1.3. Виды моделей данных

Общие положения. Ядром любой базы данных является модель данных. Модель данных представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных – совокупность структур данных и операций их обработки (логическая структура данных в БД).

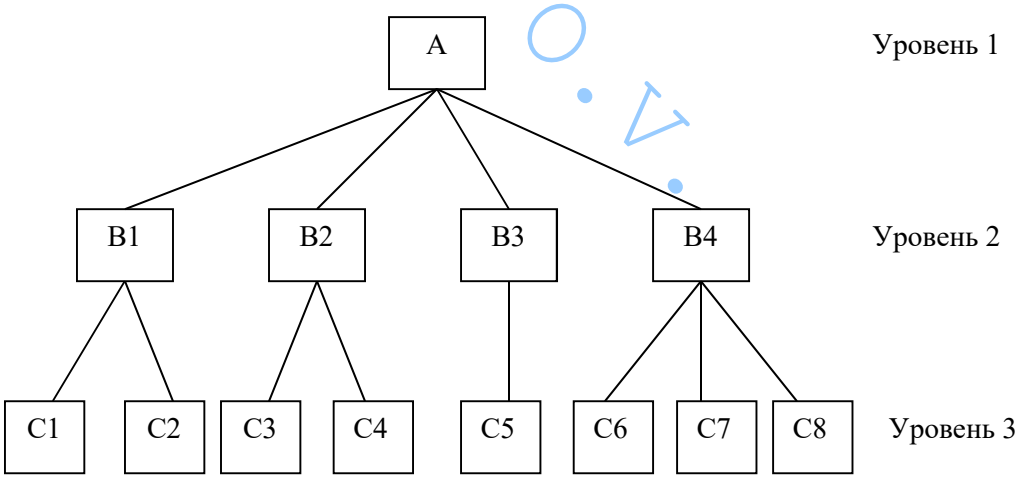
1. иерархическая модель
2. сетевая
3. реляционная (данные в виде таблиц)
4. постреляционная
5. многомерная
6. объектно-ориентированная

СУБД основывается на использовании иерархической, сетевой или реляционной модели, на комбинации этих моделей или на некотором их подмножестве.

Рассмотрим основные типы моделей данных.

Иерархическая модель

Связи между данными можно описать с помощью упорядоченного графа (дерева).



Основные понятия:

- уровень
- узел (элемент)
- связь

Узел – совокупность атрибутов - данных, описывающих некоторый объект.

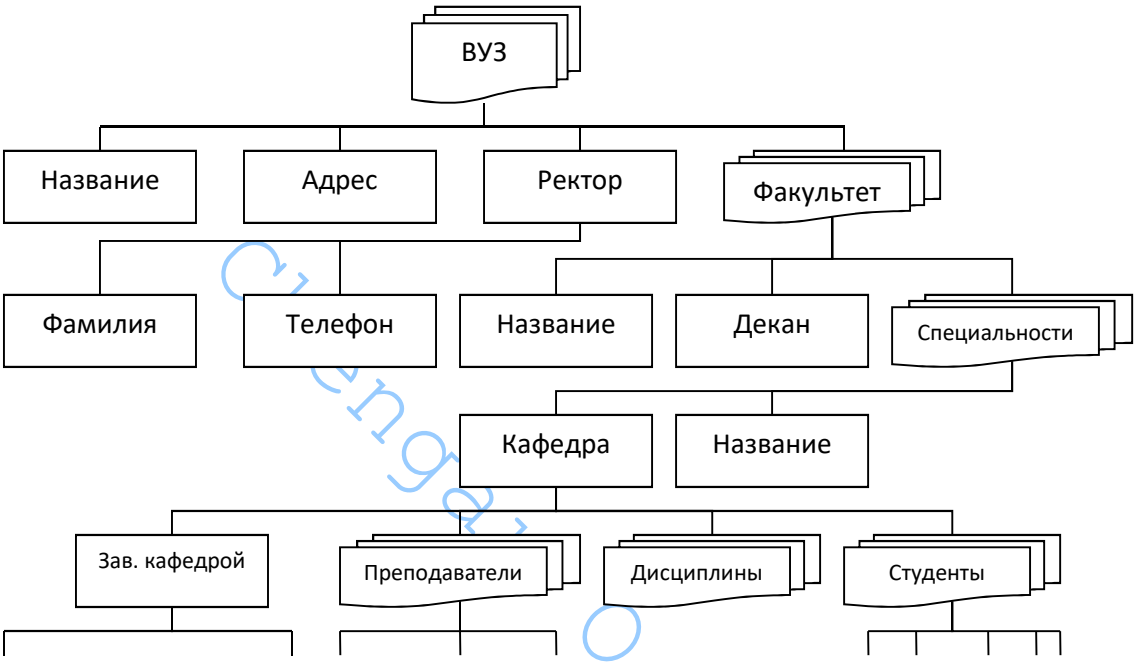
Дерево имеет один и только один корневой узел.

Основные правила контроля целостности:

Узел-потомок не может существовать без узла-родителя, а у некоторых родителей может не быть узлов-потомков.

К каждой записи (узлу БД) существует только один путь от корневой записи.

Пример.



Подобную (иерархическую) БД можно описать в программе при помощи типа запись.

Тип ВУЗ – это тип запись, реализующий дерево. Он включает в себя два элемента простого типа (Название, Адрес – тип string), элемент запись (Ректор), массив записей (элемент Факультет).

Для организации физического размещения иерархических данных в памяти ЭВМ используется представление линейными списками (Дж. Мартин).

Основные операции манипулирования иерархически организованными данными

1. Поиск указанного экземпляра БД (пример, дерево со значением 10 в поле отдел №)
2. Переход от одного дерева к другому

3. Переход от одной записи к другой внутри дерева
4. Вставка новой записи в указанную позицию
5. Удаление текущей записи и т. д.

Обход всех элементов в иерархической БД происходит сверху вниз слева направо.

Достоинства

- эффективное использование памяти ЭВМ;
- хорошие показатели времени выполнения основных операций над данными.

Недостатки

- отсутствуют механизмы поддержания целостности связи между записями различных деревьев;
- громоздкость иерархической модели для обработки информации с достаточно сложными логическими связями;
- сложность понимания для обычного пользователя.

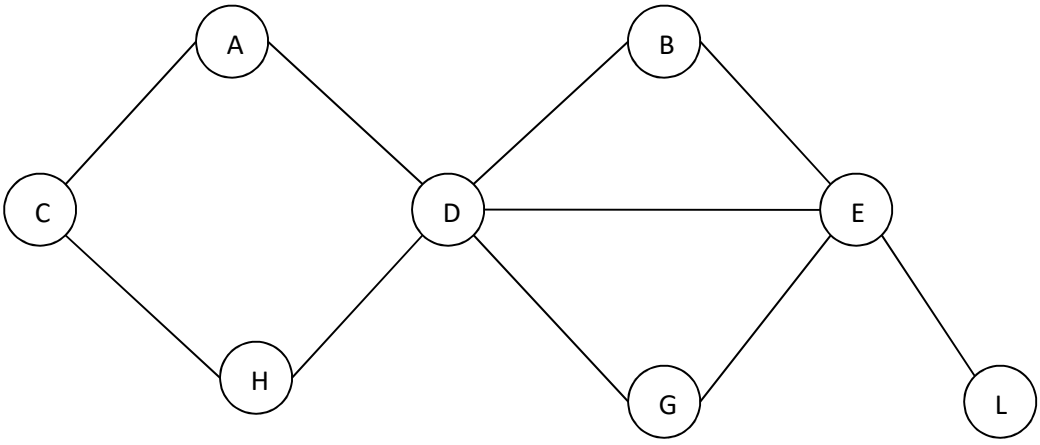
СУБД, поддерживающие иерархическую модель, - IMS, PC/Focus, Team-Up, Ока, ИНЭС, МИРИС.

Сетевая модель данных

Позволяет отображать разнообразные связи элементов данных в виде произвольного графа, тем самым обобщая иерархическую модель.

Структура называется сетевой, если в отношениях между данными структурный элемент может иметь более одного исходного.

При тех же основных понятиях (узел, элемент, связь) в сетевой структуре каждый элемент может быть связан с каждым.

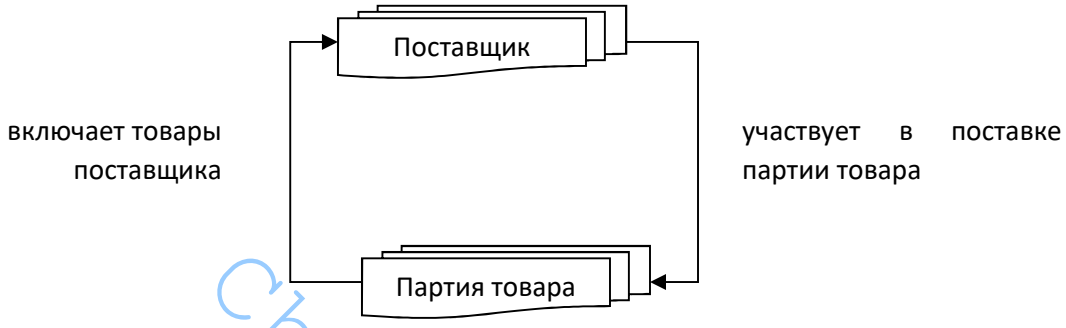


Для описания схемы сетевой БД используются группы типов запись и связь.

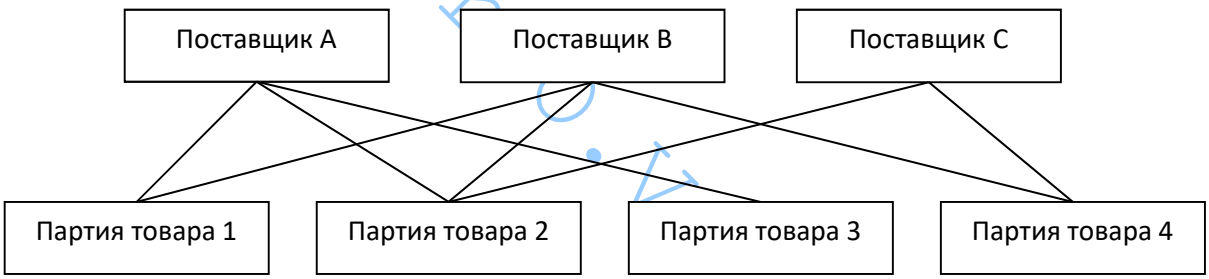
Тип связь определяется для двух типов запись – предка и потомка. Переменные типа связь являются экземплярами связи.

Сетевая БД состоит из набора записей и набора соответствующих данных. На формирование связей ограничения не накладываются.

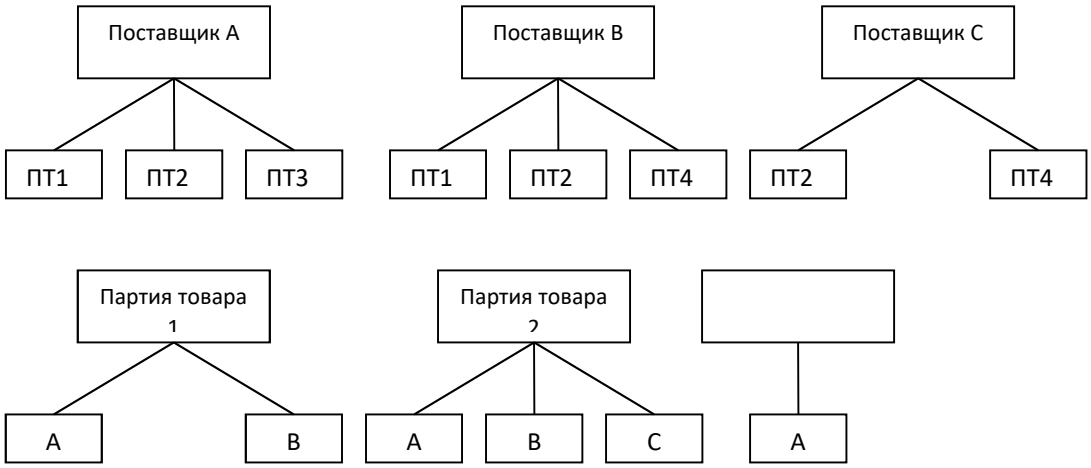
Пример:



Структура является сетевой, если каждый экземпляр поставщика может участвовать в нескольких поставках и каждая поставка может иметь несколько поставщиков.



Сетевая структура может быть представлена в виде эквивалентных древовидных структур (при этом в БД вносится избыточность).



Любая иерархическая структура может быть сведена к плоской (табличной).

Физическое размещение данных организуется теми же методами, что и в иерархической БД.

Основные операции манипулирования данными:

1. поиск записей в БД;
2. переход от предка к первому потомку;
3. переход от потомка к первому предку;
4. создание новой записи;
5. удаление или обнуление текущей записи;
6. включение (исключение) записи в связь.

Достоинства

- возможность эффективной реализации по показателям затрат памяти и оперативной обработки;
- большие возможности по сравнению с иерархической моделью в допустимости образования новой связи.

Недостатки

- высокая сложность и жесткость схемы БД;
- сложность понимания и выполнения обработки информации для обычного пользователя;
- ослаблен контроль целостности связей вследствие допустимости установки программных связей.

СУБД, поддерживающие сетевую модель: IDMS, Db_Vista III, СЕТЬ, СЕКТОР, КОМПАС.

Реляционная модель

(Эдгар Кодд)

Основывается на понятии отношений (relation).

Отношение представляет собой множество элементов, называемых кортежами.

Наглядной формой представления отношений является двумерная таблица.

Таблица соответствует объекту (сущности).

Строкам таблицы соответствуют кортежи (экземпляры объектов), а столбцам соответствуют атрибуты отношений (свойства объекта).

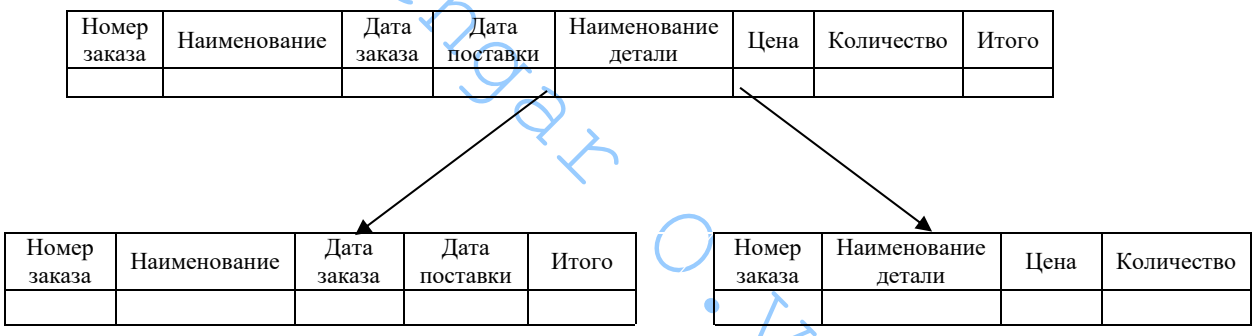
Пример

Студент

№ зач. книжки	**Фамилия**	**Имя**	**Отчество**	**Группа**
16493	Сергеев	Петр	Иванович	М-31
16593	Петрова	Анна	Игоревна	М-32

Так как в рамках одной таблицы невозможно описать более сложные структуры, используется связывание таблиц.

Физическое размещение данных в реляционной таблице на внешних носителях можно осуществить с помощью обычных файлов (плоских).



Достоинства

- простота, понятность, удобство физической реализации на ЭВМ.

Недостатки

- сложность описания иерархических и сетевых структур (избыточность данных);
- отсутствие стандартных средств идентификации отдельных записей.

СУБД, поддерживающие реляционную модель: Dbase IV, Paradox, Visual FoxPro, Oracle, Пальма, HyTech.

Постреляционная модель

Пример

INVOICES (накладные)

INVNO (номер накладной)	CASTNO (номер покупателя)
0373	8723
8374	8232
7364	8723

INVOICES_ITEMS (накладные_товары)

INVNO	GOODS	QTY
0373	Сыр	3
0373	Рыба	2
8374	Лимонад	1
8374	Сок	6
8374	Печенье	2
8364	Йогурт	1

Классическая реляционная модель предполагает неделимость данных хранящихся в полях записей таблицы (1 нормальная форма).

Постреляционная модель снимает ограничения неделимости данных. Она допускает многозначные поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей.

INVOICES

INVNO	CASTNO	GOODS	QTY
0373	8723	Сыр Рыба	3 2
8374	8232	Лимонад Сок Печенье	1 6 2
7364	8723	Йогурт	1

Достоинства

- высокая наглядность представления информации;
- более эффективное хранение данных;
- при обработке данных не требуется выполнять операцию соединения двух таблиц.

Пример: запрос на выборку данных из всех полей БД.

А) для реляционной модели

```
SELECT INVOICES.INVNO, CASTNO, GOODS, QTY
FROM
INVOICES, INVOICES_ITEMS
WHERE
INVOICES.INVNO=INVOICES_ITEMS.INVNO
```

Б) для постреляционной модели

```
SELECT INVNO, CASTNO, GOODS, QTY
FROM
INVOICES
```

Для выполнения постреляционного запроса должны быть реализованы соответствующие механизмы выборки.

Постреляционная модель поддерживает ассоциированные многозначные поля (множественные группы). Совокупность ассоциированных полей называется ассоциацией.

Предприятие	**Количество служащих ж/м**		**Средний возраст ж/м**		**Средняя зарплата ж/м**	
СевНТУ	1300	1000	35	48	290	470
МГУ	300	500	37	42	500	600

В каждой строке первое значение одного столбца ассоциации соответствует первому значению всех столбцов. На длину полей и количество полей в записях не накладывается требование постоянства.

Недостатки:

- сложность решения проблемы обеспечения целостности и непротиворечивости данных. Ограничения целостности накладываются на процедурном уровне. Для описания функций контроля значений в полях создаются специальные процедуры: коды конверсии, коды корреляции, автоматически вызываемые во время обращения к данным.

СУБД, поддерживающие реляционную модель: uniVers, Bubba, Dasdb.

Многомерная модель данных

Существует 2 направления в развитии ИС:

1. системы оперативной или транзакционной обработки информации (весьма эффективны реляционные модели);
2. системы аналитической обработки информации (системы поддержки принятия решений) (эффективны многомерные СУБД).

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

Основные понятия:

1. Агрегируемость данных – возможность рассмотрения информации на разных уровнях ее обобщения.

2. Историчность данных – предполагает обеспечение высокого уровня неизменности данных и их взаимосвязи, а также обязательность привязки данных ко времени.

3. Прогнозируемость данных – подразумевает задание функции прогнозирования и применение ее к различным временным интервалам.

Достоинства

• Многомерная модель обладает большей наглядностью и информативностью, чем информационная модель.

Пример

А) реляционное представление данных

Продажа автомобилей

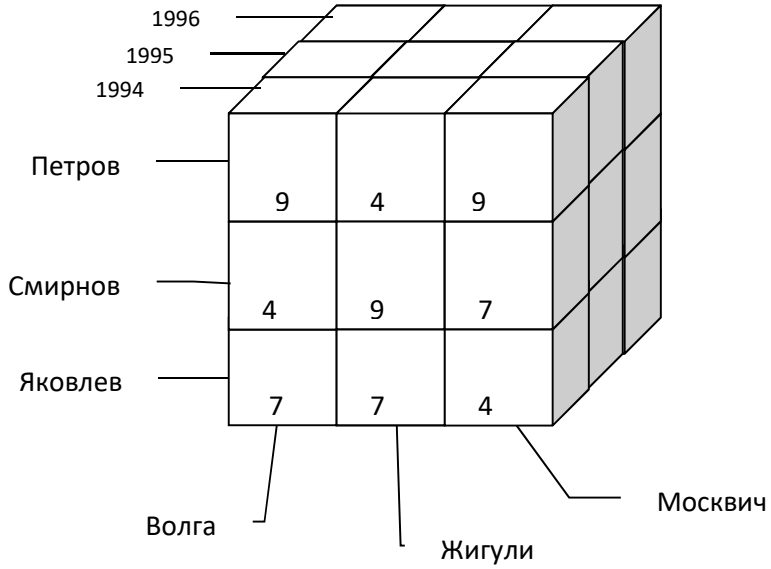
Модель	**Месяц**	**Объем**
«Жигули»	Июнь	12
«Жигули»	Июль	24
«Жигули»	Август	5
«Москвич»	Июнь	2
«Москвич»	Июль	18
«Волга»	Июль	19

Б) Многомерное представление

Модель	**Июнь**	**Июль**	**Август**
«Жигули»	12	24	5
«Москвич»	2	18	№
«Волга»	№	19	№

Измерение (Dimension) – множество однотипных данных, образующих одну из граней многомерного объекта (гиперкуба).

Ячейка гиперкуба (Cell) – поле, значение которого однозначно определяется фиксированным набором измерений (чаще всего числовой тип).



Операции

1. Срез (Slice) – выделение подмножества гиперкуба, полученного в результате фиксации одного или нескольких измерений.
2. Вращение (Rotate) – изменение порядка измерений при визуальном представлении информации.
3. Агрегация – переход к более общему представлению из гиперкуба.
4. Детализация – переход к более детальному представлению из гиперкуба.

В гиперкуб введены измерения и существует иерархия снизу вверх.
Страна

Фирма

Регион

Подразделение

Менеджер

Достоинства

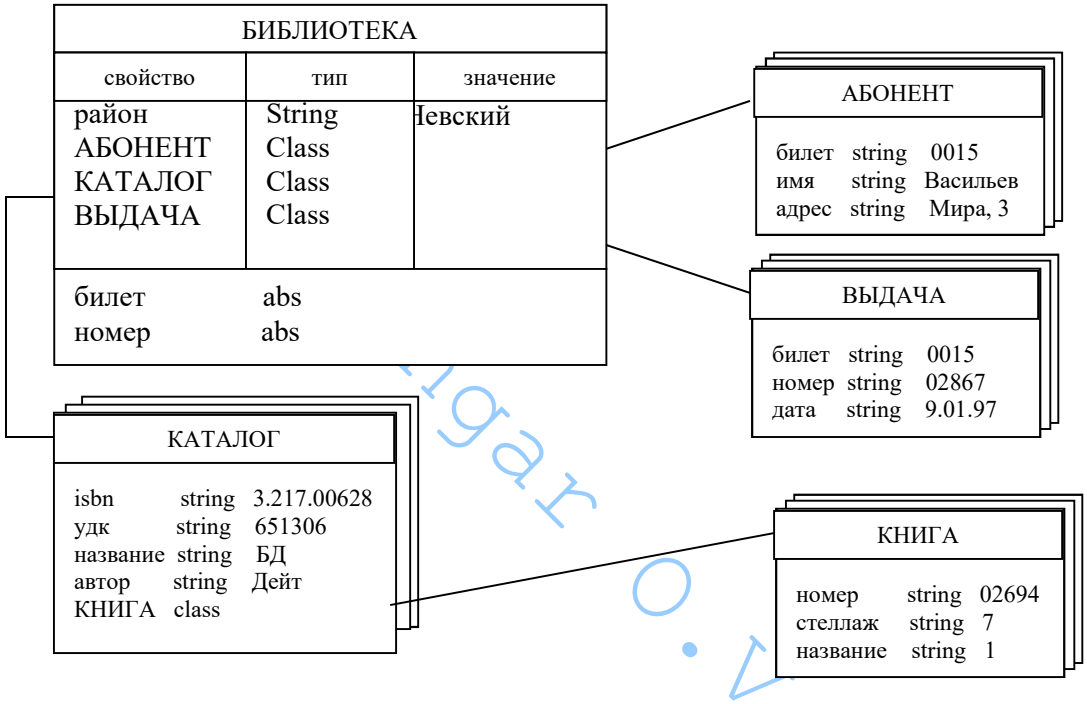
- Удобство и эффективность аналитической обработки больших объемов данных, связанных со временем.

Недостатки

• Громоздкость для простейших задач обычной оперативной обработки информации.

СУБД, поддерживающие многомерные модели: Essbase (Arbor Software), Media Multi-matrix (Speedware), Oracle Express Server, Cache (Inter System).

Объектно-ориентированные модели



1. Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты.
2. Объектом является любой экземпляр любой сущности. Это значит, что имеется возможность идентифицировать отдельные записи БД и определить функции их обработки.
3. Свойства объекта описываются некоторым стандартным типом (например, string) или типом, конструированным пользователем (class).
4. Каждый объект-экземпляр класса является потомком того объекта, в котором он определен как свойство, и наследует свойство родителя.
5. Родовые отношения между объектами в БД образуют связную иерархию объектов и реализуются путем задания соответствующих ссылок. На самом деле в программе свойства «АБОНЕНТ», «КАТАЛОГ», «ВЫДАЧА» объекта «БИБЛИОТЕКА» задаются не как класс, а как ссылка на список объектов соответствующего класса.
6. Логическая основа БД похожа на иерархическую. Основное отличие – в способах манипулирования данными.Для выполнения действий

над данными используются методы того или иного объекта (их логическая ориентация скрыта в описании класса объекта). Методы для работы с объектами класса «КНИГА» могут содержать полиморфный код, если ввести несколько потомков класса «КНИГА» (книга техническая, книга художественная, журнал). Эти классы могут иметь разный набор свойств.

7. Поиск в объектно-ориентированной БД состоит в выяснении сходства между объектом, задаваемым пользователем (цель), и объектами, хранящимися в БД.

Достоинства

- Возможность отображения информации о сложных взаимосвязях объектов (иерархические).
- Обеспечение более высокого уровня абстракции при манипулировании данными за счет объектно-ориентированных механизмов инкапсуляции, наследования, полиморфизма.
- Возможность идентификации отдельных кортежей.

Недостатки

- Высокая понятийная сложность.
- Низкая скорость выполнения запросов.

1.4. Типы данных в БД

Источник: http://site-do.ru/db/sql2.php

MySQL поддерживает несколько типов данных, которые можно условно разделить на четыре категории:

- числовые,
- строковые,
- календарные,
- данные типа NULL.

Рассмотрим их по очереди.

1) Числовой тип данных:
   - *Целое число ()*

     INT (M) или INTEGER (M)
     а также
     TINYINT (M), SMALLINT (M), MEDIUMINT (M), BIGINT (M)

     Может быть объявлено положительным с помощью ключевого слова UNSIGNED, тогда элементам столбца нельзя будет

присвоить отрицательное значение. Необязательный параметр M - количество отводимых под число символов. Необязательный атрибут ZEROFILL позволяет свободные позиции по умолчанию заполнить нулями.

- *Булево число*

  BOOL или BOOLEAN

  Булево значение. 0 - ложь (false), 1 - истина (true)

- *Число повышенной сложности*

  DECIMAL (M,D) или DEC (M,D) или NUMERIC (M,D)

  Используются для величин повышенной точности, например, для денежных данных. M - количество отводимых под число символов (максимальное значение - 64). D - количество знаков после запятой (максимальное значение - 30).

- *Вещественные числа*

  FLOAT (M,D), DOUBLE (M,D)

  Вещественное число (с плавающей точкой). Может иметь параметр UNSIGNED, запрещающий отрицательные числа, но диапазон значений от этого не изменится. M - количество отводимых под число символов. D - количество символов дробной части.

Необходимо понимать, чем больше диапазон значений у типа данных, тем больше памяти он занимает. Поэтому, если предполагается, что значения в столбце не будут превышать 100, то используйте тип TINYINT. Если при этом все значения будут положительными, то используйте атрибут UNSIGNED. Правильный выбор типа данных позволяет сэкономить место для хранения этих данных.

2) Строковый тип данных:

- *строка*

  CHAR (M), VARCHAR (M)

  Позволяет хранить строку фиксированной или переменной длины M.

- *текст*

  TEXT, MEDIUMTEXT, LONGTEXT

  Позволяют хранить большие объемы текста.

- *двоичный объект большого объема*

  BLOB, MEDIUM BLOB, LONG BLOB

  Тип данных BLOB представляет собой двоичный объект большого размера (изображений, звука, электронных документов и т.д.), который может содержать переменное количество данных.

Единственное различие между типами BLOB и TEXT состоит в том, что сортировка и сравнение данных выполняются с учетом регистра для величин BLOB и без учета регистра для величин TEXT. Другими словами, TEXT - это независимый от регистра BLOB.

Если размер задаваемого в столбце BLOB или TEXT значения превосходит максимально допустимую длину столбца, то это значение соответствующим образом усекается.
В столбцах типов BLOB и TEXT не производится удаление концевых символов, как это делается для столбцов типа VARCHAR.
Для столбцов BLOB и TEXT не может быть задан атрибут DEFAULT - значения величин по умолчанию.

- *строка из множества*

  ENUM ('value1', 'value2', ...,'valueN')
  Строки этого типа могут принимать ***только одно*** из значений указанного множества.

  SET ('value1', 'value2', ...,'valueN')
  Строки этого типа могут принимать ***любой или все элементы*** из значений указанного множества.

3) Календарный тип данных:

- *дата*
  DATE
  Предназначен для хранения даты. В качестве первого значения указывается год в формате "YYYY", через дефис - месяц в

формате "MM", а затем день в формате "DD". В качестве разделителя может выступать не только дефис, а любой символ отличный от цифры.

- *время*
  TIME
  Предназначен для хранения времени суток. Значение вводится и хранится в привычном формате - hh:mm:ss, где hh - часы, mm - минуты, ss - секунды. В качестве разделителя может выступать любой символ отличный от цифры.

- *дата, время*

  DATATIME
  Предназначен для хранения и даты и времени суток. Значение вводится и хранится в формате - YYYY-MM-DD hh:mm:ss. В качестве разделителей могут выступать любые символы отличные от цифры.
  TIMESTAMP
  Предназначен для хранения даты и времени суток в виде количества секунд, прошедших с полуночи 1 января 1970 года (начало эпохи UNIX: от '1970-01-01 00:00:00' до '2037-12-31 23:59:59').

- *год*
  YEAR (M)
  Предназначен для хранения года. M - задает формат года. Например, YEAR (2) - 70, а YEAR (4) - 1970. Если параметр M не указан, то по умолчанию считается, что он равен 4.

4) Данные типа NULL:

Вообще-то это лишь условно можно назвать типом данных. По сути это скорее указатель возможности отсутствия значения. Например, когда вы регистрируетесь на каком-либо сайте, вам предлагается заполнить форму, в которой присутствуют, как обязательные, так и необязательные поля. Понятно, что регистрация пользователя невозможна без указания логина и пароля, а вот дату рождения и пол пользователь может указать по желанию. Для того, чтобы хранить такую информацию в БД и используют два значения: NOT NULL (значение не может отсутствовать) для полей логин и пароль, NULL (значение может отсутствовать) для полей дата рождения и пол. По умолчанию всем столбцам присваивается тип NOT NULL, поэтому его можно явно не указывать.