Тема 2. Технология хранения, обработки и анализа данных.

Базы данных

- База данных - это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.
- SQL - это язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных, а также контроля доступа

- Базы данных значительно изменились с момента их появления в начале 1960-х годов. Исходными системами, которые использовались для хранения и обработки данных, были навигационные базы данных – например, иерархические базы данных (которые опирались на древовидную модель и допускали только отношение «один-ко-многим») и базы данных с сетевой структурой (более гибкая модель, допускающая множественные отноше
- Несмотря на простоту, эти ранние системы были негибкими. В 1980-х годах стали популярными реляционные базы данных, в 1990-х годах за ними последовали объектно-ориентированные базы данных. Совсем недавно вследствие роста Интернета и возникновения необходимости анализа неструктурированных данных появились базы данных NoSQL. В настоящее время облачные базы данных и автономные базы данных открывают новые возможности в отношении способов сбора, хранения, использования данных и управления ими.

- Существует множество различных типов баз данных. Выбор наилучшей базы данных для конкретной компании зависит от того, как она намеревается использовать данные.
- Реляционные базы данных

Реляционные базы данных стали преобладать в 1980-х годах. Данные в реляционной базе организованы в виде таблиц, состоящих из столбцов и строк. Реляционная СУБД обеспечивает быстрый и эффективный доступ к структурированной информации.

- Объектно-ориентированные базы данных

Информация в объектно-ориентированной базе данных представлена в форме объекта, как в объектно-ориентированном программировании.

- Распределенные базы данных

Распределенная база данных состоит из двух или более частей, расположенных на разных серверах. Такая база данных может храниться на нескольких компьютерах.

- Хранилища данных

Будучи централизованным репозиторием для данных, хранилище данных представляет собой тип базы данных, специально предназначенной для быстрого выполнения запросов и анализа.

- Графовые базы данных

Графовая база данных хранит данные в контексте сущностей и связей между сущностями.

- Oracle NoSQL Database

База данных NoSQL, или нереляционная база данных, дает возможность хранить и обрабатывать неструктурированные или слабоструктурированные данные (в отличие от реляционной базы данных, задающей структуру содержащихся в ней данных). Популярность баз данных NoSQL растет по мере распространения и усложнения веб-приложений.

- Базы данных OLTP. База данных OLTP - это база данных предназначенная для выполнения бизнес-транзакций, выполняемых множеством пользователей.
- Это лишь некоторые из десятков типов баз данных, используемых в настоящее время. Другие, менее распространенные базы данных, предназначены для очень специфических научных, финансовых и иных задач. Помимо появления новых типов, базы данных развиваются в абсолютно новых направлениях - изменяются подходы к разработке технологий, происходят значительные сдвиги, такие как внедрение облачных технологий и автоматизации. В частности, в последнее время появились следующие базы данных.

- Базы данных с открытым исходным кодом

Такие базы данных имеют открытый исходный код и могут управляться средствами как SQL, так и NoSQL.

- Облачные базы данных

Облачная база данных представляет собой набор структурированных или неструктурированных данных, размещенный на частной, публичной или гибридной платформе облачных вычислений. Существует два типа моделей облачных баз данных: традиционная база данных и база данных как услуга (DBaaS). В модели DBaaS административные задачи и обслуживание выполняются поставщиком облачных услуг.

- Многомодельные базы данных
- Многомодельная база данных объединяет разные типы моделей баз данных в единую интегрированную серверную СУБД. Это означает, что она может содержать различные типы данных.
- Документные базы данных/JSON
- Базы данных документов предназначены для хранения, извлечения и обработки документоориентированной информации и предоставляют современный способ хранения данных в формате JSON, а не в виде строк и столбцов.

- Автономные базы данных

Самоуправляемые базы данных (также называемые автономными) - это новейшие и самые революционные облачные базы данных, которые используют машинное обучение для автоматизации настройки, защиты, резервного копирования, обновления и других стандартных задач обслуживания, обычно выполняемых администраторами баз данных.

- В качестве примеров популярного программного обеспечения для управления базами данных, или СУБД, можно назвать MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, СУБД Oracle Database и dBASE.
- MySQL — это реляционная система управления базами данных с открытым исходным кодом на основе языка SQL. Она была разработана и оптимизирована для веб-приложений и может работать на многих платформах. Она обладает всеми возможностями которые требуются веб-разработчикам. База данных MySQL предназначена для обработки миллионов запросов и тысяч транзакций, поэтому ее часто выбирают компании электронной коммерции, которым требуется управлять большим количеством денежных переводов. Гибкость по мере необходимости — основная характеристика MySQL.

- Автономная база данных способна значительно расширить эти возможности. Автономные базы данных автоматизируют дорогостоящие и длительные ручные процедуры, благодаря чему бизнес-пользователи могут сосредоточиться на работе со своими данными. За счет возможностей создания и использования баз данных пользователи приобретают контроль и автономию, поддерживая при этом важные стандарты безопасности.
- Автономные базы данных — это модель будущего, представляющая исключительный интерес для компаний, которые хотят использовать лучшую из имеющихся технологий баз данных, при этом не сталкиваясь с проблемами при запуске и эксплуатации этой технологии.

- Автономные базы данных используют облачные технологии и машинное обучение для автоматизации множества стандартных задач управления базами данных, таких как настройка, защита, резервное копирование, обновление и другие повседневные задачи администрирования. Благодаря автоматизации этой рутины администраторы баз данных могут сосредоточиться на более стратегической работе. Возможности самоуправления, самозащиты и самовосстановления автономных баз данных могут радикально изменить способы управления и защиты данных, улучшая эффективность, снижая затраты и повышая безопасность.

основные модели базы данных

- Различают три основные модели базы данных – это иерархическая, сетевая и реляционная. Эти модели отличаются между собой по способу установления связей между данными.
- **Иерархический подход к организации баз данных.** Иерархические базы данных имеют форму деревьев с дугами-связями и узлами-элементами данных. Иерархическая структура предполагала неравноправие между данными – одни жестко подчинены другим. Подобные структуры, безусловно, четко удовлетворяют требованиям многих, но далеко не всех реальных задач.

- **Сетевая модель данных.** В сетевых БД наряду с вертикальными реализованы и горизонтальные связи. Однако унаследованы многие недостатки иерархической и главный из них, необходимость четко определять на физическом уровне связи данных и столь же четко следовать этой структуре связей при запросах к базе.
- **Реляционная модель данных.** Реляционная модель появилась вследствие стремления сделать базу данных как можно более гибкой. Данная модель предоставила простой и эффективный механизм поддержания связей данных.

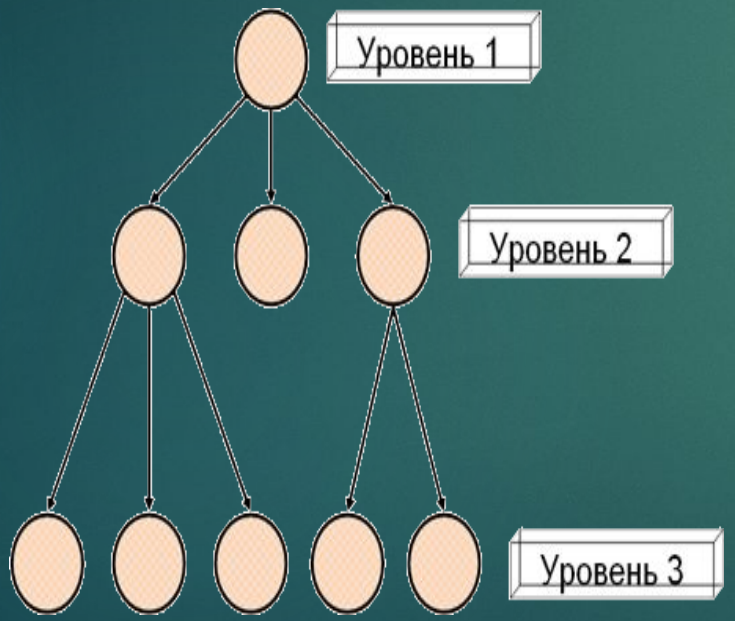
- **Объектно-ориентированная модель.** Новые области использования вычислительной техники, такие как научные исследования, автоматизированное проектирование и автоматизация учреждений, потребовали от баз данных способности хранить и обрабатывать новые объекты – текст, аудио- и видеоинформацию, а также документы. Основные трудности объектно-ориентированного моделирования данных проистекают из того, что такого развитого математического аппарата, на который могла бы опираться общая объектно-ориентированная модель данных, не существует. В большой степени, поэтому до сих пор нет базовой объектно-ориентированной модели. С другой стороны, некоторые авторы утверждают, что общая объектно-ориентированная модель данных в классическом смысле и не может быть определена по причине непригодности классического понятия модели данных к парадигме объектной ориентированности. Несмотря на преимущества объектно-ориентированных систем – реализация сложных типов данных, связь с языками программирования и т.п. – на ближайшее время превосходство реляционных СУБД гарантировано.

Реляционная модель данных

- Первый элемент реляционной модели требует от реляционной модели поддержания некоторых ограничений целостности. Одно из таких ограничений утверждает, что каждая строка в таблице должна иметь некий уникальный идентификатор, называемый первичным ключом. Второе ограничение накладывается на целостность ссылок между таблицами. Оно утверждает, что атрибуты таблицы, ссылающиеся на первичные ключи других таблиц, должны иметь одно из значений этих первичных ключей.

- Во-вторых, все данные в модели представляются в виде таблиц и только таблиц. Реляционная модель – единственная из всех обеспечивает единообразие представления данных. И сущности, и связи этих самых сущностей представляются в модели совершенно одинаково – таблицами. Правда, такой подход усложняет понимание смысла хранящейся в базе данных информации, и, как следствие, манипулирование этой информацией.
- Избежать трудностей манипулирования позволяет третий элемент модели – реляционно-полный язык. Полнота языка в приложении к реляционной модели означает, что он должен выполнять любую операцию реляционной алгебры или реляционного исчисления. Более того, язык должен описывать любой запрос в виде операций с таблицами, а не с их строками. Одним из таких языков является SQL.

Иерархическая модель базы данных



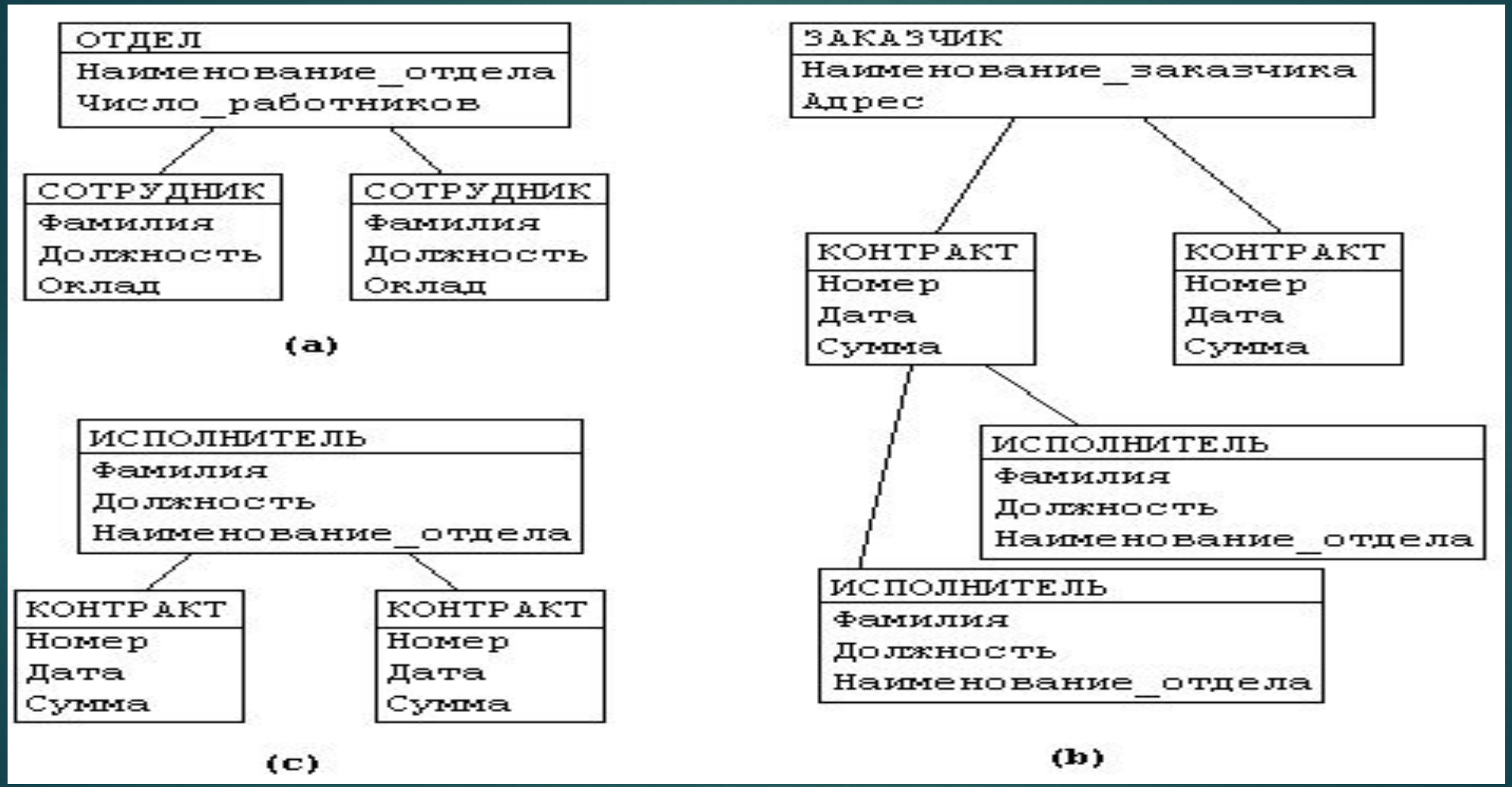
- Иерархические базы данных — самая ранняя модель представления сложной структуры данных. Информация в иерархической базе организована по принципу древовидной структуры, в виде отношений «предок-потомок». Каждая запись может иметь не более одной родительской записи и несколько подчиненных. Связи записей реализуются в виде физических указателей с одной записи на другую. Основной недостаток иерархической структуры базы данных — невозможность реализовать отношения «много-ко-многим», а также ситуации, когда запись имеет несколько предков.

- Иерархические базы данных графически могут быть представлены как перевернутое дерево, состоящее из объектов различных уровней. Верхний уровень (корень дерева) занимает один объект, второй - объекты второго уровня и так далее.
- Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект, более близкий к корню) к потомку (объект более низкого уровня), при этом объект-предок может не иметь потомков или иметь их несколько, тогда как объект-потомок обязательно имеет только одного предка. Объекты, имеющие общего предка, называются близнецами.

Термины

- Организация данных в СУБД 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.
- **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).


ОТДЕЛ
Наименование_отдела
Число_работников
СОТРУДНИК
Фамилия
Должность
Оклад
СОТРУДНИК
Фамилия
Должность
Оклад
(a)
ЗАКАЗЧИК
Наименование_заказчика
Адрес
КОНТРАКТ
Номер
Дата
Сумма
КОНТРАКТ
Номер
Дата
Сумма
ИСПОЛНИТЕЛЬ
Фамилия
Должность
Наименование_отдела
ИСПОЛНИТЕЛЬ
Фамилия
Должность
Наименование_отдела
(b)
ИСПОЛНИТЕЛЬ
Фамилия
Должность
Наименование_отдела
КОНТРАКТ
Номер
Дата
Сумма
КОНТРАКТ
Номер
Дата
Сумма
(c)

недостатки иерархических БД

- 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 (рисунок (с)). Таким образом, мы опять вынуждены дублировать информацию.

Операции над данными

- 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 10 тысяч 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".

Сетевая модель базы данных.

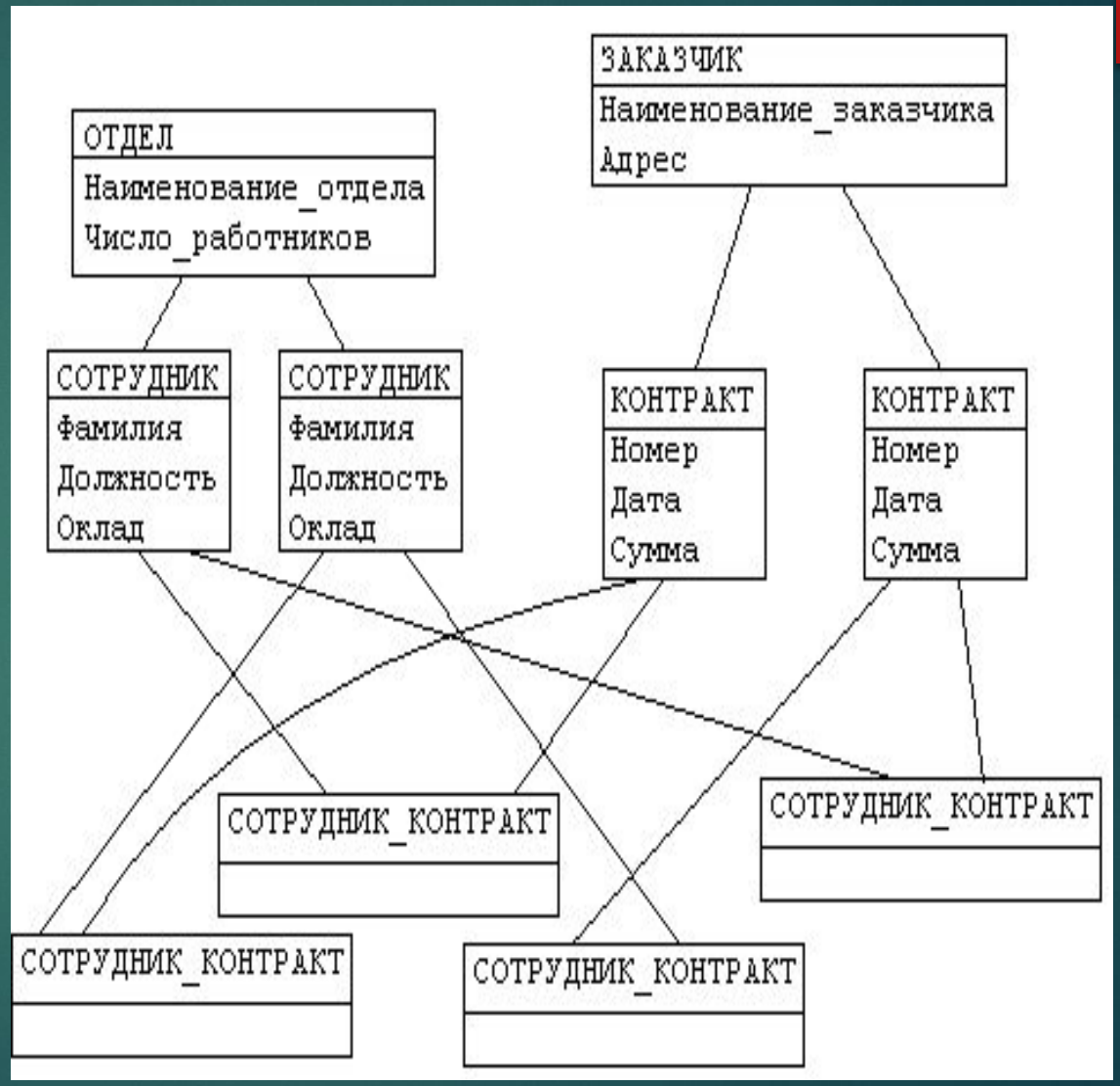
- На разработку этого стандарта большое влияние оказал американский ученый Ч.Бахман. Основные принципы сетевой модели данных были разработаны в середине 60-х годов, эталонный вариант сетевой модели данных описан в отчетах рабочей группы по языкам баз данных (COnference on DAta SYstem Languages) CODASYL (1971 г.).
- 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, например, не может работать в двух отделах).

Иерархическая структура преобразовывается в сетевую следующим образом:

деревья (a) и (b), показанные на рис. 4.2, заменяются одной сетевой структурой, в которой запись СОТРУДНИК входит в два групповых отношения;

для отображения типа M:N вводится запись СОТРУДНИК_КОНТРАКТ, которая не имеет полей и служит только для связи записей КОНТРАКТ и СОТРУДНИК, см. рис. 4.3 (Отметим, что в этой записи может храниться и полезная информация, например, доля данного сотрудника в общем вознаграждении по данному контракту)
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.

- **Классы записей**
- Принято выделять три класса членства подчиненных записей в групповых отношениях:
- 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.

- 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.
- 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.

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.

Достоинства и недостатки ранних субд

- Достоинства 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 ранних СУБД
- сложность использования;
- высокий уровень требований к знаниям о физической организации БД;
- зависимость прикладных систем от физической организации БД;
- перегруженность логики прикладных систем деталями организации доступа к БД.

- Как иерархическая, так и сетевая модель данных предполагает наличие высококвалифицированных программистов. И даже в таких случаях реализация пользовательских запросов часто затягивается на длительный срок.
- Появление объектно-ориентированных СУБД вызвано потребностями программистов на ОО-языках, которым были необходимы средства для хранения объектов, не помещавшихся в оперативной памяти компьютера. Также важна была задача сохранения состояния объектов между повторными запусками прикладной программы. Поэтому, большинство ООСУБД представляют собой библиотеку, процедуры управления данными которой включаются в прикладную программу. Примеры реализации ООСУБД как выделеного сервера базы данных крайне редки.
- Сразу же необходимо заметить, что общепринятого определения "объектно-ориентированной модели данных" не существует. Сейчас можно говорить лишь о неком "объектном" подходе к логическому представлению данных и о различных объектно-ориентированных способах его реализации.