7. НЕРЕЛЯЦИОННЫЕ МОДЕЛИ ДАННЫХ

В настоящее время в некоторых информационных системах применяются модели данных, которые исторически появились раньше реляционных - инвертированные списки, иерархические и сетевые модели. Их наиболее общие **характеристики:**

1. Понятие модели данных появилось только вместе с реляционным подходом, поэтому абстрактные представления ранних систем появились позже на основе анализа и выявления общих признаков у разных систем.

2. Для этих систем характерен навигационный принцип доступа к БД (в любой момент времени выделяется "текущая запись", с которой начинается выполнение операций с БД).

3. В качестве интерфейса применялись обычные языки программирования, расширенные функциями работы с БД, поэтому задача оптимизации доступа к БД возлагалась на пользователя.

4. Эти системы использовались в течение многих лет, накопили громадные базы данных и некоторые из них используются до сих пор, поэтому актуальна проблема совместного использования ранних и современных систем.

5. Использование более сложных внутризаписных структур данных.

Рассмотрим последнюю особенность более подробно. ***Записью*** обычно называют элемент структуры данных. Различают **записи с линейной структурой и иерархической.** В первом случае запись состоит из атомарных элементов (полей), которые следуют один за другим. Такие записи характерны для реляционных баз данных.

Во втором случае в состав записи могут входить составные единицы информации (СЕИ): векторы (массивы); повторяющиеся группы (когда в записи присутствует несколько экземпляров СЕИ, состоящих из разнотипных элементов), неповторяющиеся составные единицы информации. **Например:** в состав записи СОТРУДНИК могут входить простые элементы (ТабНомер, Фамилия), вектор ИнострЯзыки, повторяющаяся группа ПослужнойСписок (ДатаНазначения, ДатаУвольнения, МестоРаботы, Должность) и неповторяющаяся группа Адрес (Город, улица, дом, квартира, индекс). Сложная структура записей характерна для нереляционных БД.

7.1. Системы, основанные на инвертированных списках

Типичными представителями таких систем являются Datacom/DB компании Applied Data Research, Inc. и Adabas компании Software AG. Они применяются, как правило, на больших ЭВМ фирмы IBM. База данных на инвертированных списках похожа на реляционную БД, то есть также состоит из таблиц отношений, однако есть важные **отличия:**

- допускается сложная структура атрибутов (не атомарность);
- строки таблиц (записи) упорядочены в некоторой последовательности, каждой строке присваивается уникальный номер;
- пользователь может управлять логическим порядком строк в каждой таблице с помощью специального инструмента - индексов.

Некоторые атрибуты могут быть объявлены поисковыми, для каждого из них создается индекс, который содержит упорядоченные значения ключей и указатели на соответствующие записи основной таблицы (инвертированный список). Если таблицу требуется упорядочить по нескольким ключам, то создается столько же индексов. Возможна установка связи между таблицами по тем атрибутам, которые были объявлены поисковыми.

Поддерживаются два класса операций над данными:

- поиск адреса записи по некоторому пути доступа и некоторому условию,
- обновление, удаление или выборка записи с заданным адресом.

Достоинство рассмотренного метода построения базы данных:

- более быстрый поиск по сравнению с РБД (особенно поиск уникальной записи по нескольким условиям),
- возможность хранения элементов данных со сложной структурой.

Недостаток модели - отсутствие строгого математического аппарата, отсутствие средств для описания ограничений целостности БД, отсюда - большая трудоемкость программирования запросов к БД.

ПРИМЕР.

Основная таблица:

Адреса записей	**Имена столбцов (поля)**		
	Номер зачетки	Фамилия	Группа
400	5	Усачев	95е1
500	3	Пикул	95е1
600	2	Селив	96е2
700	6	Арбуз	95е1
800	7	Усачев	98е1
900	1	Пикул	98е1

Индексы и инвертированные списки:

Ключевой атрибут	Индекс	Адреса записей (инвертированные списки)
Фамилия	Арбуз Пикул Селив Усачев	700 500, 900 600 400, 800
Группа	95e1 96e2 98e1	400, 500, 799 600 800, 900

Например, для поиска записей по условию Фамилия ="Пикулин" и Группа = "98e1" достаточно найти пересечение списков с соответствующими индексами: адрес искомой записи = (500, 900) ^ (800, 900) = 900.

7.2. Иерархическая модель

Типичным, наиболее известным и распространенным представителем является Information Management System (IMS) фирмы IBM. Первая версия появилась в 1968 г., до сих пор поддерживается много баз данных, что создает существенные проблемы с переходом как на новую технологию БД, так и на новую технику.

Структура данных

Иерархическая БД состоит из упорядоченного набора структур записей, каждая структура имеет вид дерева. ***Деревом*** называется связный неориентированный граф, не содержащий циклов. Вершины дерева называются *узлами*. Один из узлов, который находится на самом верху иерархии, называют *корнем*. Остальные узлы называются *порожденными (потомками)* и связаны так, что каждый узел имеет исходный, находящийся на более высоком уровне иерархии *(предок)*. Узлы, не имеющие порожденных, называют *листьями*. Все экземпляры узла - потомка, имеющие общего предка, называют *близнецами*.

Рассмотрим пример иерархической БД (рис.7.1).

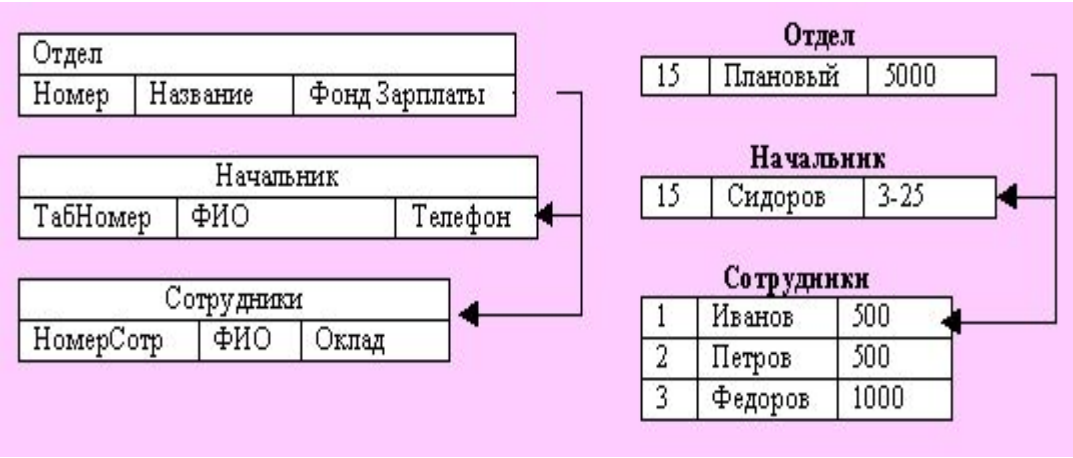

Рис. 7.1. Схема иерархической БД и один экземпляр дерева

Здесь Отдел является предком для узлов Начальник и Сотрудники, а Начальник и Сотрудники - потомки узла Отдел.

Каждому узлу соответствует своя структура записей. Например, в узле Отдел запись содержит поля Номер, Название, Фонд з\п.

Между узлами поддерживаются связи.

Справа показан один экземпляр дерева, содержащий конкретные данные для планового отдела. Для другого отдела строится другой экземпляр дерева с такой же структурой.

Для отображения другой информации о предприятии могут быть построены другие типы деревьев.

Манипулирование данными

Примерами типичных операторов манипулирования иерархически организованными данными могут быть следующие:

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

Ограничения целостности: никакой потомок не может существовать без своего родителя. Однако аналогичное условие целостности по ссылкам между узлами, не входящими в одно дерево, не поддерживается.

Достоинства иерархической модели:

- простота и естественность представления экономических данных;
- минимальный расход памяти по сравнению с другими моделями.

Недостатки иерархической модели:

- сложность отображения связей M : N без увеличения избыточности
- сложность включения информации о новых объектах и удаления устаревших данных;
- доступ к данным возможен только через корень дерева, следовательно большое время поиска данных для некоторых запросов.

Рассмотрим перечисленные недостатки **на примере.**

Построим иерархическую БД о преподавателях, студентах и сдаче экзаменов по разным дисциплинам (рис. 7.2)

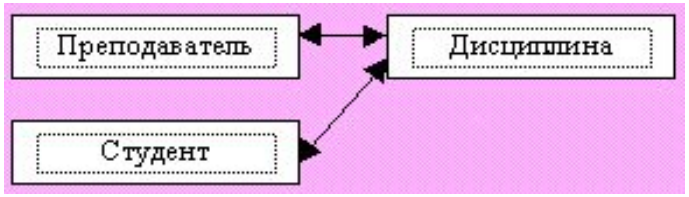

Рис.7.2. Исходная ER-модель

Можно выделить два варианта в соответствии с тем, какой объект принять за корень дерева:

1. Корень дерева - преподаватель.

o Преподаватель (Код_пр, ФИО_пр, Уч. звание, кафедра)

↓

o Студент+дисциплина (N_зач, ФИО_ст, группа, дисциплина, дата_экз, оценка)

Полученная БД удобна для представления данных о преподавателях, но неудобна для представления сведений об успеваемости студентов: если преподаватель X уволился, то необходимо информацию о нем удалить из БД, при этом удаляются и порожденные записи, следовательно теряется информация об успеваемости студентов.

2. Корень дерева - студент

- Студент (N_зач , ФИО_ст, группа)

- Дисциплина+преподаватель (Код_пр, ФИО_пр, звание, кафедра, дисциплина, дата_экз, оценка)

Построенная модель позволяет быстро находить информацию о студентах (например, о сдаче экзаменов по разным дисциплинам), но не удобна для того, чтобы узнать информацию о преподавателях. Например, если преподаватель не принимает экзаменов или ушел в длительный отпуск, то информацию о нем нельзя включить в БД. Кроме того, информация о преподавателе X дублируется столько раз, сколько у него студентов.

Переход от инфологической модели к иерархической БД

Методика перехода к иерархической БД может быть различной. Один из способов описан в учебнике [7].

Рассмотрим более простую методику, идея которой состоит в том, чтобы построить дерево на графе инфологической модели. Таких деревьев может быть несколько. При этом корень дерева - узел, в который не входят дуги (если такого узла нет, то следует ввести его). Часть связей нужно удалить и заменить их соответствующими атрибутами объектов.

Список атрибутов каждого узла определяется так:

- перечисляются все атрибуты понятия кроме тех, которые принадлежат вышестоящим узлам,
- добавляются атрибуты для моделирования связей, явно не отраженных в структуре дерева.

Дублирование атрибутов в узлах позволяет отобразить реальные связи, но увеличивает объем БД.

Пример. Рассмотрим ER-модель факультета (на рис. 7.3 показаны связи типа "является частью" и процесс сдачи экзаменов и чтения лекций).

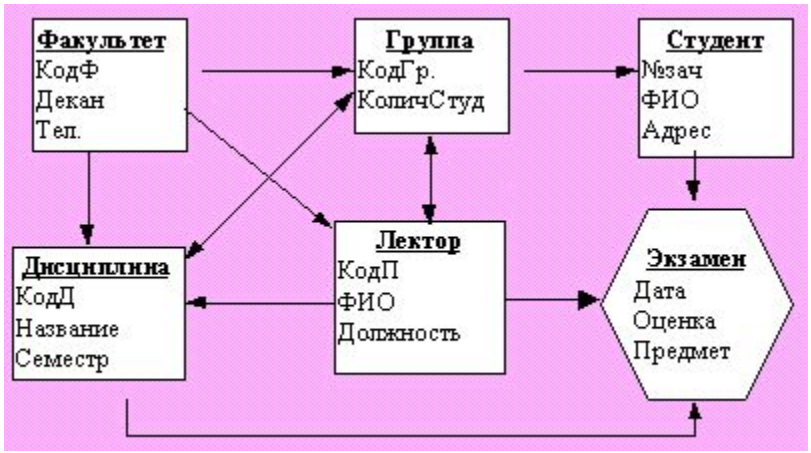

Рис. 7.3. ER-модель факультета

Можно построить несколько вариантов иерархической БД, например - рис.7.4.

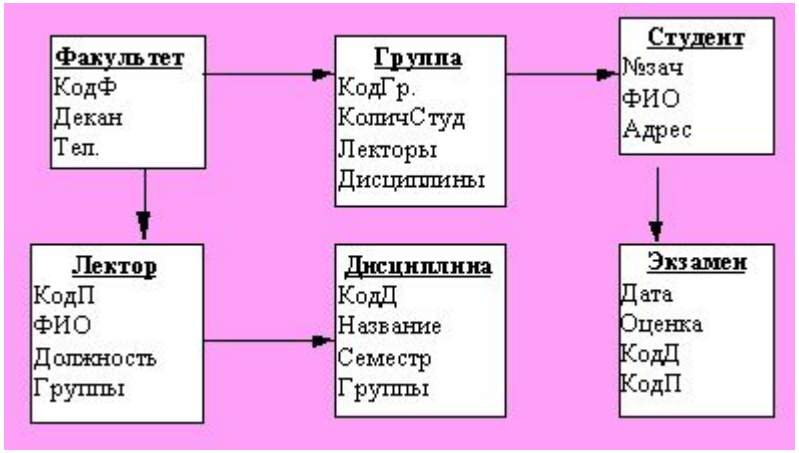

Рис. 7.4. Вариант иерархической модели

В качестве корня выбран узел “Факультет”, так как все дуги выходят из него. Обратите внимание, как изменился состав атрибутов в узлах дерева: добавлены внешние ключи вместо разорванных связей типа 1:М (например - в узле Экзамен). Вместо удаленных связей М : М добавлены векторные атрибуты "Лекторы" и "Дисциплины" в узле Группа; атрибут "Группы" в узле Лектор, атрибут "Группы" в узле Дисциплина. Налицо дублирование информации, которое можно уменьшить, но тогда усложнится обработка запросов к БД.

Если конкретная СУБД не допускает сложных внутризаписных структур, то придется вводить дополнительные узлы дерева либо пренебречь некоторыми связями в концептуальной модели, то есть потерять часть информации о предметной области.

7.3. Сетевая модель данных

Отличие сетевой модели от иерархической заключается в том, что в сетевой структуре любой элемент данных может быть связан с любым другим, то есть иерархическая модель является разновидностью сетевой.

Различают простую и сложную сетевую структуру. В ***простой сетевой структуре*** между исходным и порожденными узлами реализуется связь 1:М. ***Сложной сетевой структурой*** называют такую схему, в которой присутствует хотя бы одна связь N : M.

В настоящее время большинство СУБД поддерживают только простые сетевые структуры. Такие системы называют СУБД с равноправными (однотипными) файлами. Типичным представителем является Integrated Database Management System (IDMS) компании Cullinet Software Inc., предназначенная для использования на машинах основного класса фирмы IBM под управлением большинства операционных систем. Архитектура системы основана на предложениях Комитета по языкам программирования Conference on Data Systems Languages (CODASYL). В дальнейшем мы будем пользоваться терминологией, принятой в КОДАСИЛ.

Основная конструкция сетевой модели данных КОДАСИЛ - набор (рис.7.5).

Набор - это поименованное двухуровневое дерево, которое реализует связь между записями двух типов: владельцем набора и членом набора. Разрешаются только связи 1 : M или M : 1 , но связи M : N в явном виде не поддерживаются.

Рис. 7.5. Графическое представление набора

Для каждого набора, устанавливающего связь между предком P и потомком C должны выполняться два условия:

1. Каждый экземпляр P является предком только в одном экземпляре набора;
2. Каждый экземпляр C является потомком не более чем в одном экземпляре набора.

Основные свойства набора:

- набор имеет имя,
- в каждом наборе только один владелец,
- в каждом наборе 0, 1 или несколько членов,
- набор существует, только если существует запись - владелец.
- поскольку набор имеет имя, то два экземпляра записи могут быть в разных наборах,
- экземпляр записи может входить только в один экземпляр набора данного типа.
- в общем случае каждый набор - это вход в БД;

С помощью наборов можно строить многоуровневые деревья и простые сетевые структуры. Так как роль записи жестко не фиксируется, то в одном наборе запись (файл) может быть членом, а в другом владельцем, поэтому такая модель и называется моделью с равноправными файлами.

В сетевых моделях реальных СУБД запись может иметь любую структуру, **например,** простую линейную, как в реляционной БД, либо более сложную, включая массивы, группы, повторяющиеся группы. Совокупность однотипных записей образует ***файл***, а совокупность файлов и наборов, описанных в одной схеме, образует ***сетевую БД***. Допускаются изолированные, не связанные с другими, файлы.

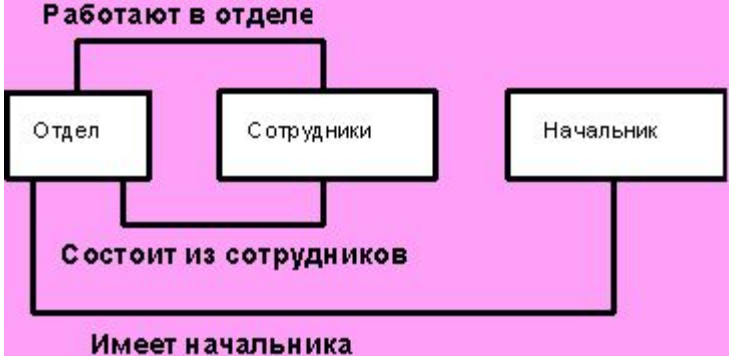

Рис. 7.6. Пример сетевой схемы БД

Типы наборов

В зависимости от способа физического хранения различают одночленные, многочленные и сингулярные наборы. ***Одночленный набор*** включает только один файл - член (рис.7.7а). ***Многочленные наборы*** состоят из трех и более файлов (рис. 7.7б), ***сингулярный набор*** - это особый набор, в котором владельцем является система. В каждом сингулярном наборе всего один экземпляр. Сингулярные наборы чаще всего применяются, чтобы получить доступ ко всем записям файла - владельца, а также чтобы объединить записи, не имеющие владельца.

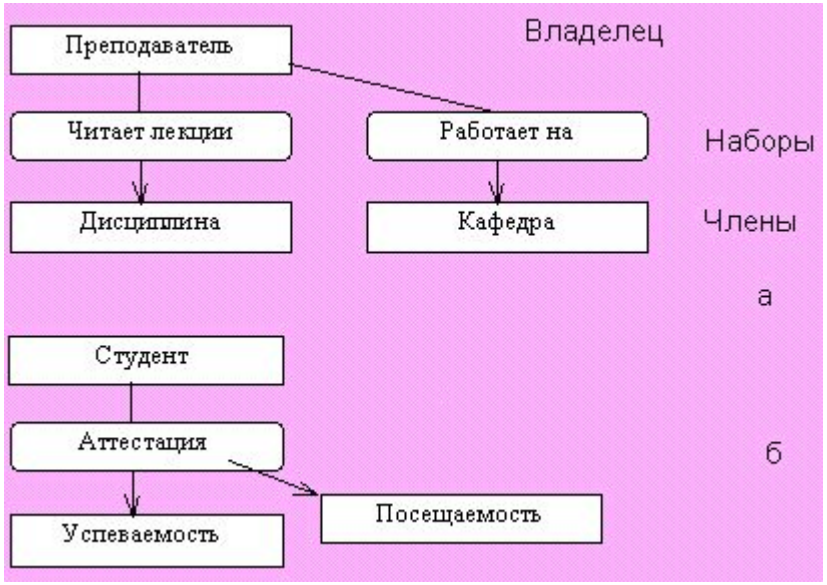

Рис. 7.7. Примеры одночленного (а) и многочленного (б) наборов

Отображение инфологической модели в сетевую модель данных

Процесс датологического проектирования сетевой БД отличается от такового для реляционных БД тем, что необходимо явно задать наборы как модели связей между объектами. Если структура каждой записи файла линейная (как в реляционной БД), то на этом отличия и кончаются. Если структура записей сложная, то связи между объектами можно отобразить двумя путями: как наборы (связи между записями) и как связи внутри записей файла.

Рис. 7.8. Пример выбора сложных внутризаписных структур

Внутризаписные сложные структуры рекомендуется применять в следующих случаях (рис. 7.8):

1. Если объект (понятие) имеет множественные свойства;
2. Если объект имеет составные свойства;
3. Если связи между объектами 1:M, где M - невелико и объект не связан с другими объектами.

В остальных случаях каждое понятие ER- модели представляется отдельным файлом.

Контрольные вопросы