Кафедра ИВС

ЛЕКЦИЯ

Тема: «Структуры, уровни представления и методы доступа к данным»

Дисциплина «Базы данных в ИС»

ОП «Информационные системы»

Авторы: ст.преп.каф.ИВС Саданова Б.М.

ст.преп.каф.ИВС Олейникова А.В.

План лекции:

1. Структуры данных, определения и классификация
2. Линейные структуры данных
3. Нелинейные структуры данных
4. Уровни представления данных
5. Методы доступа к данным
6. Индексы, их назначение
7. Типы индексов

Цель лекции: дать представление о способах организации данных в памяти ЭВМ в СУБД, основных методах доступа к данным, а также использовании индексов, позволяющих существенно сократить время реализации основных операций в БД.

Структуры данных

- **Структура данных** - организационная схема данных, в соответствии с которой они упорядочены, с тем, чтобы их можно было интерпретировать и выполнять над ними определенные операции.
- Структуры данных могут быть:

однородные	• все элементы данных представлены записями одного типа
неоднородные	• элементами одной структуры могут являться записи разных типов

Структуры данных

- Различные структуры данных предоставляют и различный доступ к своим элементам: в одних структурах доступ возможен *к любому элементу*, в других - *к строго определенному*.

- В памяти компьютера данные могут иметь *последовательное* или *связанное* представление. Соответственно различают структуры хранения, использующие *последовательное представление данных* и *связанное представление данных*.

Структуры данных

Последовательное представление:

- данные в памяти компьютера размещаются в соседних последовательно расположенных ячейках
- физический порядок следования записей полностью соответствует логическому порядку, определяемому логической структурой
- совокупность записей, размещенных в последовательно расположенных ячейках памяти, называют *последовательным списком*.

Связанное представление данных:

в каждой записи предусматривается дополнительное поле, в котором размещается указатель (ссылка).

записи располагаются в любых свободных ячейках и связываются указателями, указывающими на место расположения записи, логически следующей за данной.

структуры хранения, основанные на связанном представлении данных, называют *связанными списками*.

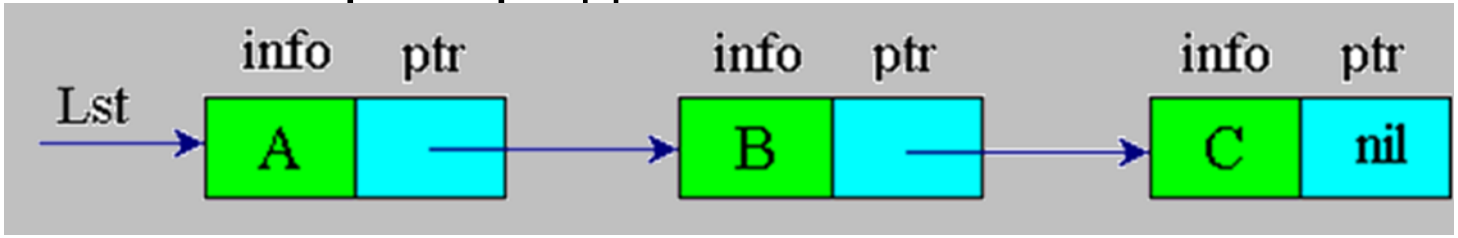
Структуры данных

- структуры хранения, основанные на связанном представлении данных, называют *связанными списками*.

- если каждая запись содержит лишь один указатель, то *список односвязный*, при большем числе указателей - *список многосвязный*

Структуры данных

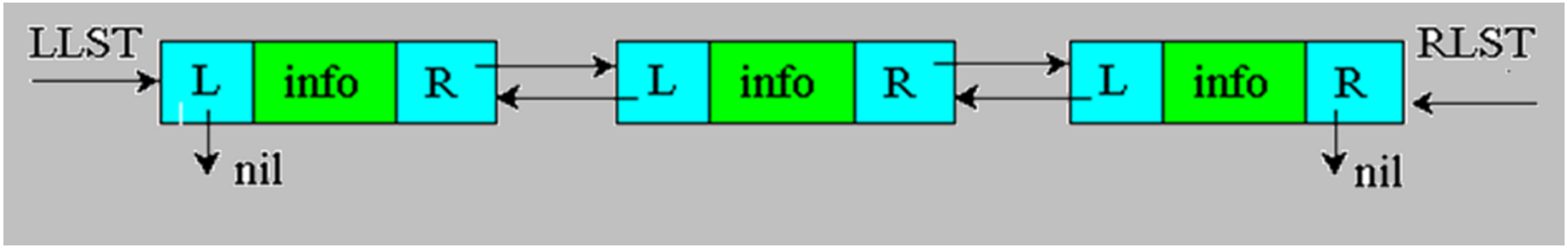
Пример односвязного списка



- Элемент односвязного списка содержит два поля: информационное поле (info) и поле указателя (ptr)
- Указатель дает только адрес последующего элемента списка
- Поле указателя последнего элемента в списке является пустым (NIL)
- LST - указатель на начало списка. Список может быть пустым, тогда LST = NIL
- Доступ к элементу списка осуществляется только от его начала, обратной связи в этом списке нет

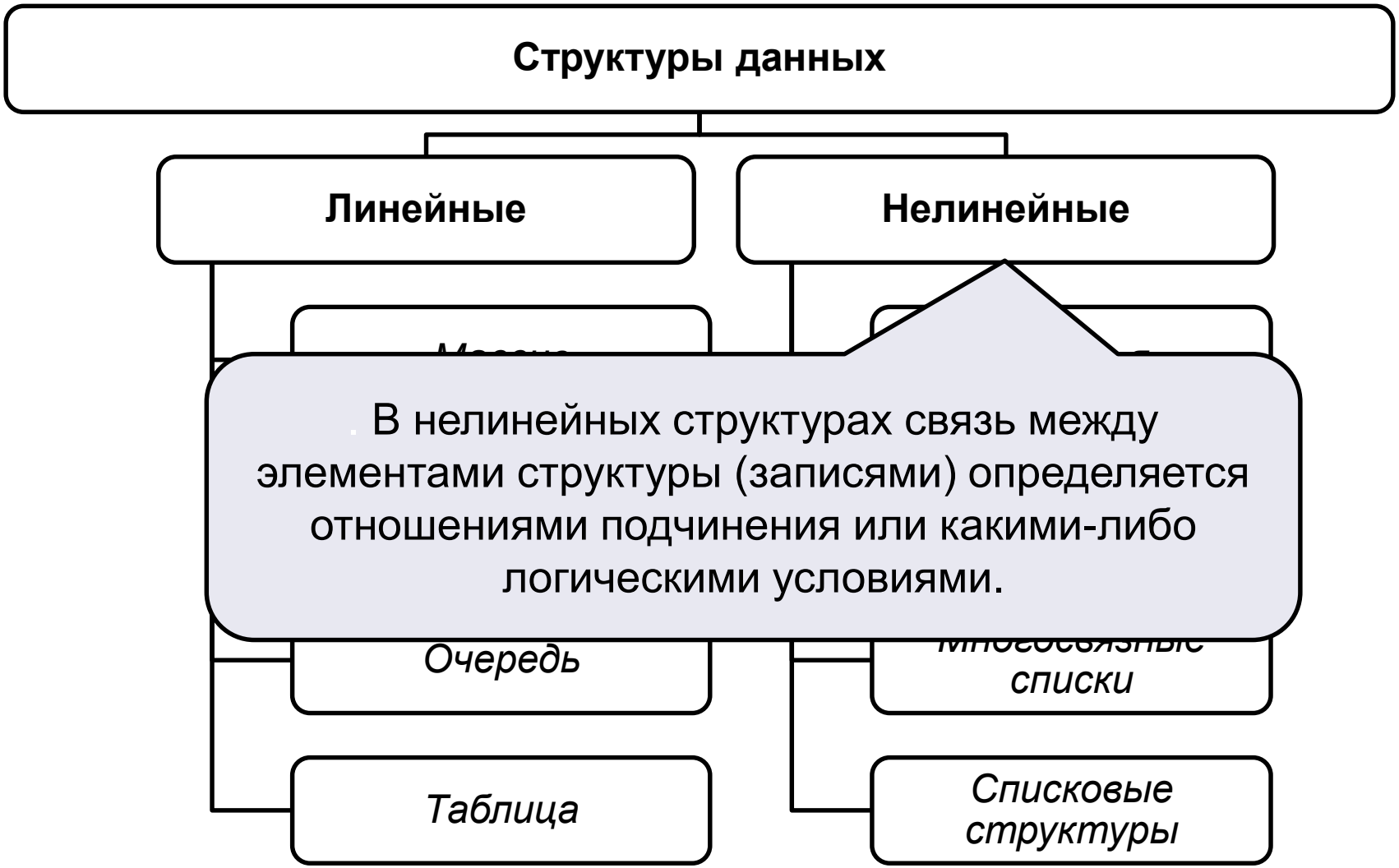
Структуры данных

Пример многосвязного списка:



- В двусвязном списке у любого элемента есть два указателя
- Один указывает на предыдущий (левый) элемент (L), другой указывает на последующий (правый) элемент (R)

Структуры данных



Линейные (статические) структуры данных

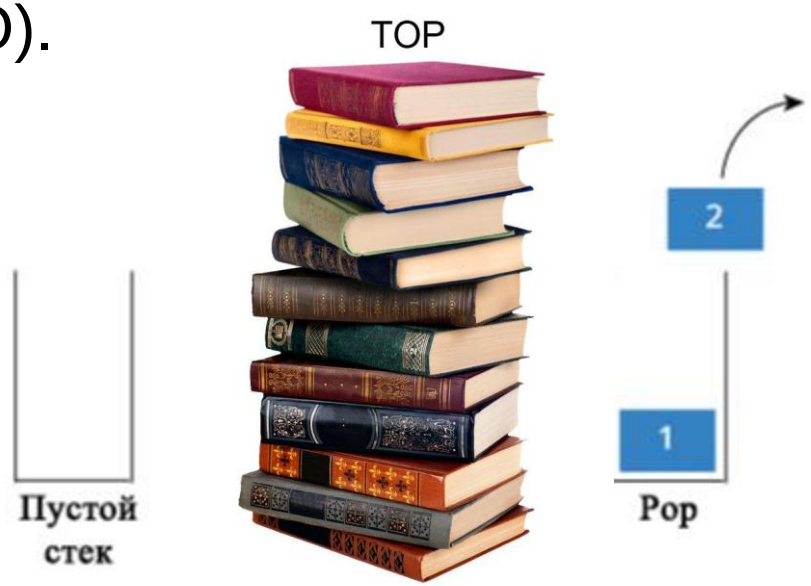
Массив

- линейная структура данных фиксированного размера, реализуемая с использованием последовательного представления данных.
- Каждый элемент массива идентифицируется одним или несколькими индексами.
- *Индекс* - это целое число, значение которого определяет позицию соответствующего элемента в массиве и используется для осуществления доступа к этому элементу.

Линейные (статические) структуры данных

Стек

- Линейная структура переменного размера.
- позволяет включать или исключать элементы, то есть объем данных в стеке может динамически расти и сокращаться во время выполнения программы.
- Информация обрабатывается по принципу "последним пришел, первым ушел« (LIFO).



Линейные (статические) структуры данных

Очередь

- Линейная структура переменного размера.
- Исключение элементов из очереди допускается с одного конца - с начала очереди.
- Включение элементов возможно лишь в противоположный конец - в конец очереди.
- Данные обрабатываются в порядке их поступления по принципу: "первым пришел, первым ушел" (FIFO).

Линейные (статические) структуры данных

Таблица

- Линейная структура данных, каждый элемент которой характеризуется определенным значением ключа и доступ к элементам которой осуществляется по ключу
- *Ключ* – поле или набор полей, однозначно идентифицирующих запись

Идентификатор	Фамилия	Имя	Отчество
232	Иванов	Иван	Иванович
453	Петров	Петр	Петрович
545	Сидоров	Савелий	Сергеевич

Ключ уникален для каждого элемента.

Нелинейные структуры данных

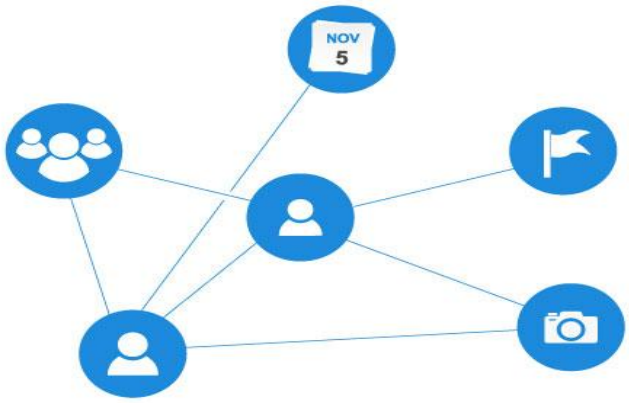
- Отношения *"один-ко-многим"* носят иерархический характер и отображаются древовидными структурами.

- Отношения *"многие-ко-многим"* носят более универсальный характер и отображаются структурами графов.

Нелинейные структуры данных

Граф

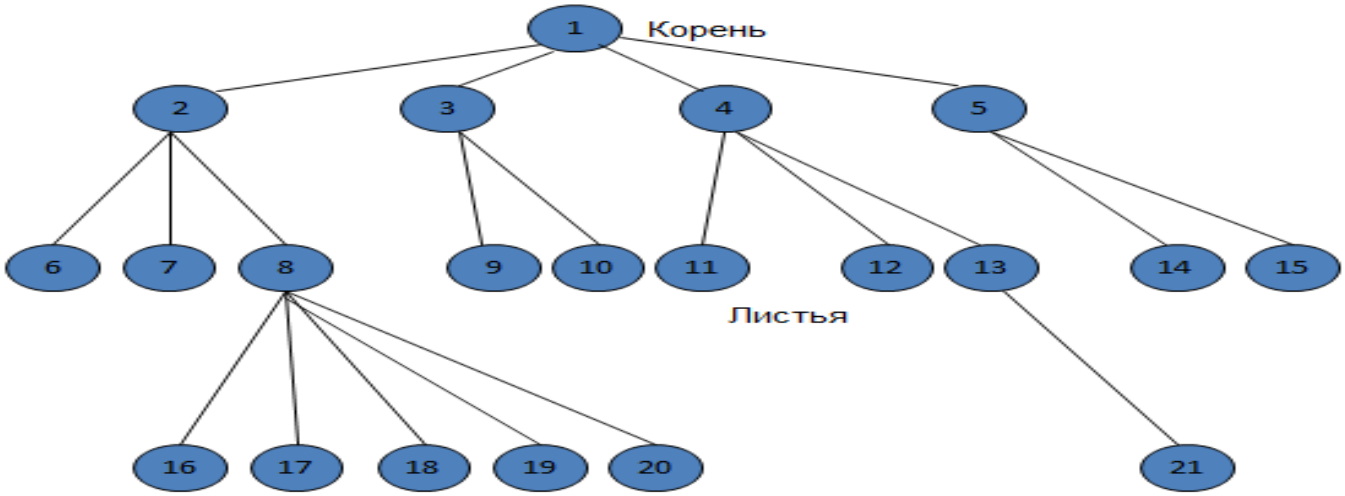
- Граф общего вида состоит из ряда вершин (узлов) и ребер, связывающих пары вершин. Если в понятия "ребро" и "вершина" вложить определенную смысловую нагрузку, то графы можно использовать для представления данных.
- Вершинам графа можно сопоставить определенные объекты, тогда ребра будут соответствовать отношениям между объектами.
- Модель данных, имеющую вид ориентированного графа, называют *сетью*.



Нелинейные структуры данных

Дерево

- Это граф с некоторыми ограничениями, т.е. ориентированный граф, не имеющий циклов.
- Узлы дерева организованы по уровням и подчинены определенной иерархии.
- Любой узел дерева связан с единственным узлом более высокого уровня - порождающим - и с m узлами ближайшего уровня - порожденными.



Нелинейные структуры данных

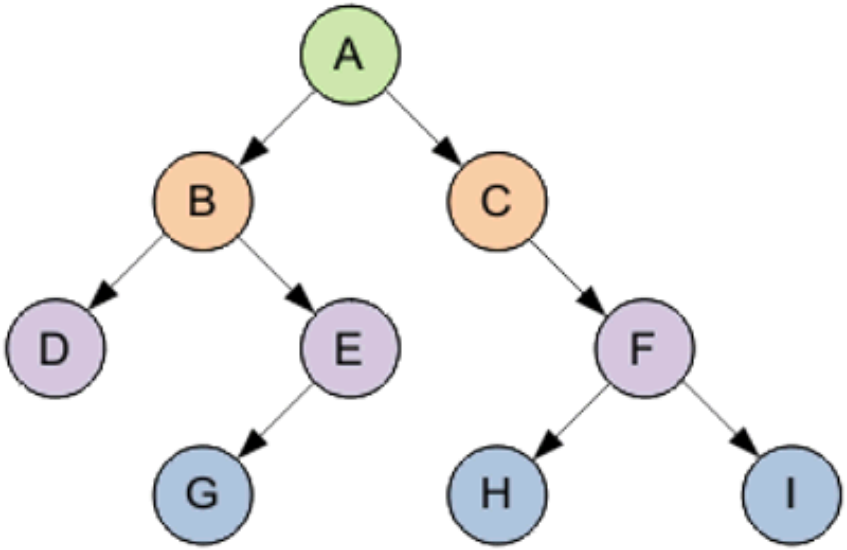
Дерево

- На самом верхнем уровне, в начале дерева имеется единственный узел, называемый *корнем*.
- Узлы, расположенные в конце каждой ветви дерева и не имеющие порожденных, называются *листьями*.
- В деревьях направление обязательно от порождающего к порожденному.
- Длина пути от корня до некоторого узла равна *уровню* этого узла.
- Количество уровней дерева определяет *высоту дерева.*
- Если в дереве между порожденными узлами, имеющими общий исходный, считается существенным их порядок, то дерево называется *упорядоченным*.

Нелинейные структуры данных

Дерево

- *Бинарное (двоичное) дерево* – это динамическая структура данных, представляющая собой дерево, в котором каждая вершина имеет не более двух потомков.



Нелинейные структуры данных

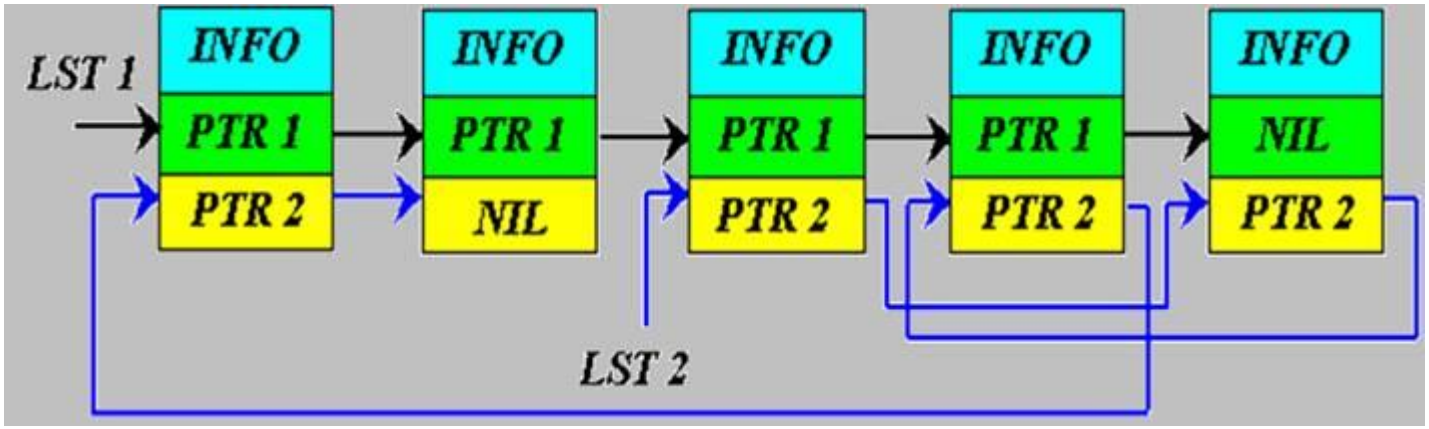
Списки

- Структура данных, в которой любой элемент может сам являться списком, называется *списковой.*
- Списковую структуру можно изобразить в виде графа. Исходная вершина графа соответствует входу в список.
- Любая другая вершина графа является либо *атомарной*, либо *структурной*.
- *Атомарные вершины* соответствуют атомарным элементам списковой структуры и содержат информацию об объектах.
- *Структурные вершины* информации об объектах не содержат, они соответствуют тем элементам, которые являются подсписками.

Нелинейные структуры данных

Списки

- Для представления в памяти компьютера списковые структуры требуют *связанного представления*.
- Выделяются объекты, записи о которых становятся элементами основного списка. Остальные объекты группируются в подсписки второго уровня, ответвляющиеся от тех элементов основного списка, с которыми они находятся в определенной логической связи.



Уровни представления данных

1. Логический уровень :

- оперируют с логическими структурами данных, отражающими реальные отношения, которые существуют между объектами (таблицами) БД и их характеристиками.
- единица хранения информации - логическая запись.
- не учитывается техническое и математическое обеспечение системы.

2. Уровень хранения :

- оперируют со структурами хранения, то есть представлениями логической структуры данных в памяти компьютера.
- единица хранения информации - логическая запись.
- одна и та же логическая структура данных может быть реализована в памяти компьютера различными структурами хранения (например, массив, запись, таблица).

Уровни представления данных

3. Физический уровень :

- оперируют с физическими структурами данных.
- решается задача реализации структуры хранения в памяти компьютера.
- единица хранения информации - физическая запись, представляющая собой участок носителя, на которой размещается одна или несколько логических записей.

Уровни представления данных

Физическая независимость от данных:

- означает, что любые изменения в физическом расположении данных или техническом обеспечении БД не должны отражаться на логических структурах и прикладных программах, то есть не должны вызывать в них изменений.

Логическая независимость от данных :

- означает, что изменения в структурах хранения не должны вызывать изменения в логических структурах данных и в прикладных программах.

Уровни представления данных

Виртуальные данные :

- существуют лишь на логическом уровне.
- пользователю эти данные представляются реально существующими и он оперирует ими при необходимости.
- Каждый раз при обращении к этим данным система определенным образом их генерирует на основании других данных, физически существующих в системе (например, представления view).

Прозрачные данные :

- представляются несуществующими на логическом уровне.
- позволяет скрыть от пользователя многие сложные механизмы, используемые при преобразовании логических структур данных в физические (например, индексы).

Методы доступа к данным

- К операциям обработки данных относятся:
  - выборка
  - изменение
  - включение данных
  - исключение данных
- В основе этих операций лежит операция доступа.
- В задачах поиска предполагается, что все данные хранятся в памяти с определенной идентификацией.

Методы доступа к данным

Задача

- организовать доступ к файлу, содержащему набор одинаковых записей, каждая из которых имеет уникальное значение ключевого поля

Решение

- последовательно просматривать

Оцен
реше

Ключи вместе с указателями на соответствующие записи в файле копируют в другую структуру, которая позволяет быстро выполнять операции сортировки и поиска.
При доступе к данным вначале в этой структуре находят соответствующее значение ключа, а затем по хранящемуся вместе с ним указателю получают запись из файла.

Методы доступа к данным

Методы, реализующие доступ к данным по ключу:

- **методы поиска по дереву**
- **методы хеширования**

При поиске в оперативной памяти часто используется бинарное дерево, т.е. упорядоченное дерево, степень которого не больше 2.

Используются когда в
известно и на время обр
оперативной памяти. С
резать, измельчать), одно
ключей на множество у
можно вычислить адре
кл

Методы доступа к данным

Алгоритм поиска по бинарному дереву :

Массив должен быть отсортирован. При поиске используется деление массива пополам

Сначала аргумент поиска сравнивается с ключом, находящимся в корне.

Если аргумент совпадает с ключом, поиск закончен, если же не совпадает, то в случае, когда аргумент оказывается меньше ключа, поиск продолжается в *левом поддереве*, а в случае, когда больше ключа - *в правом поддереве*.

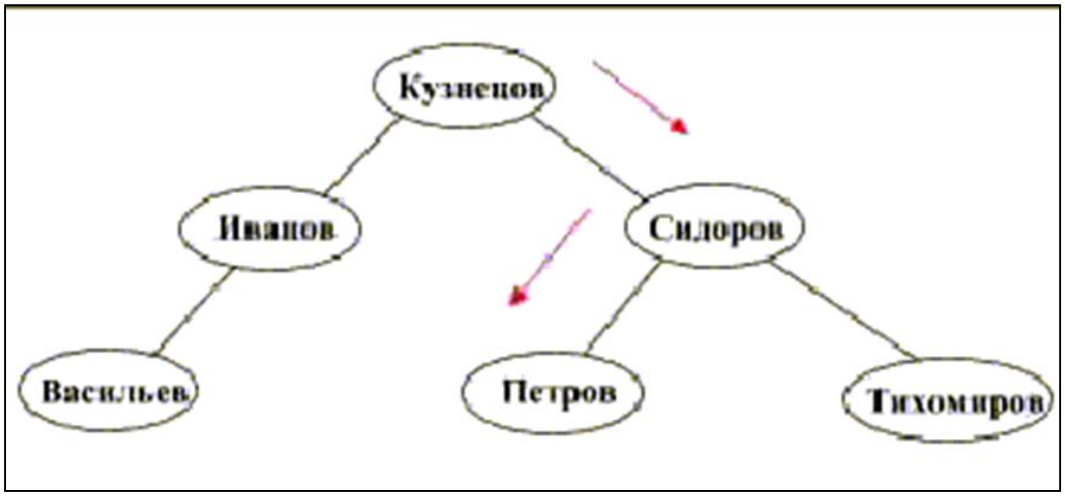
Увеличив уровень на 1, повторяют сравнение, считая текущий узел корнем.

Методы доступа к данным

Пример поиска по бинарному дереву :

- дан список студентов, содержащий их фамилии и средний балл успеваемости
- все записи имеют фиксированную длину, поэтому в качестве указателя можно использовать номер записи.
- Смещение записи в файле в этом случае будет вычисляться как ([номер_записи] -1) * [длина_записи].

Студент	Балл
Васильев	4,2
Иванов	3,4
Кузнецов	3,5
Петров	3,2
Сидоров	4,6
Тихомиров	3,8



Методы доступа к данным

Бинарные деревья

- Бинарные деревья особенно эффективны в том случае, когда множество ключей заранее неизвестно, либо когда это множество интенсивно изменяется. При переменном множестве ключей лучше иметь сбалансированное дерево.
- Бинарное дерево называют *сбалансированным (balanced),* если высота левого поддерева каждого узла отличается от высоты правого поддерева не более чем на 1.
- При поиске данных во внешней памяти очень важной является проблема сокращения числа перемещений данных из внешней памяти в оперативную.

Методы доступа к данным

Бинарные деревья

- *B-деревом порядка n* называется сильно ветвящееся дерево степени 2n+1, обладающее следующими свойствами:

✓ Каждый узел, за исключением корня, содержит не менее n и не более 2n ключей.

✓ Корень содержит не менее одного и не более 2n ключей.

✓ Все листья расположены на одном уровне.

✓ Каждый не листовой узел содержит два списка: упорядоченный по возрастанию значений список ключей и соответствующий ему список указателей (для листовых узлов список указателей отсутствует).

Методы доступа к данным

Бинарные деревья

- *Для B-дерева*:
  - ✓ сравнительно просто может быть организован последовательный доступ, т.к. все листья расположены на одном уровне;
  - ✓ при добавлении и изменении ключей все изменения ограничиваются одним узлом.
- Для доступа к пространственным (многомерным) данным используют R-деревья (R-Tree).
- *R-дерево -* это индексная структура для доступа к пространственным данным, предложенная А.Гуттманом. R-дерево допускает произвольное выполнение операций добавления, удаления и поиска данных без периодической переиндексации.

Методы доступа к данным

Хеширование

- ключевые данные, используемые для определения адреса записи организуются в виде таблицы, называемой *хеш-таблицей*.
- Хеш-функция хеширует последовательность цифр или битов, определяющих значение ключа записи или ее кода, в результате чего получается хеш-адрес, по которому записи размещаются и ищутся.
- Часто применяемая функция хеширования, реализующая преобразование ключа по методу свертки, разбивает ключ на несколько частей, которые затем суммируются таким образом, чтобы сформировалось число в требуемом диапазоне.

Методы доступа к данным

Пример хеширования

- Пусть имеются восьмизначные ключи *K=97434658* и *K=31269857* . Нужно преобразовать их в трехзначные адреса.
- Можно проделать следующие операции:
- h (97434658) = 974 + 346 + 58 = 378
- h (31269857) = 312 + 698 + 57 = 067
- Символ h означает, что обработку ключа осуществляет хеш-функция.

Методы доступа к данным

Хеш-функция

- должна обеспечивать однозначное преобразование ключа записи в ее адрес
- адреса должны равномерно распределиться по области памяти, выделенной для хранения данных.
- не должна быть слишком сложной, поскольку время, необходимое для преобразований, добавляется к времени выполнения операций ведения, поиска или обработки.
- не устраняет полностью возможность получения одинаковых адресов. Возникают *коллизии*, то есть ситуации, когда различные записи получают один и тот же адрес.

Методы доступа к данным

Хеш-функция

- Одна из самых распространенных функций хеширования основана на методе деления и определяется в виде h(K) = K mod m + 1 , где m - делитель.
- Примем m = 101 и выполним преобразование h(K) над ключами 2000, 2001 и … 2017. Ключи отобразятся в адреса 82, 83 … 99. Эти же самые адреса окажутся сгенерированными функцией h(K) для ключей 3313, 3314…3330, то есть возникает коллизия.

Методы доступа к данным

Методы разрешения коллизий:

- хеш-адрес подвергается рехешированию, т.е. определенным преобразованиям адреса. Например, может быть умножен на константу, в результате чего получится новый адрес.
- Подобные преобразования могут выполняться до тех пор, пока не будет сгенерирован адрес ячейки памяти, являющейся свободной. При поиске записи выполняется та же последовательность преобразований.
- Другой метод заключается в использовании сгенерированного адреса в качестве начальной точки для последовательного просмотра. С этого адреса начинается поиск свободного места в памяти.

Методы доступа к данным

Методы разрешения коллизий:

- Адрес, генерируемый функцией, считается адресом хранения не одной конкретной записи, а страницы памяти, в пределах которой размещаются все записи, получившие этот адрес. В пределах страницы записи могут размещаться последовательно в порядке поступления или в соответствии с каким-либо установленным порядком.
- Если со временем страница окажется заполненной, в памяти выделяется новая страница, связываемая с предыдущим указателем.

Индексы

- *Индекс* - структура данных, которая помогает СУБД быстрее обнаружить отдельные записи в файле и сократить время выполнения запросов пользователей.
- Назначение индексов - обеспечение прямого доступа к кортежу отношения (записи) по ключу. Обычно индекс определяется только для одного отношения.
- *Индекс* – это таблица, которая содержит ключевые значения для каждой записи в таблице данных и записанные в порядке, требуемом для пользователя.
- Ключевые значения определяются на основе одного или нескольких полей таблицы. Кроме того, индекс содержит уникальные ссылки на соответствующие записи в таблице.

Типы индексов

- *Первичный индекс*. Файл данных последовательно упорядочивается по полю ключа упорядочения, а на основе поля ключа упорядочения создается поле индексации, которое гарантированно имеет уникальное значение в каждой записи.
- *Вторичный индекс*. Файл данных последовательно упорядочивается по неключевому полю и на основе этого неключевого поля формируется поле индексации, поэтому в файле может быть несколько записей, соответствующих значению этого поля индексации.
- Если в файле индексации содержится только одна запись, соответствующих каждому значению индексируемого поля, то такой индекс называют *уникальным*.

Типы индексов

- Обращение к записи через индексы осуществляется в два этапа:

1. в индексной структуре находится требуемое значение атрибута и соответствующий адрес записи;

2. по этому адресу происходит обращение к внешнему запоминающему устройству (ВЗУ). Индекс загружается в ОП целиком (или хранится в ней постоянно во время работы с БД).

- Для каждой БД можно одновременно поддерживаться несколько первичных и вторичных индексов.
- Различают *одиночные* индексы и *составные*. *Составной индекс* включает два или более столбца одной таблицы.

Типы индексов

- Индексы поддерживаются динамически, т.е. после обновления БД – добавлении или удалении записей, а также при модификации полей записи, входящих в ключ – индекс приводится в соответствие с обновленной версией БД.
- Обновление индекса занимает некоторое время, поэтому существование многих индексов может замедлить работу БД.
- Файл, содержащий логические записи, называется *файлом данных*, а файл, содержащий индексные записи - *индексным файлом*.

Контрольные вопросы:

1. Дайте определение понятию структуры данных и приведите их классификацию
2. Проведите сравнительную характеристику различных линейных структур данных
3. Проведите сравнительную характеристику различных нелинейных структур данных
4. Опишите уровни представления данных и задачи каждого из них
5. Приведите описание методов, реализующих доступ к данным по ключу
6. Дайте определение понятию индекса, видов индекса

Список литературы:

1. Т. Конноли. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.: Пер.с англ. – М.: Изд.дом «Вильямс», 2012. – 1440 с.
2. К.Дейт. Введение в системы БД. изд.6-е. :Пер.с англ. - М.: Изд.дом "Вильямс", 2014. – 1200с.
3. Д.Ульман. Введение в системы баз данных. – М.: Издательство «Лори», 2015. – 853с.

Спасибо за внимание!