	JCR	JPA (initial)	JPA + CQRS + Optimization	Notes
1000 XSD writes (+ several thousand derived), no batching	265.959 sec	380.481 sec	342.463 sec	Expected (Hibernate Search indexing, moderately complex joined table structure, etc., versus simple JCR nodes)
1000 XSD writes (+ several thousand derived), 10 batches	50.283 sec	303.330 sec	266.244 sec	JCR persists each batch in a single transaction. JPA uses 1 transaction per item in the batch (which seems desirable).
query: get all XsdDocuments (not paged)	7.089 sec	3.940 sec	.408 sec	
query: get all ComplexTypeDeclarations, using a relationship path (n	7.772 sec	3.278 sec	.284 sec	
query: by UUID	.022 sec	.02 sec	.019 sec	
get metadata by UUID	.014 sec	0.061 sec	.037 sec	JCR may always be faster here, since we can simply lookup by path.
full text search (returns all XsdDocuments) (not paged)	6.057 sec	1.902 sec	.563 sec	To me, this one's the most exciting. Proper clustering, etc. will make this extremely scalable.
	JPA + CQRS + Optimization + Query Cache + Infinispan 2LC			
1000 XSD writes (+ several thousand derived), no batching	22.523 sec	Surprising, but expected. This optimized several types of repeated queries used to initialize relationship targets, etc.		
1000 XSD writes (+ several thousand derived), 10 batches	9.532 sec	Surprising, but expected. This optimized several types of repeated queries used to initialize relationship targets, etc.		
query: get all XsdDocuments (not paged)	.443 sec	Slight increase, due to cost of cache puts.		
query: get all ComplexTypeDeclarations, using a relationship path (n	.323 sec	Slight increase, due to cost of cache puts.		
query: by UUID	.041 sec	Slight increase, due to cost of cache puts.		
get metadata by UUID	.048 sec	Slight increase, due to cost of cache puts.		
full text search (returns all XsdDocuments) (not paged)	1.458 sec	Slight increase, due to cost of cache puts.		