Salesforce Large Data Volumes

JUNE 7, 2012 BY MARKCANE LEAVE A COMMENT

My own simplistic definition of an LDV scenario in the Salesforce context is in excess of 2 million records in a single object. This is a rule-of-thumb rather than prescriptive.

The following Salesforce functional areas are most affected by LDV, in terms of performance, timeouts etc. :

Reports
Search
Listviews
SOQL

Working with LDV :
In LDV cases, first and foremost you should consider options for Data Reduction such as archiving, UI mashups etc.. Where LDV is unavoidable the concept of Selectivity should be applied to the functional areas impacted most.

Data Reduction Considerations:
Archiving – consider off-platform archiving solutions
Data Warehouse – consider a data warehouse for analytics
Mashups – leave data in-situ in external systems and integrate at the UI level

Selectivity Considerations:
Selective – reduce the number of objects and fields used in a query.
Narrow Range Reporting – apply selective report filtering.
Filtering – apply restrictive filtering to Listviews and SOQL where clauses.
Custom Indexes – can be effective where query values in the indexed field represent less than 10% (or <300K) of the overall record count.
Skinny Tables – can be effective as a report accelerator at 5M records plus.
SOQL – avoid NULL values (can't be indexed)
Horizontal partitioning of data – split objects by geography for example.
Denormalisation – to remove expensive joins use Apex Triggers to resolve FK
Divisions – acts like a DB partition (by geography, ownership etc.)
Avoid over parenting – 10K limit for child records, per parent record. For example avoid one parent account having more than 10,000 contacts etc.
Data Skew – look for even distribution.

Loading Data :
Use the Bulk API to load large data volumes, via the Apex Data Loader perhaps (requires setting value to be explicitly set). The Bulk API, in simple terms, uploads the data into temporary tables then executes processing of the data (actual load into target objects) using parallel asynchronous processes. This offers potential performance improvements over the serial and synchronous combined upload+process model employed by all other loading techniques.

Data Loading Considerations;
Defer complex sharing rules
Disable Triggers and Workflow (post-process via Batch Apex)
Speed of operation; Insert then Update then Upsert (involves implicit query)
Group and sequence data to avoid parent record locking
Remember database statistics calculate overnight, wait to do performance testing
Tune the batch size (HTTP keepalives, GZIP compression)