество. Пример: A[2,5] – проекция отношения A на 2 и 5 атрибуты. В SQL: SELECT DISTINCT
- **Соединение** двух отношений по условию равенства общих атрибутов – новое отношение, кортежи которого являются конкатенациями всевозможных пар кортежей исходных отношений, для которых выполняется это условие. Пример: A[1=3]B – соединение отношений A и B по общим атрубутам A[1] и B[3]. В SQL условия соединения формулируются в разделе FROM.

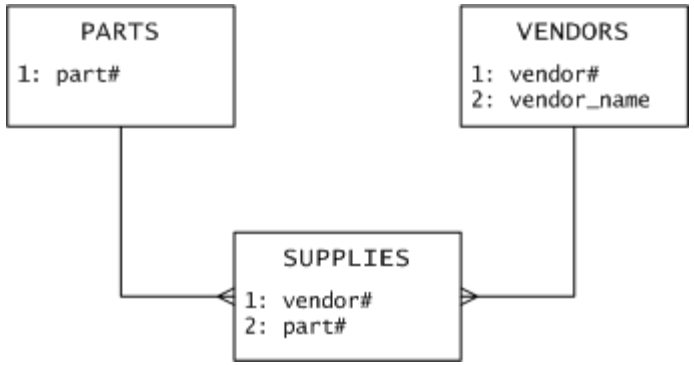
Реляционная алгебра – способ работы с отношениями

Язык SQL – создан специально для представлений операций реляционной алгебры в простом и доступном для всех виде.

- Традиционные теоретико-множественные операции (**объединение**, **пересечение**, **разность**). Для этих операций отношения должны иметь одинаковый набор атрибутов.
- **Проекция** – оставляет из всего набора атрибутов некоторое их подмножество. Пример: A[2,5] – проекция отношения A на 2 и 5 атрибуты. В SQL: SELECT DISTINCT
- **Соединение** двух отношений по условию равенства общих атрибутов – новое отношение, кортежи которого являются конкатенациями всевозможных пар кортежей исходных отношений, для которых выполняется это условие. Пример: A[1=3]B – соединение отношений A и B по общим атрубутам A[1] и B[3]. В SQL условия соединения формулируются в разделе FROM.
- **Ограничение** – оставляет лишь те кортежи, которые удовлетворяют определенному условию на атрибуты. Пример: A[1 > 2] – ограничение отношения A по условию, что атрибут 1 больше, чем атрибут 2.
  В SQL условия ограничения формулируются в разделе WHERE.

Пример использования операций реляционной алгебры

Имеются поставщики (V — VENDORS) и детали (P — PARTS), связанные многие-ко-многим поставками (S — SUPPLIES).



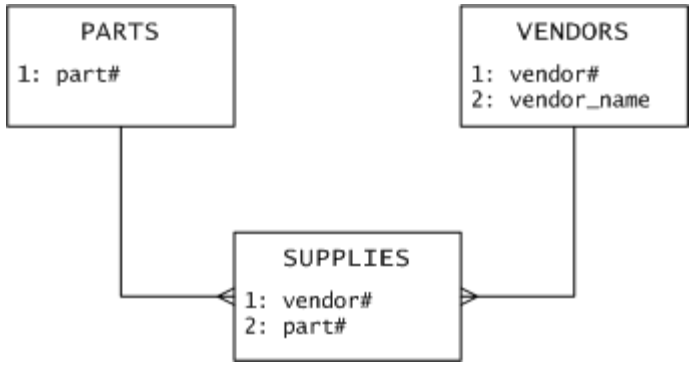
Пример использования операций реляционной алгебры

Имеются поставщики (V — VENDORS) и детали (P — PARTS), связанные многие-ко-многим поставками (S — SUPPLIES).

1. Номер поставщика, поставляющего хоть что-нибудь:

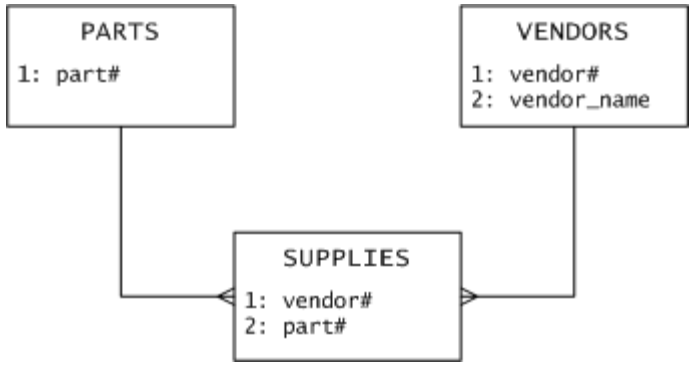
S[1]

SELECT DISTINCT vendor# FROM S



Пример использования операций реляционной алгебры

Имеются поставщики (V — VENDORS) и детали (P — PARTS), связанные многие-ко-многим поставками (S — SUPPLIES).



1. Номер поставщика, поставляющего хоть что-нибудь:

S[1]

```
SELECT DISTINCT vendor# FROM S
```

2. Имя поставщика, не поставляющего ни одной детали:

(V[1=1](V[1]-S[1]))[2]

```
SELECT vendor_name
FROM  V, (SELECT vendor# FROM V MINUS SELECT vendor# FROM S) V2
WHERE  V.vendor# = V2.vendor#
```

Практические принципы реляционной модели

Таблицы в реляционной БД являются логическими, а не физическими структурами.

Таблицы – абстракция способа физического хранения данных.
Множество деталей на уровне памяти скрыто от пользователя (размещение хранимых записей, кодировка хранимых данных, хранимые структуры доступа, …).

Практические принципы реляционной модели

- Каждое значение, содержащееся на пересечении строки и колонки, должно быть **атомарным** (неразделяемым на несколько значений).

- Значения данных в одной и той же колонке должны принадлежать к одному **типу**, доступному для исполнения в данной СУБД.

- Каждая запись в таблице **уникальна**, т. е. в таблице не существует двух записей с полностью совпадающим набором значений ее полей.

- Каждое поле имеет **уникальное имя**.

Навигационный и реляционный подходы

Основное достоинство реляционной модели – применение принципов **нечисловой (ассоциативной) обработки данных**.

Взаимосвязи между элементами данных:

- Навигационная модель – путем искусственного соединения с помощью указателей (адресов).
- Реляционная модель – в соответствии со значениями их атрибутов.

Навигационный и реляционный подходы

Основное достоинство реляционной модели – применение принципов **нечисловой (ассоциативной) обработки данных**.

Взаимосвязи между элементами данных:

- Навигационная модель – путем искусственного соединения с помощью указателей (адресов).
- Реляционная модель – в соответствии со значениями их атрибутов.

ГРУППЫ
Код Наименование

СТУДЕНТЫ
Номер зачетки Фамилия Имя Отчество Код группы

Сравнивая значения полей ГРУППЫ.Код и СТУДЕНТЫ.Код_группы, мы можем определить, в какой группе учится каждый студент, сколько студентов учится в каждой группе, вывести список студентов каждой группы и т.д.

Навигационный и реляционный подходы

- В реляционной модели запросы к БД не ограничены физическими указателями, поэтому для их реализации не нужно писать программы.

- Запросы будут продолжать работать даже после логической реорганизации базы данных.

- Реляционные приложения намного проще навигационных.

Получение реляционной схемы из ER-диаграммы

Получение реляционной схемы из ER-диаграммы

Код	Фамилия	Д.р.	Место	Ном. фак.
100	Иванов	01.03.1996	Саранск	11
103	Петров	02.12.1996	Москва	11
254	Сидоров	11.11.1996	Рузаевка	15

Номер	Название
11	Математический
15	Филологический

Получение реляционной схемы из ER-диаграммы

Код	Фамилия	Д.р.	Место	Ном. фак.
100	Иванов	01.03.1996	Саранск	11
103	Петров	02.12.1996	Москва	11
254	Сидоров	11.11.1996	Рузаевка	15

Номер	Название
11	Математический
15	Филологический

1. Каждая сущность превращается в таблицу. Имя сущности становится именем таблицы.
2. Каждый атрибут становится столбцом с тем же именем.
3. Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы.
4. Связи многие-к-одному (и один-к-одному) становятся внешними ключами. Т.е. делается копия уникального идентификатора с конца связи "один", и соответствующие столбцы составляют внешний ключ.

Реляционный подход: практическая реализация

Проект System R – начат в исследовательской лаборатории IBM в Сан-Хосе в 1973 году.

- **Язык SQL** для формулирования запросов (Чемберлен, Бойс) как интерфейс для пользователя БД. Декларативный, близок к естественному, простой синтаксис.
- **Оптимизатор** – автоматически транслировал высокоуровневый запрос в эффективный план его выполнения.
- **Компилятор** запросов – сохранял планы запросов для дальнейшего использования.

СУБД DB2 от IBM – 1983 год.

Другие пионеры реляционных баз данных:

- Майк Стоунбрейкер, университет Беркли. СУБД Ingres
- Ларри Эллисон, фирма Relational Software Inc. Первая коммерческая реляционная СУБД в 1979 году. В 1985 году компания переименована в Oracle.

Реляционный подход: дополнительные преимущества

Кроме повышения продуктивности программистов и простоты использования:

- Реляционная модель хорошо подходит к использованию в архитектуре клиент-сервер. Обмен высокоуровневыми запросами и ответами.

- Декларативный язык SQL позволяет компиляторам организовать параллельную обработку данных.

- Реляционные данные хорошо приспособлены к графическим пользовательским интерфейсам (электронные таблицы).