ВВЕДЕНИЕ

В общем смысле термин БД может применить к любой совокупности связанной информации, объединенной вместе по определенному признаку. Например, в качестве БД можно рассматривать расписание движения поездов или книгу регистрации данных о заказах покупателей и их выполнении.

Организация данных в БД, их структура определяется **моделью данных**. Любая модель данных должна содержать три компоненты:

1. ***структура данных*** - описывает точку зрения пользователя на представление данных.
2. ***набор допустимых операций***, выполняемых на структуре данных. Модель данных предполагает, как минимум, наличие языка определения данных (ЯОД), описывающего структуру их хранения, и языка манипулирования данными (ЯМД), включающего операции извлечения и модификации данных.
3. ***ограничения целостности*** - механизм поддержания соответствия данных предметной области на основе формально описанных правил.

В процессе исторического развития использовалось следующие модели данных:

- иерархическая,
- сетевая,
- реляционная,
- объектно-ориентированная.

В последнее время все большее значение приобретает объектно-ориентированный подход к представлению данных.

1. Иерархическая модель данных

1.1. Структура данных

Организация данных иерархического типа определяется в терминах: *элемент, агрегат, запись (группа), групповое отношение, база данных.*

- ***Атрибут (элемент данных)*** - наименьшая единица структуры данных. Обычно каждому элементу при описании базы данных присваивается уникальное имя. По этому имени к нему обращаются при обработке. Элемент данных также часто называют полем.

- ***Запись*** - именованная совокупность атрибутов. Использование записей позволяет за одно обращение к базе получить некоторую логически связанную совокупность данных. Именно записи изменяются, добавляются и удаляются. Тип записи определяется составом ее атрибутов. Экземпляр записи - конкретная запись с конкретным значением элементов

- ***Групповое отношение*** - иерархическое отношение между записями двух типов. Родительская запись (владелец группового отношения) называется исходной записью, а

дочерние записи (члены группового отношения) - подчиненными. Иерархическая база данных может хранить только такие древовидные структуры.

Примером такой древовидной структуры (дерева) может служить следующий рисунок, описывающий структуру университета.

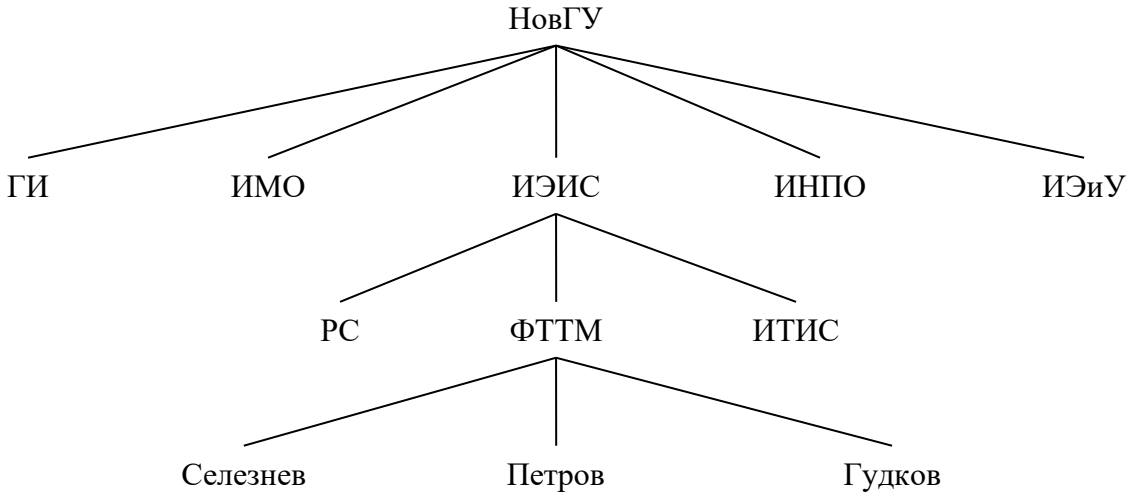

Рисунок 1 – Иерархическая структура университета

Корневая запись каждого дерева обязательно должна содержать ключ с уникальным значением. Ключи некорневых записей должны иметь уникальное значение только в рамках группового отношения. Каждая запись идентифицируется полным сцепленным ключом, под которым понимается совокупность ключей всех записей от корневой по иерархическому пути.

При графическом изображении групповые отношения изображают дугами ориентированного графа, а типы записей - вершинами (диаграмма Бахмана).

Для групповых отношений в иерархической модели обеспечивается *автоматический режим включения* и *фиксированное членство*. Это означает, что для запоминания любой некорневой записи в БД должна существовать ее родительская запись. При удалении родительской записи автоматически удаляются все подчиненные.

Пример:
*Рассмотрим следующую модель данных предприятия (см. **рис.2**): предприятие состоит из отделов, в которых работают сотрудники. В каждом отделе может работать несколько сотрудников, но сотрудник не может работать более чем в одном отделе.*
*Поэтому, для информационной системы управления персоналом необходимо создать групповое отношение, состоящее из родительской записи ОТДЕЛ (НАИМЕНОВАНИЕ_ОТДЕЛА, ЧИСЛО_РАБОТНИКОВ) и дочерней записи СОТРУДНИК (ФАМИЛИЯ, ДОЛЖНОСТЬ, ОКЛАД). Это отношение показано на рис. **(a)** (Для простоты полагается, что имеются только две дочерние записи).*
*Для автоматизации учета контрактов с заказчиками необходимо создание еще одной иерархической структуры : заказчик - контракты с ним - сотрудники, задействованные в работе над контрактом. Это дерево будет включать записи ЗАКАЗЧИК(НАИМЕНОВАНИЕ_ЗАКАЗЧИКА, АДРЕС), КОНТРАКТ(НОМЕР, ДАТА,СУММА), ИСПОЛНИТЕЛЬ (ФАМИЛИЯ, ДОЛЖНОСТЬ, НАИМЕНОВАНИЕ_ОТДЕЛА) (рис. **(b)**).*

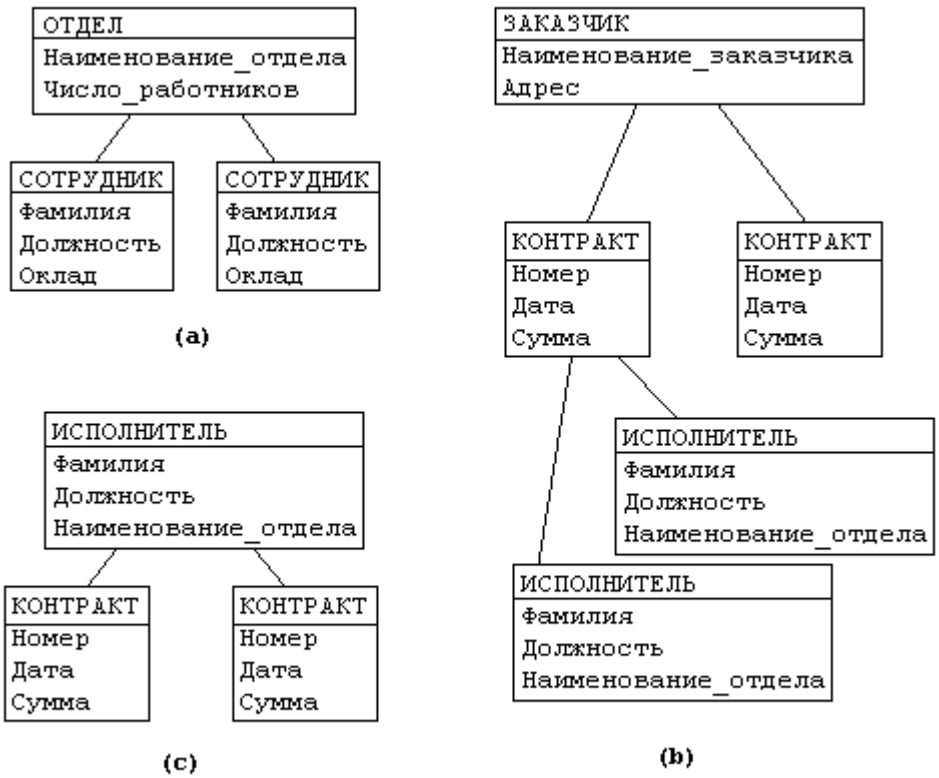

Рис. 2

Из этого примера видны недостатки иерархических БД:

- Частично дублируется информация между записями СОТРУДНИК и ИСПОЛНИТЕЛЬ (такие записи называют парными), причем в иерархической модели данных не предусмотрена поддержка соответствия между парными записями.
- Иерархическая модель реализует отношение между исходной и дочерней записью по схеме **1:N**, то есть одной родительской записи может соответствовать любое число дочерних. Допустим теперь, что исполнитель может принимать участие более чем в одном контракте (т.е. возникает связь типа **M:N**). В этом случае в базу данных необходимо ввести еще одно групповое отношение, в котором ИСПОЛНИТЕЛЬ будет являться исходной записью, а КОНТРАКТ - дочерней (рис. **(c)**). Таким образом, мы опять вынуждены дублировать информацию.

1.2. Операции над данными, определенные в иерархической модели:

- **ДОБАВИТЬ** в базу данных новую запись. Для корневой записи обязательно формирование значения ключа.
- **ИЗМЕНИТЬ** значение данных предварительно извлеченной записи. Ключевые данные не должны подвергаться изменениям.
- **УДАЛИТЬ** некоторую запись и все подчиненные ей записи.
- **ИЗВЛЕЧЬ**:

- извлечь корневую запись по ключевому значению, допускается также последовательный просмотр корневых записей
- извлечь следующую запись (следующая запись извлекается в порядке левостороннего обхода дерева)

В операции ИЗВЛЕЧЬ допускается задание условий выборки (например, извлечь сотрудников с окладом более 1 тысячи руб.)

Как видим, все операции изменения применяются только к одной "текущей" записи (которая предварительно извлечена из базы данных). Такой подход к манипулированию данных получил название "навигационного".

1.3. Ограничения целостности

Поддерживается только целостность связей между владельцами и членами группового отношения (никакой потомок не может существовать без предка). Как уже отмечалось, не обеспечивается автоматическое поддержание соответствия парных записей, входящих в различные иерархии.

2.Сетевая модель данных

2.1.Структура данных

Сетевая модель данных определяется в тех же терминах, что и иерархическая. Она состоит из множества записей, которые могут быть владельцами или членами групповых отношений. Связь между записью-владельцем и записью-членом также имеет вид **1:N**.

Основное различие этих моделей состоит в том, что в сетевой модели запись может быть членом *более чем одного* группового отношения. Согласно этой модели, каждое групповое отношение именуется и проводится различие между его типом и экземпляром. Тип группового отношения задается его именем и определяет свойства общие для всех экземпляров данного типа. Экземпляр группового отношения представляется записью-владельцем и множеством (возможно пустым) подчиненных записей. При этом имеется следующее ограничение: экземпляр записи не может быть членом двух экземпляров групповых отношений одного типа (т.е. сотрудник из примера в п.1, например, не может работать в двух отделах).

Иерархическая структура из п.1. преобразовывается в сетевую следующим образом (см. рис. 3):

- деревья (a) и (b), показанные на рис. 2, заменяются одной сетевой структурой, в которой запись СОТРУДНИК входит в два групповых отношения;
- для отображения связи типа M:N вводится запись СОТРУДНИК_КОНТРАКТ, которая не имеет полей и служит только для связи записей КОНТРАКТ и СОТРУДНИК, см. рис. 3.(Отметим, что в этой записи может храниться и полезная информация, например, доля данного сотрудника в общем вознаграждении по данному контракту.)

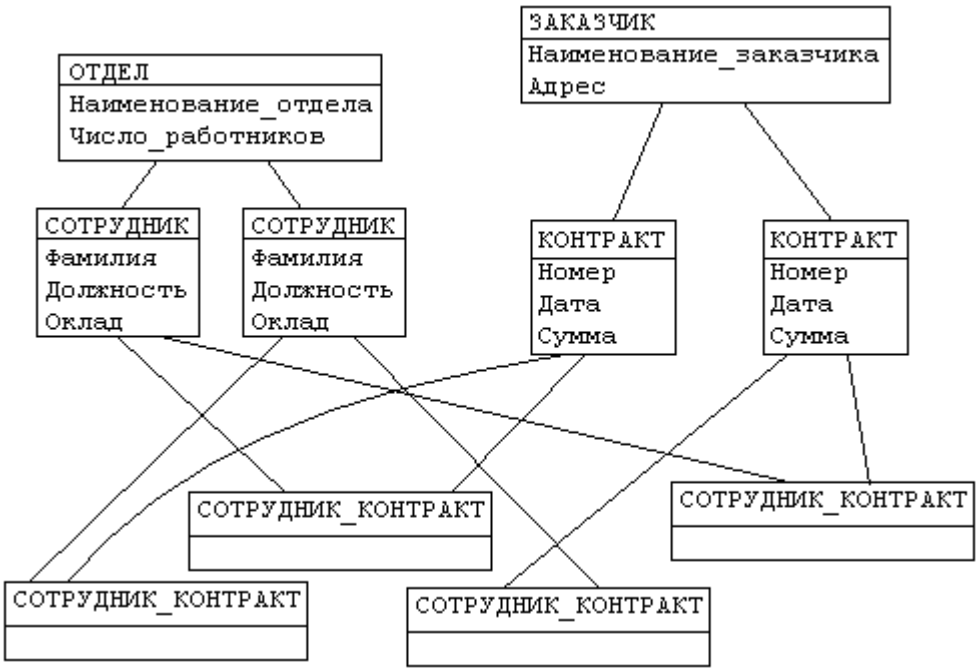

Рис. 3

Каждый экземпляр группового отношения характеризуется следующими признаками:

- **способ упорядочения подчиненных записей**:
  - произвольный,
  - хронологический /очередь/,
  - обратный хронологический /стек/,
  - сортированный.

  Если запись объявлена подчиненной в нескольких групповых отношениях, то в каждом из них может быть назначен свой способ упорядочивания.

- **режим включения подчиненных записей**:
  - автоматический - невозможно занести в БД запись без того, чтобы она была сразу же закреплена за неким владельцем;
  - ручной - позволяет запомнить в БД подчиненную запись и не включать ее немедленно в экземпляр группового отношения. Эта операция позже инициируется пользователем).
- **режим исключения** Принято выделять три класса членства подчиненных записей в групповых отношениях:
  1. *Фиксированное.* Подчиненная запись жестко связана с записью владельцем и ее можно исключить из группового отношения только удалив. При удалении записи-владельца все подчиненные записи автоматически тоже удаляются. В рассмотренном выше примере фиксированное членство предполагает групповое отношение "ЗАКЛЮЧАЕТ" между записями "КОНТРАКТ" и "ЗАКАЗЧИК", поскольку контракт не может существовать без заказчика.
  2. *Обязательное.* Допускается переключение подчиненной записи на другого владельца, но невозможно ее существование без владельца. Для удаления записи-владельца необходимо, чтобы она не имела подчиненных записей с обязательным

членством. Таким отношением связаны записи "СОТРУДНИК" и "ОТДЕЛ". Если отдел расформировывается, все его сотрудники должны быть либо переведены в другие отделы, либо уволены.

3. *Необязательное.* Можно исключить запись из группового отношения, но сохранить ее в базе данных, не прикрепляя к другому владельцу. При удалении записи-владельца ее подчиненные записи - необязательные члены сохраняются в базе, не участвуя более в групповом отношении такого типа. Примером такого группового отношения может служить "ВЫПОЛНЯЕТ" между "СОТРУДНИКИ" и "КОНТРАКТ", поскольку в организации могут существовать работники, чья деятельность не связана с выполнением каких-либо договорных обязательств перед заказчиками.

2.2. Операции над данными

- **ДОБАВИТЬ** - внести запись в БД и, в зависимости от режима включения, либо включить ее в групповое отношение, где она объявлена подчиненной, либо не включать ни в какое групповое отношение.
- **ВКЛЮЧИТЬ В ГРУППОВОЕ ОТНОШЕНИЕ** - связать существующую подчиненную запись с записью-владельцем.
- **ПЕРЕКЛЮЧИТЬ** - связать существующую подчиненную запись с другой записью-владельцем в том же групповом отношении.
- **ОБНОВИТЬ** - изменить значение элементов предварительно извлеченной записи.
- **ИЗВЛЕЧЬ** - извлечь записи последовательно по значению ключа, а также используя групповые отношения - от владельца можно перейти к записям - членам, а от подчиненной записи к владельцу набора.
- **УДАЛИТЬ** - убрать из БД запись. Если эта запись является владельцем группового отношения, то анализируется класс членства подчиненных записей. Обязательные члены должны быть предварительно исключены из группового отношения, фиксированные удалены вместе с владельцем, необязательные останутся в БД.
  ИСКЛЮЧИТЬ ИЗ ГРУППОВОГО ОТНОШЕНИЯ - разорвать связь между записью-владельцем и записью-членом.

2.3. Ограничения целостности

Как и в иерархической модели обеспечивается только поддержание целостности по ссылкам (владелец отношения - член отношения).

3. Реляционная модель данных

Реляционные БД в настоящее время являются фактически промышленным стандартом, поэтому их мы будем рассматривать подробно.

Классическая реляционная модель данных требует, чтобы данные хранились в так называемых плоских таблицах. Более точно, пользователи и приложения, обращающиеся к данным, должны работать с данными так, как если бы они размещались в таких таблицах. В упрощенном виде плоская таблица - это таблица, каждая ячейка которой может быть однозначно

идентифицирована указанием строки и столбца таблицы. Кроме того, в одном столбце все ячейки должны содержать данные одного простого типа.

Реляционная модель основана на теории множеств и математической логике. Такой фундамент обеспечивает математическую строгость реляционной модели данных.

В свою очередь, на основе реляционной модели были разработаны различные языки для доступа к реляционным данным, такие как SEQUEL, SQL, QUEL и другие. Фактическим промышленным стандартом в настоящее время стал язык SQL (Structured Query Language - язык структурированных запросов).

Различные фирмы, производители СУБД, предлагают свои реализации языка SQL. Эти реализации отличаются как друг от друга, так и от стандартизованного языка SQL. Это и хорошо, и плохо. Хорошо это тем, что конкретная реализация языка, может включать в себя более широкие возможности по сравнению со стандартизованными SQL, например, больше типов данных, большее количество команд, больше дополнительных опций имеющихся команд. Такие возможности делают работу с конкретной СУБД более эффективной. Кроме того, такие нестандартные возможности языка проходят практическую апробацию и со временем могут быть включены в стандарт. Плохо же это тем, что различия в синтаксисе реализаций SQL затрудняют перенос приложений из одной системы в другую. Например, если приложение было написано для базы данных MS SQL Server с использованием своего диалекта SQL - языка Transact-SQL, то при переносе системы в базу данных ORACLE, не все конструкции языка будут понятны соответствующему диалекту SQL - языку PL/SQL.

Взаимосвязь реляционной модели данных, стандарта языка SQL и различных его реализаций можно условно изобразить в виде следующей пирамиды:

Каждый более высокий уровень основывается на понятиях, определенных на более низком уровне. На каждом из уровней используется своя терминология. Например, на уровне теории множеств мы говорим "множество", "подмножество декартового произведения", "кортеж". На уровне реляционной модели используем термины "домен", "отношение", "кортеж". На уровне стандарта SQL и конкретных реализаций используем термины "тип данных", "таблица", "строка таблицы". И каждый раз речь идет об одном и том же.

Начнем изучение БД с основ теории множеств, необходимых для введения фундаментального понятия "отношение".