On-line File System design

# Assumptions

* File contents are not indexed or parsed, simply stored
* Therefore, file contents may not be used in a search, only the file name
* Permissions are all read-only, there is no provision for edit or delete by anyone including the owner
* Ownership is immutable and set at creation time, you cannot transfer ownership
* Files may be grouped into folders, but there is no specification for a descending hierarchy, or that folders may contain other folders, so the hierarchy is very flat
* Searches do not consider permissions, so all files/folders may be considered in a query
* Uses conventional RDBMS as a NoSQL system would likely not be any more efficient or appropriate. Cassandra would be nice for speed, but keying and relational integrity make its use as a single datastore solution untenable, and a hybrid solution would be too complex for this scenario.
* Throughput does not specify types of interactions, between simple join queries or bulk inserts, but an eventual consistency model likely appropriate

# Overall design

Gien the specification and assumptions, this system would be built around a write-slow/read-fast model where changes are made to an authoritative store and then propagated to multiple read-only stores for subsequent queries. Various mechanisms exist to execute this propagation including transaction-log trailing systems like Oracle GoldenGate, or Kafka streams.

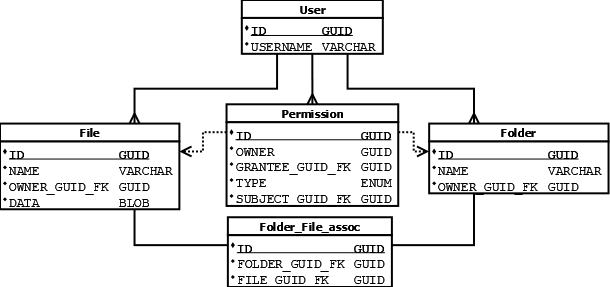
Initial authoritative storage would likely be done via RDBMS for relational integrity, management of metadata and ability to leverage current tooling. This system would act as the OLTP data layer, but would operate as a primary, or insert-only system; no queries would be done at this layer. From an RDBMS perspective, this is relatively efficient as it avoids row lock contention, complex isolation, and competition with uncontrolled queries. In a very high volume scenario, inserts could be queued up in an event stream for later processing, but this would complicate error handling. Multiple independent insert stores could be used, and using GUIDS instead of sequences makes this easier, but there should still be a unified source of truth at some level.

Once data is stored, it may be replicated/propagated to querriable ODS layers. An event stream may be more appropriate than direct duplication as it is more flexible and dynamic, and allows for non RDBMS consumers. While an RDBMS system may be an appropriate initial consumer, usage patterns may later indicate more optimal and less conventional destinations for this data.

For example, a globally distributed system may need some redundant replication, but it is likely that specific records will only be accessed in a single locale, which may provide opportunities to reduce unnecessary replication. Another example may be that a hosting country or region could mandate that those created within that locale’s borders must remain there and not be replicated outside.

Additionally, as patterns are revealed, opportunities to be more predictive about where data should and should not be replicated will become apparent, and the flexibility to route records from the source, or filter at the consumer level, will likely prove to be quite valuable. Additinally, non-RDMBS query solutions may be more effective in dynamic or high-burst traffic scenarios. Large in-memory caches, searchable stores like Elasticsearch or Redis, as well as asynchronous query results for users fetching bulk data are all options to remain flexible for at this stage.

# Data Model



## Insert Data Model

The insert data model is simple and uncomplicated.

* All records use a GUID, though a sequence ID would likely also be appropriate, a GUID can make some kinds of tracing/debugging more straightforward.
* No columns are nullable
* File data is stored as a BLOB
* As an insert-only system, only primary and foreign keys are indexed.
* Note that the Permission table uses an ENUM to indicate if it relates to a Folder or a File
* Permission.OWNER may be obsolete

## Read-Only Data Model

This model is initially the same as the insert one, but with additional indexes for searchable columns like names.

# Querying

## Example queries

### Search for files or folders

SELECT GUID, NAME from FILE where NAME LIKE ‘%...%’;

SELECT GUID, NAME from FOLDER where NAME LIKE ‘%...%’;

SELECT GUID, NAME from FILE where OWNER\_GUID\_FK = ‘guid’;

SELECT GUID, NAME from FILE f, USER u where u.USERNAME = ‘username’ and f.OWNER\_GUID\_FK = u.ID;

Grant permission to another user to view file

INSERT INTO Permission (new GUID, my GUID, grantee GUID, ‘FILE’, file GUID);

View files I have permission to view

SELECT GUID, NAME from file f, permission p where (

(f.OWNER\_GUID\_FK = my GUID)

OR

(p.GRANTEE = my GUID and p.TYPE = ‘FILE’ and f.ID = p.SUBJECT\_GUID\_FK)

)