Лекция 6. Этап логического проектирования. Понятие “модель данных”. Сетевая, иерархическая реляционная модели данных. Их отличия, достоинства и недостатки.

6.1 Понятие модели данных

На этапе концептуального проектирования формируются целостные представления о предметной области, а именно: объекты, интересующие нас в предметной области, их свойства и взаимоотношения.
Однако главной особенностью концептуальной модели является то, что она создается без учета способов хранения информации и доступа к ней (в этом состоит суть).
Следовательно, теперь необходимо каким-либо способом отобразить данную концептуальную модель предметной области в некоторую СУБД. При создании логической схемы базы данных исходят из возможностей имеющихся средств ввода, обработки и хранения данных.

> **Этап логического проектирования** заключается в создании набора структур данных, соответствующего конкретной концептуальной модели, и реализуемого в выбранной СУБД.

На этом этапе создается **логическая схема базы данных**.
Основное понятие этапа – **модель данных.**

> **Модель данных** (грубо) – способ представления данных в компьютере.

Очень Полезный Комментарий относительно употребления термина «*модель предметной области*» для информационных систем:

Математические модели и методы, такие как дифференциальные уравнения и численные (или аналитические) методы их решения, модели и методы оптимизации и т.п. создаются и применяются для того, чтобы получать неизвестные (трудноизмеримые, экстраполяционные, интерполяционные) значения функций (решения математических моделей), согласующиеся с заложенными в модель знаниями о законах реального мира.

Цель: получение *неизвестных* знаний на основании законов, считающихся известными в предметной области

У математических моделей предметных областей для представления в информационных системах задача другая: представить структуры предметной области и *реальные* данные из предметных областей наиболее адекватно и полно. Такие математические модели относятся к классу формальных логических моделей, т.е. моделей, построенных на основании математической логики и исчисления предикатов.

Цель таких моделей: *хранение, эффективный поиск, извлечение знаний* из реальных, зафиксированных, *известных*, данных, причем для этого используется аппарат математической логики.

Любая модель данных состоит из трех частей:

1) **структура данных** (как данные хранятся)
2) **ограничения целостности данных** (какие *внутренние* ограничения накладываются структурами данных, и как можно представить *внешние* ограничения, накладываемые предметной областью)
3) **манипулирование данными** (какие операции используются в модели для обращения к данным).

Замечание: Лучше всего о моделях данных написано в книге Цикритзиса и Лоховски [10].

Исторически сложилось так, что в течение долгого периода времени использовалось несколько моделей данных, которые применяются для проектирования баз данных и сейчас:

- сетевая;
- иерархическая;
- реляционная.

Однако за последние десять лет развились новые подходы к проектированию данных и сегодня наравне с предыдущими тремя моделями данных используются и активно разрабатываются такие:

- постреляционные модели данных (разрабатываются для новых типов данных, таких как время, пространство, графические образы, сложные объекты и др.),
- модели данных, базирующиеся на объектно-ориентированном подходе.

Рассмотрим сначала более “старые” модели данных.

6.2 Сетевая модель данных

Базируется на представлении данных в виде графа. **Вершины графа** используются для обозначения типов сущностей. **Дуги графа** обозначают **связи** между типами сущностей.

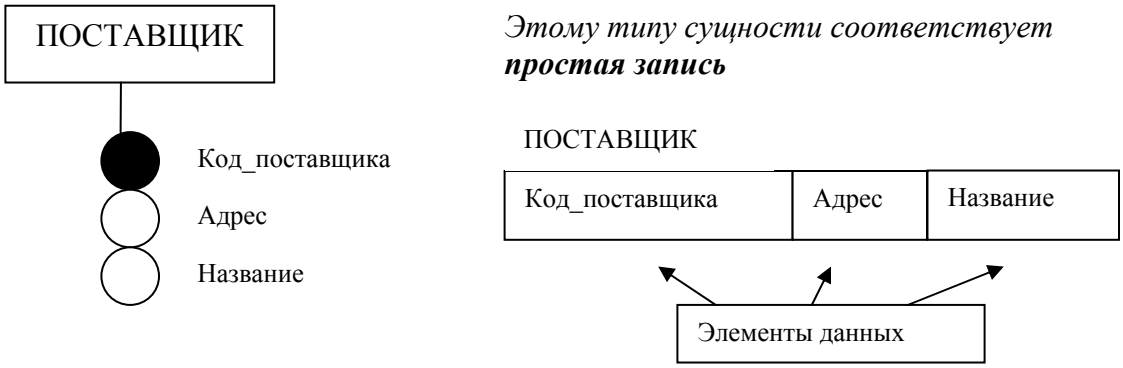
6.2.1. Структуры данных

Сетевая модель данных предлагает такие **структуры данных** для представления предметной области:

1) **элемент данных** (соответствует свойству сущности)
2) **записи** (соответствует типу сущности)
3) **агрегат** данных
4) **набор** (представляет связь между типами сущностей)
5) **база данных**

Проиллюстрируем эти структуры на примерах.

Пример 1.



Этому типу сущности соответствует ***простая запись***

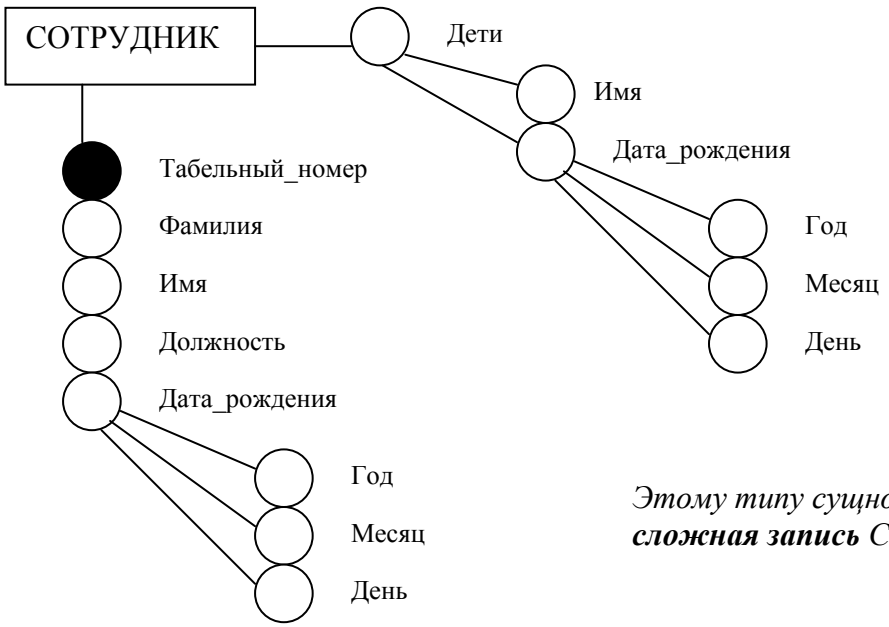
Понятие ”агрегат данных” используется при построении сложной иерархической структуры элементов данных.

Агрегат данных – несколько элементов данных, каждый из которых может состоять из нескольких элементов данных.

Обычно агрегат данных имеет несколько уровней вложенности других агрегатов данных.

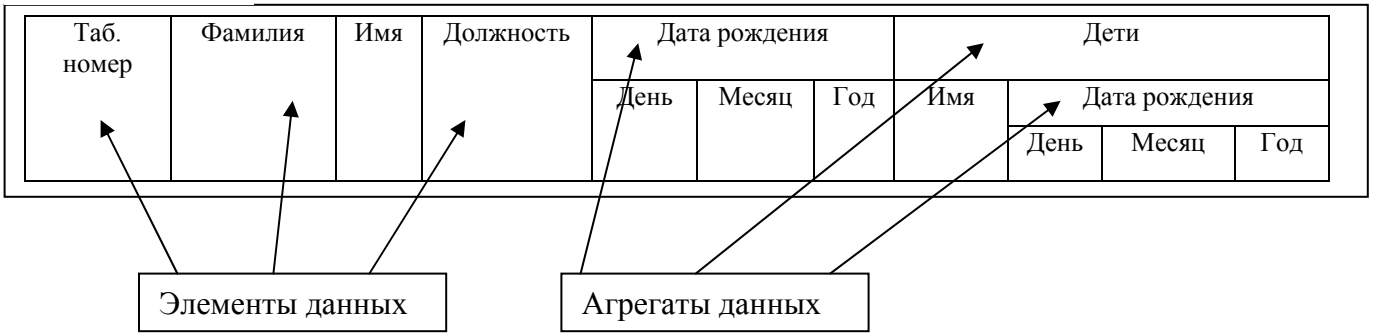
В примере 2: табельный номер, фамилия, имя, должность, - элементы данных, дата рождения, дети – агрегат первого уровня.
Агрегат первого уровня - “дата рождения” состоит из агрегатов “день”, “месяц”, “год”.
Агрегат второго уровня - "дети";
"год", "месяц", "день" рождения детей – агрегат третьего уровня.
Таким образом, агрегат данных – обобщенное понятие для элемента данных.

Пример 2.



Этому типу сущностей соответствует ***сложная запись*** *СОТРУДНИК.*

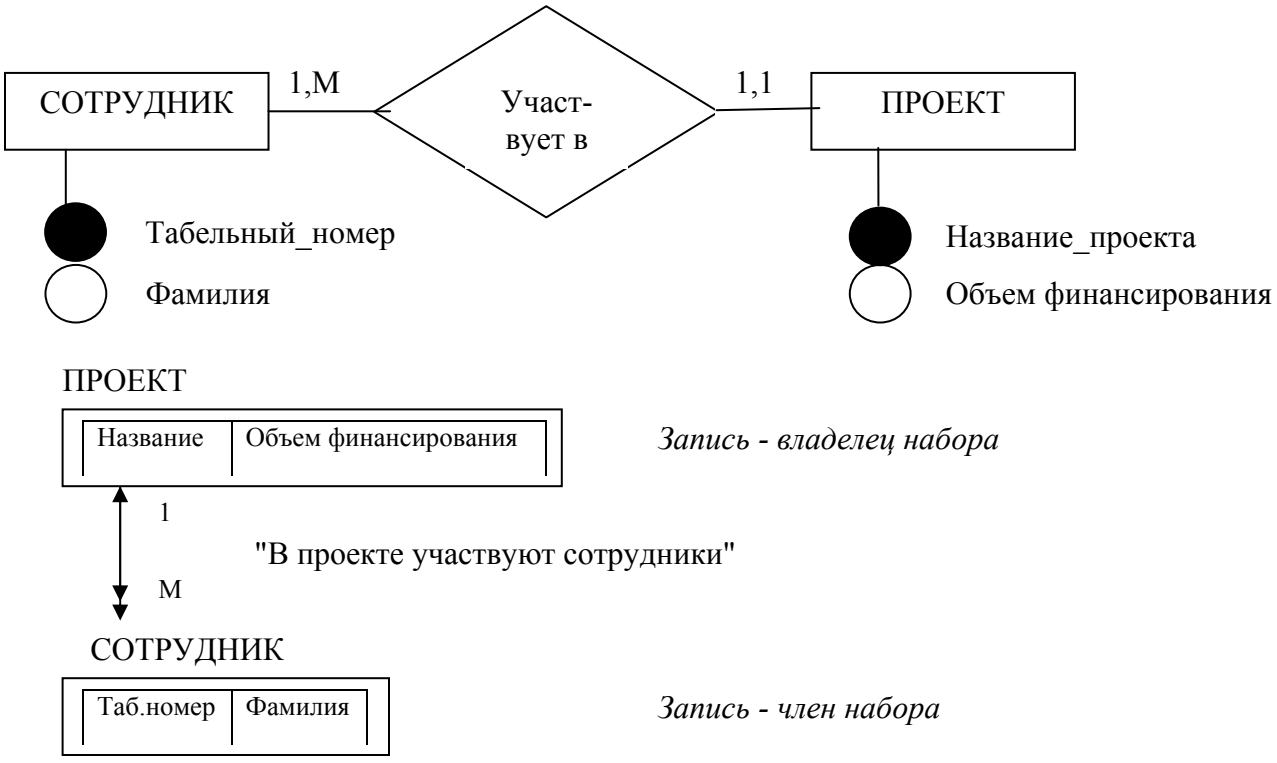
СОТРУДНИК



Для представления связей между записями используются *наборы*.

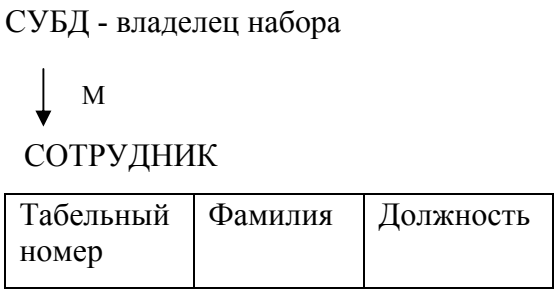
Набор изображается поименованной дугой между соответствующими типами записей. Дуга исходит из записи – владельца набора (та сторона, у которой максимальная мощность связи больше) и заходит в запись – член набора(сторона, у которой максимальная мощность связи меньше).
Экземпляр набора **должен** содержать один экземпляр записи – владельца набора, и может содержать несколько экземпляров записей – членов набора.

Пример 3.



Существует особый вид наборов – **сингулярный набор**.
Сингулярный набор – тип набора без записи – владельца (владельцем набора является СУБД)

Пример 4.



Сингулярный набор

Наконец, **базы данных** формируются таким образом:

1) база данных может состоять из любого количества типов записей и наборов
2) между двумя типами записей может существовать несколько наборов
3) типы записи может быть владельцем и одновременно членом нескольких наборов
4) тип записи может иметь более одного типа записи – владельца.

6.2.2 Ограничения целостности

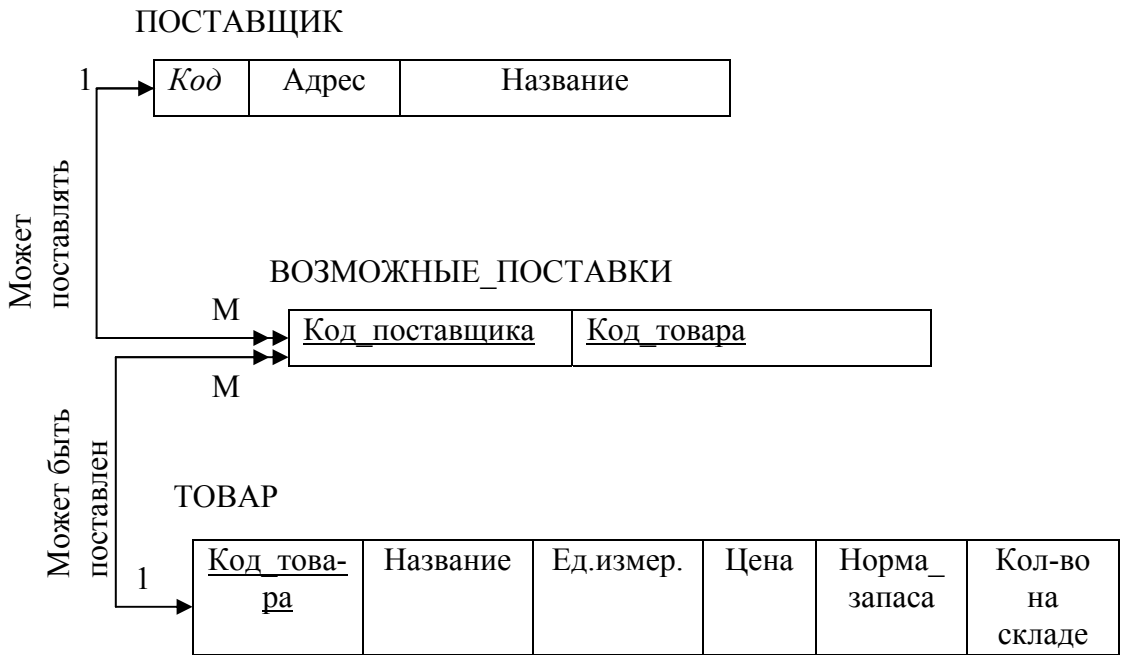
В сетевой модели CODASYL основным *внутренним* ограничением целостности является ограничения на связи:

Существуют и поддерживаются связи 1:1 и 1:М.

Т.е. “в конкретном экземпляре набора экземпляр записи – члена набора может иметь *не более* 1 экземпляра записи – владельца набора”.

Пример 5. Пример частичной логической схемы базы данных в сетевой модели данных (на основе внешней схемы “Поставки” из лекции 5).

Пример 6. Для представления связи M:N вводится вспомогательный тип записи и два набора 1:M.



6.2.3 Манипулирование данными.

В сетевой модели операции для работы с данными являются **ориентированными на записи** (т.е. операции переходов (навигаций) по графу данных) или **навигационными**.

FIND-найти записи, удовлетворяющие некоторому набору условий
GET-чтение нужного экземпляра записи
STORE- запись
MODIFY-изменения
ERASE-удаление

По способу получения результата различают навигационные и спецификационные операции. Навигационные операции считаются низкоуровневыми (процедурными), спецификационные – высокоуровневыми (непроцедурными) операциями, потому что для нахождения нужной записи с использованием навигации по графу данных нужно подробно указать путь к данному или соответствующую процедуру поиска по графу нужной записи.
Результат навигационной операции – одна запись. Следовательно, если нужно обработать все записи, то необходимо указать циклы повторения процедур (а в процедурных языках манипулирования данными вся работа по обработке запроса к сетевой базе данных ложится на плечи программиста).

6.3 Иерархическая (древовидная) модель данных.

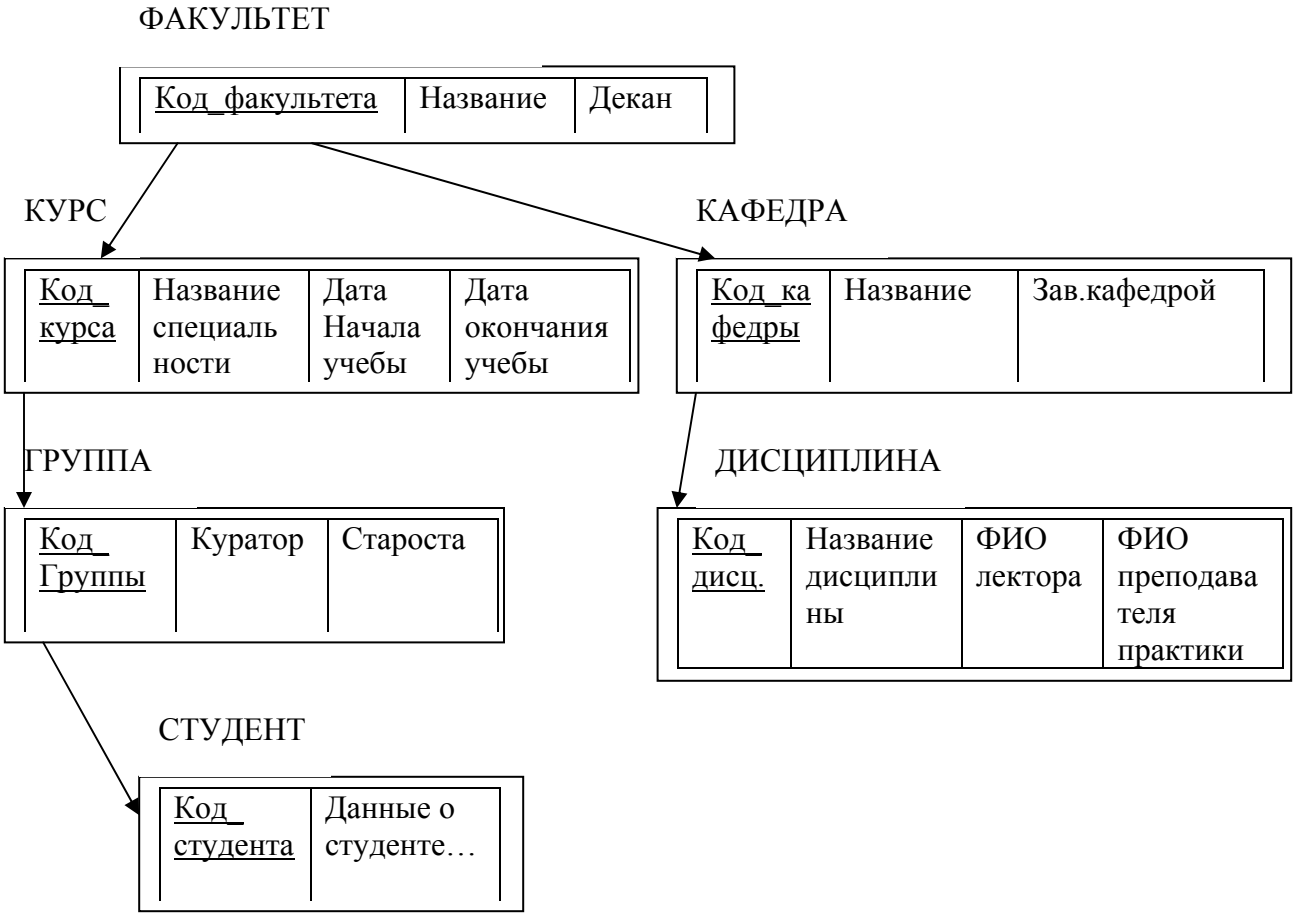
Как и в сетевой, в иерархической модели данных используются графы. Вершины соответствуют сущностям, дуги – связям. Однако, в отличие от сети, иерархическая модель данных представляет собой **дерево**, **имеющее единственный корень**, т.е. существует единственный корневой тип записи (существует единственная запись - владелец).

Само дерево определяется так :

1. существует единственная запись –корневая, в которую не заходит ни одно ребро
2. во все остальные вершины заходит одно ребро, а исходит произвольное (0...N) число ребер
3. нет циклов и петель.

Исходя из информации о построении дерева можно сразу судить о внутренних ограничениях целостности в древовидной модели данных.

Пример 7.
Описывая предметную область «Факультет», можно представить ее в виде иерархической структуры.



Связи не именуются, т.к. в дереве не должно быть несколько дуг между двумя вершинами. Каждая запись определяется однозначно полным сцепленным ключом, который является совокупностью ключей всех записей от корневой до нужной записи. Следовательно, экземпляры записи нижнего уровня в дереве могут быть определены не полностью, если не известны экземпляры соответствующих исходных записей.

Такая ситуация называется **зависимостью от пути**.

Ограничения целостности и операции манипулирования данными такие же, как и в сетевой модели данных.

Более сложный пример иерархии рассмотрен в Приложении 1.
Примеры СУБД, ориентированных на иерархические и сетевые структуры: **IDS/360**, **TDMS**.

6.4 Достоинства и недостатки сетевой и древовидной моделей данных

Общим плюсом можно считать то, что иерархические и сетевые структуры создаются относительно просто.
К минусам относится то, что:
В иерархической модели данных

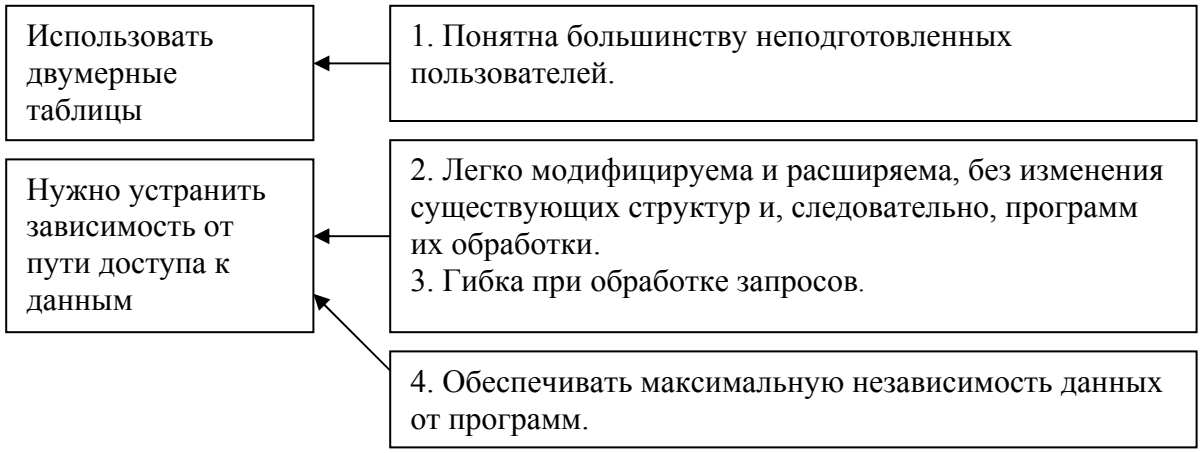
1. Большинство реальных предметных областей не обладает иерархической структурой. Следовательно, древовидные структуры применяются эффективно лишь тогда, когда предметная область хорошо формализуется в виде иерархии объектов.
2. Следствие из п.1. Для описания неиерархических предметных областей, в рамках иерархии создают искусственно более сложные структуры. Это усложняет обновление структуры предметной области.

Для иерархической и сетевой моделей данных характерно то, что в обеих моделях данных для поиска нужной записи, для работы с записями вообще используют навигационные операции (ориентированные на записи), что требует от программиста использования процедур позаписной обработки данных. Следовательно, модификация этих процедур сопряжена со значительными затратами рабочего времени.
Также, по мере усложнения модели данных количество связей между записями увеличивается, и все более запутывается. Следовательно, схему сложно понять при большом количестве связей и сущностей предметной области.
В этой связи характерно высказывание Дж.Мартина: “Базам данных постоянно грозит опасность стать громоздкими, застывшими и слишком сложными системами”

Поэтому к 70-м гг.20 века созрела необходимость альтернативы иерархиям и сетям данных. Так появилась идея создания нового типа баз данных, простых, понятных, удобных.
Автор идеи – *Эд.Ф.Кодд*.

6.5 Реляционная модель данных

Реляционная модель данных была создана для устранения недостатков сетевой и иерархической моделей данных.
Прежде всего, новая модель данных должна быть



Начать применение реляционной модели данных можно двумя путями:

- научиться представлять уже существующие сетевые и иерархические структуры данных в виде двумерных таблиц
- научиться конвертировать концептуальную схему в реляционную модель данных (т.е. из ER- диаграмм в двумерные таблицы) (более подробно - см. Приложение 2)

Рассмотрим вначале методики преобразования сетевых и иерархических структур в двумерные таблицы.
Необходимо, чтобы каждая запись каждой таблицы идентифицировалась однозначно некоторым ключом, **все составные части которого хранятся в этой же таблице** (т.е. нет необходимости знать путь к записи в базе, чтобы однозначно идентифицировать ее)

Как это сделать?
Пример 8 (Как превратить дерево в набор таблиц).
Пусть дана иерархия "Факультет" (см. пример 7).

Корню иерархии, записи "Факультет", соответствует первая таблица «ФАКУЛЬТЕТ»:

Код_факультета	Название	Декан

Вторая таблица "КУРС"

Во вторую таблицу (порожденную) добавляется поле ключа ФАКУЛЬТЕТА ("Код_факультета"). Запись о каждом курсе уникально идентифицируется не просто номером курса (3, 4, ...), а еще и кодом факультета. Первичный ключ таблицы "Код_факультета" + "Код_курса" - сцепленный.

Код_факультета	**Код_курса**	Название специальности	Дата Начала учебы	Дата окончания учебы

Третья таблица "ГРУППА". Номер конкретной группы зависит от курса и факультета, следовательно, нужно добавить ключи "Код_курса" и "Код_факультета" к первичному ключу "Код_группы".

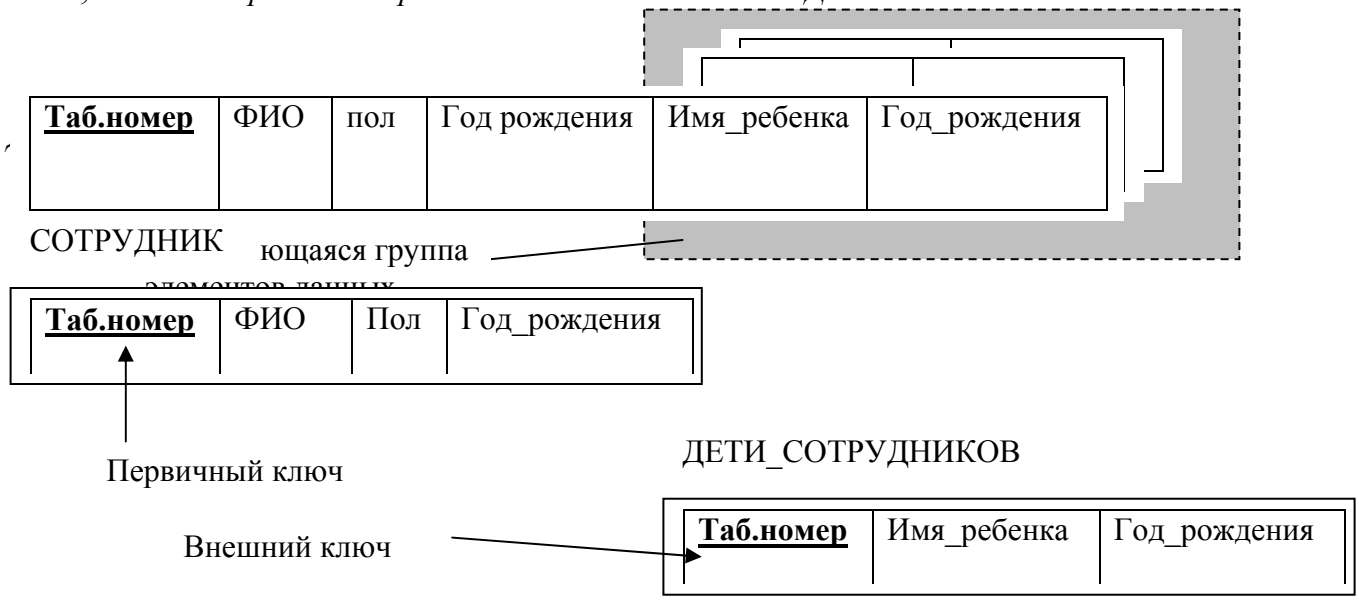
Код_факультета	**Код_курса**	**Код_группы**	Куратор	Староста

Четвертая таблица "СТУДЕНТ"

Код_факультета	**Код_курса**	**Код_группы**	**Код_студент**	Данные о студенте

Аналогично поступаем со всеми остальными типами записей в иерархии.

Пример 9.(Как превратить сеть в набор таблиц?).
Сеть, одной из вершин которой является запись "СОТРУДНИК".



Общий вывод:
Иерархии и деревья можно представить в виде набора двухмерных таблиц. Следовательно, реляционная модель данных позволяет использовать те базы данных, которые были накоплены в сетях и деревьях.

6.5.1 Структура реляционных данных

Данные представляются в виде двумерных **таблиц** определенного вида, называемых **реляционными отношениями** (см. лекцию 8). **Столбцы** таблицы соответствуют различным атрибутам, **строки** таблицы соответствуют записям, характеризующим один экземпляр типа сущности или связи. Для уникальной идентификации записи используется понятие **первичного ключа**. Для каждого атрибута существует понятие **домена** – множества его допустимых значений. Каждый атрибут имеет уникальное **имя**.

6.5.2 Ограничения целостности на данные

В реляционной модели данных можно представить все типы связей 1:1, 1: N, M : N, обязательные и необязательные.

Каждая таблица должна иметь первичный ключ, для которого выполняются два соотношения:

а) не существует двух одинаковых значений ключа

б) из первичного сцепленного ключа нельзя удалить ни одного атрибута без нарушения соотношения (а)

В реляционной модели данных принято два основных *внутренних* ограничения:

1) **"Целостность по сущностям"**
Не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение.

Если первичный ключ принимает неопределенное значение, то это значит, что существует некоторый класс объектов, не обладающих индивидуальностью, или безликих.

Когда мы преобразовывали иерархическую структуру в набор реляционных таблиц, мы добавляли в таблицы, порожденные подчиненными типами записей, первичные ключи родительских записей. Это было необходимо для того, чтобы можно было обратиться к любой записи в любой таблице напрямую, не используя путь к этой записи.

Второе ограничение связано с понятием *внешнего* ключа.

> **Внешний ключ**- атрибут или набор атрибутов одного реляционного отношения R2, каждое значение которого **обязательно** должно совпадать со значением первичного ключа некоторого другого отношения R1.

2) **“Целостность по ссылкам”**
Если FK_B – внешний ключ в таблице B, PK_A – соответствующий ему первичный ключ в таблице A, то для любого значения FK_B из B верно либо:

а) существует значение PK_A, равное значению FK_B; либо
б) FK_B-полностью неопределен.

Пример 10.
В примере 9 в таблице “ДЕТИ_СОТРУДНИКОВ” есть ключ «Таб.номер». В этой таблице он является внешним ключом.

6.5.3 Манипулирование данными

Операции для работы с данными ориентированы не на записи, а на реляционные отношения, т.е. на *множества записей*. Всего операций манипулирования реляционными данными – восемь. Четыре из них – теоретико-множественные (объединение, пересечение, разность, декартово произведение), другие четыре операции – чисто реляционные – выборка, проекция, соединение, переименование отношений. Операции эти – декларативные, т.е. высокоуровневые. Навигация по данным отсутствует. Подробнее об операциях манипулирования реляционными данными – см. лекцию 11.

Приложение 1

Предположим, нам нужно описать структуру Web-узла. Каждая страница узла является некоторой вершиной дерева. На каждой странице могут находиться ссылки на другие страницы. Каждая страница характеризуется такими данными как: «название_страницы», «номер_уровня», «номер_страницы_на_уровне» (см. Рис.6.1).
ER—диаграмма, соответствующая такой структуре имеет вид как на Рис. 6.2.

Такая ситуация представляется сетью, в которой присутствует всего один тип записи, и вместо дуги – петля (см. Рис. 6.3).

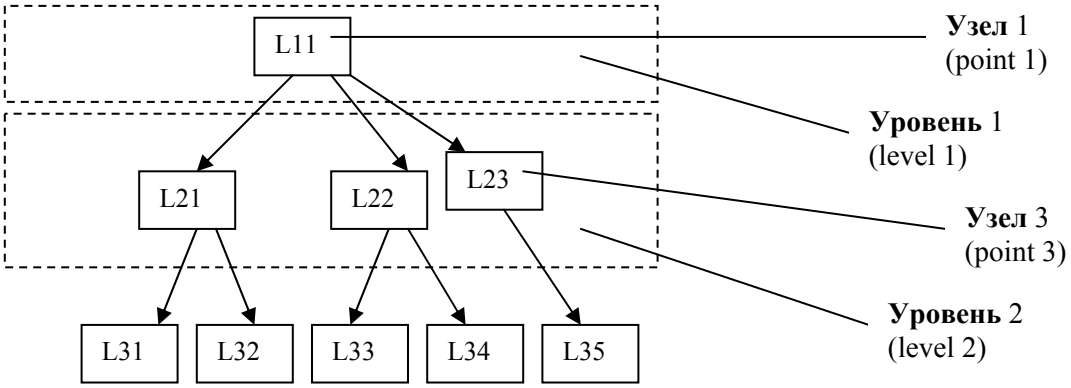


Рисунок 6.1 – Пример структуры Web-узла

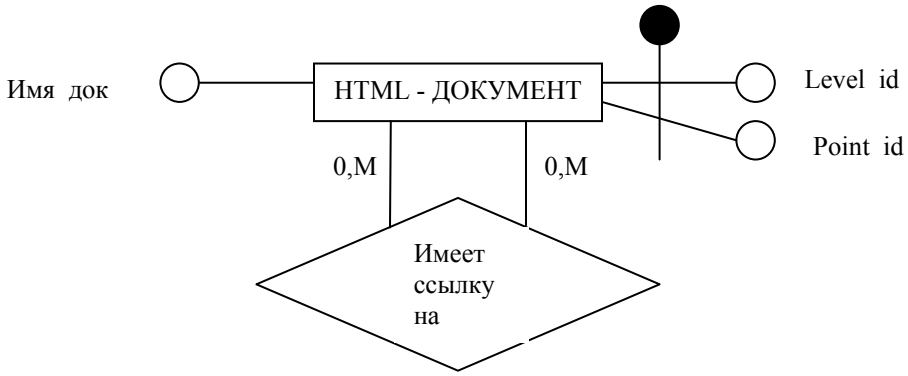


Рисунок 6.2 – Образец диаграммы Чена для Web-узла.

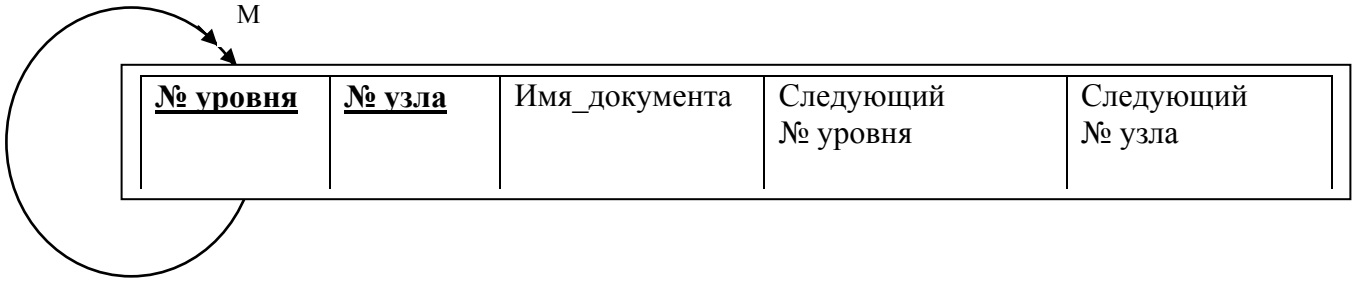
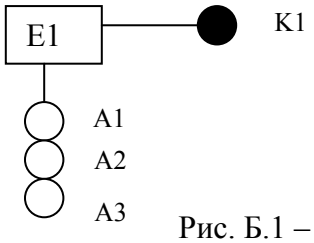


Рисунок 6.3 – Образец сетевой структуры для Web-узла.

Приложение 2

При переходе от ER-диаграмм к реляционной схеме данных используются такие правила.

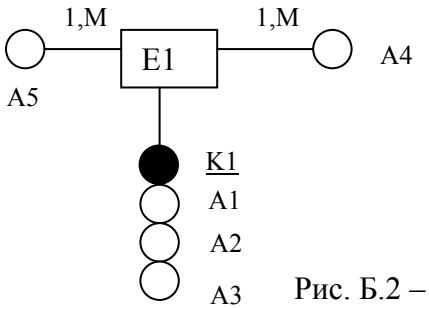
1. Для каждого простого типа сущностей и его единичных свойств строится отношение, атрибутами которого являются идентификатор типа сущностей и все его единичные свойства. Первичный ключ этого отношения – идентификатор типа сущностей.



R1 (<u>K1</u>, A1, A2, A3)

Рис. Б.1 – Тип сущности с единичными свойствами

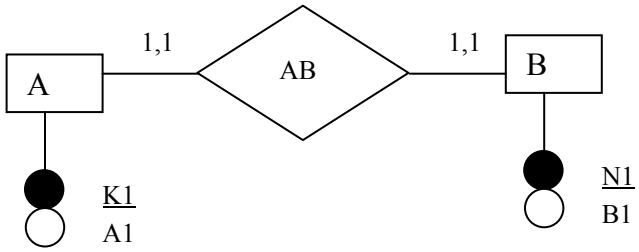
2. Если у объекта есть множественные свойства, то каждому из них ставится в соответствие отдельное отношение. Ключом этого отношения будет идентификатор типа сущностей, а остальные атрибуты – соответствуют данным множественным свойствам.



R1 (<u>K1</u>, A1, A2, A3)
и
R2 (<u>K1</u>, <u>A4</u>)
R3 (<u>K1</u>, <u>A5</u>)

Рис. Б.2 – Тип сущности с единичными и множественными свойствами

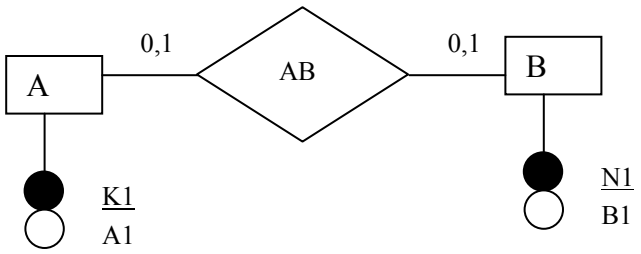
3. Если между двумя типами сущностей присутствует связь (1,1): (1,1), т.е. **обязательная с обеих сторон**, то создаются два отношения, по одному на каждый тип сущностей. Первичные ключи этих сущностей –соответствующие идентификаторы.



R1 (<u>K1</u>, A1)
и
R2 (<u>N1</u>, B2)

Рис. Б.3 – Обязательная связь 1:1 между двумя сущностями.

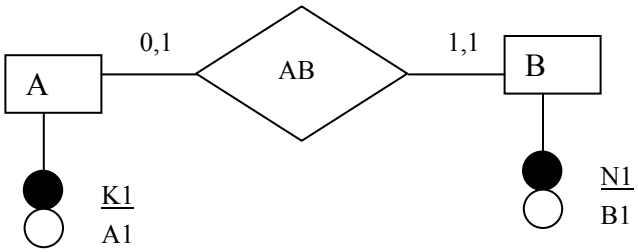
4. Если между двумя типами сущностей присутствует связь(0,1): (0,1), т.е. **необязательная с обеих сторон**, то создаются три отношения. Два из них соответствуют каждому типу сущностей. Первичные ключи этих сущностей – соответствующие идентификаторы. Третье отношение содержит идентификаторы обоих сущностей.



R1 (K1, A1)
R2 (N1, B2)
и
R3 (K1, N1)

Рис. Б.4 –Связь 0:1 между двумя сущностями.

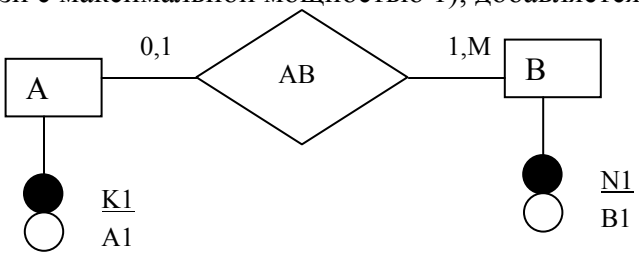
5. Если между двумя типами сущностей присутствует связь (0,1) : (1,1), т.е. **необязательная хотя бы с одной стороны**, то создаются два отношения. Первичные ключи этих сущностей – соответствующие идентификаторы. Но первичный ключ сущности, для которой минимальная мощность связи равна 0, добавляется в отношение, соответствующее той сущности, для которой минимальная мощность связи равна 1.



R1 (K1, A1)
R2 (N1, B1, K1)

Рис. Б.5 –Связь 0:1 между двумя сущностями,
необязательная с одной стороны.

6. Если между двумя типами сущностей присутствует связь (0,1):(1,M) или (1,1):(1,M), **т.е. класс принадлежности хотя бы многосвязной сущности** (сущности, участвующей в связи с максимальной мощностью M) **обязателен**, то создаются два отношения, по одному на каждую сущность. Первичные ключи этих сущностей – соответствующие идентификаторы. Но в отношение, соответствующее **односвязной сущности** (сущности, которая участвует в связи с максимальной мощностью 1), добавляется первичный ключ M-связной сущности

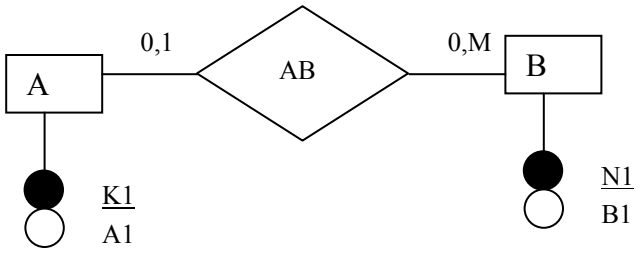


R1 (K1, A1)
R2 (N1, B1, K1)

Рис. Б.6 –Связь 1:M между двумя сущностями.
A – M-связная сущность

7. Если между двумя типами сущностей присутствует связь (0,1):(0,M) или (1,1):(0,M) т.е. **необязательная со стороны M—связной сущности**, то создаются три отношения. Два из них соответствуют каждому типу сущностей. Первичные ключи этих сущностей –

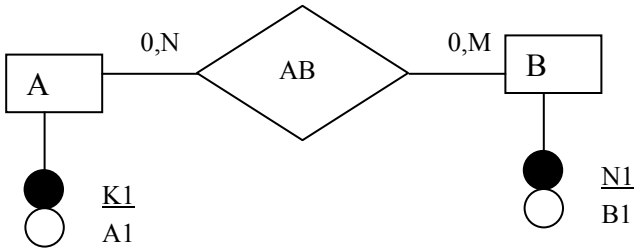
соответствующие идентификаторы. Третье отношение содержит идентификаторы обоих сущностей.



R1 (K1, A1)
R2 (N1, B1)
и
R3 (K1, N1)

Рис. Б.7 –Связь 0:M между двумя сущностями.

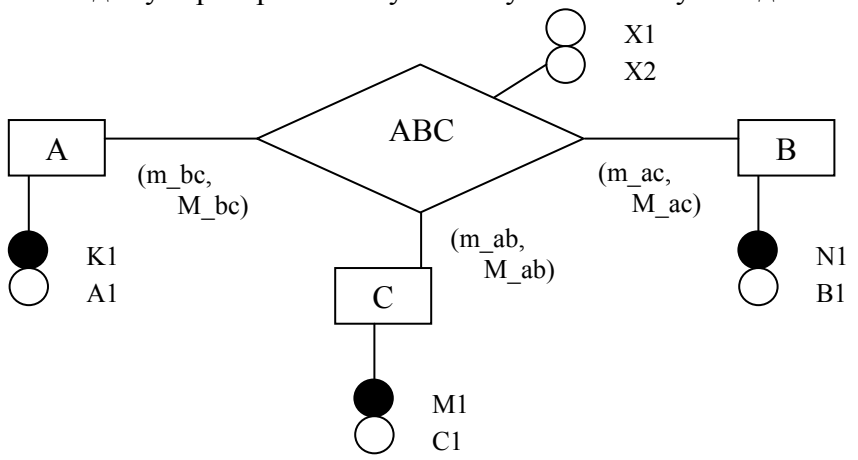
8. Если между двумя типами сущностей присутствует связь(0,N) : (0,M) или (1,N) : (1,M), то создаются три отношения. Два из них соответствуют каждому типу сущностей. Первичные ключи этих сущностей – соответствующие идентификаторы. Третье отношение содержит идентификаторы обоих сущностей.



R1 (K1, A1)
R2 (N1, B1)
и
R3 (K1, N1)

Рис. Б.8 –Связь N:M между двумя сущностями.

9. Каждому агрегированному объекту соответствует отдельное отношение.

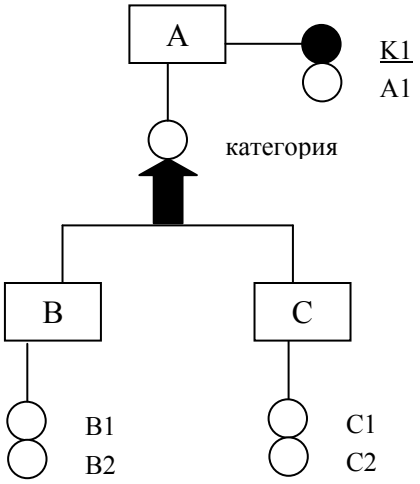


R1 (K1, A1)
R2 (N1, B1)
R3 (M1, C1)
и
R4 (K1, N1,M1, X1, X2)

Рис. Б.9 –Агрегированный объект.

10. Обобщенный объект представляется следующим образом.

Способ 1. Всему обобщенному объекту соответствует одно отношение (свойства потомков переходят родителю).
Способ 2. Каждой категории объектов нижнего уровня соответствует отдельное отношение (свойства родителя наследуются потомками).



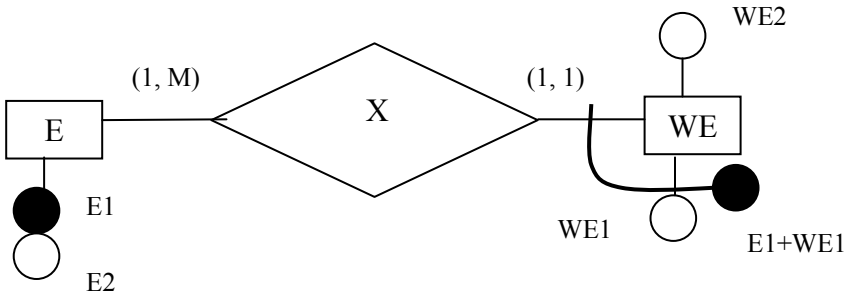
Способ 1
R1 (K1, A1, B1, B2, C1, C2, категория)

Способ 2
R1 (K1, A1, B1, B2)
R2 (K1, A1, C1, C2)

Рис. Б.10 –Обобщенный объект.

10. Если между сущностями есть связь типа «целое – часть», то она разрешается как связь M:N.

11. Если связь установлена между «сильной» и «слабой» сущностями, то создается два отношения, причем первичный ключ «сильной» сущности добавляется к атрибутам отношения, соответствующего «слабой» сущности.



R1(E1, E2)
R2(E1, WE1, WE2)

Рис. Б.11 –«Сильная»(E) и «слабая» (WE) сущности.

Приложение 3.

После преобразования ER-диаграмме из лекции 5 (Рис.5.8) будет соответствовать такая реляционная схема:

МЕНЕДЖЕР (кодМенеджера, ФИО)
ПОСТАВЩИК (кодПоставщика, названиеПоставщика, адресПоставщика, телефон)
ПОКУПАТЕЛЬ (кодПокупателя, названиеПокупателя, адресПокупателя, телефон)
СКЛАД (кодСклада, названиеСклада, типСклада)
ЕДИЗМЕР (едИзм)

ТОВАР (номенклНомер, едИзм, название)
ПРИХОДНАЯ_НАКЛАДНАЯ (номерПН, дата, кодПоставщика, кодМенеджера)
РАСХОДНАЯ_НАКЛАДНАЯ (номерРН, дата, кодПокупателя, кодМенеджера)
ПРИХОДНЫЙ_СКЛАДСКОЙ_ОРДЕР (номерПСО, дата, номерПН, кодСклада, номенклНомер, едИзм, количество, вхЦена)
РАСХОДНЫЙ_СКЛАДСКОЙ_ОРДЕР (номерРСО, дата, номерРН, кодСклада, номенклНомер, едИзм, количество, отпЦена)

Примечание: в каждом реляционном отношении подчеркиванием выделены атрибуты первичного ключа.