



JCR Query Cheatsheet for AEM

(AEM 6.5 and AEM as a Cloud Service)

Please post any feedback to the [AEM Forums](#)



Adobe Experience Manager



Feature	Use	QueryBuilder	XPath	SQL-2	Index definition	Note
Path restrictions	always	path=/content	/jcr:root/content	select * from [nt:base] where isdescendantnode('/ content')	includedPaths=/content queryPaths=/content (set them the same value!)	Just include paths, do not use "exclud- edPaths"
Detailed path restrictions	if possible	path=/content/dam/wknd type=dam:Asset	/jcr:root/content/dam/wknd// element(*,dam:Asset)	select * from [dam:Asset] as a where isdescendantnode(a, '/ content/dam/wknd')	evaluatePathRestrictions=true	Path restrictions make sense if you need to search just a fraction of the overall indexed data, and this distinction can be made by path.
Node type restrictions	always	path=/content/dam type=dam:Asset	/jcr:root/content/dam// element(*,dam:Asset)	select * from [dam:Asset] as a where isdescendantnode(a, '/ content/dam')	indexRules dam:Asset properties	You should always have a nodetype restriction! Almost all indexes are only used if a matching nodetype restriction is provided.
Node name restrictions	if needed	path=/content/wknd type=cq:PageContent nodename=jcr:content	/jcr:root/content/wknd// element(*, cq:PageContent) [fn:name()='jcr:content']	select * from [cq:PageContent] as a where name()='jcr:content' and isdescendantnode(a, '/ content/wknd')	indexNodeName=true	
Equality restrictions	always	path=/content type=cq:PageContent l_property=jcr:title l_property.value=France	/jcr:root/content// element(*, cq:PageContent)[@jcr:title='France']	select * from [cq:PageContent] as a where [jcr:title]='France' and isdescendantnode(a, '/ content')	name=jcr:title propertyIndex=true	Each condition in the query should be indexed
Fulltext conditions	if needed	path=/ type=cq:Page fulltext=ski touring	/jcr:root//element(*, cq:Page)[jcr:contains(., 'ski touring')]	select * from [cq:Page] where contains(*, 'ski touring')	analyze=true	Prefer fulltext conditions on one field, but if multiple fields should be searched consider using aggregates in the index. Requires Lucene grammar for search terms.
Conditions with functions	if needed		/jcr:root/content/wknd// element(*, cq:Page) [fn:lower-case(@jcr:title) = 'france']	select * from [cq:Page] as a where lower(jcr:title)='france' and isdescendantnode(a, '/ content/wknd')	function=fn:lower-case(@ jcr:title)	See the Oak documentation for the full list of supported functions
Like conditions with prefix match	if needed	path=/content type=cq:Page l_property=jcr:title l_property.value=Fra% l_property.operation=like	/jcr:root/content// element(*, cq:Page) [jcr:like(@jcr:title, 'Fra%')]	select * from [cq:Page] as a where [jcr:title] like 'Fra%' and isdescendantnode(a, '/ content')	name=jcr:title propertyIndex=true	Just use prefix match. Instead of "%text%" use a fulltext search condition.



JCR Query Cheatsheet for AEM

(AEM 6.5 and AEM as a Cloud Service)

Please post any feedback to the [AEM Forums](#)



Adobe Experience Manager



Feature	Use	QueryBuilder	XPath	SQL-2	Index definition	Note
Or conditions on the same property	if needed	path=/content/dam type=dam:Asset 1_property=dc:format 1_p.or=true 1_property.1_value=image/jpeg 1_property.2_value=image/png	/jcr:root/content/dam// element(*, dam:Asset)[@dc:format = 'image/jpeg' or @dc:format = 'image/png']	select * from [dam:Asset] as a where ([dc:format]='image/jpeg' or [dc:format] = 'image/png') and isdescendantnode(a, '/ content/dam')	name=dc:format propertyIndex=true	"Or" conditions on same properties are converted into an "in" select. Unlike a union query it is still a single request.
Or conditions on distinct properties	if needed	path=/content/dam type=dam:Asset 1_property=dc:description 1_property.operation=exists 2_property=dc:title 2_property.value=Camp Summer Night p.or=true	/jcr:root/content/dam// element(*, dam:Asset)[@dc:description or @dc:title = 'Camp Summer Night']	select * from [dam:Asset] as a where isdescendantnode(a, '/content/dam') and [dc:description] is not null or [dc:title] = 'Camp Summer Night'		If the properties are part of different indexes then a union will be used, which has some overhead. In this case consider switching to a fulltext query.
Ordering	if needed	path=/content/dam type=dam:Asset 1_property=dam:MIMETYPE 1_property.value=image/jpeg orderby=@dam:size	/jcr:root/content/dam// element(*, dam:Asset)[@dam:MIMETYPE='image/jpeg'] order by @dam:size	select * from [dam:Asset] as a where [dam:MIMETYPE] = 'image/jpeg' and isdescendantnode(a, ' /content/dam') order by [dam:size]	ordered=true on all properties which are used for ordering	
Limit the result set, guess total size	always	p.limit=100 p.guessTotal=true	query.setLimit(100) via Java API			If you can make reasonable assumptions about the maximum number of results, use this information. Use keyset pagination if required.

Good indexes

- extend existing indexes if possible
- adhere to the naming conventions as described in the [documentation](#)
- are stored in /oak:index
- have the node type oak:QueryIndexDefinition
- use type = lucene
- use includedPaths and do not use excludedPaths
- set queryPaths to the same value as used in

includedPaths

- don't use entryCount (but prefer index tags) to select specific indexes
- use aggregation when using fulltext queries in multiple nodes
- index each condition in the query
- avoid nullCheckEnabled
- use async = ["async", "nrt"]

Additional resources

- [Oak query overview](#) (including SQL-2 and XPath grammar)
- [Oak documentation: Lucene Indexes](#)
- [Oak: Query troubleshooting](#)
- [Indexing in AEM as a Cloud Service](#)
- [Oak Index Generator](#)
- [Keyset pagination](#)