Лекция №11 10.11.23

Графовая СУБД Neo4j (Продолжение)

Neo4j - данная графовая СУБД позволяет разрабатывать системы, основанные на графовой модели данных.

Основные объекты:

1. **Вершины** - используются для представления сущностей (самый простой граф состоит из 1 вершины). Вершина может иметь именованные значения (указываются в виде свойств).
2. **Связи** - организуют узлы, соединяя их. Связь соединяет 2 узла - начальный и конечный. Связи также могут иметь свойства.
3. **Свойства** - именованные значения. Могут быть числовыми, двоичными, строковыми, могут содержать списки.
4. **Метки** - представляют собой графы, сгруппированные в наборы. Служат для упрощения написания запросов. Вершина может быть помечена любым количеством меток. Используется для задания ограничений и добавления индексов для свойств.

Для графовых баз данных существует свой язык - `Cypher`.

Команды языка Cypher:

1. `MATCH` - поиск данных по шаблону.
2. `START` - начальная точка или точка входа.
3. `CREATE` - создание узлов, отношений и свойств.
4. `MERGE` - проверка на существование заданного шаблона.
5. `SET` - обновление свойств, узлов и связей.
6. `DELETE` - удаление узлов и связей.
7. `REMOVE` - удаление свойств и элементов.
8. `FOREACH` - обновление данных в списке.

9. `RETURN` - определение, что включить в результаты запроса.
10. `ORDER BY` - упорядочить вывод (используется вместе с 9 командой).
11. `LIMIT` - ограничение строк в результате.
12. `WITH` - объединить части запроса.
13. `UNION` - объединить результат.

Создание узлов

Создание единственного узла:

```
CREATE (node_name)
```

Создание всех узлов:

```
CREATE (node_name)
MATCH (n) RETURN n
```

Создание нескольких узлов:

```
CREATE (node_name), (s...), (s...)
MATCH (n) RETURN n
```

Создать узел:

```
CREATE (node: label)
```

Пример создания узла:

```
CREATE (n: player)
```

Создать узел со свойствами:

```
CREATE (node: label {key 1: value, key 1: value, ...})
```

Пример создания узла со свойствами:

```
CREATE (n: player {name: 'Иванов', yo: '1960'})
RETURN N
```

Создание взаимосвязи

Создание взаимосвязи:

```
CREATE(node1) - [:relation Type] -> (node 2)
```

Пример создания взаимосвязи:

```
MATCH (a: label), (b: label)
WHERE a.name = "..." AND b.name = "..."
CREATE (a) - [:relation Type] -> (b)
```

Создание взаимосвязи с метками или свойствами:

```
CREATE(node1) - [:relation Type {key 1: value, key 1: value, ...})] -> (node 2)
```

Создание свойств и меток

Создание нового свойства в узле:

```
MATCH (node: label {key 1: value, key 1: value, ...})
SET node.<имя свойства> = "..."
```

Создание нескольких свойств в узле:

```
MATCH (node: label {key 1: value, key 1: value, ...})
```

```
SET node.<имя свойства> = "...", "...", ...
```

Аналогично этому создаются новые метки.

Удаление свойств и меток

Удаление всех узлов и отношений:

```
MATCH (n) DETACH DELETE n
```

Удаление отдельного узла:

```
MATCH (n) DETACH DELETE n
```

Удаление отдельного свойства:

```
MATCH ... REMOVE n.<имя свйоства>
```

Аналогично этому удаляются метки.

Выборки

Выборки по метке:

```
MATCH (node: label)
RETURN n
```

Выборки на основе отношений:

```
MATCH (node: label) <- [:relation Type] - (n)
RETURN n
```

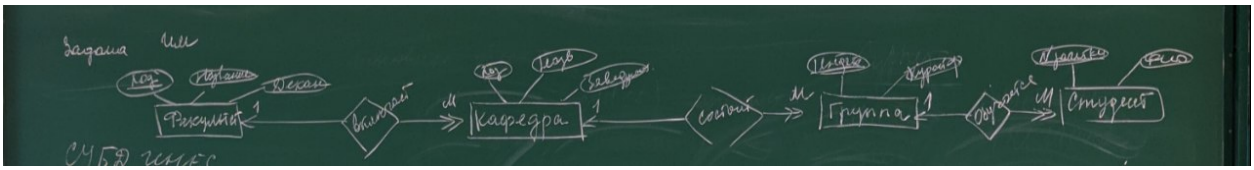
Поиск

Поиск по агрегированию:

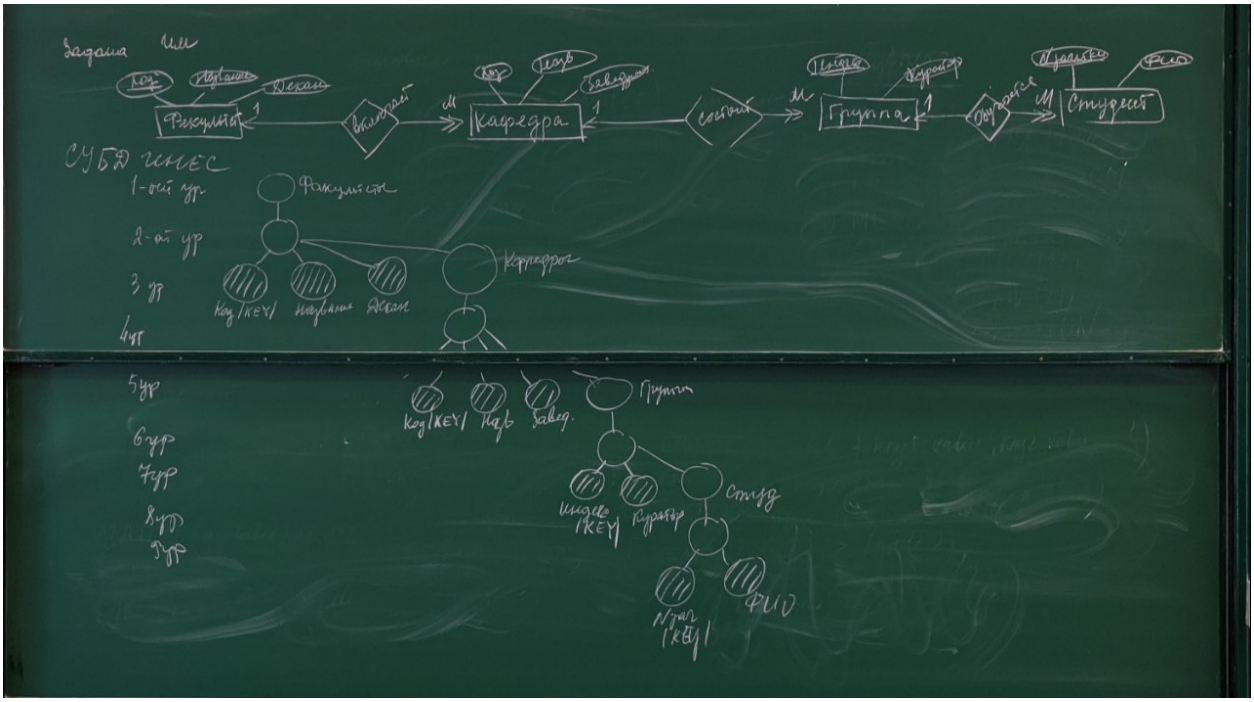
```
ORDER BY n.<имя свойства>
```

Пример

Задана информационная модель:



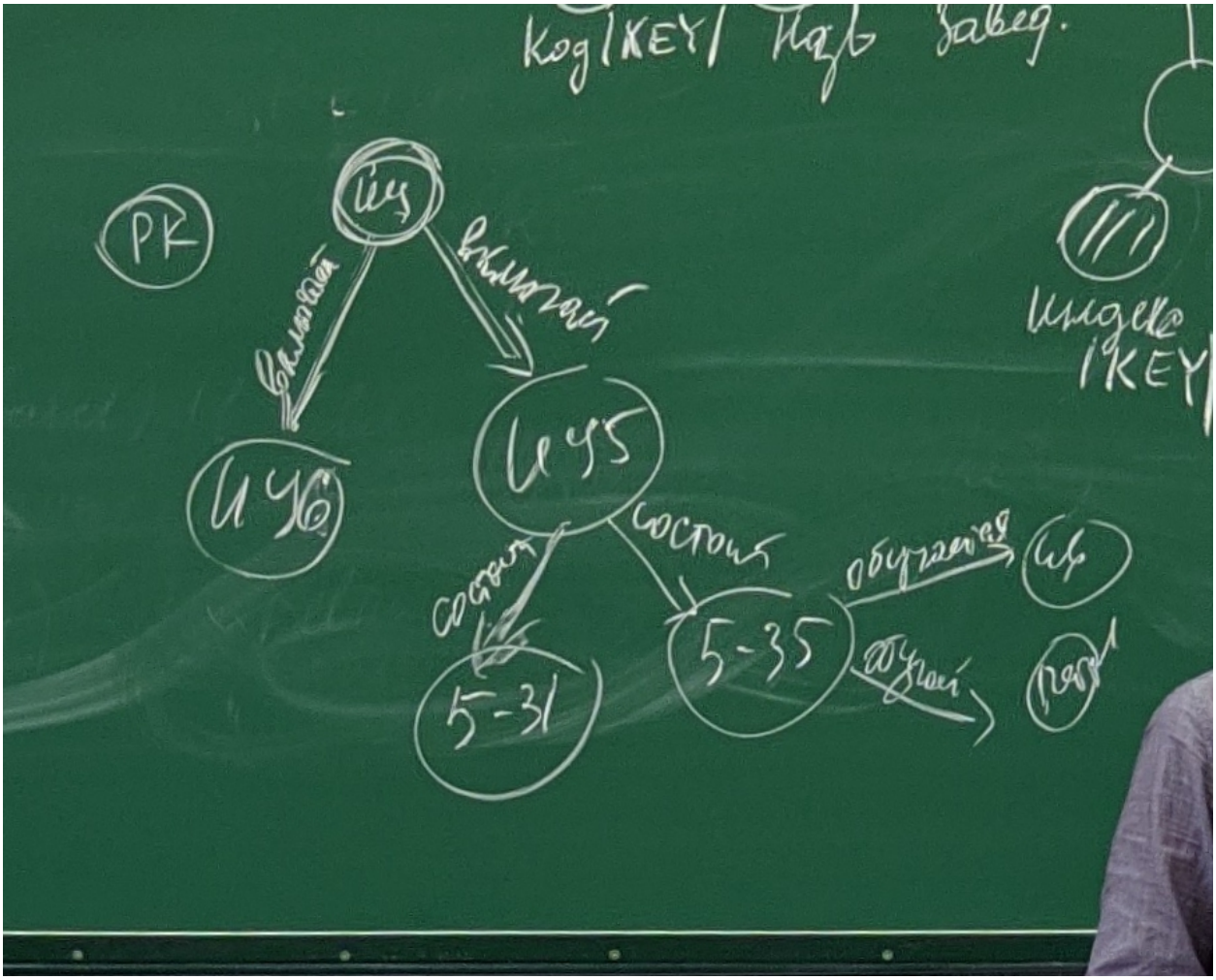
Необходимо спроектировать даталогическую модель `СУБД ИНЭС`:



Создание графовой БД:

```
CREATE (: факультет{код: "ИУ", название: "Инфор", декан: "П"})
CREATE (: факультет{код: "РК", название: "...", декан: "И"})
CREATE (: кафедра {код: "ИУ5"})
CREATE (: кафедра {код: "ИУ6"})
MATCH (a: факультет), (b: кафедра)
WHERE a.код = "ИУ" and b.код = "ИУ5"
CREATE (a) - [:включает] -> (b)
```

```
MATCH (n: кафедра)
WHERE (n.код = "ИУ5")
SET n.название = "СОИиУ"
```



Функциональная модель IDEF0

Методология SADT - методология для построения функциональной модели предметной области.

В `SADT` входят:

1. `IDEF0` - методология для создания модели, которая отражает функции системы, а также потоки информации.
2. `IDEF1` - методология для построения информационной модели, которая отражает содержание информационных потоков.

3. `IDEF2` - методология для построения динамической модели, которая отражает поведение функции во времени.

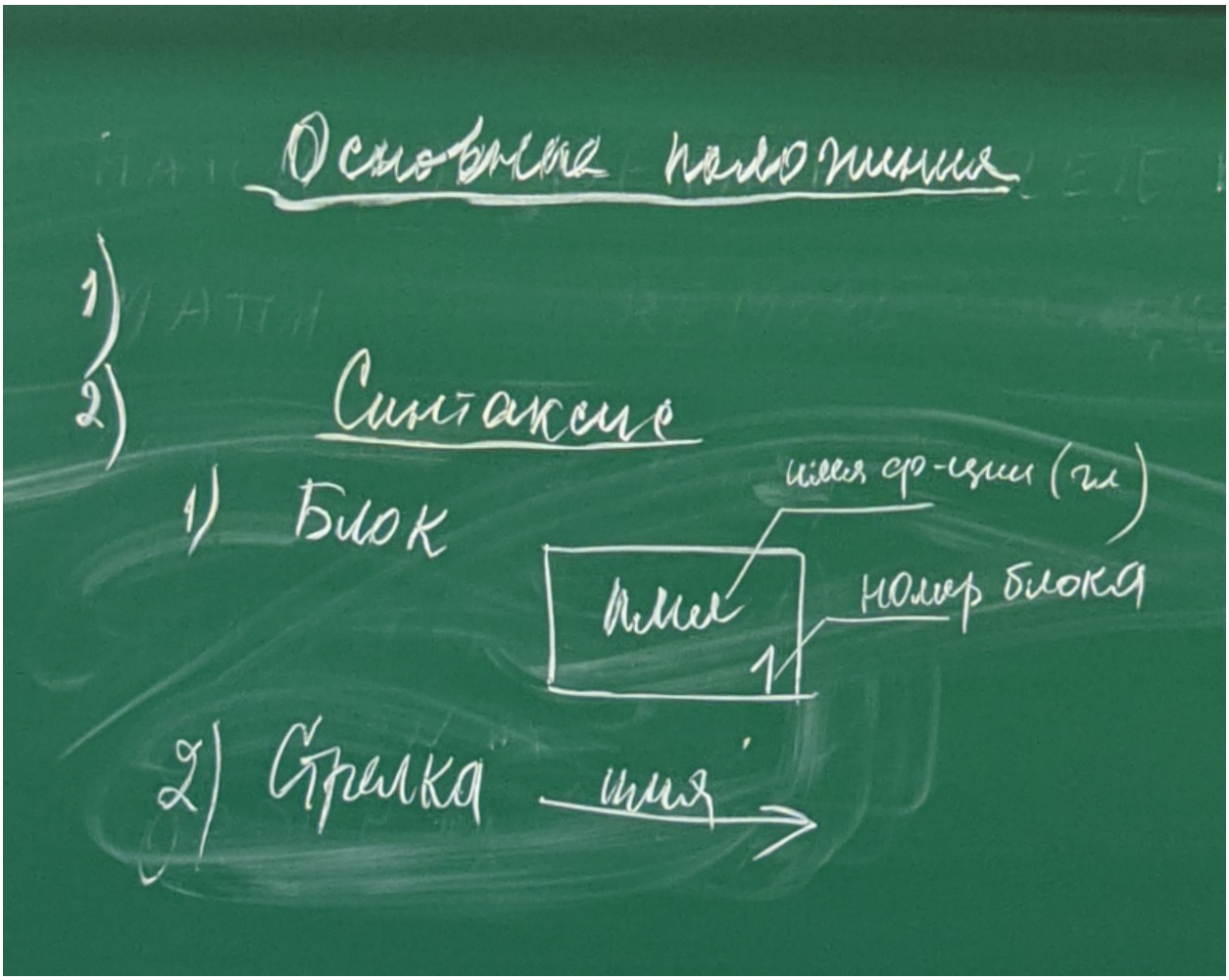
В настоящее время используют `IDEF0` и `IDEF1`.

Основные положения:

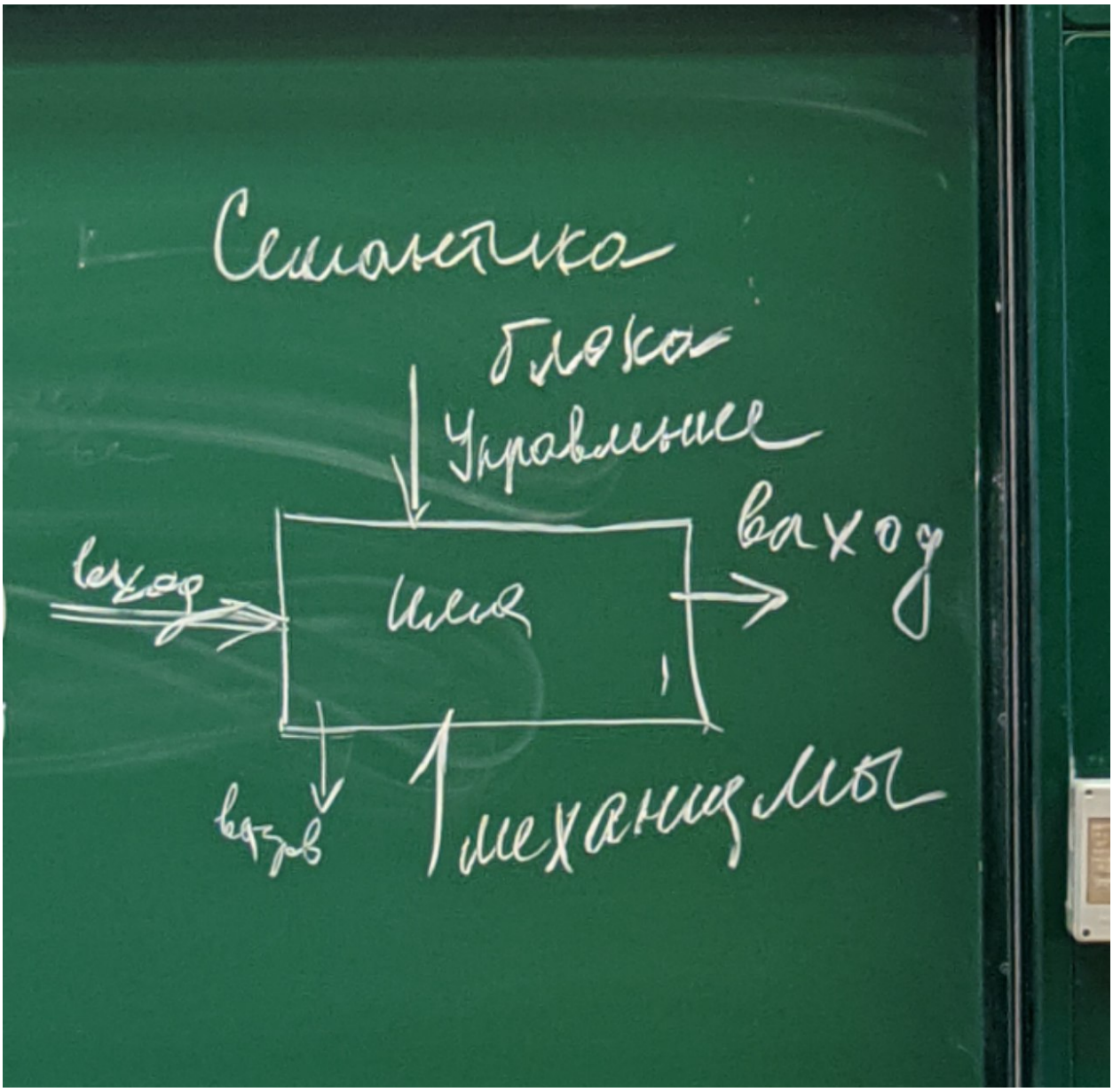
1. Представление системы в виде взаимосвязанных блоков.
2. Блоки отражают процессы, операции и действия.
3. Графическое представление.
4. Точность и лаконичность.
5. Обеспечение передачи информации.
6. Строгие формализованные правила.
7. Пошаговая итеративная разработка.
8. Отделение организации от функций.

Синтаксис:

1. `Блок` - описывает функции.
2. `Стрелка` - описывает данные.



Семантика блока:



Вход - объект модели, отражающий модели информацию, который используется функцией для получения выхода.

Управление - объект модели, отражающий правила, ограничения и стандарты.

Выход - объект модели, отражающий объект производящей функции.

Механизм - объект модели, отражающий ресурсы, которые выполняют функцию.

Вызов - объект модели, указывающий на другую функцию.

Правила построения функциональной модели IDEF0