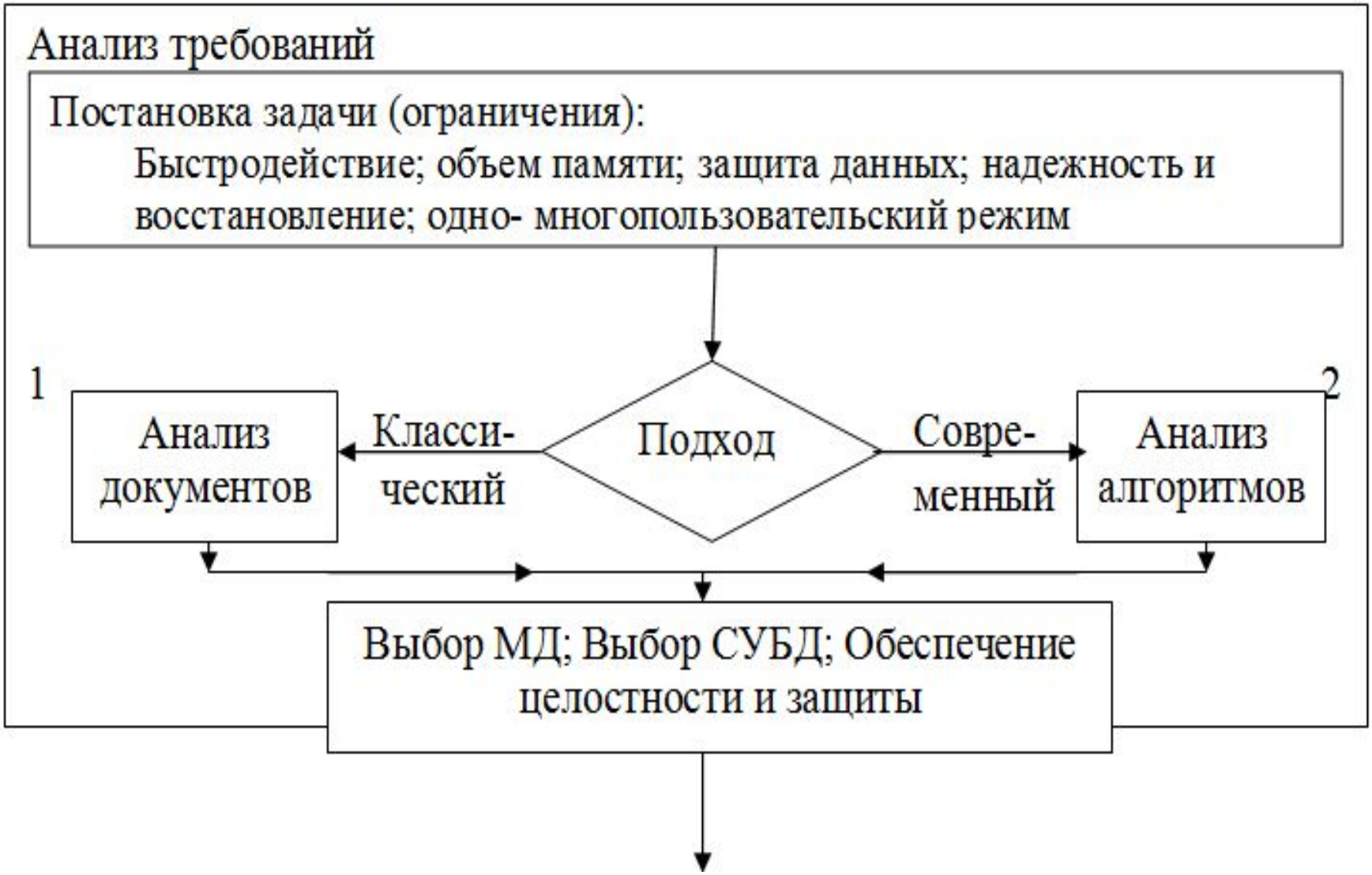
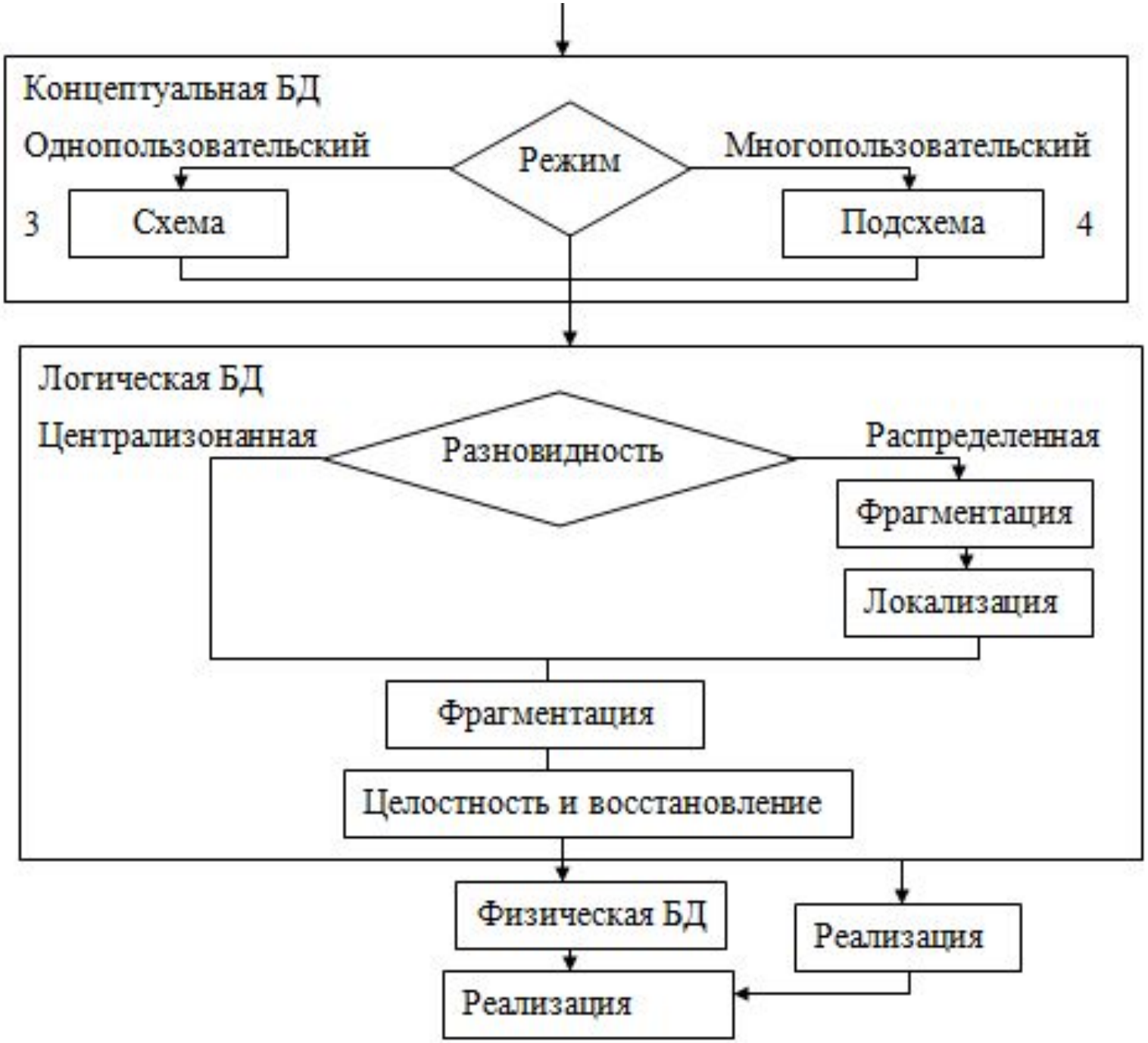
Проектирование баз данных

Существует много разновидностей методологии рассмотрения баз данных.

На рис показана совокупность процедур проектирования централизованной БД, которые можно объединить в **четыре** этапа.


Анализ требований
Постановка задачи (ограничения):
Быстродействие; объем памяти; защита данных; надежность и восстановление; одно- многопользовательский режим
1
Анализ документов
Класси-ческий
Подход
Совре-менный
2
Анализ алгоритмов
Выбор МД; Выбор СУБД; Обеспечение целостности и защиты


Концептуальная БД
Однопользовательский
Режим
Многопользовательский
3
Схема
Подсхема
4
Логическая БД
Централизонанная
Разновидность
Распределенная
Фрагментация
Локализация
Фрагментация
Целостность и восстановление
Физическая БД
Реализация
Реализация

1. Первый этап – формулирование и анализ

На **этапе формулирования и анализа** требований *устанавливаются цели организации, определяются требования к БД.* Они состоят из:

- Общих требований.
- Специфических требований (используется методика интервьюирования (опроса) персонала различных уровней управления).

Все требования документируются в форме, доступной конечному пользователю и проектировщику БД.

2. Этап концептуального проектирования

Этап **концептуального проектирования** *заключается в описании и синтезе информационных требований пользователей в первоначальный проект БД.*

Исходными данными могут быть:

- Совокупность документов пользователя при классическом подходе
- Алгоритмы приложений (алгоритмы бизнеса) при современном подходе.

Результатом этого этапа является высокоуровневое представление (в виде системы таблиц БД) информационных требований пользователей на основе различных подходов.

Требования пользователей включают в себя:

1. Выбор модели БД.
2. Создание структуры БД, которая заполняется данными с помощью команд, систем меню, экранных форм или в режиме просмотра таблиц БД.

Обеспечивается защита и целостность данных с помощью СУБД .

3. Этап логического проектирования

В **процессе логического проектирования** *высокоуровневое представление данных преобразуется в структуру используемой СУБД.*

Основной целью этапа является устранение избыточности данных с использованием специальных правил нормализации.

Цель ***нормализации*** – минимизировать повторение данных и возможные структурные изменения БД при процедурах обновления.

Это достигается разделением (декомпозицией) одной таблицы в две или несколько с последующим использованием при запросах операции навигации.

Полученная логическая структура БД может быть оценена **количественно** с помощью различных характеристик:

- число обращений к логическим записям;
- объем данных в каждом приложении;
- общий объем данных.

На основе этих оценок логическая структура может быть усовершенствована с целью достижения большей эффективности.

Процедура управления БД.

- проста в однопользовательском режиме.
- В многопользовательском режиме и в распределенных БД процедура сильно усложняется. При одновременном доступе нескольких пользователей без принятия специальных мер возможно нарушение целостности.

Для устранения этого явления используют **систему транзакций** и режим блокировки таблиц или отдельных записей.

Транзакция – процесс изменения файла, записи или базы данных, вызванный передачей одного входного сообщения.

4. Этап физического проектирования

На **этапе физического проектирования** решаются вопросы, связанные с производительностью системы, определяются структуры хранения данных и методы доступа.

Количественными критерии

- Время ответа на запрос;
- Стоимость модификации;
- Стоимость памяти;
- Время на создание;
- стоимость на реорганизацию.

Затруднение может вызывать противоречие критериев друг другу.

Качественные критерии

- Гибкость;
- Адаптивность;
- Доступность для новых пользователей;
- Совместимость с другими системами;
- Возможность конвертирования в другую вычислительную среду;
- Возможность восстановления;
- Возможность распределения и расширения.

Процесс проектирования является длительным и трудоемким и обычно продолжается несколько месяцев.

Основными ресурсами проектировщика БД являются его собственная интуиция и опыт, поэтому качество решения во многих случаях может оказаться низким.

Основными **причинами низкой эффективности проектируемых БД** могут быть:

- недостаточно глубокий анализ требований (начальные этапы проектирования);
- большая длительность процесса структурирования, делающая этот процесс утомительным и трудно выполнимым при ручной обработке.

В этих условиях важное значение приобретают вопросы автоматизации разработки.

2.1. Классический подход к проектированию баз данных

Этот подход восходит к автоматизированным системам управления информационно-поискового типа, основной целью которых являлась автоматизация документооборота (совокупности документов, циркулирующих на предприятии).

I. При **проектировании** с использованием классического подхода исходят из следующих положений.

1. На входе БД имелась *одна система документов*, которая при использовании БД *трансформировалась в другую (выходную) систему документов (таблиц, файлов).*

2. Запрос к БД мог осуществляться регулярно (по регламенту) или по запросу. (Запрос предполагал получение любой структуры документов из имеющихся в БД полей данных).
3. Данные БД использовались затем для задач принятия решений, становясь информацией. (Считалось, что алгоритмы задач более подвижны, чем данные. В силу этого создание универсальной БД позволит решить любую задачу).

II. В проектировании БД, в соответствии с классической методологией выделялись следующие этапы:

- анализ требований;
- концептуальное моделирование;
- логическое моделирование;
- физическое моделирование;
- реализация.

При этом, как показала практика, акцент следует делать на начальные этапы, на которых имеет место наибольшее число трудно исправимых ошибок

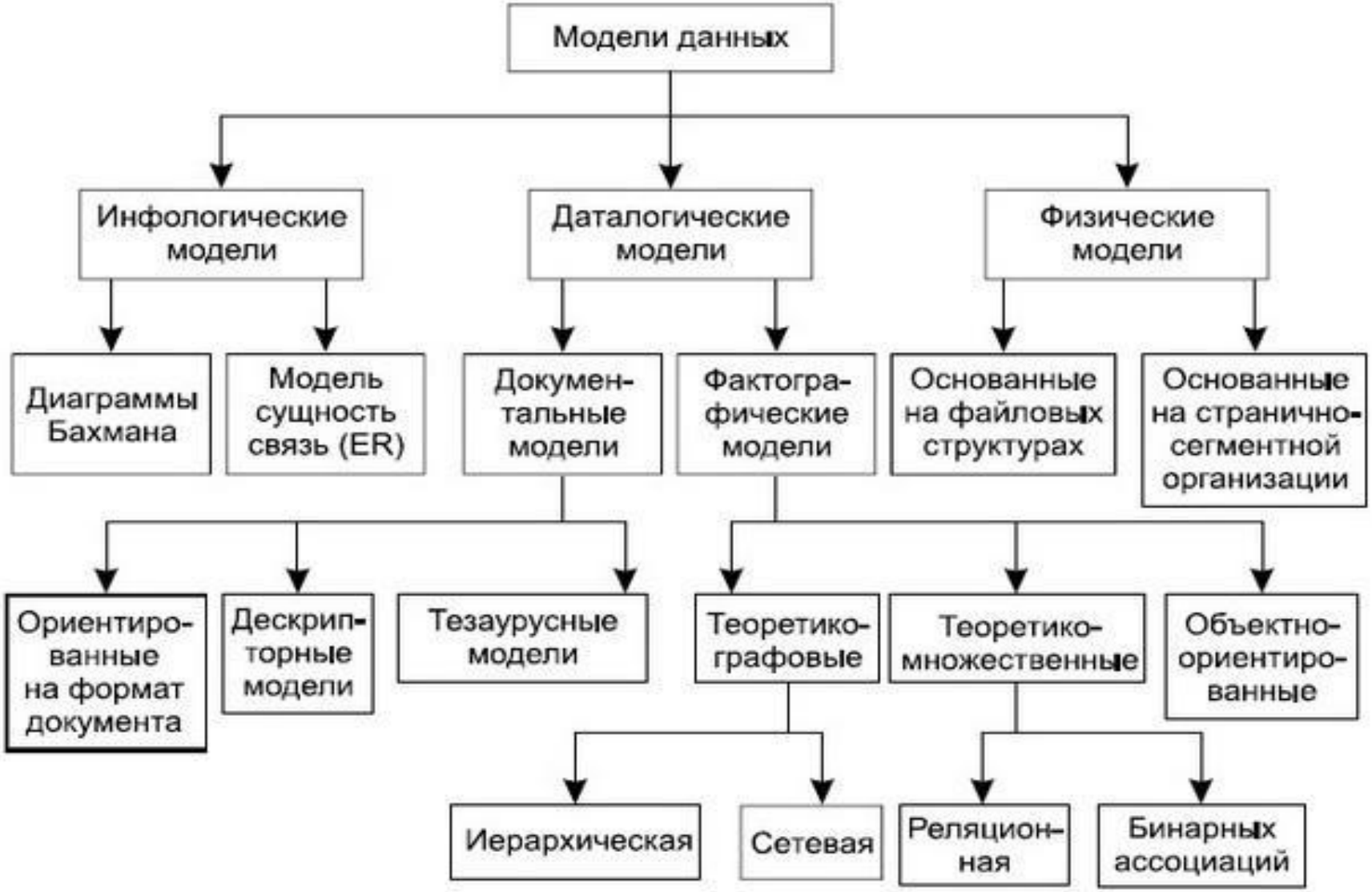
Этап	Процент выявленных ошибок	Стоимость исправления ошибок (по пятибалльной шкале)
Анализ физической реализации и первоначальные разработки	40	5
Детальное проектирование	50	3
Программирование	10	1, 2
Тестирование	–	1

3. МОДЕЛИ ДАННЫХ

- Хранимые в базе данные имеют определенную логическую структуру – иными словами, описываются некоторой ***моделью представления данных*** (моделью данных), поддерживаемой СУБД.

В некоторых СУБД поддерживаются одновременно несколько моделей данных. Например, в системе ИНТЕРБАЗА для приложений применяется сетевой язык манипулирования данными, а в пользовательском интерфейсе реализованы языки SQL и QBE

3.1. Классификация моделей данных



3.1.1. Инфологическая модель базы

Инфологическая модель базы данных представляет собой описание объектов (сущностей), с набором атрибутов и связей между ними, которые выявляются в процессе исследования как входных, так и выходных данных.

Самая распространенная модель в инфологическом моделировании это модель "сущность-связь" или ER-модель, к главным компонентам её относятся - сущности и связи.

Сущности - содержание объекта, о котором набирают необходимую информацию.

Экземпляром сущности представляется - чёткий объект.

ER-модель, предложенная П. Ченом в 1976 г., является наиболее известным представителем класса семантических (концептуальных, инфологических) моделей предметной области и представляется в графической форме, с использованием оригинальной нотации

П. Чена, называемой ER-диаграмма

Основные преимущества ER-моделей:

- наглядность;
- модели позволяют проектировать базы данных с большим количеством объектов и атрибутов;
- ER-модели реализованы во многих системах автоматизированного проектирования баз данных (например, ERWin).

Основные элементы ER-моделей:

- объекты (сущности);
- атрибуты объектов;
- связи между объектами.

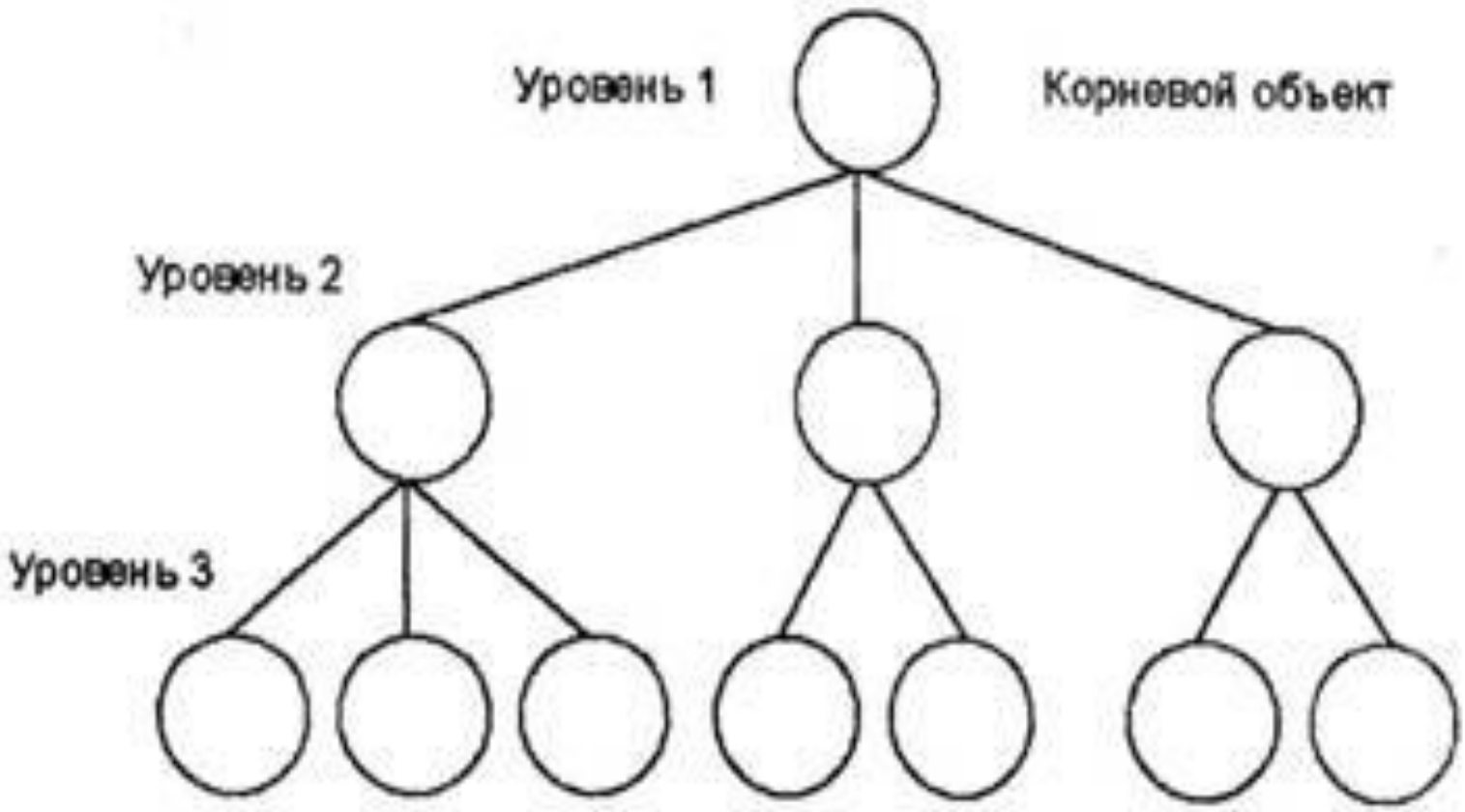
Связь между сущностями характеризуется:

- типом связи (1:1, 1:N, N:M);
- классом принадлежности. Класс может быть обязательным и необязательным. Если каждый экземпляр сущности участвует в связи, то класс принадлежности — обязательный, иначе — необязательный.

Диаграмма Бахмана фактически является графом типов БД. Любопытно, что граф знаков (каждой вершине соответствует конкретная запись, а дуге – конкретный набор) является лесом, т.е. графом, каждая компонента связности которого является деревом.

3.2. Иерархическая модель

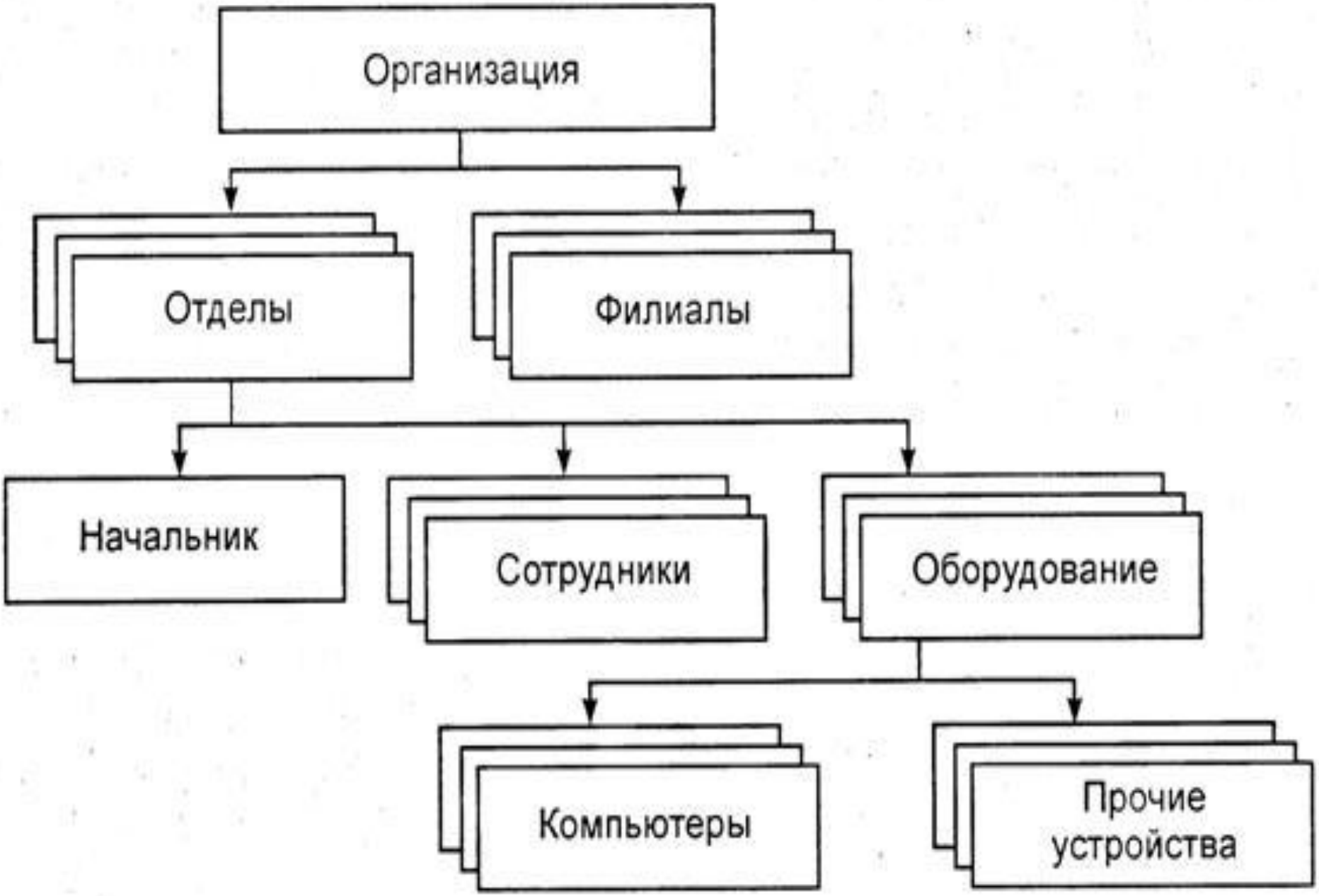
В иерархической модели связи между данными можно описать с помощью упорядоченного графа (или дерева).



Тип ***дерево*** является составным.

включает в себя подтипы (***поддеревья***), каждый из которых, в свою очередь, является типом «дерево».

Каждый из типов **дерево** состоит из одного ***корневого*** типа и упорядоченного набора (возможно, пустого) подчиненных типов.


Организация
Отделы
Филиалы
Начальник
Сотрудники
Оборудование
Компьютеры
Прочие устройства

Корневым называется тип, который имеет подчиненные типы и сам не является подтипом.

Подчиненный тип (подтип) является *потомком* по отношению к типу, который выступает для него в роли предка (родителя).

Потомки одного и того же типа являются *близнецами* по отношению друг к другу.

В целом тип «дерево» представляет собой иерархически организованный набор типов ***запись***. Поля записей хранят собственно числовые или символьные значения, составляющие основное содержание БД. Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо.

достоинства иерархической модели данных

- эффективное использование памяти ЭВМ
- неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией.

Недостатки иерархической модели

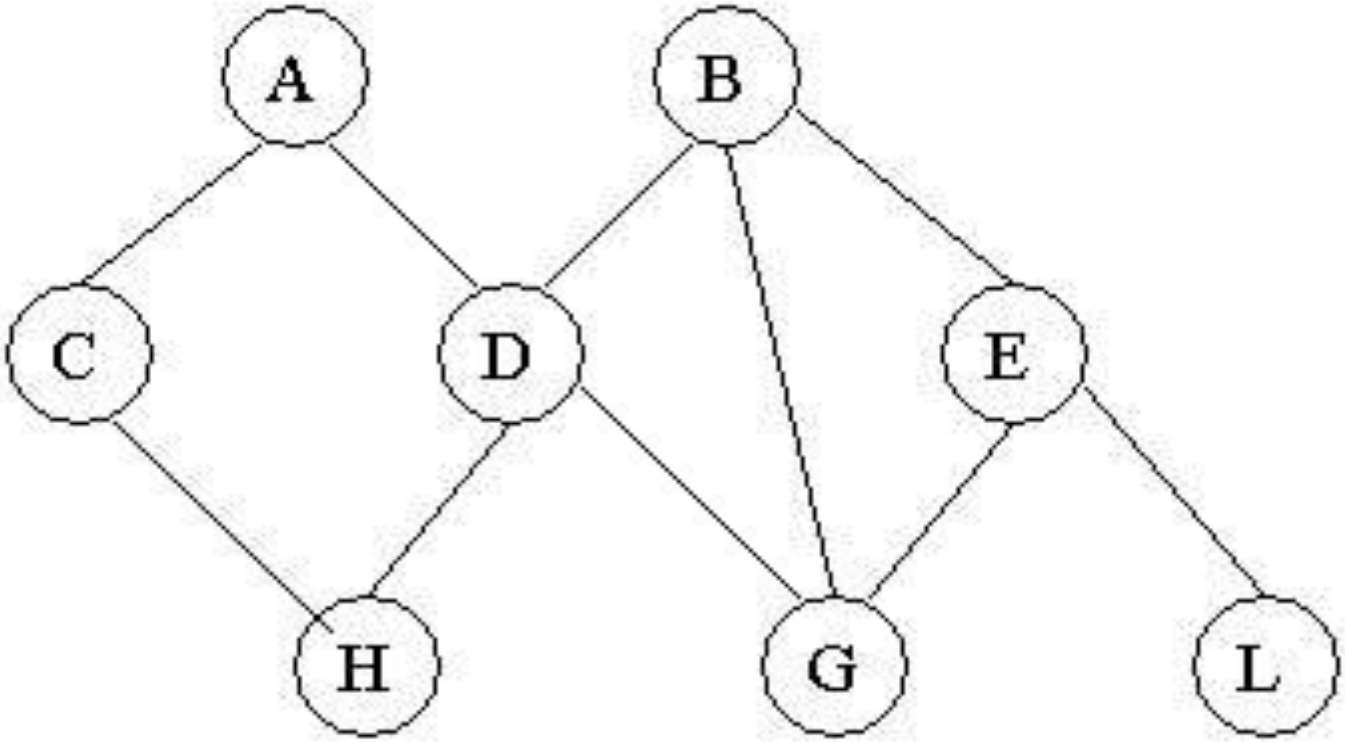
- громоздкость для обработки информации с достаточно сложными логическими связями,
- сложность понимания для обычного пользователя.

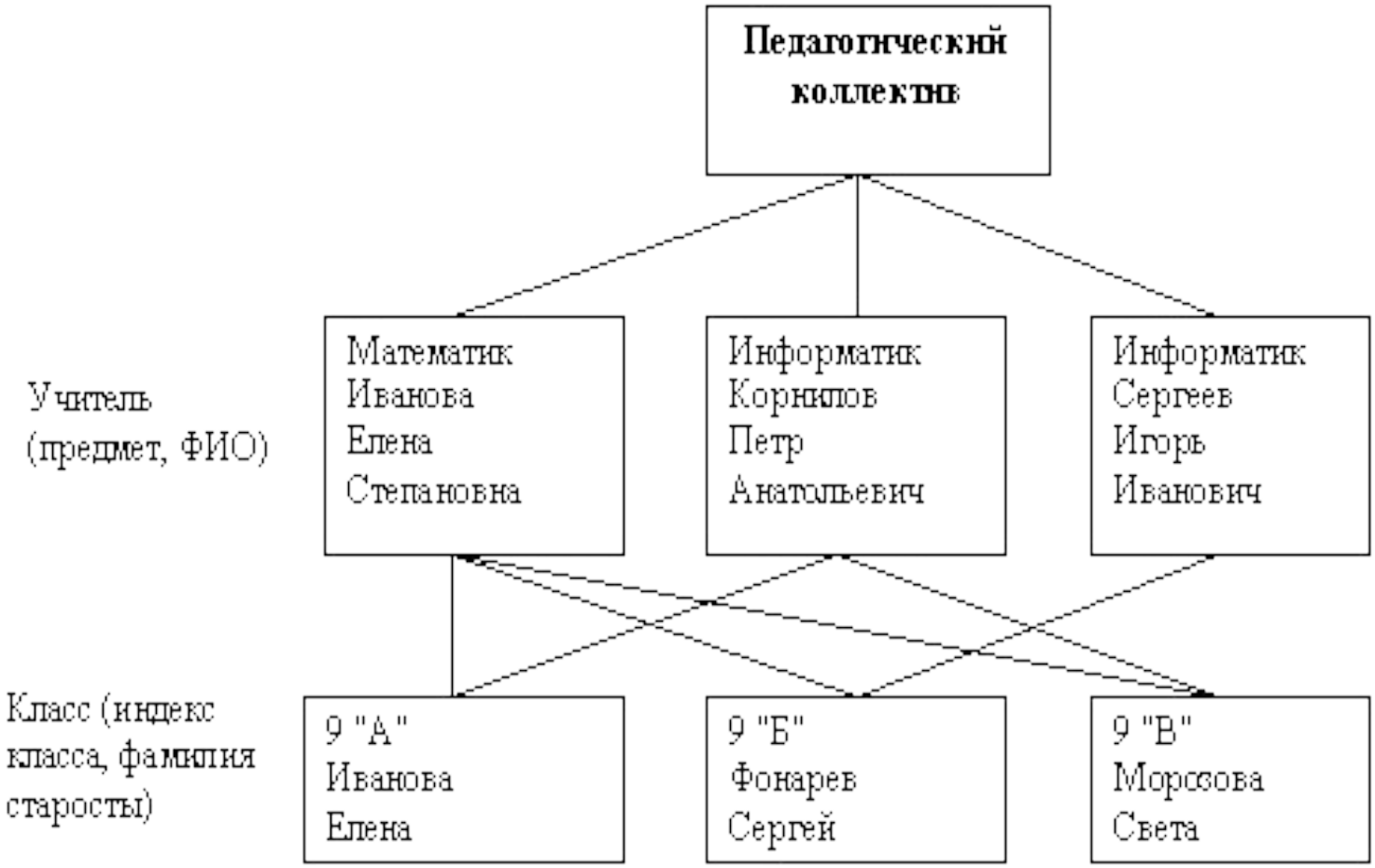
На иерархической модели данных основано сравнительно ограниченное количество СУБД, в числе которых можно назвать зарубежные системы IMS, PC/Focus, Team-Up и Data Edge, а также отечественные системы Ока, Р1НЭС и МИРИС.

3.3. Сетевая модель

Разница между иерархической моделью данных и сетевой том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных у потомка может иметься любое число предков.

- Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель данных.
- Наиболее полно концепция сетевых БД впервые была изложена в Предложениях группы КОДАСИЛ (KODASYL).


A
B
C
D
E
H
G
L


Педагогический коллектив
Учитель (предмет, ФИО)
Математик Иванова Елена Степановна
Информатик Корнилов Петр Анатольевич
Информатик Сергеев Игорь Иванович
Класс (индекс класса, фамилия старосты)
9 "А" Иванова Елена
9 "Б" Фонарев Сергей
9 "В" Морозова Света

Операций манипулирования данными баз сетевого

- поиск записи в БД;
- переход от предка к первому потомку;
- переход от потомка к предку;
- создание новой записи;
- удаление текущей записи;
- обновление текущей записи;
- включение записи в связь;
- исключение записи из связи;
- изменение связей и т. д.

Достоинство сетевой модели данных

- возможность эффективной реализации по показателям затрат памяти и оперативности.
- В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей.

Недостатки сетевой модели данных

- высокая сложность и жесткость схемы БД, построенной на ее основе,
- сложность для понимания и выполнения обработки информации в БД обычным пользователем.
- ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Системы на основе сетевой модели не получили широкого распространения на практике.

Наиболее известными сетевыми СУБД являются следующие: IDMS, db_VistaIII, СЕТЬ, СЕТОР и КОМПАС.

3. Реляционная модель

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии отношение (relation).

Отношение представляет собой множество элементов, называемых кортежами.

Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица.

Таблица имеет:

- строки (записи)
- столбцы (колонки).

Каждая строка таблицы имеет одинаковую структуру и состоит из полей.

Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики:

- фамилия,
- имя
- отчество,
- пол,
- возраст
- образование.

Поскольку в рамках одной таблицы не удается описать более сложные логические структуры данных из предметной области, применяют ***связывание*** таблиц.

Достоинство реляционной модели данных

- простота,
- понятности
- удобстве физической реализации на ЭВМ.
- Именно простота и понятность для пользователя явились основной причиной их широкого использования.

Проблемы эффективности обработки данных этого типа технически вполне разрешимыми.

Основными ***недостатками*** реляционной модели

- отсутствие стандартных средств идентификации отдельных записей
- сложность описания иерархических и сетевых связей.

Примерами зарубежных реляционных СУБД для ПЭВМ являются следующие:

- dBaselll Plus и dBase IY (фирма Ashton-Tate),
- DB2 (IBM),
- R:BASE (Microrim),
- FoxPro ранних версий и FoxBase (Fox Software),
- Paradox и dBASE for Windows (Borland),
- FoxPro более поздних версий,
- Visual FoxPro и Access (Microsoft),
- Clarion (Clarion Software),
- Ingres (ASK Computer Systems) и Oracle (Oracle).

К отечественным СУБД реляционного типа относятся системы:

- ПАЛЬМА (ИК АН УССР),
- система HyTech (МИФИ).

3.5. Постреляционная модель

***Постреляционная** модель* данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц.

Постреляционная модель данных допускает многозначные поля, значения которых состоят из подзначений.

Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Пример

Информации о накладных и товарах для сравнения приведено представление одних и тех же данных с помощью реляционной *(а)* и постреляционной (*б*) моделей.

- Таблица НАКЛАДНЫЕ содержит данные о номерах накладных (№ накл) и номерах покупателей (№ пок).
- Ттаблица НАКЛАДНЫЕ-ТОВАРЫ содержатся данные о каждой из накладных: номер накладной (№ накл), название товара (Наим_тов) и количество товара (Кол-во).
- Таблица Накладная связана с таблицей НАКЛАДНЫЕ-ТОВАРЫ по полю № накл.

а) НАКЛАДНЫЕ

№ накл	№ пок
0373	8723
8374	8232
7364	8723

НАКЛАДНЫЕ-ТОВАРЫ

№ накл	Наим_тов	Кол-во
0373	Сыр	3
0373	Рыба	2
8374	Лимонад	1
8374	Сок	6
8374	Печенье	2
7364	Йогурт	1

б) Накладная

№ накл	№ пок	Наим_тов	Кол-во
0373	8723	Сыр	3
		Рыба	2
8374	8232	Лимонад	1
		Сок	6
		Печенье	2
7364	8723	Йогурт	1

На длину и количество полей в записях таблицы не накладывается требование постоянства.

Это означает, что структура данных и таблиц имеет большую гибкость.

Поскольку постреляционная модель допускает хранение в таблицах ненормализованных данных, возникает проблема обеспечения целостности и непротиворечивости данных.

Эта проблема решается включением в СУБД механизмов, подобных хранимым процедурам в клиент-серверных системах.

Достоинством постреляционной модели

- возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей.

Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

- ***Недостатки*** постреляционной модели сложность решения проблемы обеспечения целостности
- непротиворечивости хранимых данных.

Постреляционная модель данных поддерживается СУБД uniVers. К числу других СУБД, основанных на постреляционной модели данных, относятся также системы Bubba и Dasdb.

3.6. Многомерная модель

Многомерный подход к представлению данных в базе появился практически одновременно с реляционным, но реально работающих многомерных СУБД (МСУБД) до настоящего времени было очень мало.

С середины 90-х годов интерес к ним стал приобретать массовый характер.

Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия решения.

В развитии концепций ИС можно выделить следующие два направления:

- системы оперативной (транзакционной) обработки;
- системы аналитической обработки (системы поддержки принятия решений).

Реляционные СУБД предназначались для информационных систем *оперативной* обработки информации и в этой области были весьма эффективны.

Более эффективными для аналитической обработки оказываются многомерные СУБД (МСУБД).

Многомерные СУБД являются узкоспециализированными СУБД, предназначенными для *интерактивной аналитической обработки информации*.

Раскроем основные понятия, используемые в этих СУБД:

- агрегируемость,
- историчность
- прогнозируемость данных.

Агрегируемость данных означает рассмотрение информации на различных уровнях ее обобщения. В информационных системах степень детальности представления информации для пользователя зависит от его уровня:

- аналитик,
- пользователь-оператор,
- управляющий, руководитель.

Историчность данных предполагает обеспечение высокого уровня статичности (неизменности) собственно данных и их взаимосвязей, а также обязательность привязки данных ко времени.

Временная привязка данных необходима для частого выполнения запросов, имеющих значения времени и даты в составе выборки.

Прогнозируемость данных подразумевает задание функций прогнозирования и применение их к различным временным интервалам.

Многомерность модели данных означает не многомерность визуализации цифровых данных, а многомерное логическое представление структуры информации при описании и в операциях манипулирования данными.

По сравнению с реляционной моделью многомерная организация данных обладает более высокой ***наглядностью*** и ***информативностью***.

Модель	Месяц	Объем
Жигули	июнь	12
Жигули	июль	24
Жигули	август	5
Москвич	июнь	2
Москвич	июль	18
Волга	июль	19

Модель	Июнь	Июль	Август
Жигули	12	24	5
Москвич	2	18	
Волга		19	

Основные понятия многомерных моделей данных

Измерение (Dimension) – это множество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются Дни, Месяцы, Кварталы и Годы.

В качестве географических измерений широко употребляются Города, Районы, регионы и Страны.

В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка (Cell) или показатель – это поле, значение которого однозначно определяется фиксированным набором измерений.

Тип поля чаще всего определен как цифровой.

В зависимости от того, как формируются значения некоторой ячейки, она может быть:

- переменной (значения изменяются и могут быть загружены из внешнего источника данных или сформированы программно)
- формулой (значения, подобно формульным ячейкам электронных таблиц, вычисляются по заранее заданным формулам).

Измерения:
Время (год) — 2012, 2013, 2014
Менеджер — Петров, Смирнов, Яковлев
Модель — «Волга», «Жигули», «Москвич»
Показатель: Объем продаж

В многомерной модели данных применяется ряд специальных операций

- формирование среза,
- вращение,
- агрегация и детализация.

Срез (Slice) - подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений.

Например, если ограничить значения измерения Модель автомобиля в гиперкубе маркой «Жигули», то получится двухмерная таблица продаж этой марки автомобиля различными менеджерами по годам.

Операция ***вращение*** (Rotate) применяется при двухмерном представлении данных и заключается в изменении порядка измерений при визуальном представлении данных.

Операции ***агрегация*** (Drill Up) и «детализация» (Drill Down) означают соответственно переход к более общему и к более детальному представлению информации пользователю из гиперкуба.

Основным ***достоинством*** многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем.

Недостатком является ее громоздкость для простейших задач обычной оперативной обработки информации.

Примерами систем, поддерживающих многомерные модели данных, являются:

- Essbase (Arbor Software),
- Media Multi-matrix (Speedware),
- Oracle Express Server (Oracle)
- Cache (InterSystems).

3.7. Объектно-ориентированная модель

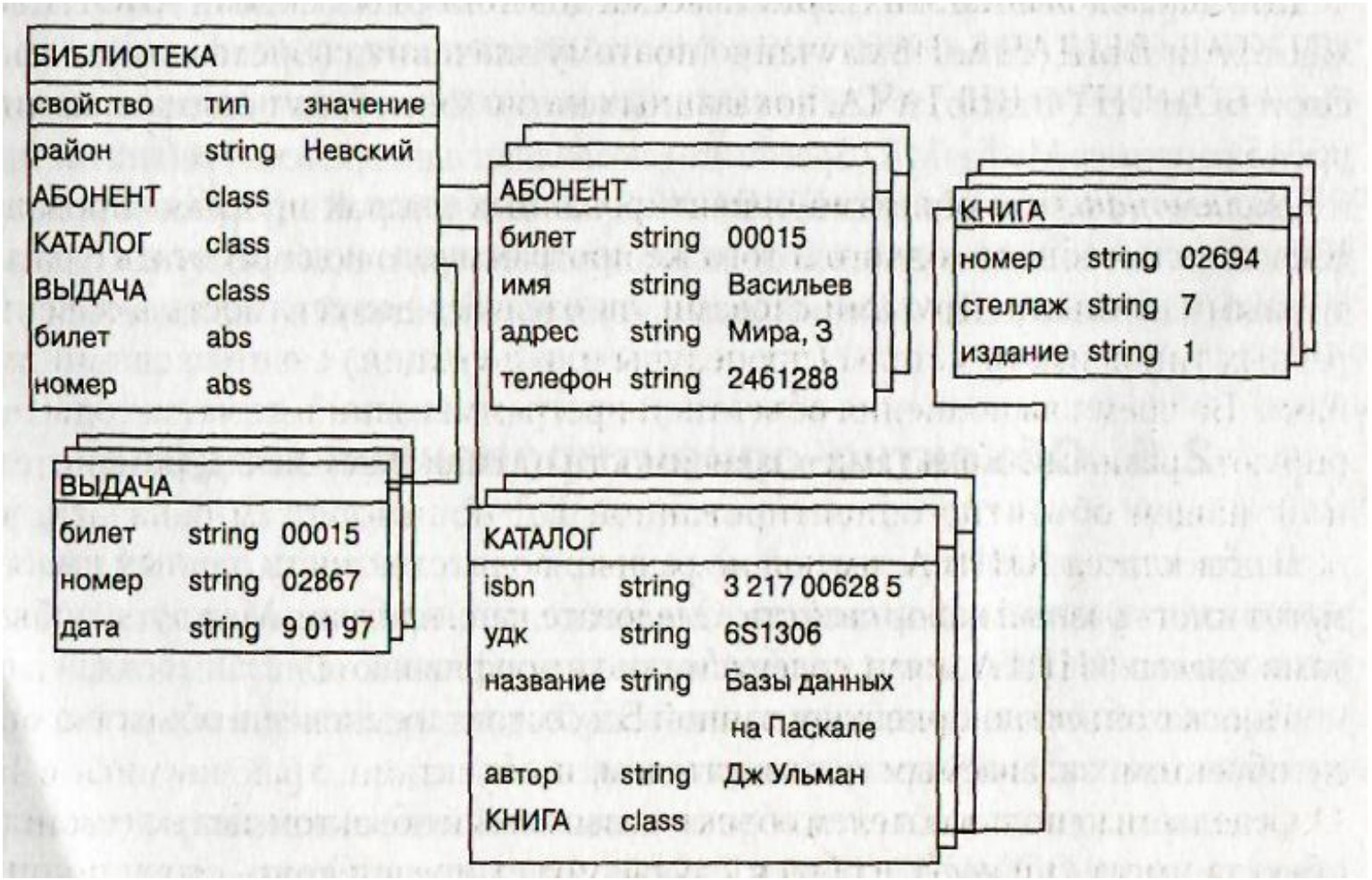
В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы.

Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом:

- строковым – string
- типом, конструируемым пользователем (определяется как class).

Логической структура объектно-ориентированной БД на примере библиотечного дела



Здесь объект типа БИБЛИОТЕКА является родительским для объектов-экземпляров классов АБОНЕНТ, КАТАЛОГ и ВЫДАЧА.

Различные объекты типа КНИГА могут иметь одного или разных родителей.

Объекты типа КНИГА, имеющие одного и того же родителя, должны различаться по крайней мере инвентарным номером (уникален для каждого экземпляра книги), но имеют одинаковые значения свойств *isbn,ydK, название* и *автор.*

Понятия объектно-ориентированной модели БД.

Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено.

Так, если в объект типа КАТАЛОГ добавить свойство, задающее телефон автора книги и имеющее название *телефон,* то мы получим одноименные свойства у объектов АБОНЕНТ и КАТАЛОГ.

Смысл такого свойства будет определяться тем объектом, в который оно инкапсулировано.

Наследование распространяет область видимости свойства на всех потомков объекта.

Так, всем объектам типа КНИГА, являющимся потомками объекта типа КАТАЛОГ, можно приписать свойства объекта-родителя:

- *isbn,*
- *удк,*
- *название*
- *автор.*

Полиморфизм - способность одного и того же программного кода работать с разнотипными данными.

Он означает допустимость в объектах разных типов иметь методы (процедуры или функции) с одинаковыми именами. Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента.

Применительно к нашей объектно-ориентированной БД полиморфизм означает, что объекты класса КНИГА, имеющие разных родителей из класса КАТАЛОГ, могут иметь разный набор свойств. Следовательно, программы работы с объектами класса КНИГА могут содержать полиморфный код.