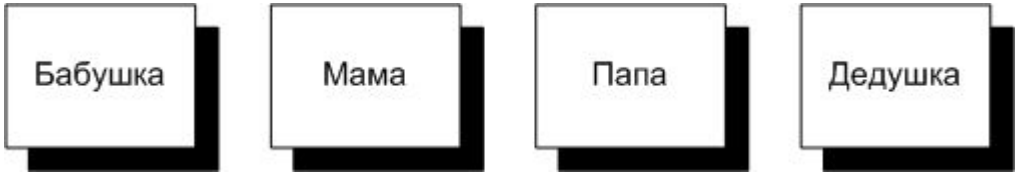
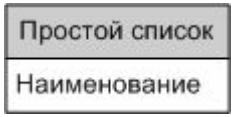
Способы хранения и алгоритмы обработки различных структур в реляционной БД

Списки, деревья, графы

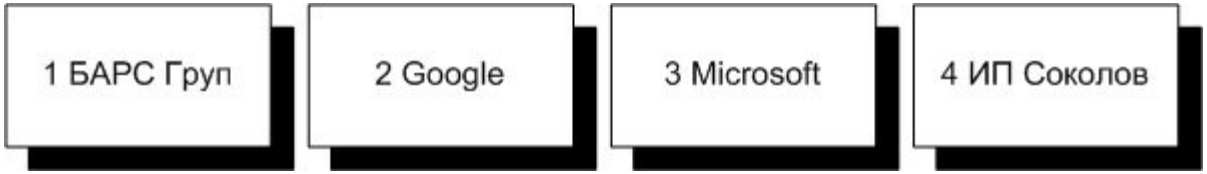
Списки

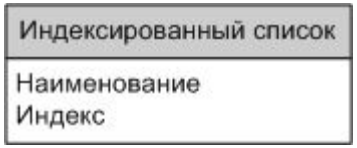
- Простой



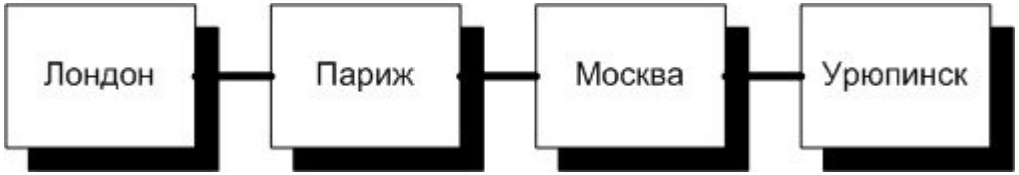


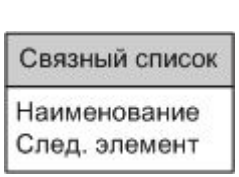
- Индексированный список

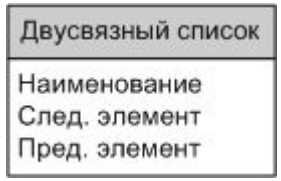




- Связный и двусвязный список





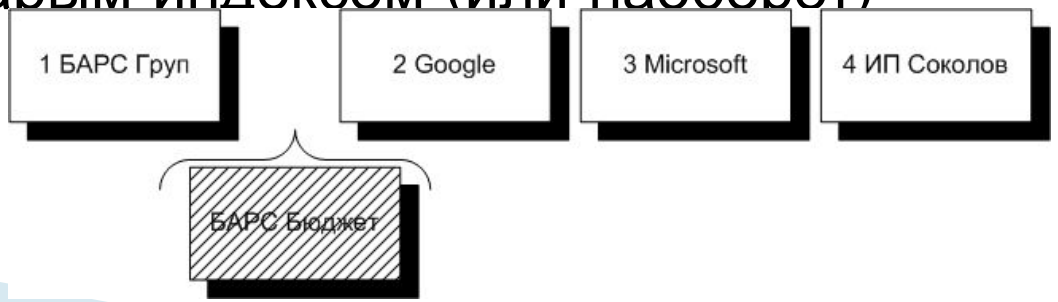


Операции со списками

- Добавление элемента в начало/конец/внутрь
- Перемещение элемента вперед/назад
- Удаление элемента
- Выборка диапазона элементов (с/по)
- Поиск предыдущего/следующего элемента

Списки. Индексированный список

- Добавление:
  - Находится нужный индекс и делается вставка с пересчетом индексов последующих элементов списка
- Перемещение:
  - Старый индекс = Индекс перемещаемого элемента
  - Индекс перемещаемого элемента = Индекс новой позиции
  - Изменяем индексы элементов между Индексом новой позиции и Старым индексом (или наоборот)



Списки. Индексированный список

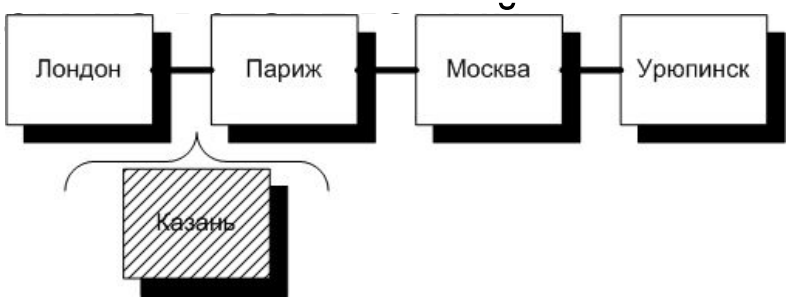
- Удаление:
  - Изменяем индексы последующих элементов после удаляемого
- Выборка диапазона:
  - Запрос элементов с индексами между индексами крайних элементов
- Следующий/предыдущий:
  - Запрос элемента с индексом на единицу большим/меньшим

Списки. Индексированный список

- Преимущества:
  - Простота выборок
  - Простота хранения
- Недостатки:
  - Изменение списка требует небольшой обработки остальных элементов
- Варианты:
  - Индекс может быть например датой

Списки. Связный список

- Добавление в начало:
  - Во вставляемом элементе указываем на первый и помечаем его первым
- Добавление в конец:
  - Находим последний элемент и указываем на вставляемый
- Добавление внутрь:
  - Находим элемент после которого нужно добавить
  - Во вставляемом элементе указываем следующий из найденного
  - В найденном элементе указыва

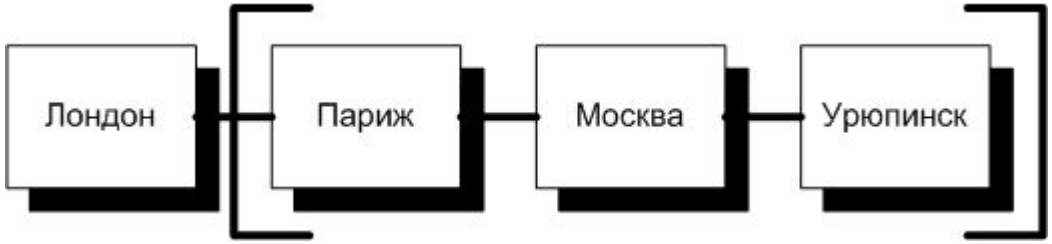


Списки. Связный список

- Перемещение:
  - Находим предыдущий элемент от перемещаемого
  - В найденном элементе указываем на следующий элемент из перемещаемого
  - Если не нашли нет, значит помечаем следующий элемент как первый
  - Делаем вставку перемещаемого элемента в нужную позицию списка
- Удаление:
  - Находим предыдущий элемент от удаляемого
  - В найденном элементе указываем на следующий элемент из удаляемого
  - Если не нашли нет, значит помечаем следующий элемент как первый

Списки. Связный список

- Выборка диапазона:
  - В цикле начиная с начала диапазона по одному выбираем следующий элемент пока не конец диапазона
- Следующий:
  - Выбираем из указателя в элементе
- Предыдущий:
  - Если это не первый элемент, то выбрать элемент в котором есть указатель на текущий



Списки. Связный список

- Преимущества:
  - Относительная простота изменения списка
- Недостатки:
  - Сложная навигация по списку и выборки
- Варианты:
  - Связующим полем может быть дата

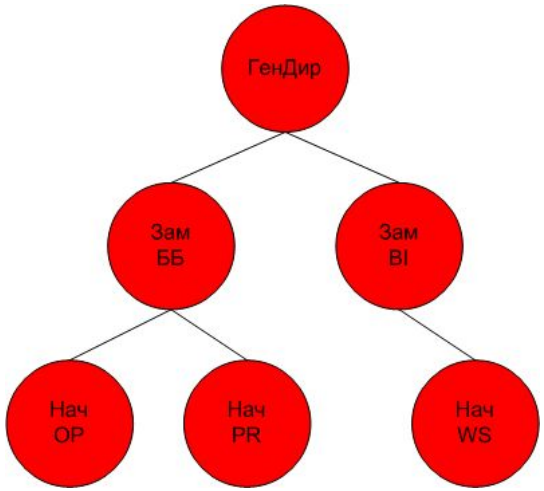
Деревья

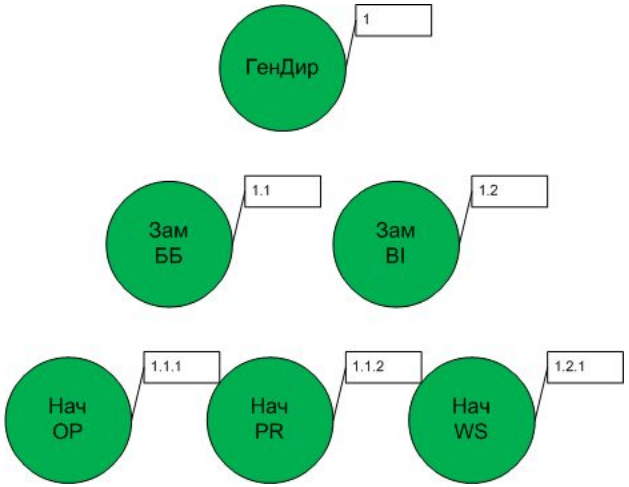
- Смежные вершины (Adjacency List)
- Материализованный путь (Materialized Path)
- Вложенные множества (Nested Set)

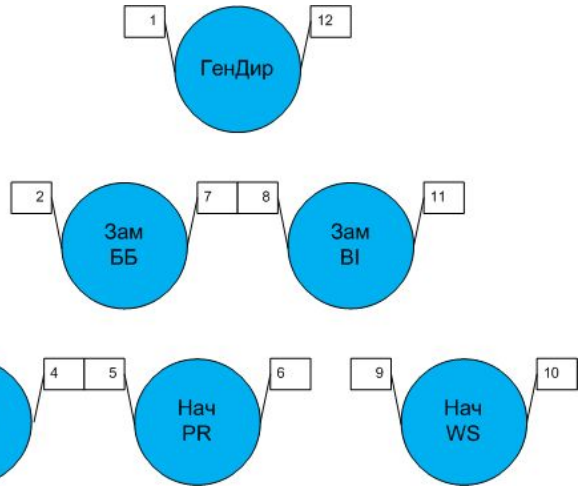
Узел
Наименование Род. узел

Узел
Наименование Полный путь

Узел
Наименование Левое число Правое число





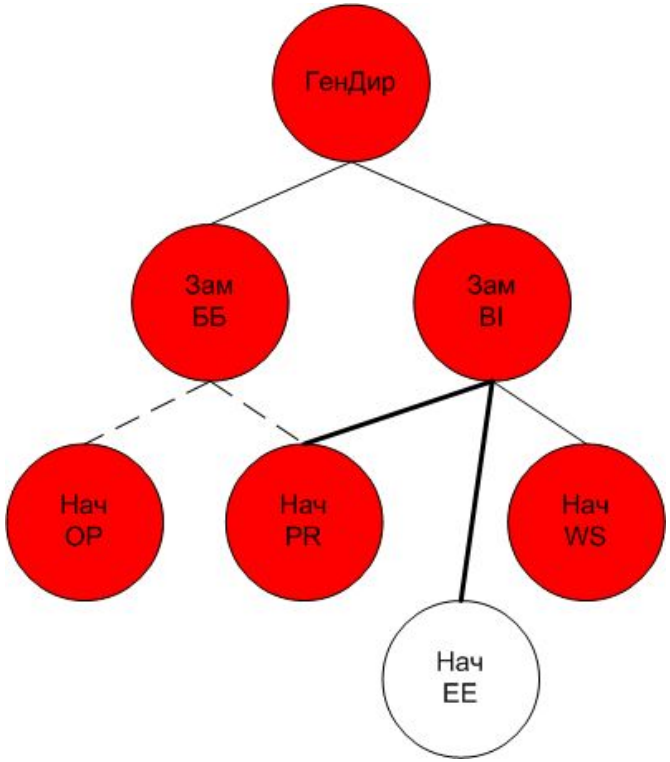


Операции с деревьями

- Добавление узла
- Перемещение узла
- Удаление узла
- Выборка пути от узла до корня
- Выборка ветки дерева
- Выборка узлов уровня
- Выборка конечных узлов
- Выборка соседних узлов

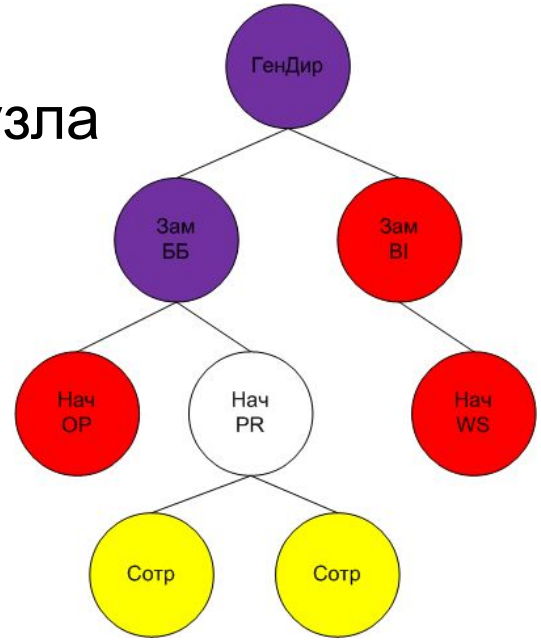
Деревья. Смежные вершины

- Добавление:
  - По аналогии со связным списком
- Перемещение:
  - Удаление из связного списка
  - Вставка в связный список
- Удаление:
  - Удаление из связного списка



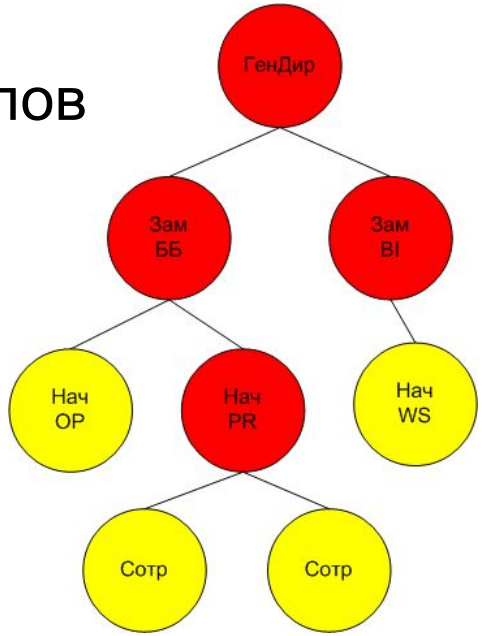
Деревья. Смежные вершины

- Выборка пути от узла до корня
  - В цикле выбираем родителя текущего узла, затем родителя делаем текущим узлом и продолжаем до корня
- Выборка ветки дерева относительно узла:
  - Выбираем всех детей для текущего узла затем в цикле по детям выполняем рекурсивно такой же выбор детей



Деревья. Смежные вершины

- Выборка конечных узлов:
  1) Выбираем текущим узел корня дерева
  2) Ищем всех, у кого являемся родителем
  3) Если не нашли таких узлов, то добавляем текущий узел в выбранные
  4) Если нашли, то в цикле рекурсивно выполняем с п.2) для найденных узлов



Деревья. Смежные вершины

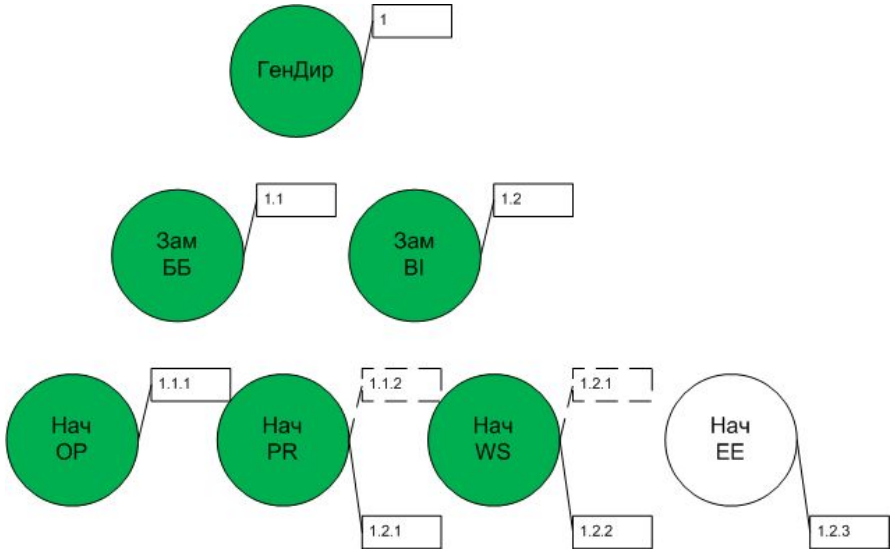
- Выборка уровня дерева:
  1) Выбираем текущим узел корня дерева
  2) Если достигнут нужный уровень, то добавляем узел в выбранные
  3) Ищем всех, у кого являемся родителем, затем в цикле рекурсивно выполняем с п.2) для найденных узлов
- Выборка соседних узлов:
  - Выбрать узлы у которых родитель совпадает с текущим узлом

Деревья. Смежные вершины

- Преимущества:
  - Простота хранения
  - Простота изменения
- Недостатки:
  - Сложность работы с ветками
  - Сложность работы с уровнями
  - Сложность определения принадлежности к родителям/детям

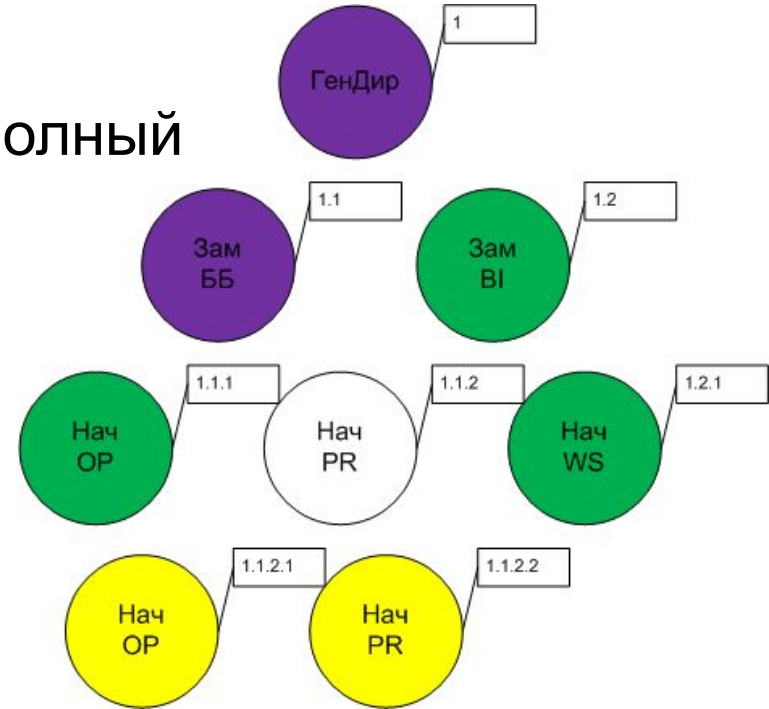
Деревья. Материализованный путь

- Добавление:
  - Находим место и путь в дереве для вставки
  - В добавляемом узле указывается полный путь
- Перемещение:
  - Находим новое место и путь в дереве для вставки
  - У перемещаемого узла и всех его дочерних узлов изменяется полный путь
- Удаление:
  - Удаление узла



Деревья. Материализованный путь

- Выборка пути от узла до корня
  - Выбираем те узлы дерева, в которых полный путь является началом полного пути заданного узла
- Выборка ветки дерева относительно узла:
  - Выбираем те узлы, у которых полный путь начинается с полного пути заданного узла



Деревья. Материализованный путь

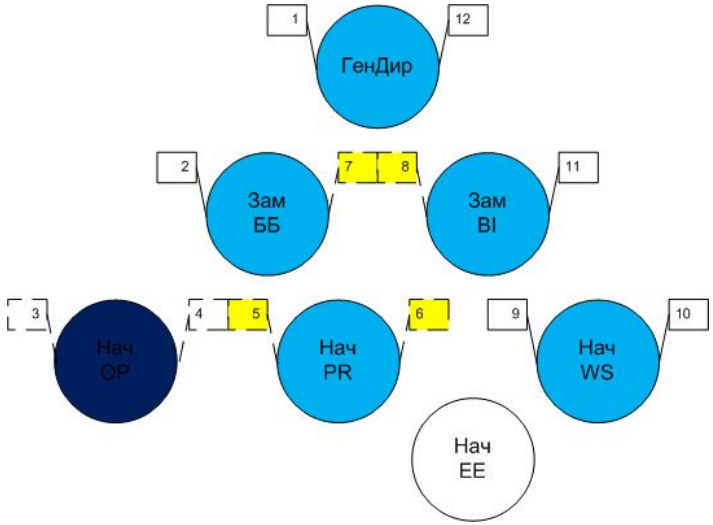
- Выборка конечных узлов:
  - Выбрать те узлы, по которым не найдется узлов с полным путем начинающимся с полного пути текущего узла
- Выборка уровня дерева:
  - Выбрать узлы, у которых в полном пути присутствует элемент искомого уровня
- Выборка соседних узлов:
  - Выбрать узлы, у которых полный путь совпадает с заданным узлом за исключением последнего элемента пути

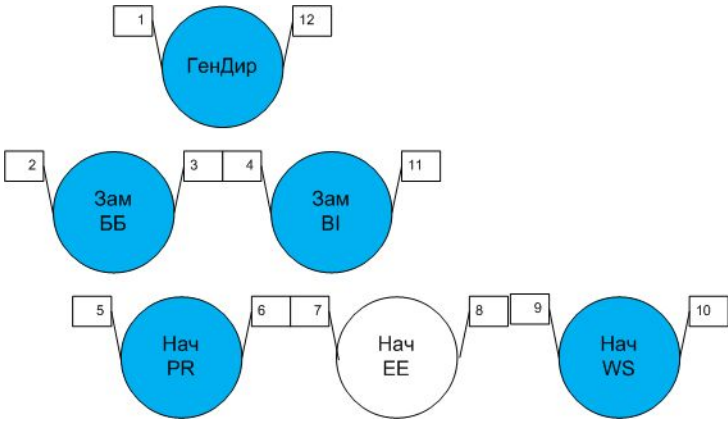
Деревья. Материализованный путь

- Преимущества:
  - Простота работы с ветками
  - Простота определения принадлежности к родителям/детям
- Недостатки:
  - Сложность работы с уровнями
  - Небольшая сложность в изменении
- Варианты:
  - Добавить атрибут уровня

Деревья. Вложенные множества

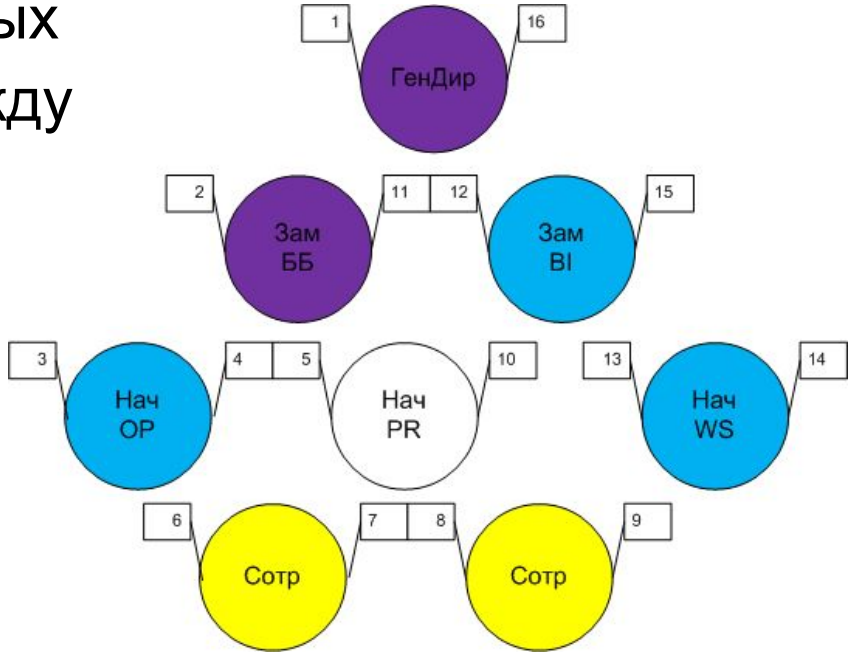
- Добавление:
  - Пересчет узлов дерева находящихся правее добавляемого узла
- Перемещение:
  - Пересчет узлов дерева находящихся между начальной позицией перемещаемого элемента и конечной позицией (или наоборот)
- Удаление:
  - Пересчет узлов дерева находящихся правее удаляемого узла





Деревья. Вложенные множества

- Выборка пути от узла до корня
  - Выбираем те узлы дерева, у которых правое число больше правого числа и левое число меньше левого числа заданного узла
- Выборка ветки дерева относительно узла:
  - Выбираем те узлы, у которых левое число находится между левым и правым числом заданного узла



Деревья. Вложенные множества

- Выборка конечных узлов:
  - Выбрать те узлы, у которых разница между левым и правым числом равна 1
- Выборка уровня дерева:
  - Алгоритма нет! (я не знаю)
- Выборка соседних узлов:
  - Постепенно выбирать узлы, у которых левое число отличается на 1 от правого числа начального элемента и у которых правое число отличается на 1 от левого числа начального элемента

Деревья. Вложенные множества

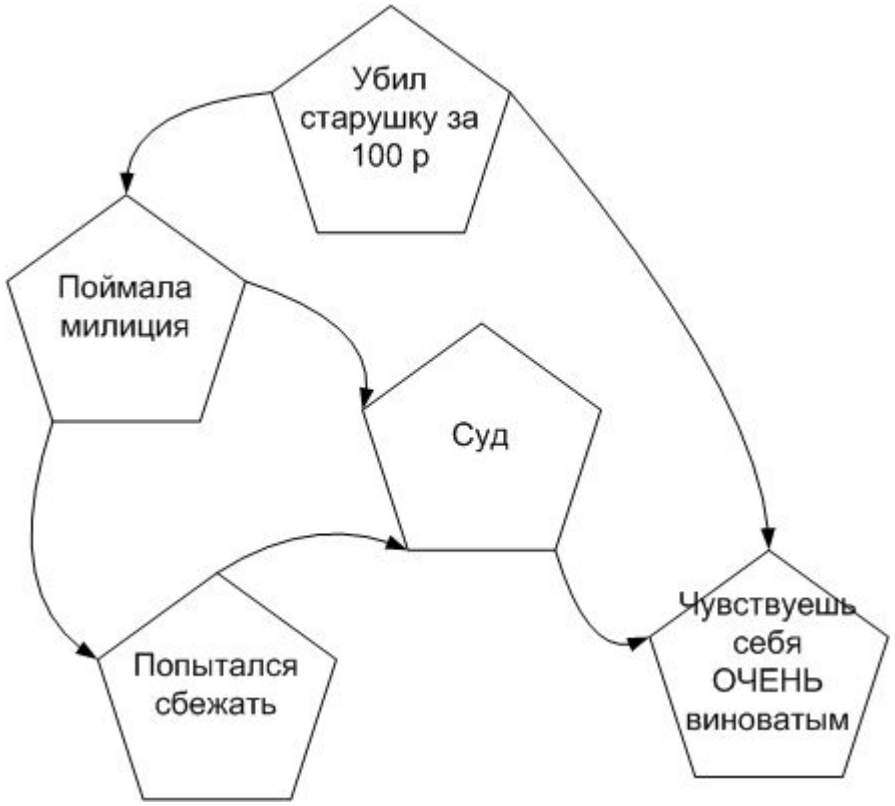
- Преимущества:
  - Простота работы с ветками
  - Простота определения принадлежности к родителям/детям
- Недостатки:
  - Сложность работы с уровнями
  - Сложность в изменении
- Варианты:
  - Добавить атрибут уровня

Графы

- Список ребер

Вершина	Наименование
A	Убил старушку
B	Поймала милиция
C	Суд
D	Попытался бежать
E	Виновен

Ребро	Из вершины	В вершину
1	A	B
2	A	E
3	B	C
4	B	D
5	D	C
6	C	E



Операции с графами

- Добавление вершины/ребра
- Перемещение вершины/ребра
- Удаление вершины/ребра
- Получение всех вершин/ребер родителей
- Получение всех вершин/ребер потомков
- Поиск пути между вершинами

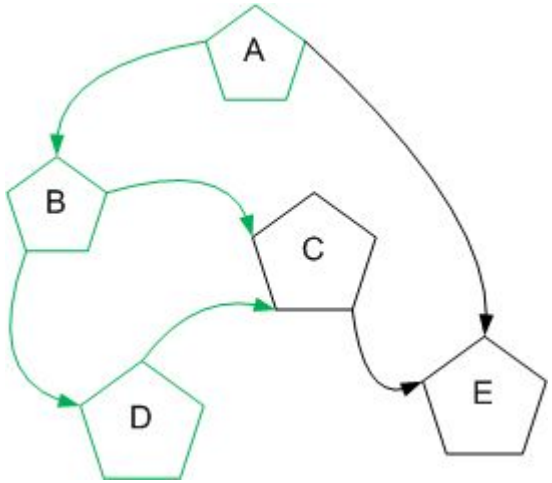
Графы. Список ребер

- Добавление:
  - Добавление ребер ведущих к и из вершины
- Перемещение:
  - Удаление ребер ведущих к и из перемещаемой вершине
  - Добавление ребер ведущих к и из новой позиции вершины
- Удаление:
  - Удаление ребер ведущих к и из удаляемой вершины

Графы. Список ребер

- Получение родителей:
  1) Находим вершины приводящие к текущей вершине, включаем их в список просмотренных вершин
  2) В цикле для найденных вершин за исключением тех, которые уже есть в списке просмотренных вершин, выполняем п.1) и продолжаем пока ест вершины

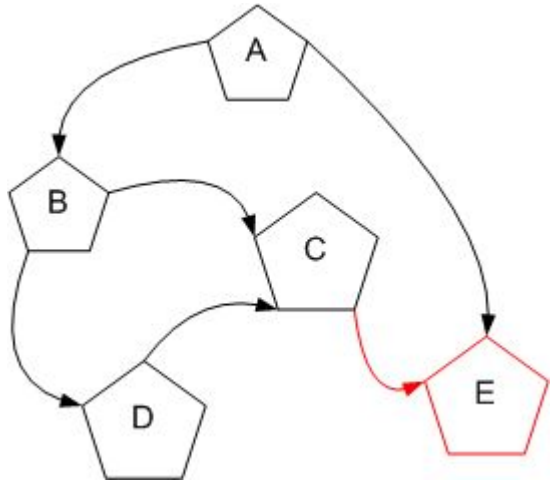
Ребро	Из вершины	В вершину
1	A	B
2	A	E
3	B	C
4	B	D
5	D	C
6	C	E



Графы. Список ребер

- Получение потомков:
  1) Находим вершины выводящие из текущей вершины, включаем их в список просмотренных вершин
  2) В цикле для найденных вершин за исключением тех, которые уже есть в списке просмотренных вершин, выполняем п.1) и продолжаем пока есть вершины

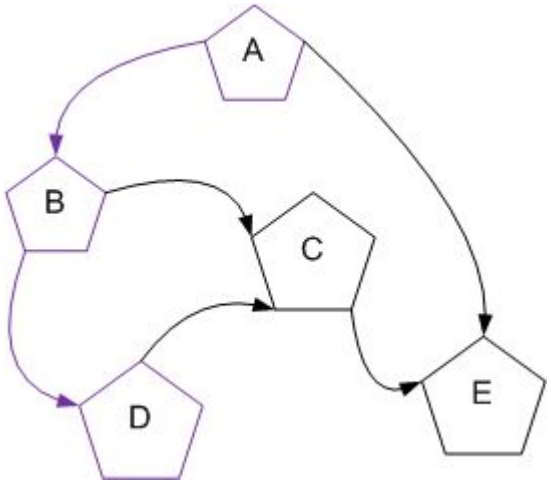
Ребро	Из вершины	В вершину
1	A	B
2	A	E
3	B	C
4	B	D
5	D	C
6	C	E



Графы. Список ребер

- Получение пути:
  - Аналогично процессу получения родителей для начальной вершины, но ищем до получения конечной вершины
  - Или аналогично процессу получения потомков для начальной вершины, но ищем до получения конечной вершины

Ребро	Из вершины	В вершину
1	A	B
2	A	E
3	B	C
4	B	D
5	D	C
6	C	E

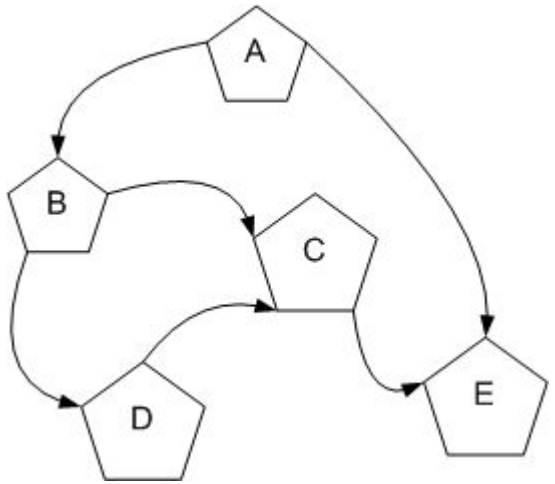


Графы. Список ребер

- Преимущества:
  - Простота ведения
- Недостатки:
  - Сложность поиска пути
  - Сложность поиска родителей и потомков
- Варианты:
  - Список достижимости

Графы. Список достижимости

Ребро	Из вершины	В вершину	Начальное ребро
1	A	B	
2	B	C	
3	A	C	1
4	B	D	
5	A	D	1
6	D	C	
7	C	E	
8	B	E	2
9	A	E	3
10	D	E	6
11	A	E	



Графы. Список достижимости

- Добавление:
  - Добавление ребра
  - Добавление всех путей ведущих по этому ребру
- Перемещение:
  - Удаление ребра с зависимыми путями
  - Добавление ребра с зависимыми путями
- Удаление:
  - Удаление ребра и всех зависимых путей

Ребро	Из вершины	В вершину	Начальное ребро
1	A	B	
2	B	C	
3	A	C	1
4	B	D	
5	A	D	1
6	D	C	
7	C	E	
8	B	E	2
9	A	E	3
10	D	E	6
11	A	E	

Графы. Список достижимости

- Получение родителе:
  - Выбрать все «ребра» и их вершины по которым достижима заданная вершина
- Получение потомков:
  - Выбрать все «ребра» и их вершины которые достижимы из заданной вершины
- Получение пути:
  - Найти «ребро» в котором присутствуют обе вершины
  - В цикле построить полный путь

Ребро	Из вершины	В вершину	Начальное ребро
1	A	B	
2	B	C	
3	A	C	1
4	B	D	
5	A	D	1
6	D	C	
7	C	E	
8	B	E	2
9	A	E	3
10	D	E	6
11	A	E	

Графы. Список достижимости

- Преимущества:
  - Пути строятся при ведении графа
  - Простота навигации по графу
- Недостатки:
  - Небольшая сложность ведения графа

Конец

Храните данные в базах данных

… и регулярно делайте backup!