

Лекция 8.

Модели данных. Реляционные и нереляционные СУБД

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?

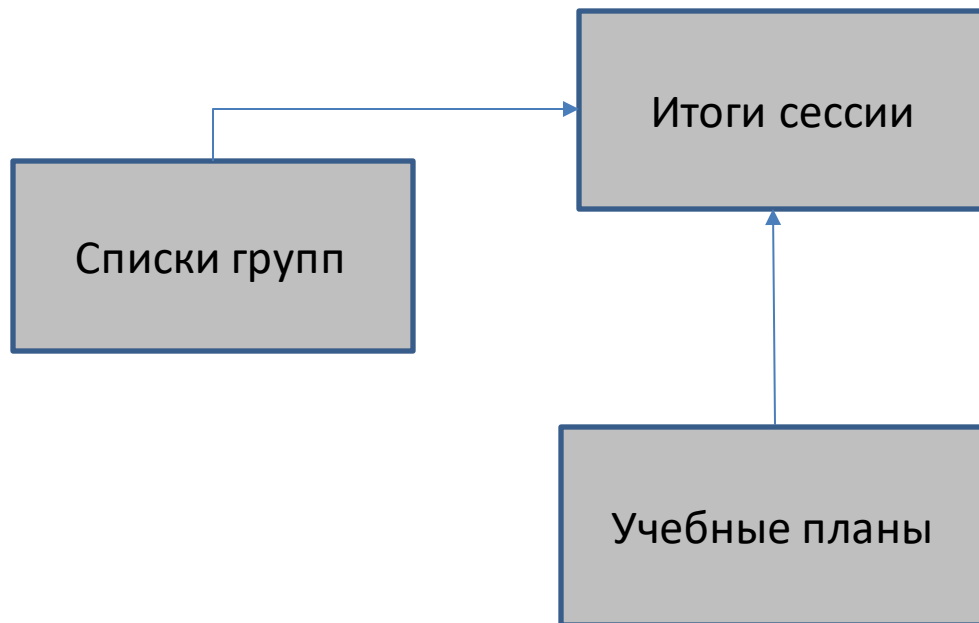
Студенты проходят предметы по плану и сдают экзамены

Списки групп
(Word)

Учебные планы
(Excel)

Итоги сессии
(PDF)

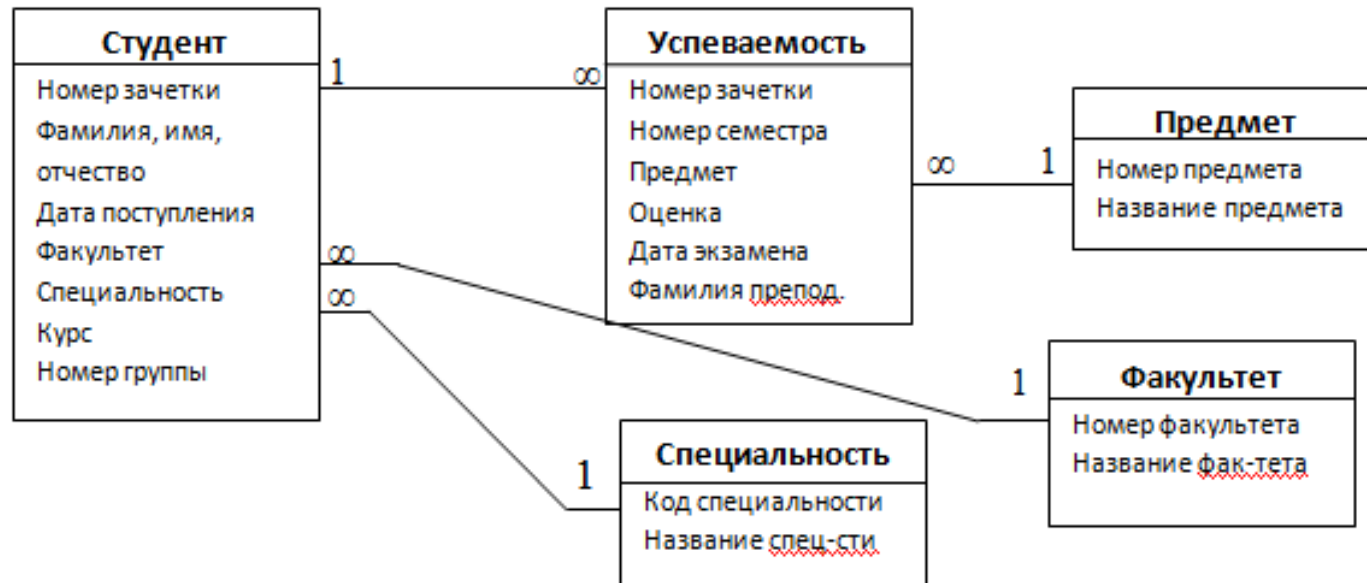
Набор файлов и база данных – в чем отличие?



1. Структура данных

2. Связи между данными

Студенты проходят предметы по плану и сдают экзамены



Структура данных + Связи между данными

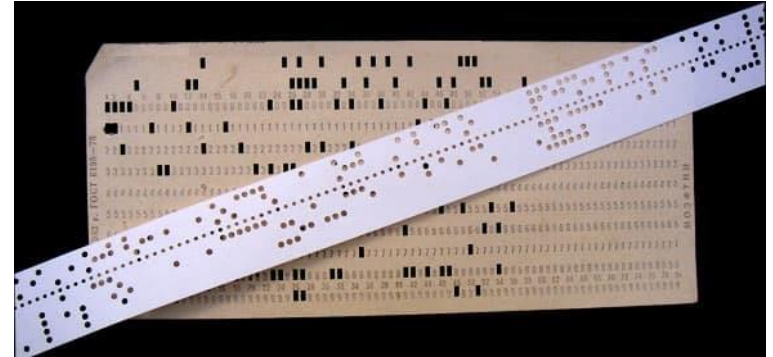
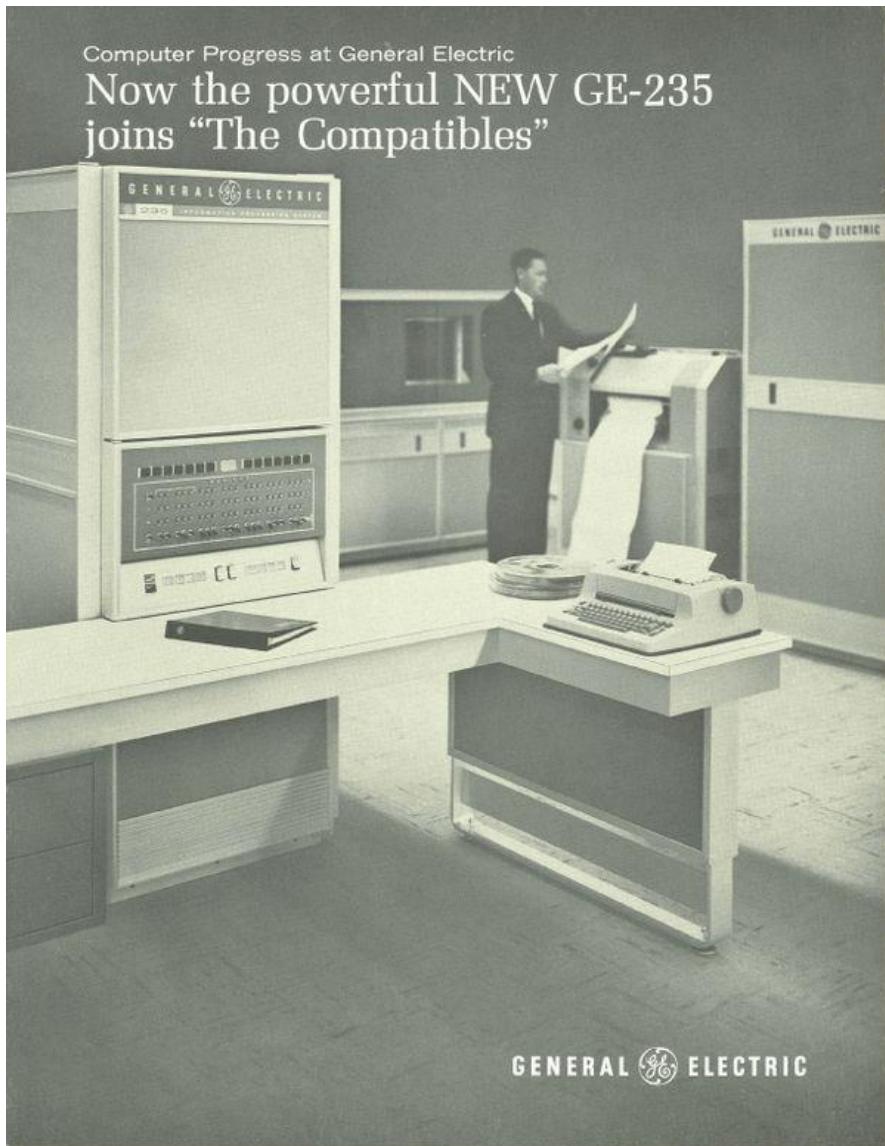
Эволюция моделей данных

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**).

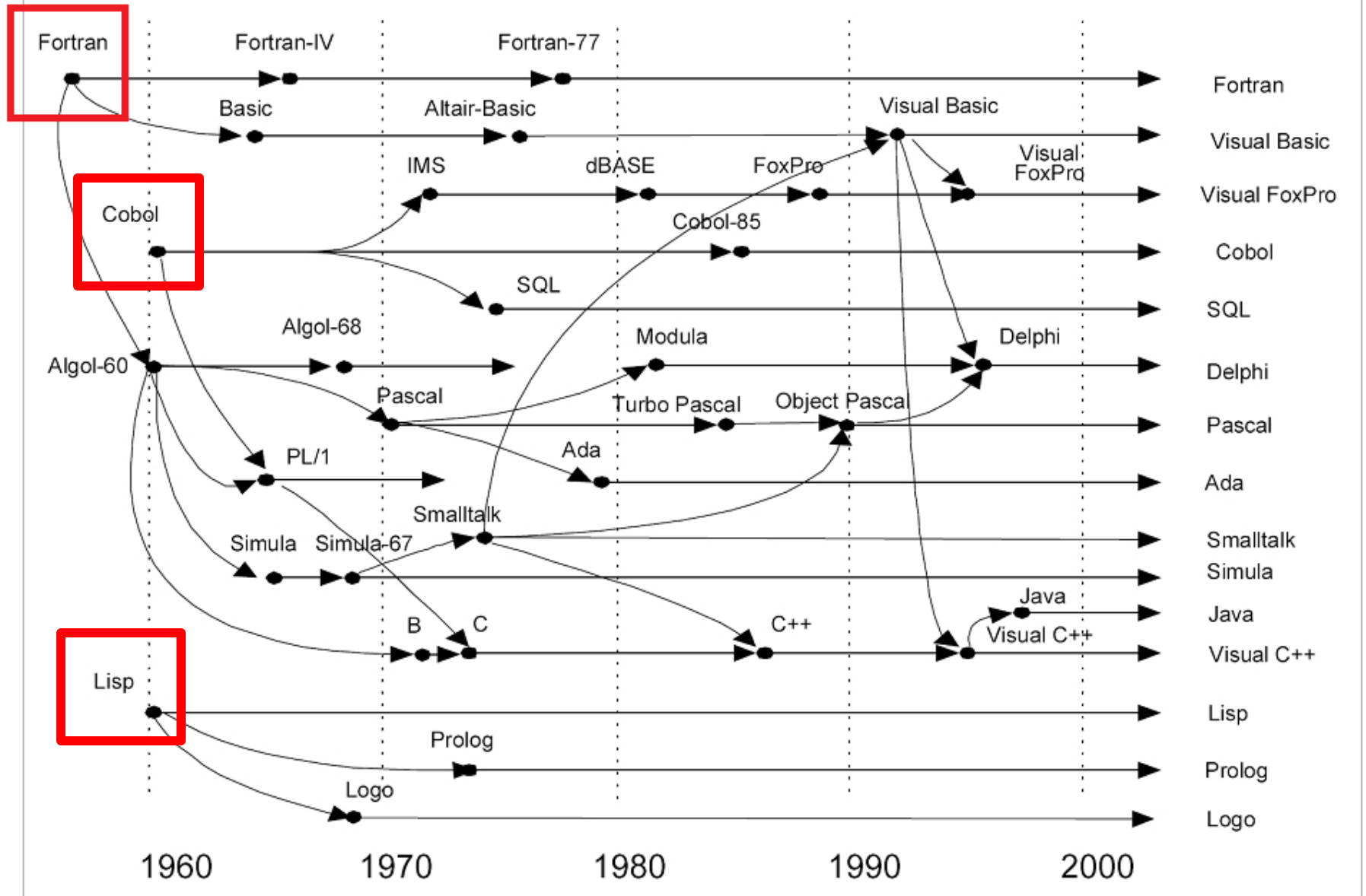
Годы	Используемые модели данных
1960-е	Сетевая и иерархическая (<i>навигационный подход, императивное программирование</i>)
1970-е	1. Сетевая и иерархическая 2. Реляционная (<i>математическая база, декларативное программирование</i>)
1980-е	1. Реляционная 2. Сетевая и иерархическая
1990-е	Реляционная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

Компьютеры в 1961 году

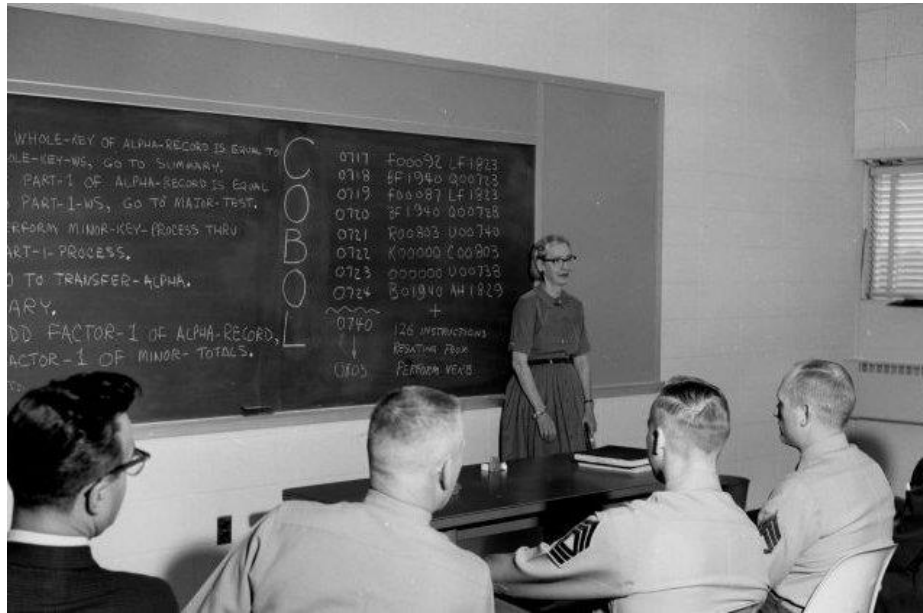


- Мэйнфреймы, пакетная обработка программ
- ЯП Fortran, Algol, Cobol
- Нет универсальных ОС
- Нет файловых систем
- ЭВМ начинают использоваться для бизнес-задач
- ЭВМ доступны только крупным организациям, нет обычных пользователей

ГЕНЕАЛОГИЧЕСКОЕ ДЕРЕВО ЯЗЫКОВ ПРОГРАММИРОВАНИЯ



COBOL – первый язык для бизнес-приложений



1959 год

Грейс Хоппер –
руководитель проекта
(язык Flow-Matic)

Common **B**usines **O**riented
Language

- Структуры данных (записи)
- Файлы

```
PERFORM UNTIL EndOfStudentFile
    ADD 1 TO MonthCount(MOBirth)
    READ StudentFile
    AT END SET EndOfStudentFile TO TRUE
END-READ
END-PERFORM
```


Первые СУБД



- 1961-1964 годы, для мэйнфрейма GE-235
- Чарльз Бахман (1924-2017), работал в General Electric
- СУБД Integrated Data Store (IDS).
- Получил за нее премию Тьюринга в 1973 году.

Впервые доступ к данным осуществлялся не непосредственно, а через описание данных в виде иерархической системы «указателей», помогающих осуществить доступ к конкретной ячейке памяти.

Вся БД упаковывалась в один файл, таблицы размещения данных создавались вручную.

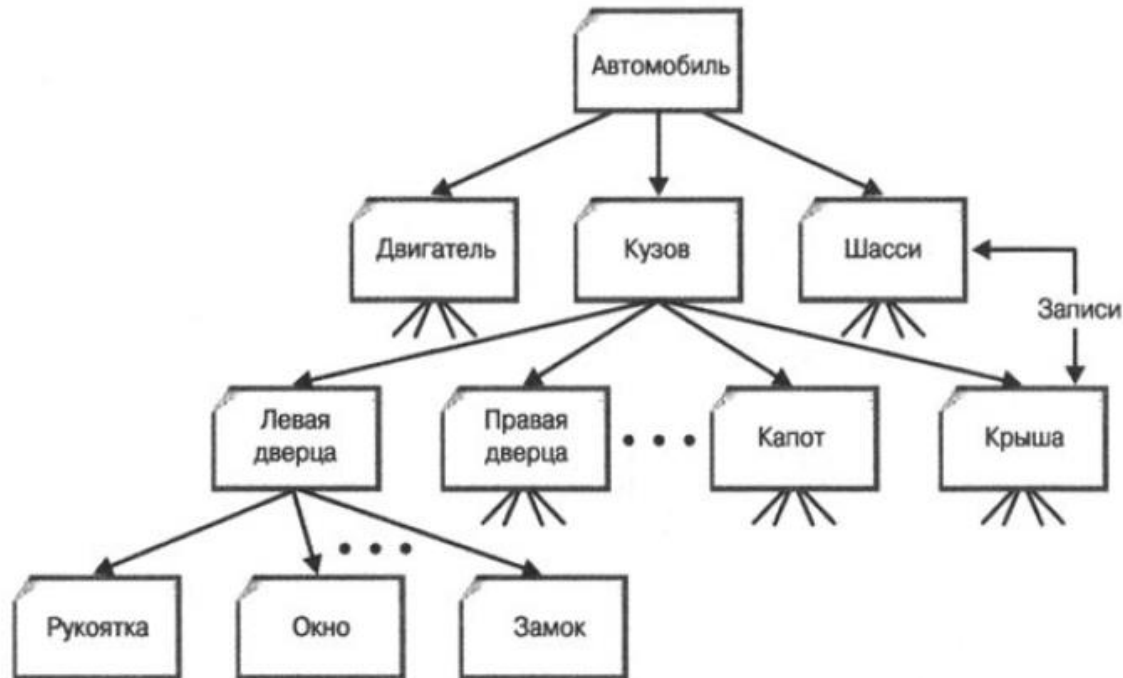
Первые СУБД

- Компания Goodrich переписала IDS на язык ISL (Intermediate System Language) и переименовала СУБД в **IDMS** (Integrated Database Management System). После этого IDMS можно было легко переносить на другие мейнфреймы IBM.
- IDMS приобрела компания Cullinane Джона Кулинана - первая успешная фирма по продаже ПО для мейнфреймов (с 1968 года).
- **IBM IMS** (Information Management System). Первая коммерческая СУБД, 1968 год (обработка спецификации деталей для ракеты Apollo).

Первые СУБД

Около 10 лет все программы, работающие с данными, писали в навигационном стиле на языке Cobol с интегрированными командами для работы с БД.

Иерархическая модель



Задачи планирования производства

- Простая структура
- Отношения «предок-потомок»
- Скорость
- Жесткость наборов отношений

Схема иерархической базы данных

Дерево, вершинами которого являются типы сегментов.



Иерархическая база данных — это совокупность деревьев, корнями которых являются экземпляры корневого типа сегмента.

Каждое такое дерево называется **набором** и образует **физическую запись** иерархической базы данных.

От записи к записи можно перемещаться, только если они объединены набором.

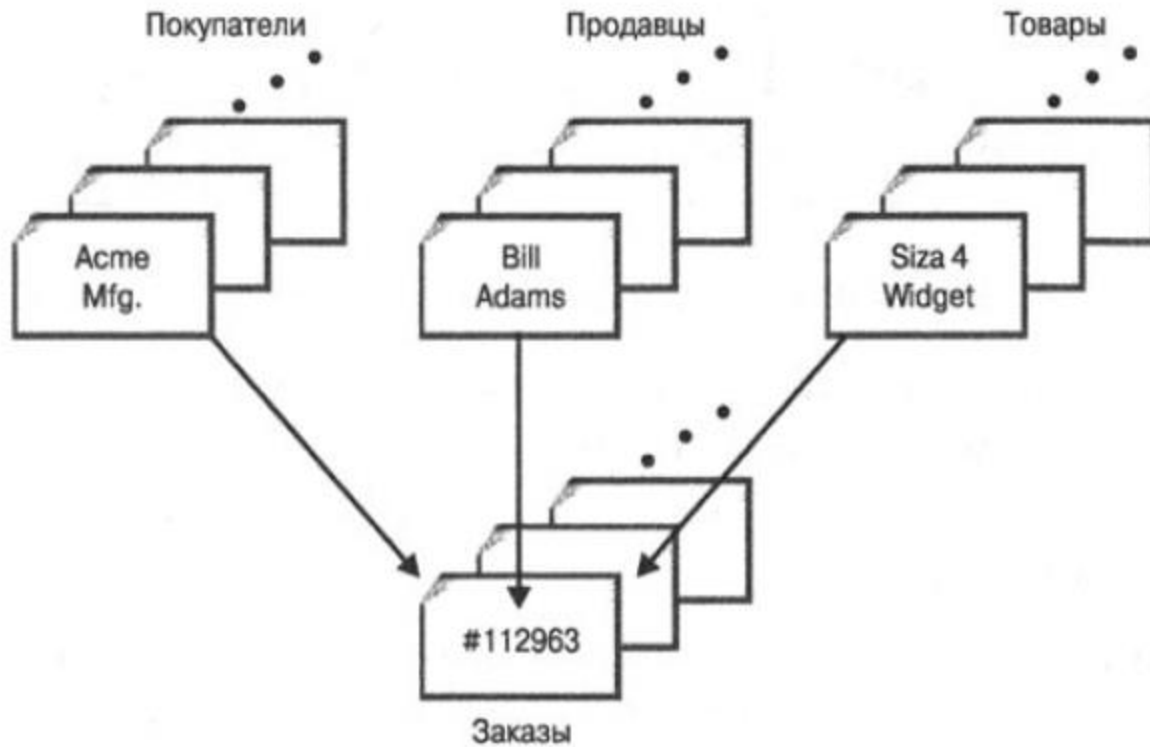
Типичные операторы для манипулирования данными

- Найти указанное дерево БД (например, аналитический отдел);
- Перейти от одного дерева к другому;
- Перейти от одной записи к другой внутри дерева (например, от отдела - к первому сотруднику);
- Перейти от одной записи к другой в порядке обхода иерархии;
- Вставить новую запись в указанную позицию;
- Удалить текущую запись.

Иерархическая модель данных: резюме

- Данные в иерархической БД упорядочены: всегда есть первая запись и список следующих и предыдущих.
- В БД хранятся не только данные, но и связи между ними (наборы). Наборы – существенный и обязательный объект, т.к. от записи к записи можно перемещаться, только если они объединены набором.
- Навигационная БД предполагает исключительно процедурный стиль программирования – разработчик сам определяет алгоритм получения интересующих его данных в рамках существующих в БД связей; изменятся связи – перестанут работать программы.

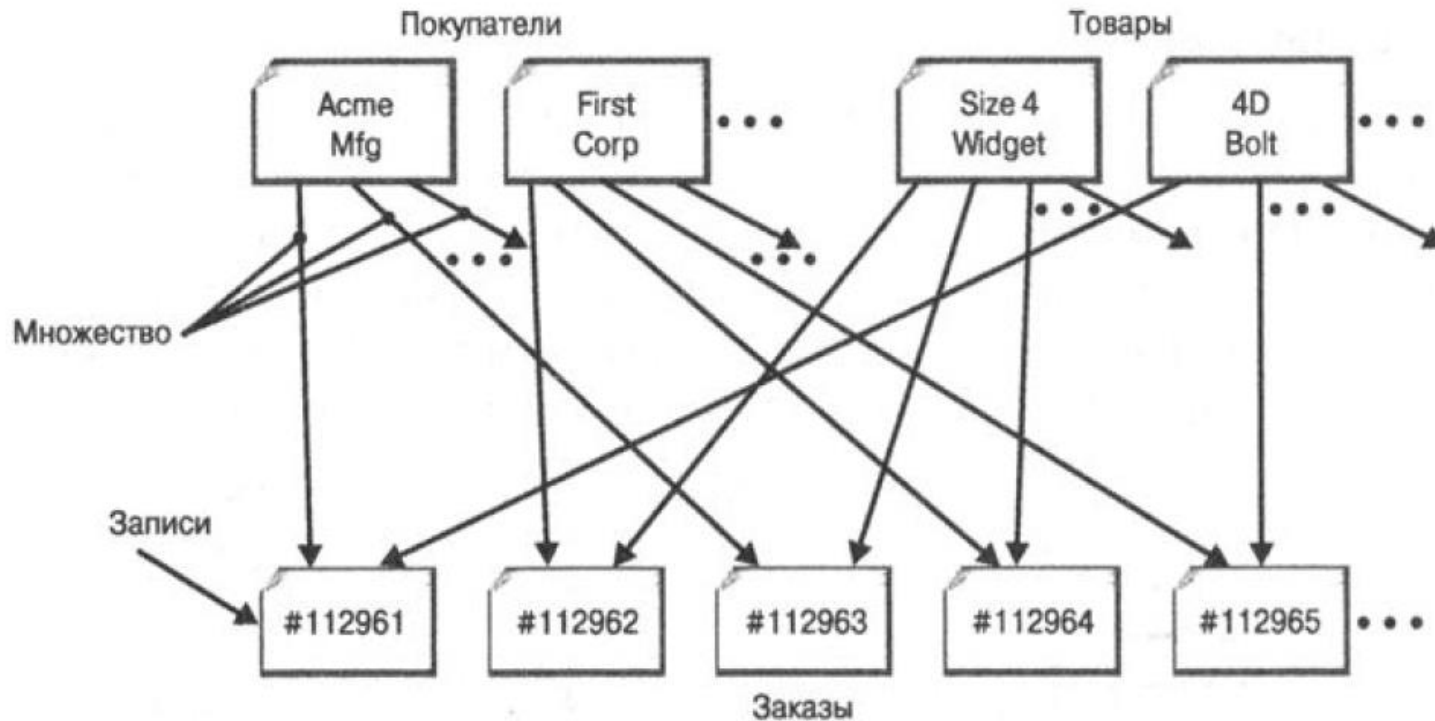
Сетевая модель



Задачи учета заказов

Множественные отношения «предок-потомок»

Сетевая модель

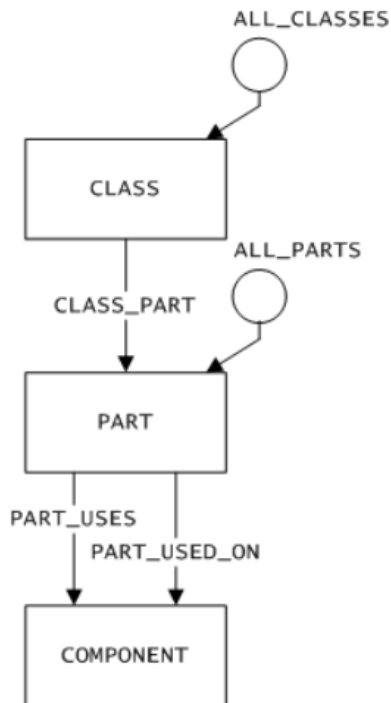


- Гибкость (по сравнению с иерархической)
- Стандартизация (CODASYL)
- Производительность
- Жесткость наборов отношений

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.
...
```

Иерархическая модель данных: плюсы и минусы

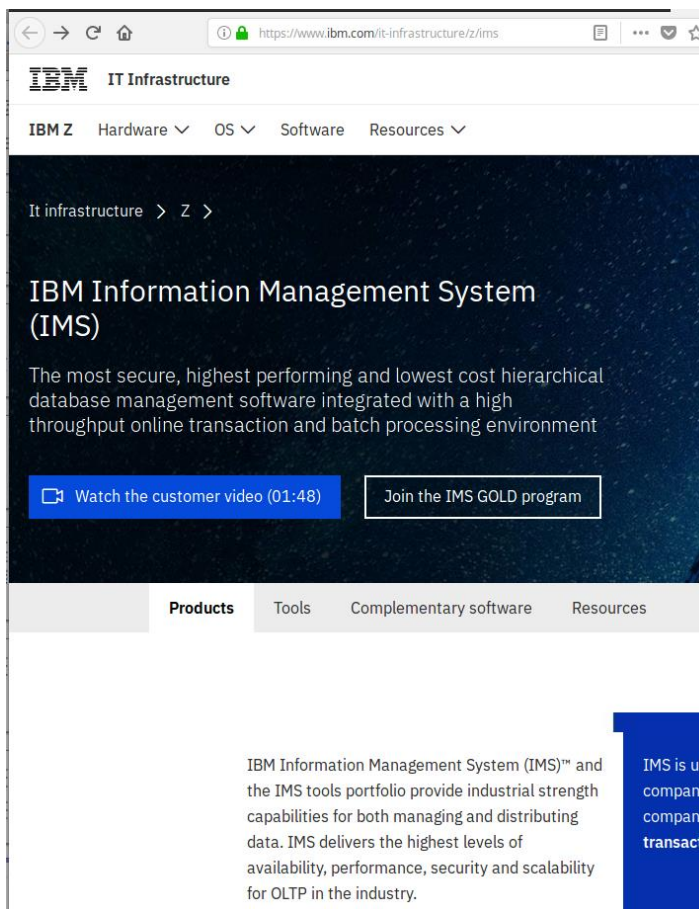
Сильные места ранних СУБД:

- Развитые средства управления данными во внешней памяти на низком уровне;
- Возможность построения вручную эффективных прикладных систем;
- Возможность экономии памяти за счет разделения подобъектов (в сетевых системах).

Недостатки:

- Слишком сложно пользоваться (требуется написание большого количества кода, возрастает вероятность допустить ошибку);
- Фактически необходимы знания о физической организации данных;
- Прикладные системы зависят от этой организации, их логика перегружена деталями организации доступа к БД.

Иерархические БД сегодня



The screenshot shows the IBM IT Infrastructure website. The main heading is "IBM Information Management System (IMS)". Below it, a description states: "The most secure, highest performing and lowest cost hierarchical database management software integrated with a high throughput online transaction and batch processing environment". There are two buttons: "Watch the customer video (01:48)" and "Join the IMS GOLD program". At the bottom, there is a navigation bar with "Products", "Tools", "Complementary software", and "Resources".

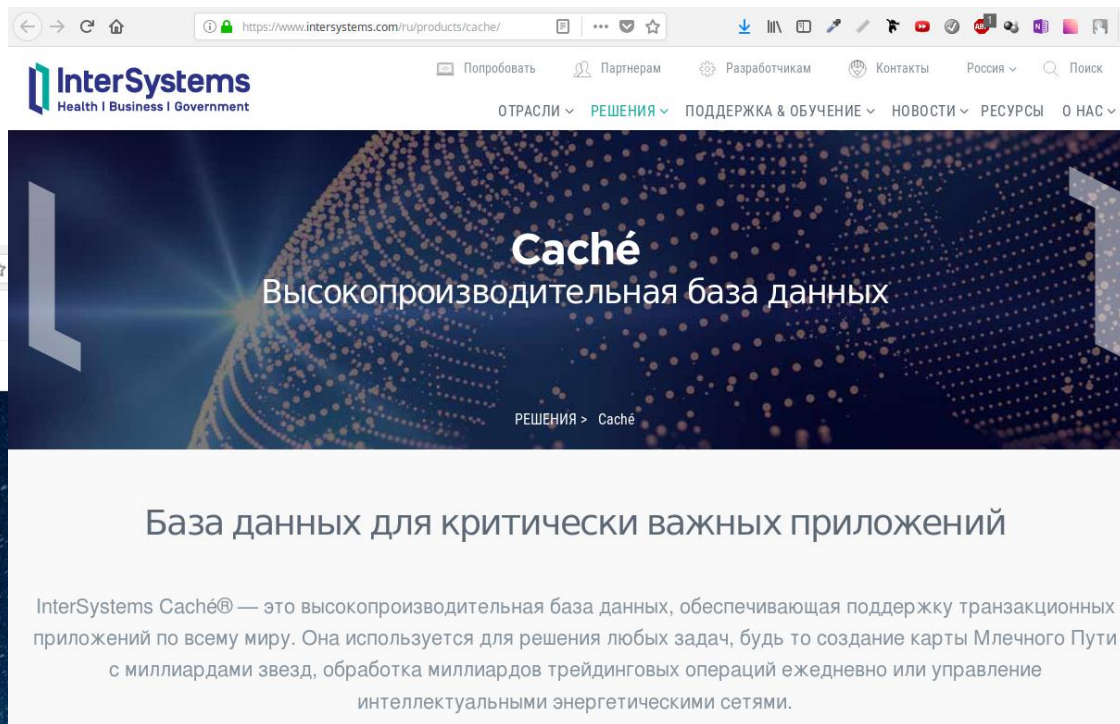
IBM Information Management System (IMS)

The most secure, highest performing and lowest cost hierarchical database management software integrated with a high throughput online transaction and batch processing environment

[Watch the customer video \(01:48\)](#) [Join the IMS GOLD program](#)

Products Tools Complementary software Resources

IBM Information Management System (IMS)™ and the IMS tools portfolio provide industrial strength capabilities for both managing and distributing data. IMS delivers the highest levels of availability, performance, security and scalability for OLTP in the industry.



The screenshot shows the InterSystems Cache website. The main heading is "Cache" with the subtitle "Высокопроизводительная база данных". Below it, a description states: "База данных для критически важных приложений". There is a button "Watch the customer video (01:48)".

InterSystems Cache® — это высокопроизводительная база данных, обеспечивающая поддержку транзакционных приложений по всему миру. Она используется для решения любых задач, будь то создание карты Млечного Пути с миллиардами звезд, обработка миллиардов трейдинговых операций ежедневно или управление интеллектуальными энергетическими сетями.

[Watch the customer video \(01:48\)](#)

IMS is used by many of the Fortune 1000 companies worldwide. Collectively these companies process more than **265 billion transactions per day** in IMS – securely.

Компьютеры в начале 1970-х годов



- ЭВМ используются в различных сферах
- Мини-компьютеры (PDP фирмы DEC) доступны небольшим предприятиям
- Компьютеры становятся интерактивными
- Развиваются многопользовательские ОС и файловые системы
- Множество языков программирования
- Появляются идеи об использовании компьютеров непрофессионалами



Навигационная модель vs реляционной

Проблемы **навигационной** модели:

- БД – инструмент программиста (трудно использовать, долгая реализация запросов пользователей).
- 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

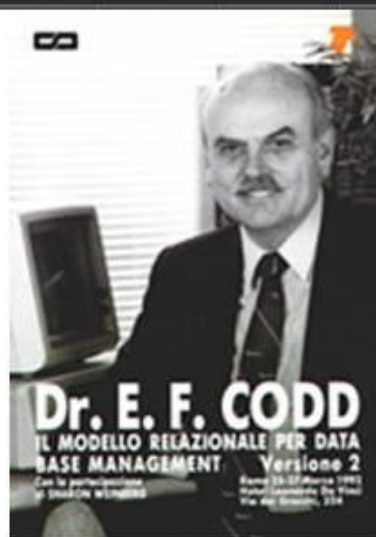
E. F. Codd

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

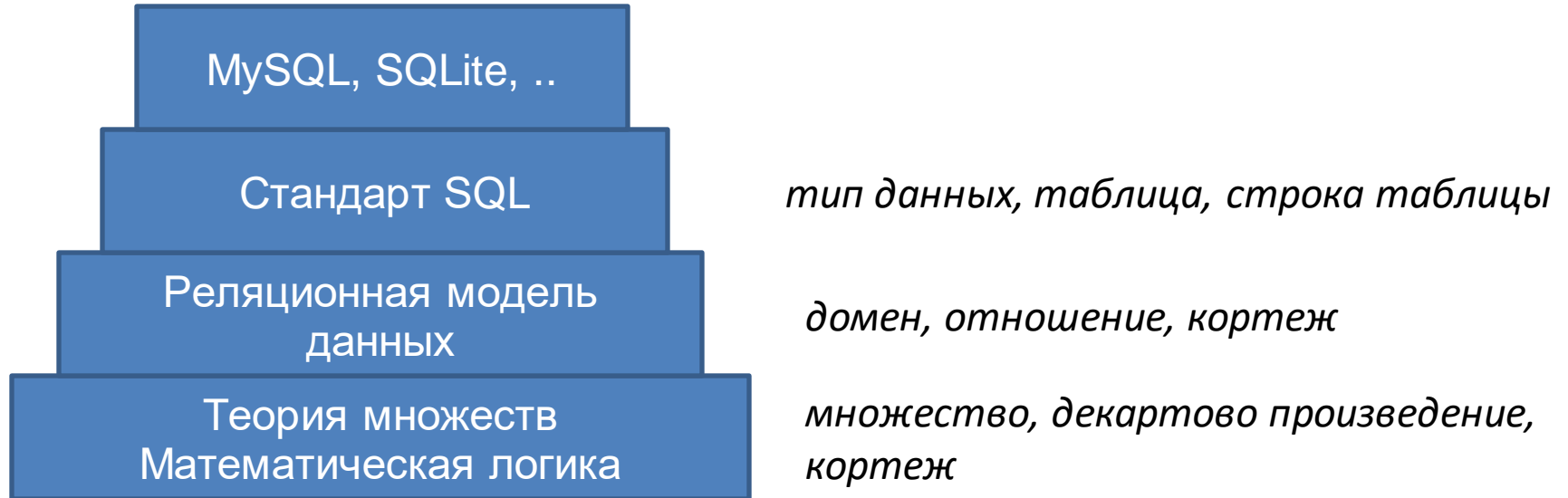
Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

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)

Математические основы реляционной модели



Математические основы реляционной модели

a_1^1	a_2^1	...	a_n^1	a_1^1	a_2^1	...	a_n^1	...	a_1^1	a_2^1	...	a_n^1
a_1^2	a_2^2	...	a_n^2	a_1^2	a_2^2	...	a_n^2		a_1^2	a_2^2	...	a_n^2
...
a_1^m	a_2^m	...	a_n^m	a_1^m	a_2^m	...	a_n^m		a_1^m	a_2^m	...	a_n^m
a_1^1	a_2^1	...	a_n^1	a_1^1	a_2^1	...	a_n^1	...	a_1^1	a_2^1	...	a_n^1
a_1^2	a_2^2	...	a_n^2	a_1^2	a_2^2	...	a_n^2		a_1^2	a_2^2	...	a_n^2
...
a_1^m	a_2^m	...	a_n^m	a_1^m	a_2^m	...	a_n^m		a_1^m	a_2^m	...	a_n^m
...						
a_1^1	a_2^1	...	a_n^1	a_1^1	a_2^1	...	a_n^1	...	a_1^1	a_2^1	...	a_n^1
a_1^2	a_2^2	...	a_n^2	a_1^2	a_2^2	...	a_n^2		a_1^2	a_2^2	...	a_n^2
...
a_1^m	a_2^m	...	a_n^m	a_1^m	a_2^m	...	a_n^m		a_1^m	a_2^m	...	a_n^m

Отношение степени n - подмножество R декартового произведения множеств $A_1 \times A_2 \times \dots \times A_n$.

- 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 $A = \{\text{Вова, Петя, Маша, Лена}\}$. 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 (несчастливая любовь).

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

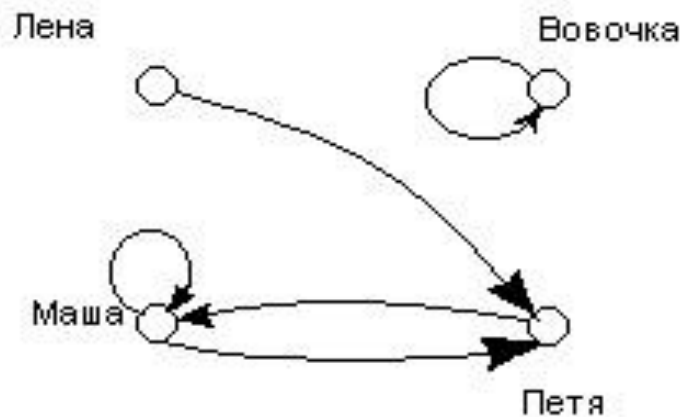
Как представить это отношение, чтобы с ним удобно было работать на компьютере?

Способ 1. Произвольный текст. 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 (несчастливая любовь).

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 строки, 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, \dots, D_n , называется подмножество из $D_1 \times D_2 \times \dots \times D_n$.
- Множества D_1, D_2, \dots, 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 $\{d_1, d_2, \dots, d_n\}$ – **кортежи**.
- Число n – **степень** отношения.

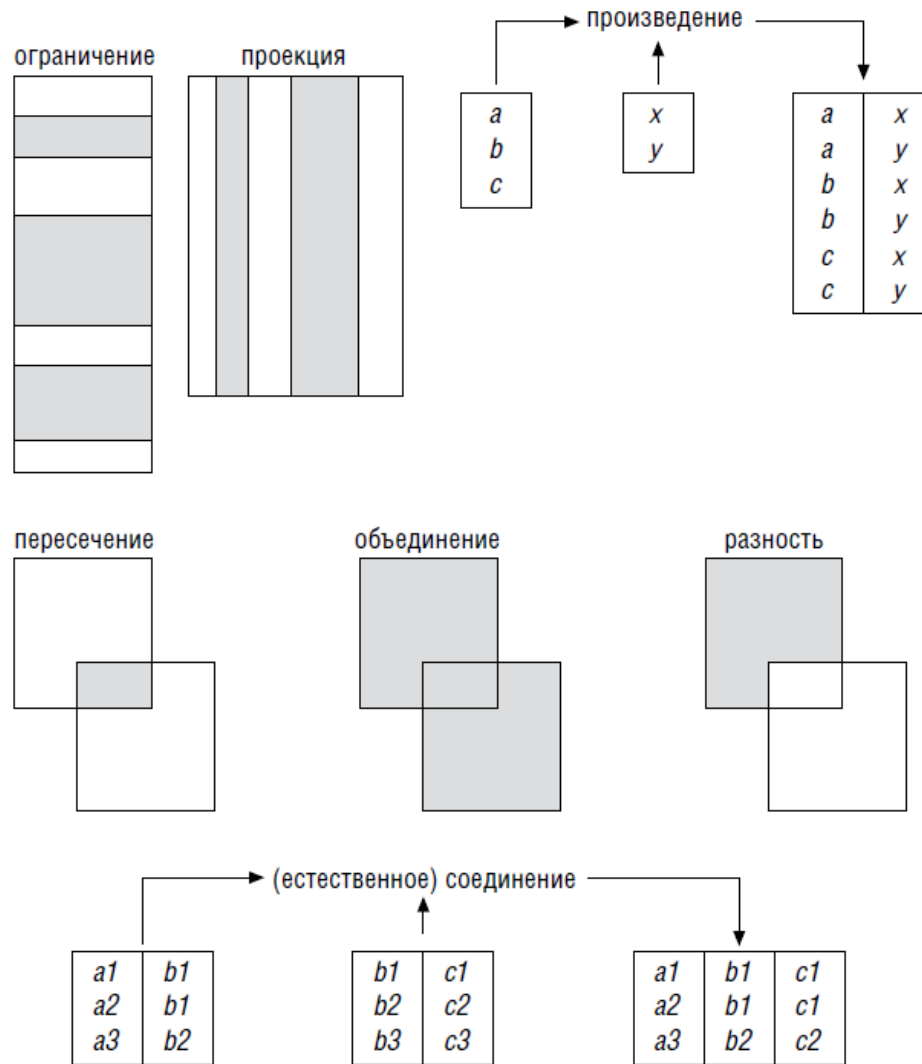
Основные понятия реляционной модели



Рис. 2.1. Наглядное представление основных понятий реляционной модели

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

Реляционная алгебра – способ работы с отношениями



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.

- **Проекция** – оставляет из всего набора атрибутов некоторое их подмножество.

$A[2,5]$ – проекция отношения A на 2 и 5 атрибуты

`SELECT DISTINCT f1, f2 FROM A` (по именам столбцов)

Реляционная алгебра – способ работы с отношениями

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

- **Соединение** двух отношений по условию равенства общих атрибутов – новое отношение, кортежи которого являются конкатенациями всевозможных пар кортежей исходных отношений, для которых выполняется это условие.

$A[1=3]B$ – соединение отношений A и B по общим атрибутам 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.

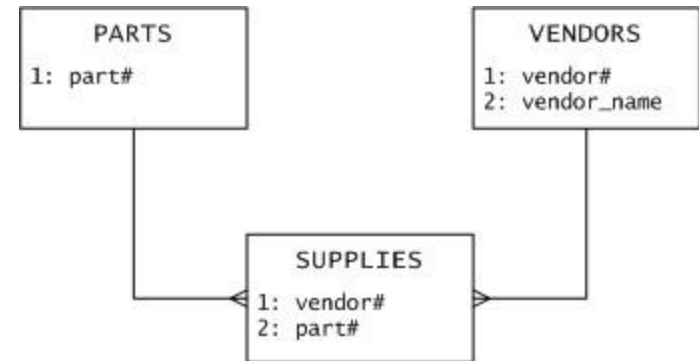
- **Ограничение** – оставляет лишь те кортежи, которые удовлетворяют определенному условию на атрибуты.

$A[1 > 2]$ – 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).



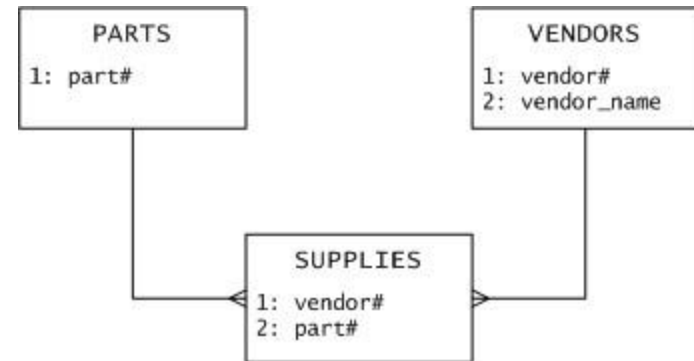
Пример использования операций реляционной алгебры

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



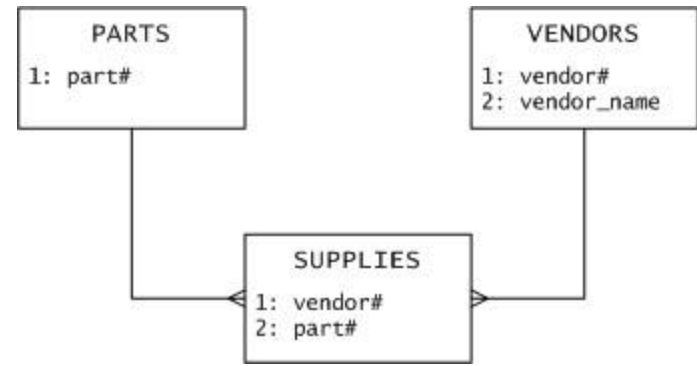
Пример использования операций реляционной алгебры

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

1. Номер поставщика, поставляющего хоть что-нибудь:

$S[1]$

$\text{SELECT DISTINCT vendor\# FROM S}$



2. Имя поставщика, не поставившего ни одной детали:

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

$\text{SELECT vendor_name}$
 $\text{FROM V, (SELECT vendor\# FROM V EXCEPT SELECT vendor\# FROM S) V2}$
 $\text{WHERE V.vendor\# = V2.vendor\#}$

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**.

- 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.

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

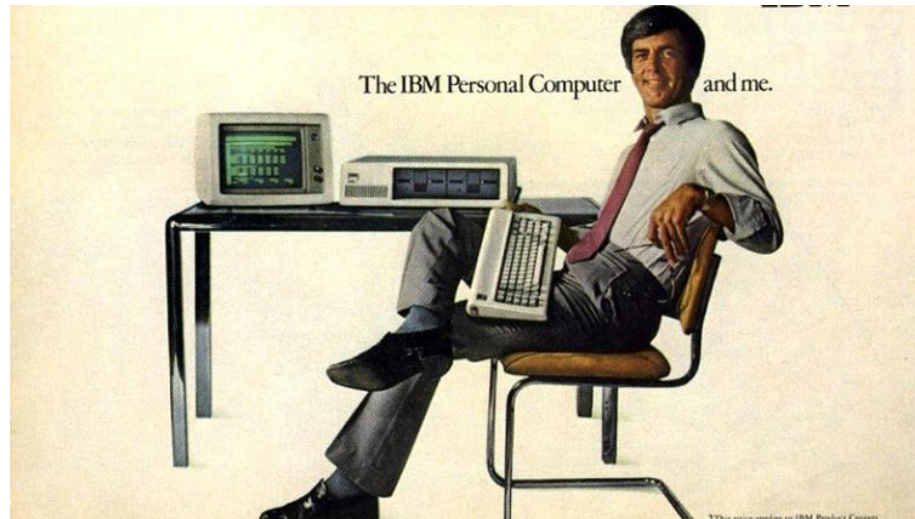
Кроме повышения продуктивности программистов и простоты использования:

- 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).

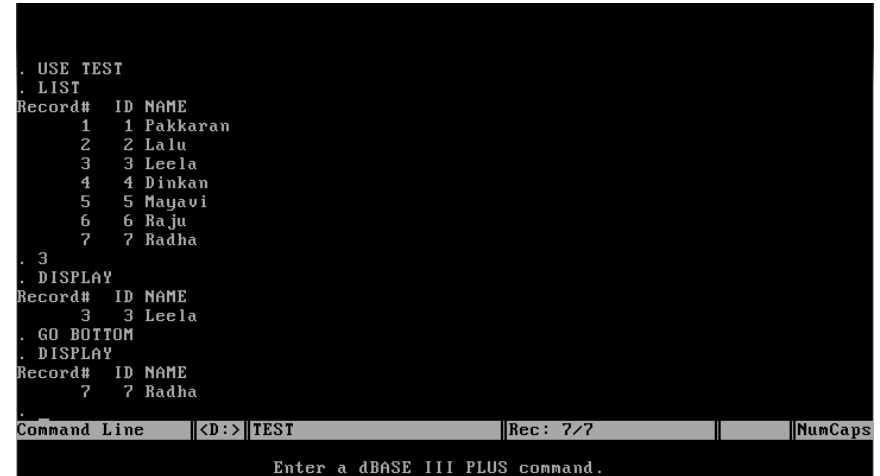
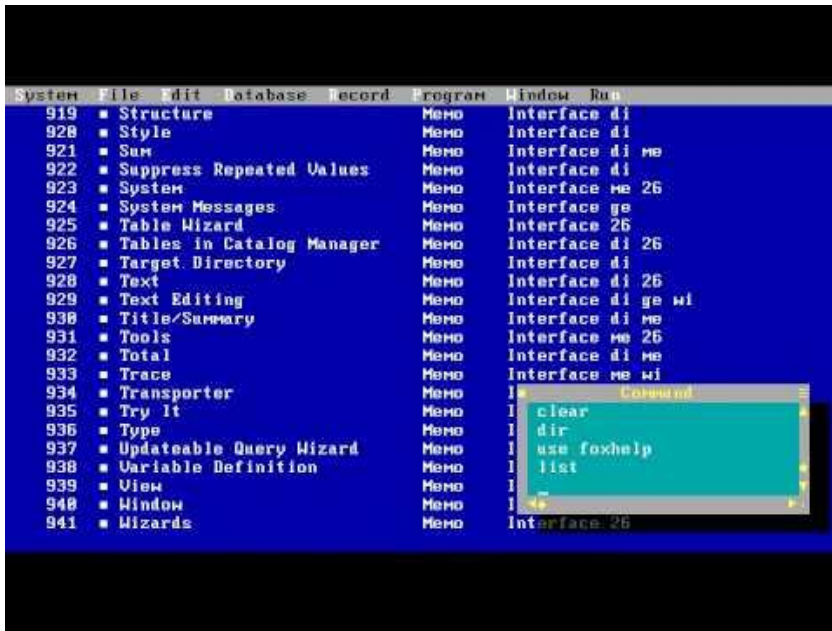
Компьютеры в начале 1980-х годов



- Появляются персональные микро-компьютеры
- Ресурсы микро-компьютера очень ограничены, сложная ОС для него не нужна
- С компьютерами работает все больше непрофессионалов



Навигационный подход к реляционной модели



- Формат DBF
- СУБД dBase для CP/M (1980)
- FoxPro, Clipper, Paradox, ...

