**Язык Cypher. Выборка. Агрегатные, строковые и числовые функции.**

Общая структура запроса для чтения:

**[MATCH WHERE]**

**[OPTIONAL MATCH WHERE]**

**[WITH [ORDER BY] [SKIP] [LIMIT]]**

**RETURN [ORDER BY] [SKIP] [LIMIT]**

**OPTIONAL MATCH**

**OPTIONAL MATCH (n) - [r] -> (m)**

Схож с MATCH, но может использовать **null** в случае отсутствия компонентов шаблона.

**WHERE**

**WHERE n.property <> $value**

Предложение WHERE используется для фильтрации. Отметим, что WHERE всегда является частью предложений MATCH, OPTIONAL MATCH или WITH. Помещение его после другого предложения в запросе изменит то, что он делает.

**WHERE EXISTS {**

**MATCH (n)-->(m) WHERE n.age = m.age**

**}**

Использование подзапроса для фильтрации.

**RETURN, DISTINCT, ORDER BY, SKIP, LIMIT**

**RETURN \***

Возвращает значение всех переменных.

**RETURN n AS columnName**

Задания псевдонима **columnName** (alias) для переменной **n**.

**RETURN DISTINCT n**

Возвращает только уникальные значения переменной **n**.

**ORDER BY n.property**

Сортировка результата.

**ORDER BY n.property DESC**

Сортировка результата по убыванию свойства **property**.

**SKIP $skipNumber**

Определяет, с какой позиции результаты запроса попадут в выходной набор.

**LIMIT $limitNumber**

Ограничение количества выходных строк.

**SKIP $skipNumber LIMIT $limitNumber**

Пропускаем несколько первых выходных результатов и ограничиваем итоговый выходной набор.

**RETURN count(\*)**

Количество одинаковых результатов в выходных наборов.

**WITH**

**MATCH (user)-[:FRIEND]-(friend)**

**WHERE user.name = $name**

**WITH user, count(friend) AS friends**

**WHERE friends > 10**

**RETURN user**

Синтаксис WITH схож с RETURN. Он явно разделяет части запроса, позволяя вам объявить, какие переменные переносятся в следующую часть.

**MATCH (user)-[:FRIEND]-(friend)**

**WITH user, count(friend) AS friends**

**ORDER BY friends DESC**

**SKIP 1**

**LIMIT 3**

**RETURN user**

ORDER BY, SKIP и LIMIT может также использоваться с WITH.

**Aggregating functions**

**count(\*)**

Считает количество найденных строк.

**count(variable)**

Считает количество найденных ненулевых значений переменных.

**count(DISTINCT variable)**

Внутри любой агрегатной функции может применяться оператор **DISTINCT**, который может удалять повторы значений переменной.

**collect(n.property)**

Создание списка из значений свойств. **Null** игнорируются.

**sum(n.property)**

Суммируют численные значения свойства. Схожие функции **avg()**, **min()**, **max()**.

**percentileDisc(n.property, $percentile)**

Дискретный процентиль. Непрерывный процентиль - это percentileCont (). Аргумент процентиля составляет от 0,0 до 1,0.

**null**

**null** используется для представления отсутствующих или незаданных значений. **null** не равен **null**, т.е. отсутствие информации о значении двух переменных не означает, что они имеют равные значения. В соответствии с данной парадигмой выражение **null = null** возвращает **null**, а не **true**. Для проверки на то, что значение выражение есть **null** используется выражение **IS NULL**. Арифметические выражения, сравнения и вызов функций (за исключением **coalesce**) вернет **null**, если любой ее аргументы является **null**.

При попытке обращения к отсутствующему элементу в списке или к свойству вернет null. В предложении **OPTIONAL MATCH** **null** будет использовано для заполнения отсутствующих частей в шаблоне.

**String functions**

**toString($expression)**

Переводит выражение в строку.

**replace($original, $search, $replacement)**

Заменяет все вхождения **$search** в **$original** на **$replacement**. Все аргументы должны быть выражениями.

**substring($original, $begin, $subLength)**

Выделяет подстроку в строке **$original** начиная с позиции **$begin** длиной **$subLength**. Аргументы **$subLength** не является обязательным и может быть опущен.

**left($original, $subLength),**

**right($original, $subLength)**

Выделяет подстроку в строке **$original** начиная с первой позиции (**left**)или с последней в обратном порядке (**right**)длиной **$subLength**. Аргументы **$subLength** не является обязательным и может быть опущен.

**trim($original)**

**lTrim($original)**

**rTrim($original)**

Удаление пробелов.

**toUpper($original), toLower($original)**

Первеод символов букв либо в верхний регистр, либо в нижний регистр для строки **$original**.

**split($original, $delimiter)**

Заменяем строку **$original** с разделителем **$delimiter** на список.

**reverse($original)**

Возвращает строку с обратным порядком символов.

**size($string)**

Подсчитывает количество символов в строке.

**Mathematical functions**

**abs($expr)**

Абсолютное значение полученного выражения.

**rand()**

Возвращает случайное число от 0 (включительно) до 1 (не включительно), т.е. [0,1). Возвращает новое значение для каждого вызова.

**round($expr)**

**ceil()**

**floor()**

Округляет до ближайшего целого (**round**) или до ближайшего большего (**ceil**) или меньшего целого (**floor**).

**sqrt($expr)**

Вычисляет квадратный корень

**sign($expr)**

Моделирует математическую функцию sign, т.е. -1, если **$expr<0**; 0, если **$expr=0**; иначе 1

**sin($expr)**

Остальные тригонометрические функции также включены cos(), tan(), cot(), asin(), acos(), atan(), atan2(), и haversin(). Аргументы для тригонометрических функций по умолчанию воспринимаются как заданные в радианах.

**degrees($expr), radians($expr), pi()**

Функции для преобразования радианов в градусы (**degrees($expr)**) и наоборот (**radians($expr)**). Функция **pi()** возвращает значениеπ.

**log10($expr), log($expr), exp($expr), e()**

Функции для вычисления десятичного **log10($expr)** и натурального **log($expr)** логарифма. Функция **exp($expr)** служит для вычисления экспоненты, а функция **e()** возвращает значения числа e.

**Relationship functions**

**type(a\_relationship)**

Строковое представления типа отношений.

**startNode(a\_relationship)**

Стартовое поле отношения.

**endNode(a\_relationship)**

Конечное поле отношения.

**id(a\_relationship)**

Внутреннее id отношения.