Выступление: Queries in AEM. JCR-SQL2

слайд 1 Thema:

Добрый день, меня зовут Павел Носов и я хотел бы представить вам презентацию на тему Queries in AEM. JCR-SQL2. Я из кампании Hight Torque Software

слайд 2 Agenda:

И хотелось бы рассмотреть две темы, общая Грамматика, которую должны поддерживать все имплементации JCR2 и фичи присущие только Jackrabbit/CRX или же проще говоря AEM.

слайд 3 Grammar:

Начнём с рассмотрения грамматики JCR-SQL2. При рассмотрении будем опираться на синтаксические диаграммы(railraod diagram) построенные по спецификации JCR 2.0 specification. Найти их можно по ссылкам представленным на слайде. По скольку CQ/AEM уважают JCR спецификации то эти диаграммы описываю тот синтаксический минимум, который должен быть представлен в реализации JCR, и значит на них можно опираться при построении SQL2-JCR запросов. По факту синтаксис поддерживаемый CQ/AEM шире чем тот что представлен на диаграммах. Но об этом позже.

слaйд 4 Grammar:Query structure:

На слайде представлена синтаксическая диаграмма запроса JCR-SQL2. Как видно из диаграммы основная часть Запрос состоит из ключевого слова SELECT, затем column(перечисление полей), затем ключевое слово FROM, затем selector(селектор). Все последующие элементы опциональны. Из них можно выделить: join; constrain (ограничения), которому предшествует ключевое слово WHERE; ordering(сортировки), которому предшествует ключевое слово ORDER BY

Рассмотрим все составляющие элементы по очереди.

слайд 5 Grammar:Selector:

На слайде представлена синтаксическая диаграмма элемента запроса selector. selector необходимо для выбора подмножества nodes в workspace.

В качестве значения для nodeTypeName может выступать зарегистрированный nodeType у которого значение аттрибута queryable=true.

слайд 6 Grammar:Selector(2):

Зарегистрированные nodeType можно найти при помощи инструмента http://{host}:{port}/crx/explorer/nodetypes/index.jsp . А вот как узнать является ли значение аттрибута queryable=true для меня так и осталось загадкой. Если кто-нибудь знает — скажите мне :). Если значение nodeTypeName валидно, тогда selector выберет все ноды в workspace к которым имеет доступ сессия с которой выполняется запрос, согласно со следующими правилами:

1) нод имеет jcr:primaryType равный nodeTypeName

2) или jcr:primaryType равный подтипу nodeTypeName

3) или jcr:mixinTypes равный nodeTypeName

4) или jcr:mixinTypes равный подтипу nodeTypeName

selectorName это алиса, который можно задать для выбранного множества, который затем может быть использован в запросе. SelectorName в пределах запроса должны быть уникальны. Для задания алиаса используется ключевое слово AS.

http://www.day.com/specs/jcr/2.0/6\_Query.html#6.7.3 Selector

слайд 7 Grammar:Selector(3):

На слайде представлена таблица с примером selector а алиасом и без

слайд 8 Grammar:Column:

На слайде представлена диаграмма элемента запроса column. colomn необходима для задания колонок, входящих в таблицу генерируемую при таблично отображении результатов поиска. Значением selectorName может выступать один из алиасов объявленных для элемента запроса selector. Значением для properyName являются имена пропертей нода. Для каждой проперти можно задать алиас при помощи ключевого слова AS. Значение алиаса указывается вместо columnName.

слайд 9 Grammar:Column(2):

В таблице представлены примеры column. Видно, что если проперти имеет имя, в которое входят символы кроме [a-z0-9-\_], то имя проперти берётся в [].

слайд 10 Column and Table View:

Для демонстрации того как строиться таблица результатов создадим компонент. [ПОКАЗАТЬ КОД КОМПОНЕНТА]. Для того чтобы получить данный для отображения таблицы компонент использует код представленный на слайде.

слайд 11 Column and Table View(2):

Рассмотрим первый запрос в котором в качестве блока column используется … \* ….

слайд 12 Column and Table View(3):

Как видно имена столбцов результирующей таблицы формируются из nodeType + '.' + propertyName

слайд 13 Column and Table View(4):

Рассмотрим второй запрос в котором в качестве блока column используется … selectorName.\* ….

слайд 14 Column and Table View(5):

Как видно имена столбцов результирующей таблица

формируются из selectorName + '.' + propertyName

слайд 15 Column and Table View(6):

Рассмотрим третий запрос в котором в качестве блока column используется … selectorName.propertyName AS columnName ….

слайд 16 Column and Table View(7):

Как видно имена столбцов результирующей таблица columnName

слайд 17 Grammar:Join:

На данном слайде представлена синтаксическая диаграмма блока join. Поддержка join в иплементация опционально. Но в AEM поддерживаются INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN. Давайте разберёмся в чем разница между разными join.

слайд 18 Grammar:Join(2):

Вот что написано в JCR2 спецификации

слайд 19 Grammar:Join(3):

Я ничего не понял

слайд 20 Grammar:Join(4):

слайд 21 Grammar:Join(4):

А вот тоже самое но с картинками

слайд 22 Grammar:Join(6):

В качестве JoinCondition может выступать EquilJoinCondition (проверка на равенство пропертей объединяемых подмножеств), SameNodeJoinCondition, ChildNodeJoinCondition, DescendantNodeJoinCondition.

При EquiJoinCondition все элементы обязательные, если один из них пропустить - запрос будет не валидным.

слайд 23 Grammar: Join:ISSAMENODE:

В качестве JoinCondition может выступать SameNodeJoinCondition. Если selectorPathName пропущена, тогда условие равенства нод проверяется вырожением эквивалентным [ПРОЧИТАТЬ СО СЛАЙДА]

selector1Node.isSame(selector2Node)

Если selectorPathName присутствует, тогда для проверки условия используется [ПРОЧИТАТЬ СО СЛАЙДА]

selector1Node.isSame(selector2Node.getNode(selector2Path))

слайд 24 Grammar: Join:ISSAMENODE(2):

запрос с selectorPathName

слайд 25 Grammar: Join:ISSAMENODE(3):

еще один запрос с selectorPathName

слайд 26 Grammar: Join:ISCHILDNODE:

В качестве JoinCondition может выступать ChildNodeJoinCondition. Для проверки используется выражение эквивалентное:

childSelectorNode.getParent().isSame(parentSelectorNode)

где childSelectorNode это нод выбранный селектором childSelectorName и parentSelectorNode это нод выбранный селектором parentSelectorName

слайд 27 Grammar: Join:ISDESCENDANTNODE:

В качестве JoinCondition может выступать DescendantNodeJoinCondition. Для проверки используется выражение эквивалентное:

descendantSelectorNode.getAncestor(n).  
isSame(ancestorSelectorNode) &&  
descendantSelectorNode.getDepth() > n

где n >= 0, и descendantSelectorNode — это нод выбранный селектором descendantSelectorName и ancestorSelectorNode — это нод выбранный селектором ancestorSelectorName

слайд 28 Grammar: Constraint, AndCondition:

На слайде представлены диаграммы блоков constraint и andCondition

слайд 29 Grammar: Condition:

На слайде представлена диаграмма блока Condition. Видно что в качестве блока condition может быть использован блок comparison, NOT comparison, проверка наличия или отсутствия проперти, FullTextSearch, SameNode, ChildNode, DescendantNode.

слайд 30 Grammar: Condition(2):

Все возможные варианты представлены в таблице на слайде. Пройдёмся по некоторым из них

SameNode, ChildNode, DescendantNode работаю точно так же как и для join блока, за исключением того что перый параметр не обязателен если в запросе участвует только один селектор, а второй параметр это литерал представляющий абсолютный путь.

слайд 31 Grammar: Condition(3):

Убедимся в этом выполнив запросы:

SELECT \* FROM [cq:Page] AS page INNER JOIN [nt:base] AS node ON ISDESCENDANTNODE(node, page) WHERE ISDESCENDANTNODE(page, node.test)

SELECT \* FROM [cq:Page] AS page INNER JOIN [nt:base] AS node ON ISDESCENDANTNODE(node, page) WHERE ISDESCENDANTNODE(page, '/content/geometrixx/en/toolbar')

слайд 32 Grammar: Comparison:

На слайде представлена диаграмма блока Comparison. Особое внимание хотелось бы уделить оператору LIKE. При его использовании в литерале в правой части выражения можно использовать спецсимволы '%' и '\_'. '%' - означает 0 или больше символов, '\_' - ровно один любой символ.

слайд 33 Grammar: Comparison(2):

Убедимся в этом выполнив запросы:

SELECT \* FROM [nt:base] WHERE ISDESCENDANTNODE('/content/geometrixx/en/toolbar') AND [jcr:path] LIKE '/content/geometrixx/en/toolbar/jcr\_content'

SELECT \* FROM [nt:base] WHERE ISDESCENDANTNODE('/content/geometrixx/en/toolbar') AND [jcr:path] LIKE '/content/geometrixx/en/toolbar/jcr\_content%'

слайд 34 Grammar: Static Operands:

На слайде представлена диаграмма блока static operands. Данное выражение позволяет каставать литерал к типам данных. В итоге данное выражение можно использовать в правой части сравнения

слайд 35 Grammar: Static Operands(2):

Особое внимание уделим операции приведения строки к дате. Рассмотрим следующий запрос:

SELECT \* FROM [nt:base] WHERE ISDESCENDANTNODE('/content/geometrixx/en/toolbar') AND [cq:lastModified] > CAST('2016-04-26T23:51:02.000+03:00' AS DATE)

слайд 36 Grammar: Static Operands(3):

Стоит ли напомнить что CQ имеет особенный формат даты

слайд 37 Grammar: Literals:

На слайде представлена синтаксическая диаграмма литералов

слайд 38 Grammar: Dynamic Operand:

На слайде представлена синтаксическая диаграмма Динамических Операторов.

слайд 39 Grammar: Dynamic Operand(2):

Выделим из них оператор NAME(). При помощи него можно получить имя нода выделенного селектором:

SELECT \* FROM [nt:base] WHERE ISDESCENDANTNODE('/content/geometrixx/en/toolbar') AND NAME() LIKE '%par%'

слайд 40 Grammar: Ordering:

На данном слайде представлена диаграмма блока Ordering.

слайд 41 AEM Specific:

Перейдём к особенностям реализации диалекта в AEM.

слайд 42 rep:excerpt():

Нет описания

слайд 43 rep:spellcheck():

Нет описания

слайд 44 rep:similar():

Нет описания

слайд 45 Searching with synonyms :

Нет описания

слайд 46 rep:suggest():

Нет описания

слайд 47 OAK Specific: IN:

AEM имплементация SQL2 с версии 6.х поддерживает использование IN оператора. Правда вложенные запросы не поддерживаются. Рассмотрим запрос:

SELECT \* FROM [nt:base] WHERE ISDESCENDANTNODE('/content/geometrixx/en/toolbar') AND NAME() IN ('par', 'jcr:content') ORDER BY NAME()

Как можно заметить, значения для IN это перечисление литералов.

слайд 48 rep:suggest():

Нет описания

слайд 49 OAK Specific: NATIVE:

OAK имплементация SQL2 поддерживает использование динамического оператора NATIVE. Рассмотрим запрос:

SELECT [jcr:path] FROM [nt:base] WHERE NATIVE('solr', 'mlt?q=id:UTF8TEST&mlt.fl=manu,cat&mlt.mindf=1&mlt.mintf=1')

Первый операнд — это имя индекса, второй это литерал с запросом в синтаксисе поддерживающемся индексом