Database - Support #4058

consider denormalizing tables as the default approach

05/01/2019 10:48 AM - Greg Shah

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 #4060: investigate converting extent fields to ...	**New**
Related to Database - Feature #6898: PostgreSQL schema improvements	**New**

History

#1 - 05/01/2019 10:49 AM - Greg Shah

We know that denormalizing for all tables is not always a performance improvement, so there may need to be a better approach to figuring out when to denormalize. Still, the default may be better as denormalized if it helps the majority of cases.

#2 - 05/01/2019 01:00 PM - Eric Faulhaber

I don't really want to make denormalization the default, as this can result in a huge number of columns in a table, hindering usability and potentially creating its own performance problem. However, I agree that we need to identify potentially problematic extent fields more proactively than we are today (i.e., not really at all until runtime testing reveals a bottleneck).

Perhaps a starting point in identifying tables with such fields is to create one or more custom reports which identify when extent fields are being used in ways which may produce a sub-optimal query plan, such as in where clause matches or for sorting purposes.

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

- Related to Support #4060: investigate converting extent fields to array columns added

#4 - 11/01/2022 08:28 AM - Greg Shah

- Related to Feature #6898: PostgreSQL schema improvements added

#5 - 11/01/2022 08:32 AM - Greg Shah

We've discussed offline this many times recently. I want to record some thoughts from those discussions:

- The current implementation of our persistence layer always fetches from the secondary extent table(s) which increases the number of trips to the database/number of SQL queries. This means it likely costs more in performance to use normalized columns than it does to use the denormalized.
- The normalized approach expands the storage used as compared to the denormalized form.
- The above mean that the only real benefit of denormalized columns is if somehow it is superior in terms of readability, understandability, reduction in code complexity (of SQL or of our Java code for managing these). In fact, all of these measures are significantly worse for normalized vs denormalized. The key point here is **no customer we have ever spoken with likes our normalized approach**.
- Some people may have an internalized fear of "exploding" the columns in a table. However, we should note that this is exactly what Progress is doing under the covers. The array subscripting is really just syntactic sugar to make accessing these columns a bit cleaner. We just need to get over this fear. Perhaps it would be more appropriate to talk about "expanding" instead of "exploding".

At this point, it is a holdover from Hibernate and normalized fields need to be terminated with extreme prejudice. See also #6418 and #4060.

BTW, "normalized" is not a proper term for this approach. The concept of normalization is to factor the data into a more rational form that better encodes the data relationships while reducing data redundancy. A classic example would be refactoring mailing/shipping/billing address columns from multiple tables into a common address table and removing the extra columns from those source tables). This is usually about the business meaning of the data and trying to find a more optimal form for the same meaning. The "normalized" extent columns use a weird concept that stuffs unrelated extent columns into the same secondary table purely by a match between data type and extent size. This means we are merging columns of unrelated business meaning into the same table. That is the opposite of normalization. Since we use this misnomer, it carries over to our "denormalized" form discussions as well.

#6 - 11/01/2022 08:35 AM - Greg Shah

- Related to deleted (Feature #6898: PostgreSQL schema improvements)

#7 - 11/01/2022 08:35 AM - Greg Shah

- Related to Feature #6898: PostgreSQL schema improvements added