Базы данных

Структуры данных
Системы управления базами данных
Режимы работы с базами данных
Чтение фрагментов базы данных

Структуры данных

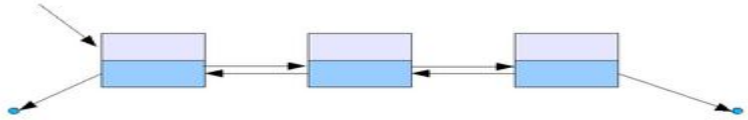
Решая конкретную задачу, необходимо выбрать множество данных, представляющих реальную ситуацию. Важную роль здесь играют свойства самих данных и операции, которые должны выполняться над ними. Обычно данные делят на простые – *неструктурированные* – (основной признак: одно имя – одно значение) и *структурированные.*

Структурированные типы данных классифицируют по следующим основным признакам:

- *По однородности* (все элементы структуры однотипны – однородная структура),
- *По упорядоченности* (между элементами определен порядок следования – упорядоченная структура),
- *По способу доступа* – прямого (каждый элемент структуры доступен пользователю в любой момент времени независимо от других элементов) и последовательного доступа (доступ к элементу возможен только после извлечения предыдущих),
- *По динамичности* – статические (с фиксированным размером данных) и динамические (размер данных устанавливается по ходу решения задачи).

Типовые структуры данных

- *Массив* – однородная упорядоченная статическая структура прямого доступа, однородный набор величин одного и того же типа (*компонентов* массива), объединенных одним общим именем *(идентификатором*) и идентифицируемых (адресуемых) вычисляемым *индексом*.
- *Стек* — структура данных с порядком доступа к элементам «последним пришёл - первым вышел» (LIFO, Last In - First Out). Добавление элемента возможно только в вершину стека (добавленный элемент становится первым в стеке), удаление - также только из вершины стека.
- *Свя́занный спи́сок* —структура данных, в котором объекты расположены в линейном порядке, порядок определяется указателями на каждый объект. Различают *односвязный* список**,** в котором можно передвигаться только в сторону конца списка (узнать адрес предыдущего элемента невозможно) и *двусвязный* список, по которому можно передвигаться в любом направлении — как к началу, так и к концу. В этом списке проще производить удаление и перестановку элементов, т.к. всегда известны адреса тех элементов списка, указатели которых направлены на изменяемый элемент.



- *Очередь* — структура данных с порядком доступа к элементам "первый пришел - первый вышел" (FIFO, First In — First Out). Добавление элемента возможно лишь в конец очереди, выборка - только из начала очереди.

Системы управления базами данных

Хранение информации – одна из важнейших функций компьютера. Одним из распространенных средств такого хранения являются базы данных.

База данных (БД) – совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

Создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляется централизованно с помощью специального программного инструментария – системы управления базами данных.

Система управления базами данных (СУБД) – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

Основные функции СУБД :

- управление данными во внешней памяти;
- управление буферами оперативной памяти;
- управление транзакциями;

(Транзакция - в информатике - совокупность операций над данными, которая, с точки зрения обработки данных, либо выполняется полностью, либо совсем не выполняется.

Транзакция - в информационных системах - последовательность логически связанных действий, переводящих информационную систему из одного состояния в другое. Транзакция либо должна завершиться полностью, либо система должна быть возвращена в исходное состояние).

- журнализация и восстановление БД после сбоев;
- поддержание языков БД.

Состав реляционной СУБД

Логически в современной реляционной СУБД можно выделить:

- *ядро* СУБД (т.н. Data Base Engine); его основными логическими компонентами являются: менеджер данных, менеджер буферов, менеджер транзакций и менеджер журнала;
- *компилятор* языка БД (обычно SQL);
- подсистему поддержки *времени выполнения*;
- набор *утилит*.

Язык SQL - это основное средство общения с реляционными базами данных.

SQL (Structured Query Language) - структурированный язык запросов.

SQL состоит из трех компонентов:

DDL (Язык Определения Данных) – Язык Описания Схемы, состоит из команд, которые создают объекты (таблицы, индексы, и так далее) в базе данных.

DML (Язык Манипулирования Данными) – это набор команд, которые определяют, какие значения представлены в таблицах в любой момент времени.

DCD (Язык Управления Данными) состоит из средств, которые определяют, разрешить ли пользователю выполнять определенные действия или нет

Основные понятия БД

- *Концептуальная модель* БД описывает сущности, их свойства и связи между ними; не зависит от конкретной СУБД.
- *Сущность* (entity) – это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна. В диаграммах сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа. Примеры сущностей: ФАКУЛЬТЕТ ГРУППА СТУДЕНТ

Каждый экземпляр сущности (объект) должен быть отличим от любого другого экземпляра той же сущности. Пример экземпляров сущности ФАКУЛЬТЕТ: ПС, ФМ, АТ и т.п., сущности СТУДЕНТ: Иванов А.П., Петрова Н.Н. и т.п.

- *Связь* (relationship) – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). Возможны связи на основе отношений один-к-одному, один-ко-многим, многие-ко-многим.

Связь «содержит»: ГРУППА содержит много СТУДЕНТОВ. Каждый СТУДЕНТ входит только в одну ГРУППУ

Свойства сущностей

Сущности имеют свойства, которые называются *атрибутами* (attribute).

Например, атрибуты

сущности ФАКУЛЬТЕТ: название, год создания;

сущности ГРУППА: номер;

сущности СТУДЕНТ: фамилия, имя, отчество, номер студенческого билета, номер паспорта, год рождения, месяц рождения, день рождения.

Любой атрибут принимает значения из некоторого множества допустимых значений, называемого *доменом атрибута*.

Например,

домен атрибута «год создания»: целые положительные числа;

домен атрибута «имя»: строка, не содержащая пробелов;

домен атрибута «год рождения»: целые положительные числа.

домен атрибута «месяц рождения»: январь, февраль, март … декабрь.

домен атрибута «день рождения»: целые числа от 1 до 31.

Ключ сущности

*Ключ сущности (*entity key), *первичный ключ* - это атрибут (или множество атрибутов) уникальным образом идентифицирующих экземпляр сущности (объект). Например, ключ сущности СТУДЕНТ – номер студенческого билета, ключ ФАКУЛЬТЕТА – название.

Если ключ состоит из одного атрибута, его называют *простым* ключом.

Если ключ сущности состоит из нескольких атрибутов, его называют *составным* ключом. Например, для сущности ДОМ с атрибутами «улица», «этажность», «год постройки», «номер дома», первичным ключом будет «улица»+ «номер дома»

Виды моделей данных

Организация данных рассматривается с позиций той или иной модели данных.

Модель данных является ядром любой базы данных.

С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных — совокупность структур данных, ограничений целостности и операций манипулирования данными.

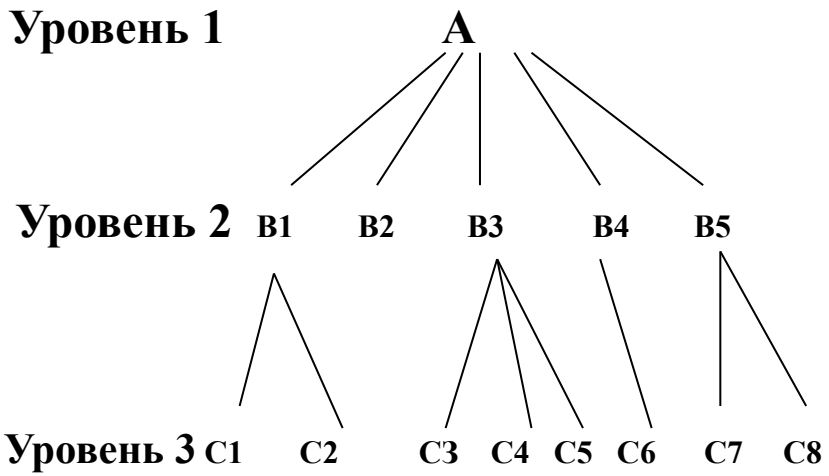
Модели используются для представления данных в информационных системах.

Различают три типа моделей данных, которые имеют множества допустимых информационных конструкций:

- *иерархическая ,*
- *сетевая,*
- *реляционная.*

Иерархическая модель данных

- Иерархическая структура представляет совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево), вид которого представлен на рис



Основные понятия иерархической структуры

Это – узел, уровень и связь.

- *Узел* — это совокупность атрибутов данных, описывающих некоторый объект.

На схеме иерархического дерева узлы представляются вершинами графа.

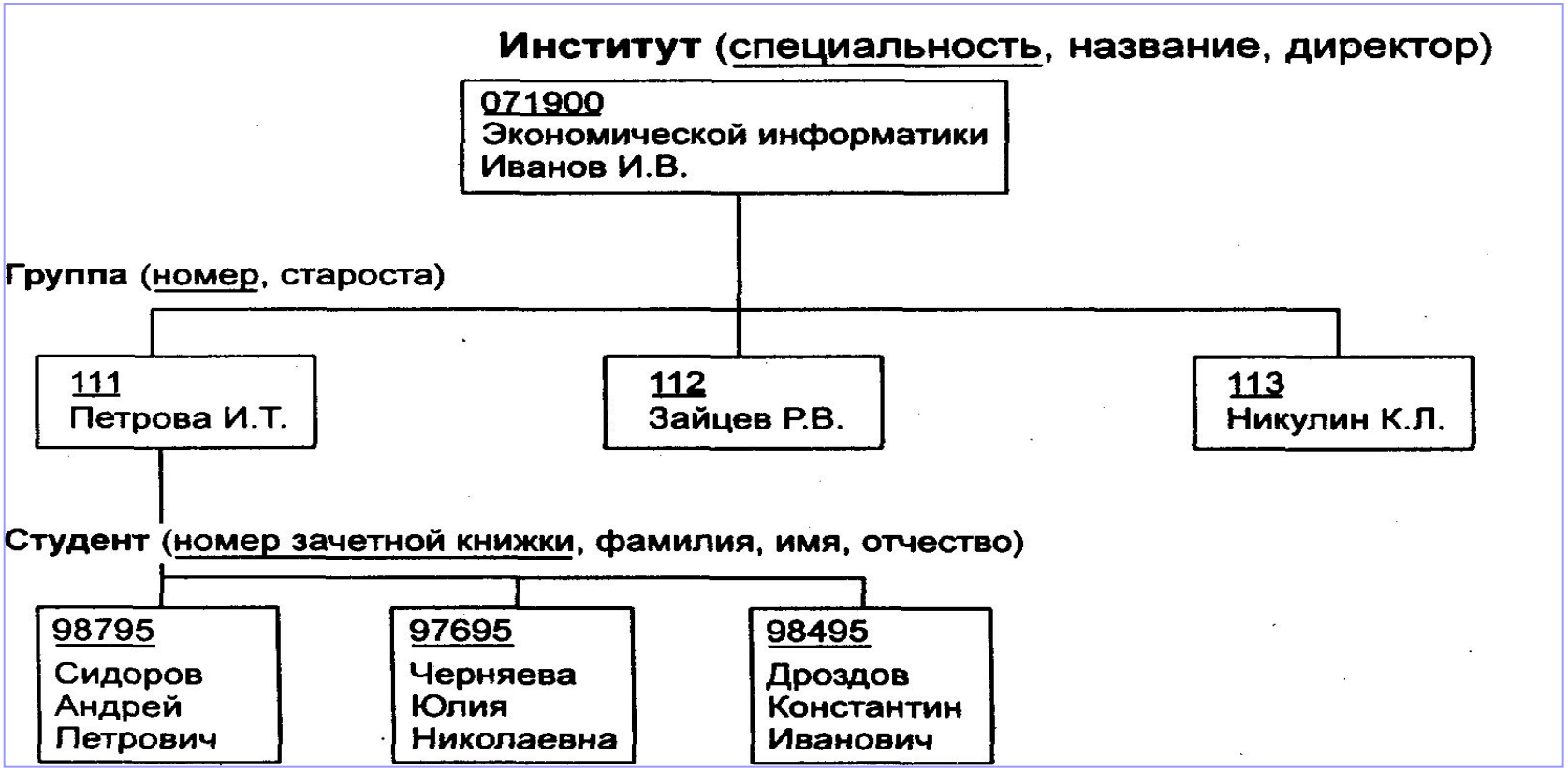
Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне.

- Иерархическое дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) *уровне*.

Зависимые (подчиненные) узлы находятся на втором, третьем и т.д. уровнях.

К каждой записи базы данных существует только один (иерархический) путь от корневой записи. Например, как видно из рис., для записи С4 путь проходит через записи В3 к А.

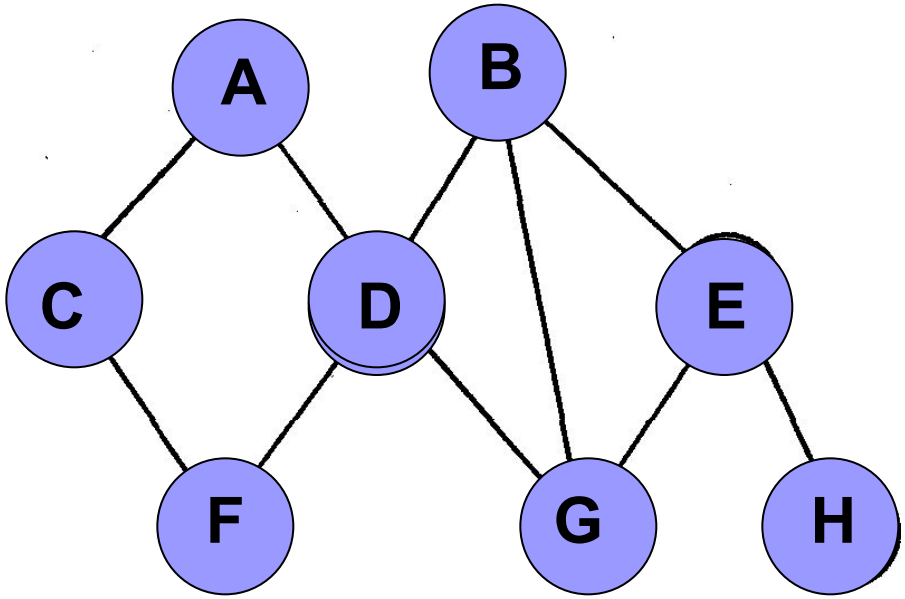
Пример иерархической структуры



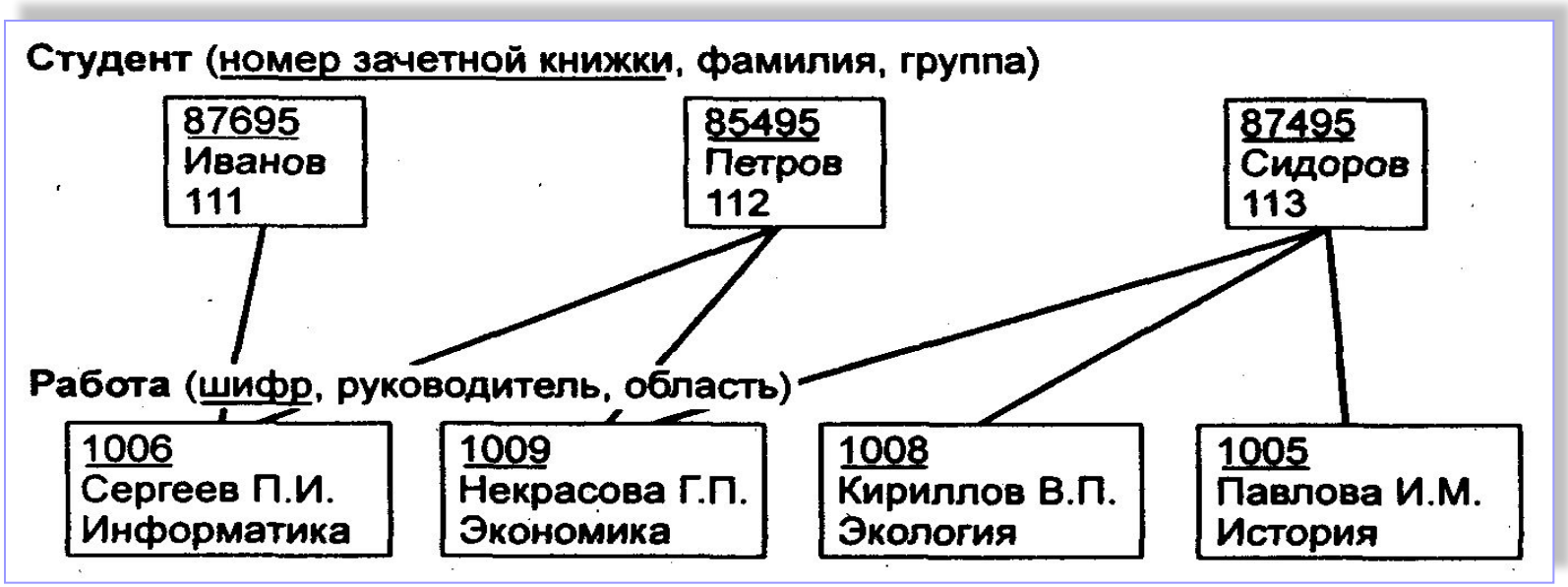
Для рассматриваемого примера иерархическая структура правомерна, так как каждый студент учится в определенной (только одной) группе, которая относится к определенному (только одному) институту.

Сетевая модель данных

- В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.
- На рисунке изображена сетевая структура базы данных в виде графа.



Пример сетевой структуры БД



Примером сложной сетевой структуры может служить структура базы данных, содержащей сведения о студентах, участвующих в научно-исследовательских работах (НИРС). Возможно участие одного студента в нескольких НИРС, а также участие нескольких студентов в разработке одной НИРС. Графическое изображение описанной в примере сетевой структуры состоит только из двух типов записей.

Понятие *реляционный* (англ. *relation* — отношение) связано с разработками известного американского специалиста в области систем баз данных Е. Кодда

Реляционная модель данных

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц.

Каждая *реляционная таблица* представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы — один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя (заголовки столбцов являются названиями полей в записях);
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Отношение – это плоская таблица, содержащая N столбцов, среди которых нет одинаковых. N – это *степень отношения*, или *арность* отношения.

Столбец отношения соответствует атрибуту сущности.

Кортеж – строка отношения (соответствует записи в таблице).

Пример реляционной модели

№ личного дела	Фамилия	Имя	Отчество	Дата рождения	Группа
16493	Сергеев	Петр	Михайлович	01.01.76	112
16593	Петрова	Анна	Владимировна	15.03.75	111
16693	Анохин	Андрей	Борисович	14.04.76	112

- Отношения представлены в виде *таблиц,* строки которых соответствуют кортежам или *записям,* а столбцы — атрибутам отношений, доменам, *полям.*
- Поле, каждое значение которого однозначно определяет соответствующую запись, называется *простым ключом* (ключевым полем)
- . Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет *составной ключ.*
- В примере ключевым полем таблицы является "№ личного дела".

Нормализация БД

База данных, с логической точки зрения представляет собой совокупность взаимосвязанных отношений. При проектировании БД должна обеспечивать *целостность данных*, т.е. достоверность и непротиворечивость хранимых данных, причем эти свойства не должны утрачиваться в процессе работы с данными (после многочисленных изменений, удалений и дополнений данных).

Для обеспечения целостности схема БД должна удовлетворять определенным требованиям: каждый факт, должен храниться только один раз, поскольку дублирование может привести (и на практике непременно приводит, как только проект приобретает реальную сложность) к несогласованности между копиями одной и той же информации. Следует избегать любых неоднозначностей, а также избыточности хранимой информации.

Приведение структуры БД в соответствие этим ограничениям – это *нормализация БД.* Нормализация БД предполагает нормализацию входящих в нее отношений.

Режимы работы с базами данных

- *Проектировочный*. Режим работы со схемой БД. В этом режиме проектировщик БД создает в ней новые объекты (например, таблицы), задает их структуру, изменяет свойства существующих полей или добавляет новые, устанавливает дополнительные связи между таблицами. У него есть на это соответствующие права. Он работает со структурой всей базы в целом и поэтому имеет полный доступ к любым ее элементам. Если база достаточно сложная, то у нее может быть целая группа разработчиков, каждый из которых имеет права на модификацию определенной части схемы базы.
- *Эксплуатационый* (*пользовательский*). Режим работы с данными. Пользователь БД которое наполняет ее информацией, обрабатывает данные с помощью запросов и получает результат в виде результирующих таблиц или отчетов в соответствии с предоставленными ему правами. Доступ к структуре базы в этом случае закрыт.

Чтение фрагментов базы данных

- Элементарным фрагментом базы данных является *запись*. Запись соответствует кортежу в отношении и строке в таблице. Запись состоит из *полей.* Поле соответствует атрибуту в отношении и столбцу в таблице.
- Чтобы прочитать записи выполняют запросы к базе данных.
- *Запрос* – команда, которая требует от СУБД, чтобы она вывела определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера или терминала, ее можно также послать принтеру, сохранить в файле (или как объект в памяти компьютера), представить как вводную информацию для другой команды или процесса.
- В большинстве современных реляционных СУБД запросы выполняются посредством языка SQL (Структурированный Язык Запросов)

Технология работы с БД

Каждая СУБД имеет свои особенности, которые необходимо учитывать. Однако, имея представление о функциональных возможностях любой СУБД, можно представить обобщенную технологию работы пользователя в этой среде.

Основные этапы работы с СУБД можно представить в виде схемы: