2 ДАТАЛОГИЧЕСКИЕ МОДЕЛИ ДАННЫХ

Прежде чем перейти к детальному и последовательному изучению реляционных систем БД, остановимся коротко на ранних СУБД. В этом есть смысл по трем причинам: во-первых, эти системы исторически предшествовали реляционным, и для правильного понимания причин повсеместного перехода к реляционным системам нужно знать хотя бы что-нибудь про их предшественников; во-вторых, внутренняя организация реляционных систем во многом основана на использовании методов ранних систем; в-третьих, некоторое знание в области ранних систем будет полезно для понимания путей развития постреляционных СУБД.

Мы ограничиваемся рассмотрением только общих подходов к организации трех типов ранних систем, а именно: систем, основанных на инвертированных списках, иерархических и сетевых систем управления базами данных. Начнем с некоторых наиболее общих характеристик ранних систем.

Эти системы активно использовались в течение многих лет, дольше, чем используется какая-либо из реляционных СУБД. На самом деле некоторые из ранних систем используются даже в наше время, накоплены громадные базы данных, и одной из актуальных проблем информационных систем является использование этих систем совместно с современными системами.

Все ранние системы не основывались на каких-либо абстрактных моделях. Понятие модели данных фактически вошло в обиход специалистов в области БД только вместе с реляционным подходом. Абстрактные представления ранних систем появились позже на основе анализа и выявления общих признаков у различных конкретных систем.

В ранних системах доступ к БД производился на уровне записей. Пользователи этих систем осуществляли явную навигацию в БД, используя языки программирования, расширенные функциями СУБД. Интерактивный доступ к БД поддерживался только путем создания соответствующих прикладных программ с собственным интерфейсом.

Навигационная природа ранних систем и доступ к данным на уровне записей заставляли пользователя самого производить всю оптимизацию доступа к БД, без какой-либо поддержки системы.

2.1 Системы на инвертированных списках

К числу наиболее известных и типичных представителей таких систем относятся Datacom/DB компании Applied Data Research, Inc. (ADR), ориентированная на использование на машинах основного класса фирмы IBM, и Adabas компании Software AG.

Организация доступа к данным на основе инвертированных списков используется практически во всех современных реляционных СУБД, но в этих системах пользователи не имеют непосредственного доступа к

инвертированным спискам (индексам). Кстати, когда мы будем рассматривать внутренние интерфейсы реляционных СУБД, вы увидите, что они очень близки к пользовательским интерфейсам систем, основанных на инвертированных списках.

2.1.1 Структуры данных

База данных, организованная с помощью инвертированных списков, похожа на реляционную БД, но с тем отличием, что хранимые таблицы и пути доступа к ним видны пользователям. При этом:

- Строки таблиц упорядочены системой в некоторой физической последовательности.
- Физическая упорядоченность строк всех таблиц может определяться и для всей БД (так делается, например, в Datacom/DB).

Для каждой таблицы можно определить произвольное число ключей поиска, для которых строятся индексы. Эти индексы автоматически поддерживаются системой, но явно видны пользователям.

2.1.2 Манипулирование данными

Поддерживаются два класса операторов:

а) Операторы, устанавливающие адрес записи, среди которых:

- прямые поисковые операторы (например, найти первую запись таблицы по некоторому пути доступа);
- операторы, находящие запись в терминах относительной позиции от предыдущей записи по некоторому пути доступа.

б) Операторы над адресуемыми записями

Типичный набор операторов:

LOCATE FIRST – найти первую запись таблицы T в физическом порядке; возвращает адрес записи;

LOCATE FIRST WITH SEARCH KEY EQUAL – найти первую запись таблицы T с заданным значением ключа поиска K; возвращает адрес записи;

LOCATE NEXT – найти первую запись, следующую за записью с заданным адресом в заданном пути доступа; возвращает адрес записи;

LOCATE NEXT WITH SEARCH KEY EQUAL – найти следующую запись таблицы T в порядке пути поиска с заданным значением K; должно быть соответствие между используемым способом сканирования и ключом K; возвращает адрес записи;

LOCATE FIRST WITH SEARCH KEY GREATER – найти первую запись таблицы T в порядке ключа поиска K со значением ключевого поля, большим заданного значения K; возвращает адрес записи;

RETRIVE – выбрать запись с указанным адресом;

UPDATE – обновить запись с указанным адресом;

DELETE – удалить запись с указанным адресом;

STORE – включить запись в указанную таблицу; операция генерирует адрес записи.

Общие правила определения целостности БД отсутствуют. В некоторых системах поддерживаются ограничения уникальности значений некоторых полей, но в основном все возлагается на прикладную программу.

2.2 Иерархические модели

Типичным представителем иерархической модели данных (наиболее известным и распространенным) 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.

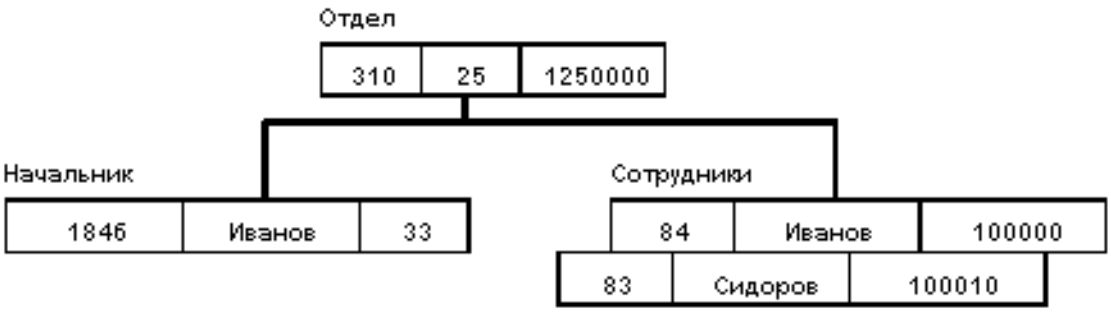
Иерархическая БД состоит из упорядоченного набора деревьев; более точно, из упорядоченного набора нескольких экземпляров одного типа дерева.

Тип дерева состоит из одного «корневого» типа записи и упорядоченного набора из нуля или более типов поддеревьев (каждое из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор типов записи.

Пример типа дерева (схемы иерархической БД):

Здесь тип записи Отдел является предком для типов Начальник и Сотрудники, а Начальник и Сотрудники – потомки Отдел. Между типами записи поддерживаются связи.

База данных с такой схемой могла бы выглядеть следующим образом (мы показываем один экземпляр дерева):



Все экземпляры данного типа потомка с общим экземпляром типа предка называются *близнецами*. Для БД определен полный порядок обхода – сверху-вниз, слева-направо.

В IMS использовалась оригинальная и нестандартная терминология: «сегмент» вместо «запись», а под «записью БД» понималось все дерево сегментов.

Примерами типичных операторов манипулирования иерархически организованными данными могут быть следующие:

- найти указанное дерево БД (например, отдел 310);
- 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.

Автоматически поддерживается целостность ссылок между предками и потомками. *Основное правило: никакой потомок не может существовать без своего родителя.* Заметим, что аналогичное поддержание целостности по ссылкам между записями, не входящими в одну иерархию, не поддерживается (примером такой «внешней» ссылки может быть содержимое поля Каф_Номер в экземпляре типа записи Куратор).

2.3 Сетевые модели

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 Data Base Task Group (DBTG) Комитета по языкам программирования Conference on Data Systems Languages (CODASYL), организации, ответственной за определение языка программирования Кобол. Отчет DBTG был опубликован в 1971 г., а в 70-х годах появилось несколько систем, среди которых IDMS.

Сетевой подход к организации данных является расширением иерархического. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков.

Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно, из набора экземпляров каждого типа из заданного в схеме БД набора типов записи и набора экземпляров каждого типа из заданного набора типов связи.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора экземпляров типа записи потомка. Для данного типа

связи L с типом записи предка P и типом записи потомка C должны выполняться следующие два условия:

- каждый экземпляр типа P является предком только в одном экземпляре L;
- каждый экземпляр C является потомком не более, чем в одном экземпляре L.

На формирование типов связи не накладываются особые ограничения; возможны, например, следующие ситуации:

- тип записи потомка в одном типе связи L1 может быть типом записи предка в другом типе связи L2 (как в иерархии).
- данный тип записи P может быть типом записи предка в любом числе типов связи.
- данный тип записи P может быть типом записи потомка в любом числе типов связи.

Может существовать любое число типов связи с одним и тем же типом записи предка и одним и тем же типом записи потомка. Если L1 и L2 – два типа связи с одним и тем же типом записи предка P и одним и тем же типом записи потомка C, то правила, по которым образуется родство, в разных связях могут различаться.

- Типы записи X и Y могут быть предком и потомком в одной связи и потомком и предком – в другой.
- Предок и потомок могут быть одного типа записи.

Примерный набор операций может быть следующим:

- найти конкретную запись в наборе однотипных записей (инженера Сидорова);
- перейти от предка к первому потомку по некоторой связи (к первому сотруднику отдела 310);
- перейти к следующему потомку в некоторой связи (от Сидорова к Иванову);
- перейти от потомка к предку по некоторой связи (найти отдел Сидорова);
- создать новую запись;
- уничтожить запись;
- модифицировать запись;
- включить в связь;
- исключить из связи;
- переставить в другую связь и т.д.

Простой пример сетевой схемы БД:

2.3.1 Достоинства и недостатки

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;
– фактически необходимы знания о физической организации;
– прикладные системы зависят от этой организации;
– их логика перегружена деталями организации доступа к БД.