Data model design
Purpose of this technote

Adobe Campaign relies on external databases. It has no limitation on the size of tables. To optimize performance large tables need to have a specific design.

The technote explains how to optimize the database design for larger volumes.

Size of tables

- A **small** size table is similar to the delivery table.
- A **medium** size table is the same as the size of the recipient table. It has one record per customer.
- A **large** size table is similar to the broad log table. It has many records per customer.

For example, if your database contains 10 million recipients, the broad log table will contain about 100 to 200 million messages, and the delivery table will contain a few thousand records.

Example

In this example, the Transactions and Transaction Item tables are big: more than 10 million.

The Product and Store tables are smaller: less than 10,000.

The product label and reference have been placed in the Product table.

The Transaction Item table only has a link to the Product table, which is numerical.

Choice of Data Types
A large table should have mostly numeric fields and contain links to reference tables.

You can use "expr" attributes to avoid duplicating fields. For instance the Recipient table uses an expression for the domain, which is already present in the email field.

The "XML" type is a good trick to avoid creating too many fields. But it also takes up disk space as it uses a CLOB column in the database.

**Choice of fields**

The drivers to ensure that a field is required to be stored in a table are: targeting purpose or personalization purpose. In other words, if a field is neither used to send a personalized email nor used as a criterion in a query, you have to consider that this field is useless and takes up disk space for nothing.

Keep in mind that FDA (Federated Data Access, an optional feature that allows to access external data) covers the need to add a field "on-the-fly" during a campaign process. You don't need to import everything if you have FDA.

**Choice of keys**

Efficient keys are essential for performance. Numeric data types should always be preferred as keys for tables.

For SQLServer database, you could consider using "clustered index" if performance is needed. Since Adobe does not handle this, you will need to create it in SQL.

**Indexes**

Indexes are essential for performance. When you declare a key in the schema, Adobe will automatically create an index on the fields of the key.

You can also declare additional indexes for queries that don't use the key.

Indexes should be limited in size and number because this impacts the performance during the insertion of data.

**Links and Cardinality**

When you design a link, make sure the target record is unique when a 1-1 relationship has been declared. Otherwise the join may return multiple records when only one is expected. This will result in errors during delivery preparation when "the query returns more rows than expected". Set the link name to the same name as the target schema.

Beware of the "own" cardinality on big tables. Deleting records that have wide tables in "own" cardinality can stop the instance. The table is locked and the deletions are made one by one. So it's best to use "neutral" cardinality on child tables that have big volumes.

Declaring a link as an external join is not good for performance. The zero ID record emulates the external join functionality. It is not necessary to declare external joins if the link uses the autopk.

**Partitioning**

Partitioning at the database level could be a solution to optimize queries on certain tables with large volumes. Strategies differ with the objectives of the table.

- For response or reporting aggregates, partition broad logs and transactions by week or month to optimize access to the latest data.
- For distributed marketing, partition the recipient table by region or agency to optimize access to local data.

**Dedicated tablespaces**
The tablespace attribute in the schema allows you to specify a dedicated tablespace for a table.

The installation wizard allows you to store objects by type (data, temporary and index).

Dedicated tablespaces are better for partitioning, security rules, and allow fluid and flexible administration, better optimization, and performance.

One-to-many relationships

Data design will impact usability and functionality. If you design your data model with lots of one-to-many relationships, it makes it more difficult for users to construct meaningful logic in the application. One-to-many filter logic can be difficult for non-technical marketers to correctly construct and understand.

It's good to have all the essential fields in one table because it makes it easier for users to build queries. Sometimes it's also good for performance to duplicate some fields across tables if it can avoid a join.

Certain built-in functionalities will not be able to reference one-to-many relationships, e.g. Offer Weighting formula and Deliveries.

Cleanup task

You can declare the “deleteStatus” attribute in a schema. It's more efficient to mark the record as deleted, then postpone the deletion in the cleanup task.