RDB via Snapshots Scanner

The *RDBMS Scanner* component performs full and incremental scans over a database. This version of RDBMS scanner performs full scanning using a SQL statement to extract the data from one or more *data tables*. This scanner use snapshot files to determine which content has been updated.

Updated content is submitted to the configured pipeline in AspireObjects attached to Jobs, with every column extracted from the data tables added to the AspireObject. Updated content is split into three types - *add* , *update*, and *delete*. Each type of content is published on a different event so that it may be handled by different Aspire pipelines.

This RDBMS Scanner version is also capable of fetching ACLs data by two different ways: defining a SQL statement or defining a the *column name* that contains the ACLs.

The scanner reacts to an incoming job. This job may instruct the scanner to *start*, *stop*, *pause* or *resume*. The *start* job will contain the database connection parameters and SQL for data extraction, although this may be defaulted in the component configuration in the application. xml file. When pausing or stopping, the scanner will wait until all the jobs it published have completed before itself completing.

RDB via Snapshots Scanner	
Factory Name	com.searchtechnologies.aspire:aspire-rdb-connector
subType	defaut
Inputs	AspireObject from a content source submitter holding all the information required for a crawl
Outputs	Jobs from the crawl

? Unknown Attachment

Configuration

This section lists all configuration parameters available to configure the RDB via Snapshots Scanner component.

General Scanner Component Configuration

Basic Scanner Configuration

Element	Type	Default	Description
snapshotDir	String	snapshots	The directory for snapshot files.
numOfSnapshotBackups	int	2	The number of snapshots to keep after processing.
waitForSubJobsTimeout	long	600000 (=10 mins)	Scanner timeout while waiting for published jobs to complete.
maxOutstandingTimeStatistics	long	1m	The max about of time to wait before updating the statistics file. Whichever happens first between this property and maxOutstandingUpdatesStatistics will trigger an update to the statistics file.
maxOutstandingUpdatesStatistics	long	1000	The max number of files to process before updating the statistics file. Whichever happens first between this property and maxOutstandingTimeStatistics will trigger an update to the statistics file.
usesDomain	boolean	true	Indicates if the group expansion request will use a domain\user format (useful for connectors that does not support domain in the group expander).

Branch Handler Configuration

This component publishes to the *onAdd*, *onDelete* and *onUpdate*, so a branch must be configured for each of these three events.

Element	Type	Description
branches/branch/@event	string	The event to configure - *onAdd*, *onDelete* or *onUpdate*.
branches/branch/@pipelineManager	string	The name of the pipeline manager to publish to. Can be relative.
branches/branch/@pipeline	string	The name of the pipeline to publish to. If missing, publishes to the default pipeline for the pipeline manager.
branches/branch/@allowRemote	boolean	Indicates if this pipeline can be found on remote servers (see Distributed Processing for details).
branches/branch/@batching	boolean	Indicates if the jobs processed by this pipeline should be marked for batch processing (useful for publishers or other components that support batch processing).
branches/branch/@batchSize	int	The max size of the batches that the branch handler will created.
branches/branch/@batchTimeout	long	Time to wait before the batch is closed if the batchSize hasn't been reached.

branches/branch /@simultaneousBatches	int	The max number of simultanous batches that will be handled by the branch handler.

File System Specific Configuration

Element	Type	Default	Description
rdb	string		The Multi RDBMS Connection Pool component to use for RDB connections.

Configuration Example

```
<component name="RDBMSRDB" subType="default" factoryName="aspire-multiple-rdb">
  <debug>true</debug>
  <timeout>30m</timeout>
  <purgePoll>5m</purgePoll>
</component>

<component name="Scanner" subType="default" factoryName="aspire-rdb-connector">
  <debug>true</debug>
  <rdb>./RDBMSRDB</rdb>
  <branches>
    <branch event="onAdd" pipelineManager="../ProcessPipelineManager"
      pipeline="addUpdatePipeline" allowRemote="true" batching="true"
      batchSize="50" batchTimeout="60000" simultaneousBatches="2" />
    <branch event="onUpdate" pipelineManager="../ProcessPipelineManager"
      pipeline="addUpdatePipeline" allowRemote="true" batching="true"
      batchSize="50" batchTimeout="60000" simultaneousBatches="2" />
    <branch event="onDelete" pipelineManager="../ProcessPipelineManager"
      pipeline="deletePipeline" allowRemote="true" batching="true"
      batchSize="50" batchTimeout="60000" simultaneousBatches="2" />
  </branches>
</component>
```

Source Configuration

Scanner Control Configuration

The following table describes the list of attributes that the AspireObject of the incoming scanner job requires to correctly execute and control the flow of a scan process.

Element	Type	Options	Description
@action	string	start, stop, pause, resume, abort	Control command to tell the scanner which operation to perform. Use **start** option to launch a new crawl.
@actionProperties	string	full, incremental	When a **start** @action is received, it will tell the scanner to either run a **full** or an **incremental** crawl.
@normalizedCSName	string		Unique identifier name for the content source that will be crawled.
displayName	string		Display or friendly name for the content source that will be crawled.

Header Example

```
  <doc action="start" actionProperties="full" actionType="manual" crawlId="0" dbId="0" jobNumber="0"
normalizedCSName="FeedOne_Connector"
   scheduleId="0" scheduler="##AspireSystemScheduler##" sourceName="ContentSourceName">
    ...
    <displayName>testSource</displayName>
    ...
  </doc>
```

All configuration properties described in this section are relative to /doc/connectorSource of the Aspire Object of the incoming Job.

Property	Type	Default	Description
rdbmsJDBCUrl	string		The JDBC URL for your RDBMS server and database. For example, "jdbc:mysql://192.168.40.27/mydb" (MySQL). This will vary depending on the type of RDBMS.
rdbmsJDBCUser	string		The name of a database user with read-only access to all of the tables which need to be indexed, and write access to the necessary update tables (if update management is handled through the RDB).
rdbmsJDBCPassword	string		The database password

rdbmsJDBCDriverJar	string		Path to the JDBC driver JAR file for your RDBMS. Typically this is placed in the "lib" directory inside your Aspire Home, for example "lib/myjdbcdriver.jar".
rdbmsJDBCDriverClass	string		(Optional)The name of the JDBC driver class if the class name from the META-INF/services/java.sql.Driver file in the driver JAR file should not be used, or that file does not exist in the driver JAR file.
retrievePerBatch	boolean		Determine if the discoverySQL + extractionSQL have to be used (batching mode). If false then the Scanner will perform a single SQL execution using the fullSQL query.
fullSQL	string		The SELECT query to be run for retrieve all documents. This query is used for full or incremental scan. The WHERE clause can be used to specify any required condition for crawling the desired documents. For slicing the discoverySQL, add {SLICES} tag to the WHERE clause. For more details, check the wiki.
discoverySQL	string		The SELECT query to be run for discovering documents. This query is used for full or incremental scan. The WHERE clause can be used to specify any required condition for crawling the desired documents. For slicing the discoverySQL, add {SLICES} tag to the WHERE clause. For more details, check the wiki.
extractionSQL	string		The SELECT query for extracting data for each document found. It is required to include a WHERE clause that contains, at least, the 'IN' expression. For example: "(...) WHERE idColumnName IN {IDS}", where idColumnName corresponds to unique key field name. {IDS} will be automatically replaced by the connector with the corresponding unique key values.
idColumn	string		The name of the column which contains unique identifiers for each row. This is used for both full and incremental crawls and must match the name returned by the SQL. If the column is aliased using the SQL "AS" construct, you should provide the alias name here.
idColumnString	boolean	false	True if the ID column is a string.
useSlices	boolean	false	True if you want to divide the Full crawl SQL into multiple slices. Only works when the id column is an integer.
slicesNum	integer	10	The number of SQL slices to split Full crawl SQL. Only works when **idColumn** is an integer.
indexContainers	boolean	false	True if the Tables will be indexed in the crawl.
waitForSubJobs	integer	600000	Timeout value for the crawler's sub-jobs, in milliseconds.
aclMethod	string	SQL	Possible values: column or SQL. Sets whether the ACLs will be retrieved from a column in the extractionSQL or the fullSQL, or if it will be retrieved by the aclSQL.
aclSQL	string		Retrieves the ACLs for the documents, the result should be in the following format: "domain\user@NT". There must be one row for every ACL on the document

Scanner Configuration Example

```
<doc action="start" actionProperties="full" normalizedCSName="My_RDB_Source">
  <connectorSource>
    <useSlices>false</useSlices>
    <retrievePerBatch>true</retrievePerBatch>
    <discoverySQL>Select ID from main_data</discoverySQL>
    <extractionSQL>Select ID from main_data WHERE ID IN {IDS}</extractionSQL>
    <idColumn>ID</idColumn>
    <idColumnString>false</idColumnString>
    <indexContainers>false</indexContainers>
    <waitForSubJobs>600000</waitForSubJobs>
    <aclMethod>SQL</aclMethod>
    <aclSQL>Select 'mydomain\\user@NT' as ACL from main_data</aclSQL>
    <displayName>RBMS snapshots</displayName>
    <dbUrl>jdbc:mysql://127.0.0.1:3306/test</dbUrl>
    <dbUser>root</dbUser>
    <dbPassword>encrypted:771FA372D6F46E9F0C058B359683E8CB</dbPassword>
    <dbDriverJar>lib/mysql-connector-java-5.1.27-bin.jar</dbDriverJar>
    <dbProps/>
    <dbDriverClass/>
  </connectorSource>
  <displayName>My RDB Source</displayName>
</doc>
```

Output

```
<doc>
  <url>5</url>
  <id>5</id>
  <displayUrl>5</displayUrl>
  <snapshotUrl>001 5</snapshotUrl>
  <repItemType>aspire/row</repItemType>
  <fetchId>5</fetchId>
  <fetchUrl>5</fetchUrl>
  <rdb>../MultiConnectionPool</rdb>
  <docType>item</docType>
  <sourceName>RBMS_snapshots</sourceName>
  <sourceType>database</sourceType>
  <connectorSource>
    <useSlices>false</useSlices>
    <retrievePerBatch>true</retrievePerBatch>
    <discoverySQL>Select ID from main_data</discoverySQL>
    <extractionSQL>Select ID from main_data WHERE ID IN {IDS}</extractionSQL>
    <idColumn>ID</idColumn>
    <idColumnString>false</idColumnString>
    <indexContainers>false</indexContainers>
    <waitForSubJobs>600000</waitForSubJobs>
    <aclMethod>SQL</aclMethod>
    <aclSQL>Select 'mydomain\\user@NT' as ACL from main_data</aclSQL>
    <displayName>RBMS snapshots</displayName>
    <dbUrl>jdbc:mysql://127.0.0.1:3306/test</dbUrl>
    <dbUser>root</dbUser>
    <dbPassword>encrypted:771FA372D6F46E9F0C058B359683E8CB</dbPassword>
    <dbDriverJar>lib/mysql-connector-java-5.1.27-bin.jar</dbDriverJar>
    <dbProps/>
    <dbDriverClass/>
    <contentSourceId>RBMS_snapshots</contentSourceId>
  </connectorSource>
  <action>add</action>
  <connectorSpecific>
    <field name="id">5</field>
  </connectorSpecific>
  <acls>
    <acl access="allow" domain="mydomain" entity="group" fullname="mydomain\user@NT" name="user@NT" scope="
global"/>
  </acls>
</doc>
```

Scanner Operation

Retrieve data per batch

This mode uses SQL taken from the job (*<connectorSource/discoverySQL>*, *<connectorSource/extractSQL>* or configuration) and execute them against the database configured via a Multi RDBMS Connection Pool stage. Each resulting row is formed into an AspireObject using the column names as document elements, and this document is submitted to a pipeline manager using the event configured for inserts. As the document is created, the value of the column identified in the job (*<connectorSource/idColumn>*) is noted as the primary key of the document. The value *insert* will be placed in the *action* attribute of the document.

Column names from the extractSQL query are added to the AspireObject inside the "connectorSpecific" field. If the column names are standard Aspire Object fields, they are added to the root level. See Connector Metadata for further details on which are standard fields.

Example AspireObject from a retrieve data per batch

```
<doc>
  <url>5</url>
  <id>5</id>
  <displayUrl>5</displayUrl>
  <snapshotUrl>001 5</snapshotUrl>
  <repItemType>aspire/row</repItemType>
  <fetchId>5</fetchId>
  <fetchUrl>5</fetchUrl>
  <rdb>../MultiConnectionPool</rdb>
  <docType>item</docType>
  <sourceName>RBMS_snapshots</sourceName>
  <sourceType>database</sourceType>
  <connectorSource>
    <useSlices>false</useSlices>
    <retrievePerBatch>true</retrievePerBatch>
    <discoverySQL>Select ID from main_data</discoverySQL>
    <extractionSQL>Select ID from main_data WHERE ID IN {IDS}</extractionSQL>
    <idColumn>ID</idColumn>
    <idColumnString>false</idColumnString>
    <indexContainers>false</indexContainers>
    <waitForSubJobs>600000</waitForSubJobs>
    <aclMethod>SQL</aclMethod>
    <aclSQL>Select 'mydomain\\user@NT' as ACL from main_data</aclSQL>
    <displayName>RBMS snapshots</displayName>
    <dbUrl>jdbc:mysql://127.0.0.1:3306/test</dbUrl>
    <dbUser>root</dbUser>
    <dbPassword>encrypted:771FA372D6F46E9F0C058B359683E8CB</dbPassword>
    <dbDriverJar>lib/mysql-connector-java-5.1.27-bin.jar</dbDriverJar>
    <dbProps/>
    <dbDriverClass/>
    <contentSourceId>RBMS_snapshots</contentSourceId>
  </connectorSource>
  <action>add</action>
  <connectorSpecific>
    <field name="id">5</field>
  </connectorSpecific>
  <acls>
    <acl access="allow" domain="mydomain" entity="group" fullname="mydomain\user@NT" name="user@NT" scope="
global"/>
  </acls>
</doc>
```

Retrieve everything

This mode uses SQL taken from the job (<*connectorSource/fullSQL*> or configuration) and execute them against the database configured via a Multi RDBMS Connection Pool stage. Each resulting row is formed into an AspireObject using the column names as document elements, and this document is submitted to a pipeline manager using the event configured for inserts. As the document is created, the value of the column identified in the job (<*connectorSource/idColumn*>) is noted as the primary key of the document. The value *insert* will be placed in the *action* attribute of the document.

Column names from SQL queries are added to the AspireObject inside the "connectorSpecific" field. If the column names are standard AspireObject fields, they are added to the root level. See Connector AspireObject Metadata for further details on which are standard fields.

```
<doc>
  <url>5</url>
  <id>5</id>
  <displayUrl>5</displayUrl>
  <snapshotUrl>001 5</snapshotUrl>
  <repItemType>aspire/row</repItemType>
  <fetchId>5</fetchId>
  <fetchUrl>5</fetchUrl>
  <rdb>../MultiConnectionPool</rdb>
  <docType>item</docType>
  <sourceName>RBMS_snapshots</sourceName>
  <sourceType>database</sourceType>
  <connectorSource>
    <useSlices>false</useSlices>
    <retrievePerBatch>false</retrievePerBatch>
    <fullSQL>Select * from main_data</fullSQL>
    <idColumn>ID</idColumn>
    <idColumnString>false</idColumnString>
    <indexContainers>false</indexContainers>
    <waitForSubJobs>600000</waitForSubJobs>
    <aclMethod>SQL</aclMethod>
    <aclSQL>Select 'mydomain\\user@NT' as ACL from main_data</aclSQL>
    <displayName>RBMS snapshots</displayName>
    <dbUrl>jdbc:mysql://127.0.0.1:3306/test</dbUrl>
    <dbUser>root</dbUser>
    <dbPassword>encrypted:771FA372D6F46E9F0C058B359683E8CB</dbPassword>
    <dbDriverJar>lib/mysql-connector-java-5.1.27-bin.jar</dbDriverJar>
    <dbProps/>
    <dbDriverClass/>
    <contentSourceId>RBMS_snapshots</contentSourceId>
  </connectorSource>
  <action>add</action>
  <connectorSpecific>
    <field name="id">5</field>
  </connectorSpecific>
  <acls>
    <acl access="allow" domain="mydomain" entity="group" fullname="mydomain\user@NT" name="user@NT" scope="
global"/>
  </acls>
</doc>
```

Slicing Full SQL

This feature consists in splitting the Full crawl SQL into multiple queries, for the scanning process to be faster. Slicing the *fullSQL* should significantly improve the performance when scanning large databases. Only works when **idColumn is an integer**.

Full crawl SQL Example (with slicing)

Having the following fullSQL:

```
  SELECT C.page_id, C.modified_date,
    FROM mw_content C, mw_published P, mw_old T
    WHERE P.rev_id = C.page_latest AND T.old_id = P.rev_text_id;
```

For the connector to support slices you should add the **{SLICES}** tag into your *WHERE* clause in your *fullSQL* as shown in the next example:

```
  SELECT C.page_id, C.modified_date,
    FROM mw_content C, mw_published P, mw_old T
    WHERE P.rev_id = C.page_latest AND T.old_id = P.rev_text_id AND {SLICES};
```

In this case, there already was a *WHERE* clause, so we added "*AND {SLICES}*". In the case you don't need any condition, your query must contain *W HERE {SLICES}* for slices to work.

Slicing Retrieve data per batch SQL

This feature consists in splitting the discoverySQL into multiple queries, for the scanning process to be faster. Slicing the *extractionSQL* should significantly improve the performance when scanning large databases. Only works when **idColumn is an integer**.

Explore SQL Example (with slicing)

Having the following discoverySQL:

```
  SELECT C.page_id, C.modified_date,
    FROM mw_content C, mw_published P, mw_old T
    WHERE P.rev_id = C.page_latest AND T.old_id = P.rev_text_id AND P.rev_id IN {IDS};
```

For the connector to support slices you should add the **{SLICES}** tag into your *WHERE* clause in your *extractionSQL* as shown in the next example:

```
SELECT C.page_id, C.modified_date,
  FROM mw_content C, mw_published P, mw_old T
  WHERE P.rev_id = C.page_latest AND T.old_id = P.rev_text_id AND P.rev_id IN {IDS} AND {SLICES};
```

In this case, there already was a *WHERE* clause, so we added "*AND {SLICES}*". In the case you don't need any condition, your query must contain *W HERE {SLICES}* for slices to work.