Теоретико-графовые модели данных

В зависимости от типа графа выделяют **иерархическую** или **сетевую** модели. Исторически эти модели появились раньше, и в настоящий момент они используются реже, чем реляционная модель данных.

Иерархическая модель данных является наиболее простой и появилась первой среди всех даталогических моделей, именно эту модель поддерживает первая из зарегистрированных промышленных СУБД IMS фирмы IBM. Основными информационными единицами в иерархической модели являются: *база данных (БД), сегмент и поле.* **Поле** данных - минимальная, неделимая единица данных, доступная пользователю с помощью СУБД. *Сегмент* в терминологии Американской Ассоциации по базам данных DBTG (Data Base Task Group) называется *записью,* при этом в рамках иерархической модели определяются два понятия: *тип сегмента* или *тип записи* и *экземпляр сегмента* или *экземпляр записи.*

Тип сегмента — это поименованная совокупность типов элементов данных (однородных записей), в него входящих. *Экземпляр сегмента* образуется из конкретных значений полей или элементов данных, в него входящих. Для возможности различия отдельных записей в данном наборе каждый тип сегмента должен иметь ключ или набор ключевых атрибутов

Ключом называется набор элементов данных, однозначно идентифицирующих экземпляр сегмента.
В иерархической модели сегменты объединяются в ориентированный древовидный граф. При этом направленные ребра графа отражают иерархические связи между сегментами: каждому экземпляру сегмента, стоящему выше по иерархии и соединенному с данным типом сегмента, соответствует несколько (множество) экземпляров данного (подчиненного) типа сегмента.
Тип сегмента, находящийся на более высоком уровне иерархии, называется логически исходным по отношению к типам сегментов, соединенным с данными направленными иерархическими ребрами, которые в свою очередь называются логически подчиненными по отношению к этому типу сегмента. Иногда исходные сегменты называют сегментами-предками, а подчиненные сегменты называют сегментами-потомками.

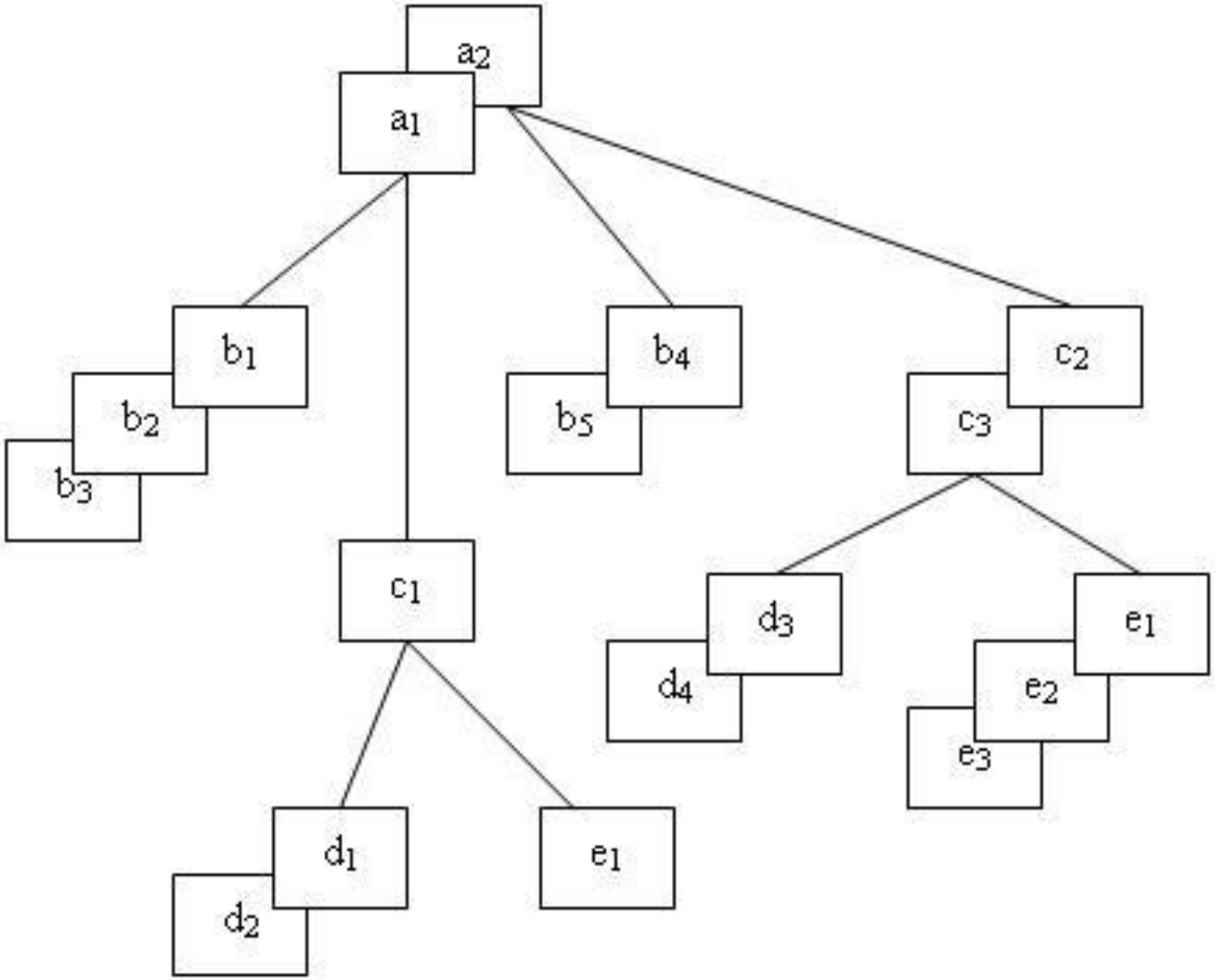
На концептуальном уровне определяется понятие схемы БД в терминологии иерархической модели.

Схема иерархической БД представляет собой совокупность отдельных деревьев, каждое дерево в рамках модели называется *физической базой данных.* Каждая физическая БД удовлетворяет следующим иерархическим ограничениям:

- в каждой физической БД существует один корневой сегмент, то есть сегмент, у которого нет логически исходного (родительского) типа сегмента;
- каждый логически исходный сегмент может быть связан с произвольным числом логически подчиненных сегментов;
- каждый логически подчиненный сегмент может быть связан только с одним логически исходным (родительским) сегментом.

Различие между сегментом и типом сегмента — такое же, как между типом переменной и самой переменной: сегмент является экземпляром типа сегмента.

Между экземплярами сегментов также существуют иерархические связи.



Каждый тип сегмента может иметь множество соответствующих ему экземпляров. Между экземплярами сегментов также существуют иерархические связи.

Рис. Пример двух экземпляров данного дерева

Язык описания данных иерархической модели

Языковые средства описания данных (DDL, Data Definition Language) и средства манипулирования данными (DML, Data Manipulation Language).

Каждая физическая база описывается набором операторов, определяющих ее логическую структуру, и структуру хранения БД.

Описание начинается с оператора DBD (Data Base Definition):

DBD Name = < имя БД>, ACCESS = < способ доступа>

Определено 5 способов доступа, определяющих организацию взаимосвязи физических записей :

HSAM — hierarchical sequential access method (иерархически последовательный метод),

HISAM — hierarchical index sequential access method (иерархически индексно-последовательный метод),

EDAM — hierarchical direct access method (иерархически прямой метод),

HID AM — hierarchical index direct access method (иерархически индексно-прямой метод),

INDEX — индексный метод.

Описание наборов данных, предназначенных для хранения БД:
DATA SET D01 = < имя оператора, определяющего хранимый набор данных>.
DEVICE =< устройство хранения БД>,
[OVFLW = < имя области переполнения>]
Так как физические записи имеют разную длину, то при модификации данных запись может увеличиться и превысить исходную длину записи. В этом случае может понадобиться дополнительное пространство хранения для размещения дополнительных данных- область переполнения.
После описания всей физической БД идет описание типов сегментов, ее составляющих, в соответствии с иерархией, которое всегда начинается с описания корневого сегмента. Общая схема описания типа сегмента такова:
SEGM NAME = < имя сегмента>. BYTES =< размер в байтах>.
FREQ = <средняя частота реализаций сегмента под одним исходным>
PARENT = <имя родительского сегмента>

Параметр FREQ определяет среднее количество экземпляров данного сегмента, связанных с одним экземпляром родительского сегмента. Для корневого сегмента это число возможных экземпляров корневого сегмента.
Для корневого сегмента параметр PARENT равен 0 (нулю). Далее для каждого сегмента дается описание полей:
FIELD NAME = {(<имя поля> [. SEQ].{U M}) | <имя поля> }.
START = < номер байта, с которого начинается значения поля >,
BYTES = <размер поля в байтах>,
TYPE = {X | P | C}
Признак SEQ — задается для ключевого поля, если экземпляры данного сегмента физически упорядочены в соответствии со значениями данного поля.
Параметр U задается, если значения ключевого поля уникальны для всех экземпляров данного сегмента, M — в противном случае.
Если поле является ключевым, то его описание задается в круглых скобках, в противном случае имя поля задается без скобок.
Параметр TYPE определяет тип данных.

Для ранних иерархических моделей были определены только три типа данных: X — шестнадцатеричный, P —упакованный десятичный, C — символьный.

Заканчивается описание схемы вызовом процедуры генерации:

DBDGEN — указывает на конец последовательности управляющих операторов описания БД;

FINISH — устанавливает ненулевой код завершения при обнаружении ошибки;

END — конец.

В системе может быть несколько физических БД (ФБД), но каждая из них описывается отдельно своим DBD и ей присваивается уникальное имя.

Каждая ФБД содержит только один корневой сегмент.

Совокупность ФБД образует **концептуальную модель данных**.

Внешние модели

Внешняя модель представляет собой совокупность поддеревьев для физических баз данных, с которыми работает данный пользователь. Представление внешней модели называется логической базой данных и определяется совокупностью блоков связи данного приложения с физическими БД, входящими в концептуальную схему БД. Блок связи — *PCB, program communication bloc* — описывает связь с одной физической БД по следующим правилам:

DBD NAME = < имя логической БД (подсхемы)> , ACCESS = LOGICAL

DATA SET = LOGICAL.

SEGM NAME = <имя сегмента в подсхеме>,

PARENT =<имя родительского сегмента в подсхеме>,

SOURSE =(Имя соответствующего сегмента ФБД. имя ФБД)

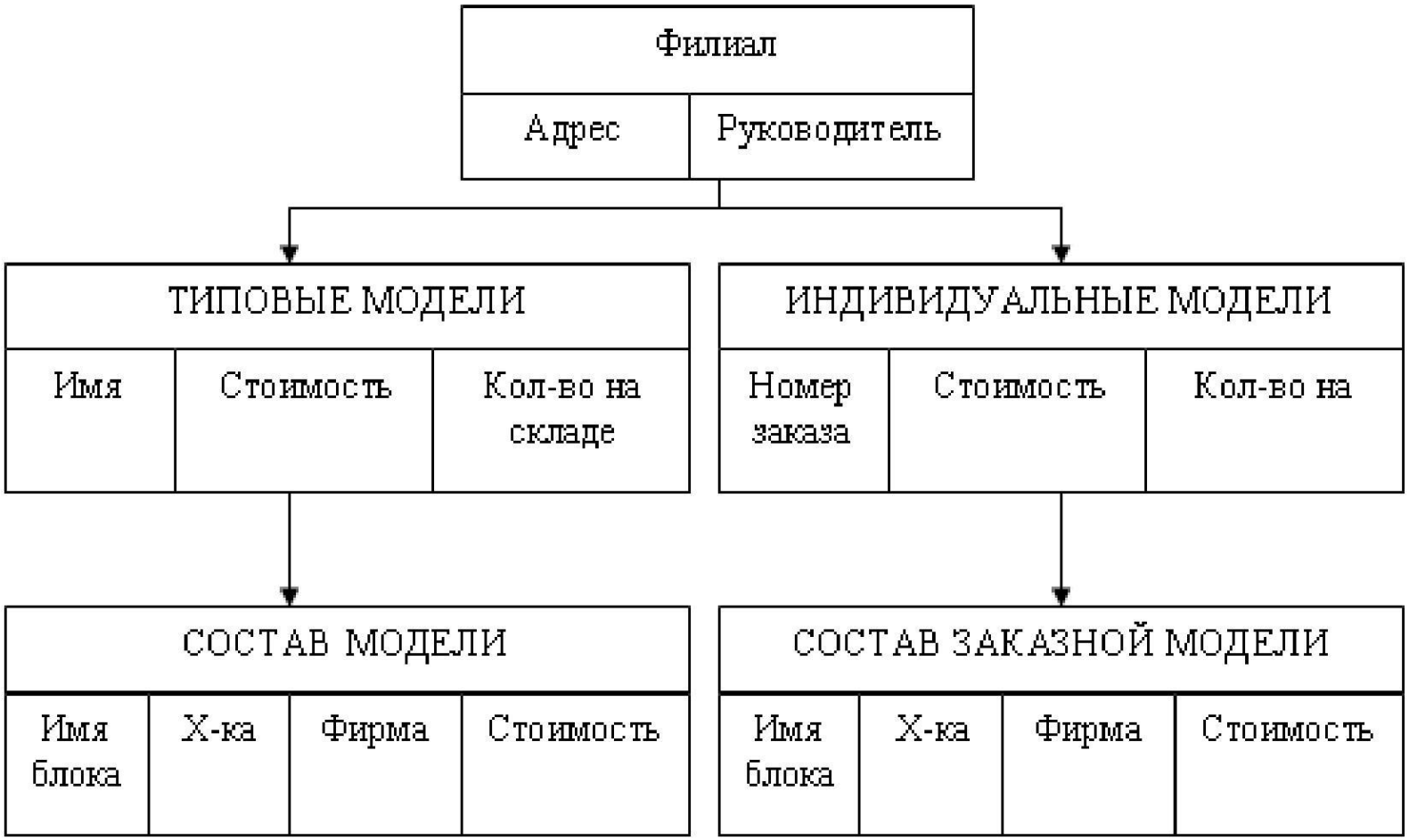
DBDGEN

FINISH

END

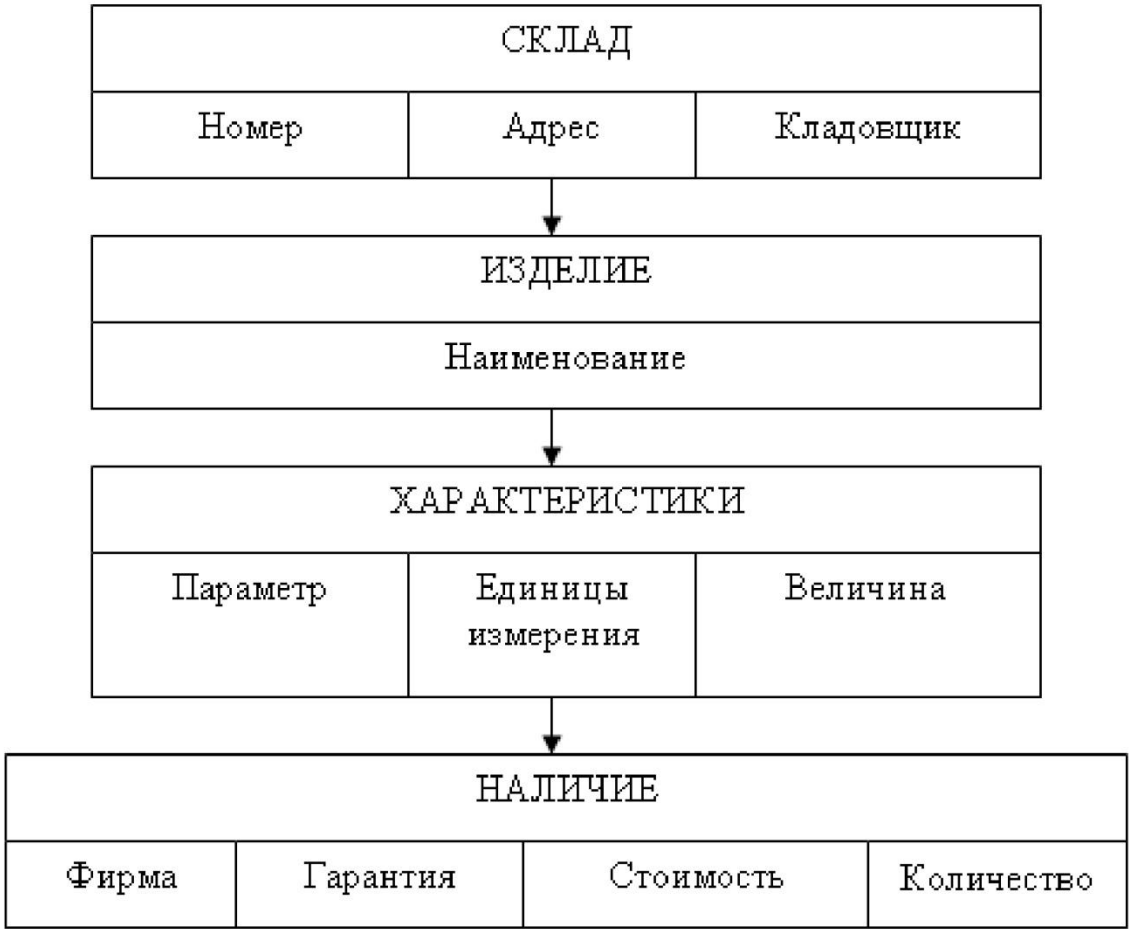
Совокупность блоков PCB образует полное внешнее представление

Пример



Указать модель, проверить наличие, полное описание индивидуальной модели.

Для индивидуальной модели нужна информация со склада



Язык манипулирования данными в иерархических базах данных
Для доступа к базе данных у пользователя должна быть сформирована специальная среда окружения, поддерживающая в явном виде имеющиеся навигационные операции.
Для этого в ней должны храниться:
шаблоны всех записей логических баз данных, доступных пользователю;
указатели на текущий экземпляр сегмента данного типа — для всех типов сегментов.
ЯМД поддерживает в явном виде навигационные операции. Эти операции связаны с перемещением указателя, который определяет текущий экземпляр конкретного сегмента.
Все операторы в ЯМД можно разделить на 3 группы.
Первую группу составляют **операторы поиска данных**.
Синтаксис:
GET UNIQUE <имя сегмента> WHERE <список поиска>;
список поиска состоит из последовательности условий вида:
<имя сегмента>.<имя поля>ОС <constant или имя другого поля данного сегмента или имя переменной>;

Пример:

Найти типовую модель стоимостью не более $600, которая существует не менее чем в 10 экземплярах.

GET UNIQUE ТИПОВЫЕ МОДЕЛИ

WHERE Типовые модели.Стоимость <= $600

AND Типовые модели,Количество на складе >= 10

Данная команда всегда ищет с начала БД и останавливается, найдя первый экземпляр сегмента, удовлетворяющий условиям поиска.

Синтаксис:

GET NEXT <имя сегмента> WHERE <список аргументов поиска>

Получить перечень винчестеров со склада1, не менее 10 с объемом 10 Гб

```
GET UNIQUE СКЛАД WHERE Склад.Номер = 1
GET  NEXT ИЗДЕЛИЕ WITHIN PARENT WHERE Изделие.
    Наименование = «Винчестер»
GET  NEXT ХАРАКТЕРИСТИКИ WITHIN PARENT WHERE
ХАРАКТЕРИСТИКИ.Параметр =10  AND
ХАРАКТЕРИСТИКИ.Единицы измерения =Гб AND
ХАРАКТЕРИСТИКИ.Величина > 10
While Not Fail DO
GET  NEXT  WITHIN PARENT
END
```

Операторы поиска данных с возможностью модификации

1.Найти и удержать единственный экземпляр сегмента. Эта операция подобна первой операции поиска GET UNIQUE, единственным отличием этой операции является то, что после выполнения этой операции пал найденным экземпляром сегмента допустимы операции модификации (изменения) данных.

Синтаксис:

GET HOLD UNIQUE <имя сегмента> WHERE <список поиска>

2.Найти и удержать следующий с теми же условиями поиска. Аналогично операции 4 эта операция дублирует вторую операции поиска GET NEXT с возможностью выполнения последующей модификации данных.

Синтаксис:

GET HOLD NEXT [WHERE <дополнительные условия>]

3.Получить и удержать следующий для того же родителя. Эта операция является аналогом операции поиска 3, но разрешает выполнение операций модификации данных после себя.

Синтаксис:

GET HOLD NEXT WITHIN PARENT [where <дополн условия>]

Операторы модификации данных

Удалить :DELETE
Обновить **:** UPDATE
Ввести новый экземпляр сегмента: INSERT <имя сегмента>

Способ доступа, который применяется в данной модели, связан с последовательным перемещением от одного экземпляра сегмента к другому. Такой способ напоминает движение летательного аппарата или корабля по заданным координатам и называется навигационным.

Сетевая модель данных

Базовыми объектами модели являются:

- элемент данных;
- агрегат данных;
- запись;
- набор данных.

Элемент данных —это минимальная информационная единица, доступная пользователю с использованием СУБД.

Агрегат данных соответствует следующему уровню обобщения в модели. В модели определены агрегаты двух типов:
агрегат типа *вектор* и агрегат типа *повторяющаяся группа.*

Агрегат данных имеет имя, и в системе допустимо обращение к агрегату по имени. Агрегат типа **вектор** соответствует линейному набору элементов данных.

Записью называется совокупность агрегатов или элементов данных, моделирующая некоторый класс объектов реального мира. Понятие записи соответствует понятию «сегмент» в иерархической модели. Для записи вводятся понятия **типа** записи и **экземпляра** записи.

Пример

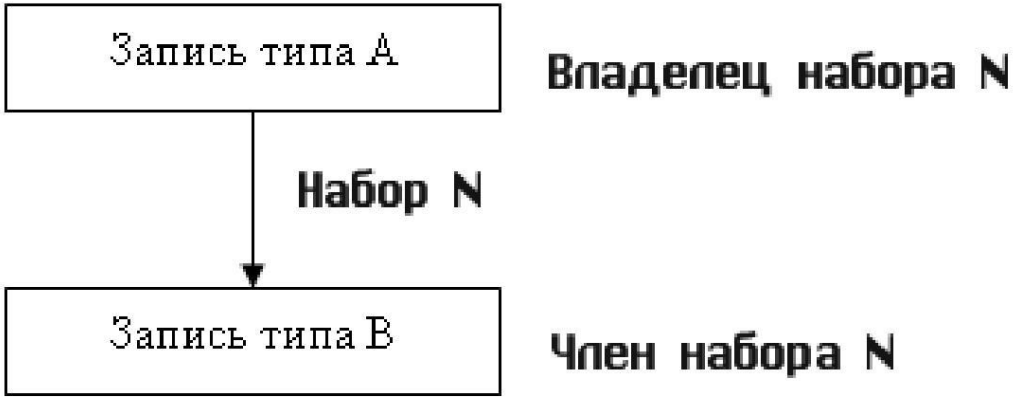
Агрегат Адрес

Адрес			
Город	Улица	дом	квартира

Агрегат типа повторяющаяся группа соответствует совокупности векторов данных.

Следующим базовым понятием в сетевой модели является понятие «Набор». Набором называется двухуровневый граф, связывающий отношением «один-ко-многим» два типа записи.
Родительский тип записи в данном наборе называется владельцем набора, а дочерний тип записи — членом того же набора.
Для любых двух типов записей может быть задано любое количество наборов, которые их связывают. Фактически наличие подобных возможностей позволяет промоделировать отношение «многие-ко-многим» между двумя объектами реального мира, что отличает от предыдущей модели.
В рамках набора возможен последовательный просмотр экземпляров членов набора, связанных с одним экземпляром владельца набора.
Между двумя типами записей может быть определено любое количество наборов: например, можно построить два взаимосвязанных набора. Существенным ограничением набора является то, что один и тот же тип записи не может быть одновременно владельцем и членом набора.

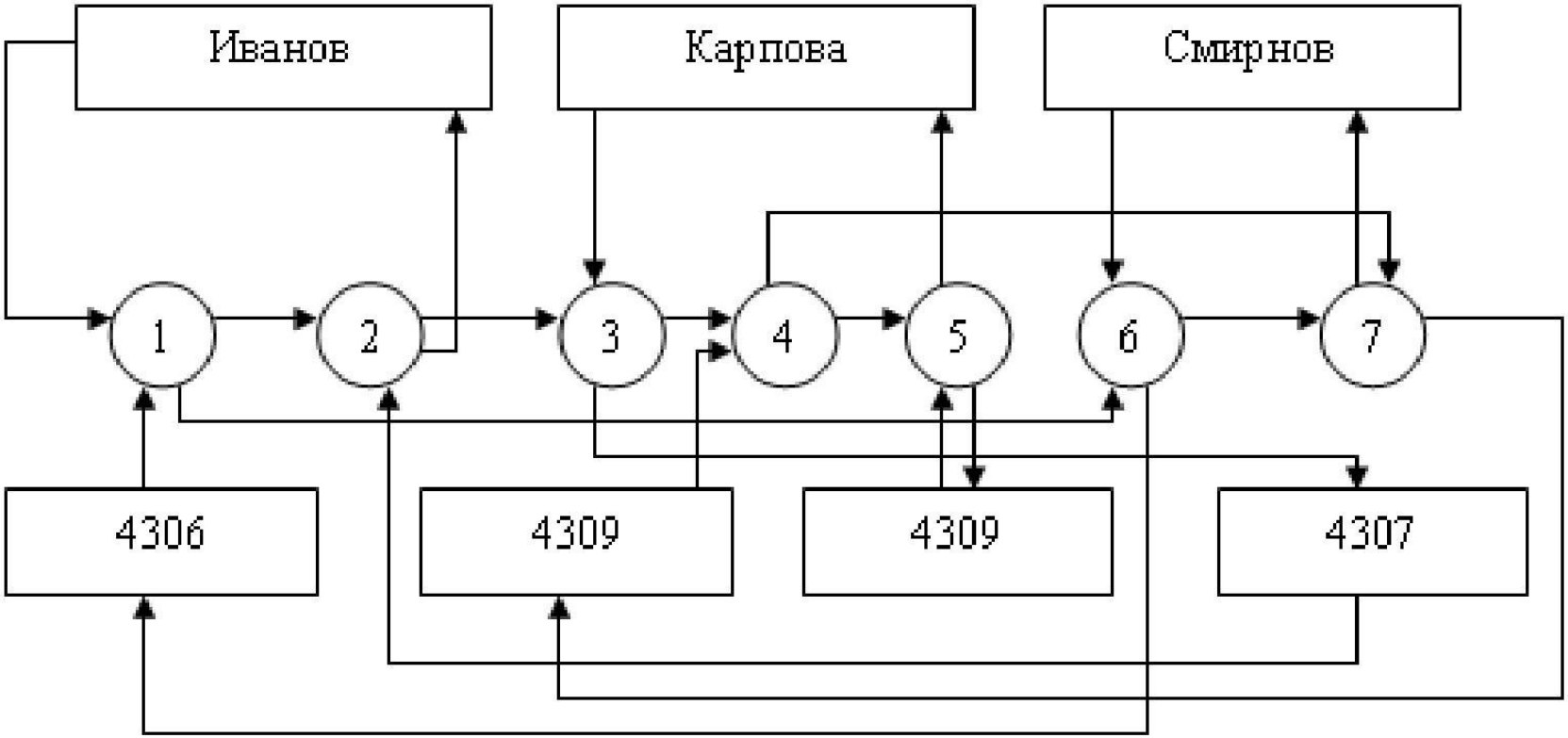
Набор



Два набора

Преподаватель	Группа	День недели	№ пары	Аудитория	Дисциплина
Иванов	4306	Понедельник	1	22-13	КИД
Иванов	4307	Понедельник	2	22-13	КИД
Карпова	4307	Вторник	2	22-14	БЗ и ЭС
Карпова	4309	Вторник	4	22-14	БЗ и ЭС
Карпова	84305	Вторник	1	22-14	БД
Смирнов	4306	Вторник	3	23-07	ГВП
Смирнов	4309	Вторник	4	23-07	ГВП

Пример взаимосвязи экземпляров двух наборов



Среди всех наборов выделяют специальный тип набора, называемый «Сингулярным набором», владельцем которого формально определена вся система. Сингулярный набор изображается в виде входящей стрелки, которая имеет собственно имя набора и имя члена набора, но у которой не определен тип записи «Владелец набора». Например, сингулярный набор M.

Сингулярные наборы обеспечивают доступ к экземплярам отдельных типов данных, т.е. если нужен произвольный доступ к записи, нужно ввести сингулярный набор.

В общем случае сетевая база данных представляет совокупность взаимосвязанных наборов, образующих на концептуальном уровне граф

Язык описания данных в сетевой модели

- Описание базы данных – области размещения
- Описания записей – элементов и агрегатов (каждого в отдельности)
- Описания наборов (каждого в отдельности)

SCHEMA IS <Имя БД>

AREA NAME IS <Имя физической области>

RECORD NAME IS <Имя записи (уникальное)>

Для каждой записи определяется способ размещения экземпляров записи данного типа:

LOCATION MODE IS{DIRECT (напрямую)

CALC <Имя программы> USING<[список перс.]>

DUPLICATE ARE [NOT] ALLOWED

VIA <Имя набора> Set (рядом с записью владельца)

SYSTEM (решать будет система)

Каждый тип записи должен быть приписан к некоторой физической области размещения:

WITHIN <Имя области размещения> AREA

После описания записи в целом идет описание внутренней структуры:

<Имя уровня> <Имя данного> <Шаблон> <Тип>

Номер уровня определяет уровень вложенности при описании элементов и агрегатов данных. Первый уровень — сама запись. Поэтому элементы или агрегаты данных имеют уровень начиная со второго. Если данное соответствует агрегату, то любая его составляющая добавляет один уровень вложенности.

Если агрегат является вектором, то он описывается как

<Номер уровня> <Имя агрегата>.<Номер уровня> <Имя 1-й сост.>

а если — повторяющейся группой, то следующим образом:

<Номер уровня> <Имя агрегата>.OCCURS <N> TIMES

где N — среднее количество элементов в группе.

Описание набора и порядка включения членов в него выглядит следующим образом:

SET NAME IS <Имя набора>:

OWNER IS (<Имя владельца> SYSTEM).

Далее указывается порядок включения новых экземпляров члена данного набора в экземпляр набора:

ORDER PERMANENT INSERTION IS {SORTED | NEXT | PREV | LAST FIRST}

После этого описывается член набора с указанием способа включения и способа исключения экземпляра — члена набора из экземпляра набора.

MEMBER IS <Имя члена набора> {AUTOMATIC | MANUAL} {MANDATORY OPTIONAL} KEY IS (ACCENDING | DESCENDING) <Имя элемента данных>

При автоматическом включении каждый новый экземпляр члена набора автоматически попадает в текущий экземпляр набора в соответствии с заданным ранее Порядком включения.

При ручном способе экземпляр члена набора сначала попадает в БД, а только потом командой CONNECT может быть включен в конкретный экземпляр набора. Если задан способ исключения MANDATORY, то экземпляр записи, исключаемый из набора, автоматически исключается и из базы данных

Язык манипулирования данными в сетевой модели

Все операции манипулирования данными в сетевой модели делятся на *навигационные операции* и *операции модификации.*

Навигационные операции осуществляют перемещение по БД путем прохождения по связям, которые поддерживаются в схеме БД. В этом случае результатом является новый единичный объект, который получает статус *текущего объекта.*

Операции **модификации** осуществляют как добавление новых экземпляров отдельных типов записей, так и экземпляров новых наборов, удаление экземпляров записей и наборов, модификацию отдельных составляющих внутри конкретных экземпляров записей

Операторы манипулирования данными в сетевой модели

Операция	**Назначение**
READY	Обеспечение доступа данного процесса или пользователя к БД (open file)
FINISH	Окончание работы с БД
FIND	Группа операций, устанавливающих указатель найденного объекта на текущий объект
GET	Передача найденного объекта в рабочую область (после FIND)
STORE	Помещение в БД записи, сформированной в рабочей области
CONNECT	Включение текущей записи в текущий экземпляр набора
DISCONNECT	Исключение текущей записи из текущего набора
MODIFY	Обновление текущей записи из рабочей области
ERASE	Удаление экземпляра текущей записи

В рабочей области пользователя хранятся шаблоны записей, программные переменные и три типа указателей текущего состояния:

- текущая запись процесса (код или ключ последней записи, с которой работала данная программа);
- текущая запись типа записи (для каждого типа записи ключ последней записи, с которой работала программа);
- текущая запись типа набор (для каждого набора с владельцем T1 и членом T2 указывается, T1 или T2 были последней обрабатываемой записью).

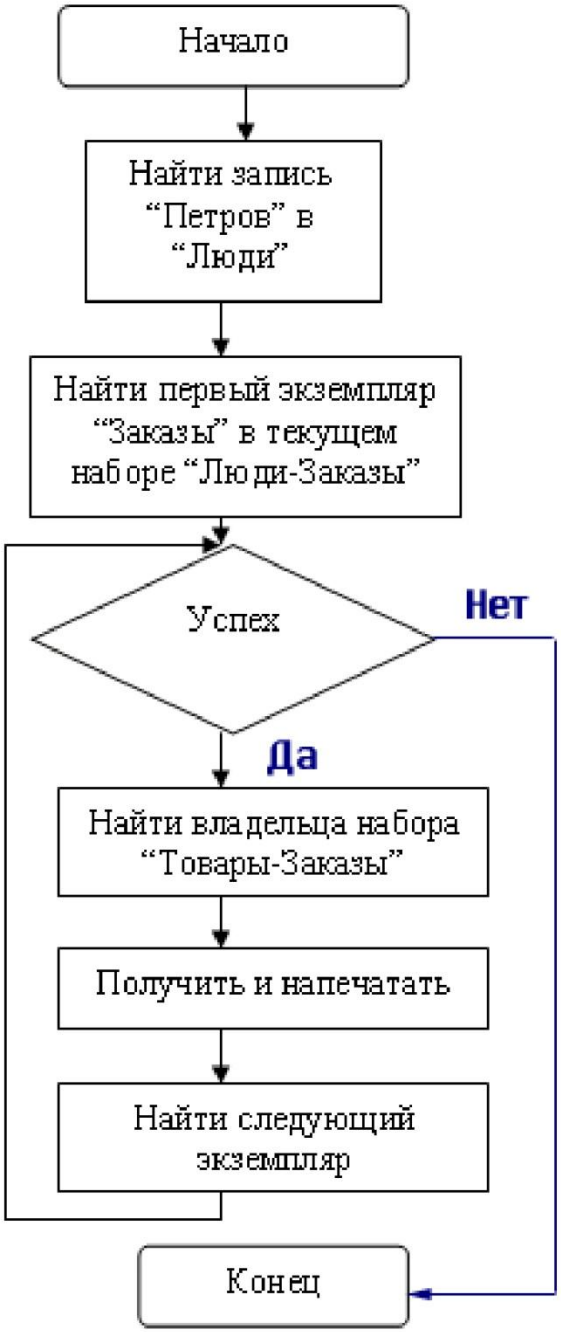
Наиболее интересна операция поиска (FIND), так как именно она отражает суть навигационных методов, применяемых в сетевой модели. Всего существует семь типов операций поиска:

1. По ключу (запись должна быть описана через CALC USING ...):

FIND <Имя записи> RECORD BY CALC KEY <Имя параметра>

2. Последовательный просмотр записей данного типа:
FIND DUPLICATE <Имя записи> RECORD BY CALC KEY
3. Найти владельца текущего экземпляра набора:
FIND OWNER OF CURRENT <Имя набора> SET
4. Последовательный просмотр записей—членов текущего экземпляра набора:
FIND (FIRST | NEXT) <Имя записи> RECORD IN CURRENT <Имя набора> SET
5. Просмотр записей—членов экземпляра набора, специфицированных рядом полей:
FIND [DUPLICATE] <Имя записи> RECORD IN CURRENT <Имя набора> SET USING <Список полей>
6. Сделать текущей записью процесса текущий экземпляр набора:
FIND CURRENT OF <Имя набора> SET
7. Установить текущую запись процесса:
FIND CURRENT OF <Имя записи> RECORD

Алгоритм и программа печати заказов, сделанных Петровым




```
ФИО = "Петров"
FIND Люди RECORD BY CALC KEY
FIND FIRST Заказы RECORD IN
CURRENT Люди-Заказы SET WHILE NOT FAIL
DO
FIND OWNER OF CURRENT
Товары-Заказы SET GET Товары
PRINT НаимТовара FIND NEXT Заказы
RECORD IN
CURRENT Люди-Заказы SET
END
```