Anchor for Vaulters

Lars Rönnbäck

Originally for the DVEE Summit in Stockholm, October 2019
http://www.anchormodeling.com/modeler/latest
```sql
-- Anchor table

-- ST_Store table (with 2 attributes)

IF Object_ID('dbo.ST_Store', 'U') IS NULL
CREATE TABLE [dbo].[ST_Store] (  
ST_ID int IDENTITY(1,1) not null,  
Metadata_ST int not null,  
constraint pAST_Store primary key (  
ST_ID asc  
)  
)
GO

-- Historized attribute table

-- ST_NAM_Store_Name table (on ST_Store)

IF Object_ID('dbo.ST_NAM_Store_Name', 'U') IS NULL
CREATE TABLE [dbo] [ST_NAM_Store_Name] (  
ST_NAM_ST_ID int not null,  
ST_NAM_Store_Name varchar(42) not null,  
ST_NAM_ChangedAt datetime not null,  
Metadata_ST_NAM int not null,  
constraint fkST_NAM_Store_Name foreign key (  
ST_NAM_ST_ID  
) references [dbo].[ST_Store][ST_ID],  
constraint pAST_NAM_Store_Name primary key (  
ST_NAM_ST_ID asc,  
ST_NAM_ChangedAt desc  
)  
)
GO

-- Static attribute table

-- ST_LOC_Store_Location table (on ST_Store)

IF Object_ID('dbo.ST_LOC_Store_Location', 'U') IS NULL
CREATE TABLE [dbo] [ST_LOC_Store_Location] (  
ST_LOC_ST_ID int not null,  
ST_LOC_Store_Location varchar(42) not null,  
Metadata_ST_LOC int not null,  
constraint fkST_LOC_Store_Location foreign key (  
ST_LOC_ST_ID  
) references [dbo].[ST_Store][ST_ID]
)  
GO
```
## Data Vault

### Business Key
- One key is the master key  
  *(less flexibility)*
- The master key is stable  
  *(makes assumptions about the future)*
- The master key is available in the hub  
  *(faster loading performance)*
- No real explanation of how to manage other ways to identify the same thing  
  *(confusing)*
- Hashing is (maybe) encouraged  
  *(the surrogate key encodes the domain)*

## Anchor

### Natural keys
- Every candidate key is treated equally  
  *(more flexibility)*
- Candidate keys may change over time  
  *(assumes nothing about the future)*
- Keys are spread out in the model  
  *(slower loading performance)*
- Every possible way to identify something is equally valid  
  *(clear, but may complicate ETL)*
- Hashing is impossible  
  *(the surrogate key cannot carry meaning)*
Data Vault

Changes at Loading

- Captured from the perspective of the database
  (no requirements on the sources)
- Temporally inconsistent data can be stored
  (relies on ETL if this is undesirable)
- Inconsistencies must be resolved at read time
  (faster write, slower query performance)
- The time when something changed in the domain is not part of the primary key
  (more complicated point in time logic)

Anchor

Changes from the Domain

- Captured from the perspective of the domain being modeled
  (sources should be change-aware, fall-back to database perspective)
- Temporally inconsistent data cannot be stored
  (database will enforce consistency)
- Inconsistencies must be resolved at write time
  (slower write, faster query performance)
- The time when something changed in the domain is part of the primary key
  (less complicated point in time logic)
Data Vault

**Grouping**

- Attributes are grouped by rate of change or by what they represent (no clear cut rule)
- To see what changed, the values from the previous row are needed (slower change detection)
- A typical query needs a few joins (can be faster, can be slower, depending on the selectivity in the query)
- Null values may exist (not optimal for sparse data)

Anchor

**6NF**

- Every attribute becomes its own table (no room for creativity)
- Changes are tracked independently of each other (faster change detection)
- A typical query needs a lot of joins (can be faster, can be slower, depending on the selectivity in the query)
- Nulls become the absence of rows (optimal for sparse data)
Data Vault

Implementation variety

- No naming convention
  (harder to swap between DWs)
- Some degrees of freedom when it comes to implementation details
  (different DWs may use different flavors)
- Largest known Data Vault implementation is for the U.S. Government at 15 petabytes.
  (not much more is public)

Anchor

Implementation consistency

- Naming convention
  (easier to swap between DWs)
- Almost no degrees of freedom when it comes to implementation details
  (different DWs have the same flavor)
- Largest known Anchor implementation is for avito.ru at 279 terabytes on HPE Vertica.
  (scientific papers available with a case study)
Data Vault

Auditability

- Links are always many-to-many
  (may rely on ETL to enforce cardinality)
- Column values can range over the span of the data type
  (may rely on ETL to enforce constraints)
- Values may disappear (become NULL)
  (less up-front complexity)
- Inconsistencies can be analyzed and managed before the next layer
  (source corrections can be done asynchronously)
- Raw data always available
  (reloading doesn’t need to touch the source)

Anchor

Correctness

- Ties always have their cardinality specified
  (the database will enforce cardinality)
- Column values may be constrained to certain intervals
  (the database will enforce constraints)
- Values are assumed exhaustive (become explicitly ”unknown”)
  (more up-front complexity)
- Inconsistencies must be managed, but negates the need for layered architectures
  (source corrections may need to be expedient)
- Raw data rarely available in full
  (reloading needs to touch the source again)
Data Vault

**Flavorful**

- Infinitely extendable  
  (great ideas can be utilized immediately)
- Extensions, such as bitemporality, may be added in any fashion  
  (less coherency)
- Models can be created in almost all modeling tools, with some code generated  
  (no lock-down, but most tools are commercial)
- Not overly dependent on query optimization features  
  (works well in most databases)
- Dangling references  
  (any load order, disables some query optimization)

**Anchor**

**Stringent**

- Controlled extensions  
  (if it doesn't adhere, it's something else)
- Extensions are added to the methodology after careful research  
  (coherent, but features may take time to appear)
- Models are normally created using the Online Anchor Modeler, lots of code generated  
  (dependent on a single tool, but it is free)
- Very particular about query optimization features being present  
  (works well only in very recent databases)
- PK – FK on every table  
  (particular load order, enables all query optimization)
Data Vault and Anchor

- Built for managing change
- Decouple mutable and immutable information
- Differentiate between entities and relationships
- Have features that reduce complexity
- Care for the core concepts in the domain
- Insert Only
- Try to live up to modern requirements
- Consider schema evolution
- Have modeling tools that generate code
- Have tools for metadata driven automation
- Both can *twine*
- Still not as widely adopted as Dimensional…
- Variations of **ENSEMBLE MODELING**