Database - Support #4060

investigate converting extent fields to array columns

05/01/2019 01:01 PM - Eric Faulhaber

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Database - Support #4058: consider denormalizing tables as the def...	**New**
Related to Database - Feature #6418: represent extent fields as arrays	**WIP**

History

#1 - 05/01/2019 01:08 PM - Eric Faulhaber

We have found on occasion that our default approach of normalizing extent fields of like extent into a secondary table and mapping these to an associated list in Hibernate can cause a performance problem with some queries. OTOH, denormalizing across the board can cause problems with table design (very wide tables when large extent fields are exploded into separate columns) and potentially cause their own performance problems. Also, the associated list approach has caused no end of lazy initialization problems with Hibernate over the years.

Rather than choosing between the normalized and denormalized approaches we use today, we should investigate the implications of converting extent fields in the most natural/analogous way: to array columns.

I had discarded this approach early on, because Hibernate has limitations in how it deals with array columns. I don't recall ATM what those limitations were (possibly related to managed flushes and dirty checking?), but it may be time to revisit this decision. Perhaps the limitations aren't really issues for us after all, given that we use these fields in very specific and well-defined ways.

#2 - 05/01/2019 01:17 PM - Eric Faulhaber

Note that this task is created in the context of investigating changes that may improve performance, but I don't know that using array columns would be any more or less performant than our current approaches. This requires investigation.

Also note that there is quite a bit of complexity in how we convert where clauses and sorting and how we preprocess where clauses at runtime, which is hard-wired to the current normalized and denormalized approaches. Thus, changing this to support array columns may be a relatively complicated proposition. So, we need to understand whether it is worth the effort from a performance standpoint, before undertaking any changes.

DBAs and BI developers may prefer using array columns from a design perspective, though, in that they would represent the closest design analog to the current schema in Progress.

#3 - 11/01/2022 08:12 AM - Greg Shah

- Related to Support #4058: consider denormalizing tables as the default approach added

#4 - 11/01/2022 08:30 AM - Greg Shah

- Related to Feature #6418: represent extent fields as arrays added