Datomic Cloud Documentation

Search

What is Datomic?

- Data Model
- Architecture
- Supported Operations
- Programming with Data and EDN

Local Dev and CI

Cloud Setup

- Start a System
- Configure Access
- Get Connected

Tutorial

- Client API
- Assertion
- Read
- Accumulate
- Read Revisited
- Retract
- History

Client API

Videos

- AWS Setup
- Edn
- Datalog
- Datoms
- HTTP Direct
- CLI tools

Schema

- Defining Attributes
- Schema Reference
- Changing Schema
- Data Modeling
- Schema Limits

Transactions

- Processing Transactions
- Transaction Data Reference
- Transaction Functions
- ACID
- Client Synchronization

Query and Pull

- Executing Queries
- Query Data Reference
- Pull
- Index-pull
- Raw Index Access

Time in Datomic

- Log API
- Time Filters

Ions

- Ions Tutorial
- Ions Reference
- Monitoring Ions

Analytics Support

- Configuration
- Connecting
- Metaschema
  Grammar SyntaxGrammarNaming ConventionsMetaschemaTablesTable OptionsColumn Joins
- SQL CLI
- Troubleshooting

Analytics Tools

- Metabase
- R
- Python
- Jupyter
- Superset
- JDBC

- Other Tools

Operation

- Planning Your System
- Start a System
- AWS Account Setup
- Access Control
- CLI Tools
- Client Applications
- High Availability (HA)
- Howto
- Query Groups
- Monitoring
- Upgrading
- Scaling
- Deleting
- Splitting Stacks

Tech Notes

- Turning Off Unused Resources
- Reserved Instances
- Lambda Provisioned Concurrency

Best Practices

Troubleshooting

FAQ

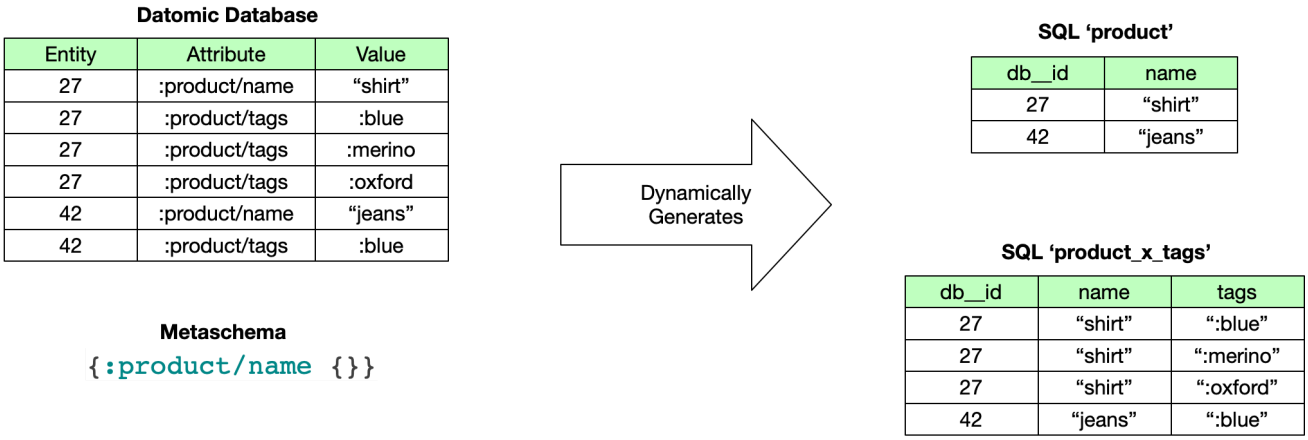
Examples

Releases

Glossary

Metaschema Reference

Metaschemas drive the mapping between Datomic entities and attributes and SQL tables & columns. For an overview, see the analytics overview page.



Even with no metaschema entries, there will always be auto-generated `db__idents` and `db__attrs` tables which contain information about `:db/idents` and schema. Use `describe` to see their columns.

Grammar Syntax

```
'' literal
"" string
[] = list or vector
{} = map {k1 v1 ...}
() grouping
? zero or one
+ one or more
* zero or more
| choice
```

Grammar

```
    (whatever-)attr       = keyword naming an attribute in Datomic schema
    sql-name              = string or symbol, alphanumeric or '_' only

    metaschema            = {:tables tables-map :joins joins-map}

    tables-map            = {(membership-attr opts-map)+}
    opts-map              = {name-opt? include-opt? exclude-opt? rename-opt?
                             rename-many-opt? scale-opt?}
    name-opt              = :name sql-name
    include-opt           = :include [attr* ns-wildcard*]
    ns-wildcard           = keyword ending in '/*'
    exclude-opt           = :exclude [attr*]
    rename-opt            = :rename {(attr sql-name)+}}
    rename-many-opt       = :rename-many-table {(card-many-attr sql-name)+}
    scale-opt             = :scale {(bigdec-attr integer)+}

    joins-map             = {(ref-attr table-name)+}
    table-name            = sql-name of table generated by tables-map
```

Naming Conventions

Fully namespaced-qualified attr names are required. Attributes without namespaces will be ignored.

Any non-alphanumeric characters in schema, tables, and catalog names will be converted to underscores (_). Thus, a system (and catalog properties file) named `my-datomic-system` will be converted to `my_datomic_system` for use in analytics tools.

Metaschema

```
metaschema            = {:tables tables-map :joins joins-map}
```

A metaschema is a map with two entries, *:tables* and *:joins*. The *:tables* entry specifies which Datomic attributes are used to create SQL tables. The *:joins* entry enables you to extend these tables by joining in additional columns through ref attributes.

Tables

```
tables-map            = {(membership-attr opts-map)+}
```

The `:tables` map maps Datomic attributes to SQL tables by defining which Datomic attributes are considered "membership" attrs.

Each entry in the `:tables` map will result in the creation of a SQL table, containing all entities that include that membership attr. The table name is derived from the namespace of the membership attr.

Table Options

```
opts-map              = {name-opt? include-opt? exclude-opt? rename-opt?
                         rename-many-opt? scale-opt?}
```

Each entry in the tables map requires an options map. The simplest options map is the empty map, `{}`. Each options map *may* contain any combination of:

- a name option
- an include option
- an exclude option
- a rename option
- a rename-many option
- a scale option

Name

```
name-opt              = :name sql-name
```

The default table name for a SQL table is derived from the namespace of the membership attr. This name can be overridden for any table(s) in the metaschema with the *:name* option. The *sql-name* must include only alphanumeric characters and the underscore (_) character.

Include and Exclude

```
include-opt           = :include [attr* ns-wildcard*]
ns-wildcard           = keyword ending in '/*'
exclude-opt           = :exclude [attr*]
```

By default, the columns of each SQL table will correspond to all Datomic attributes with the same namespace as the membership attr. Additional attributes can be included in any table(s) via the `:include` option. Namespace wildcards may also be used in the *:include* attribute list to include all attributes within that namespace in the table.

Similarly, attributes from the membership attr namespace can be omitted from the SQL table using the `:exclude` option.

Renaming Attributes

```
rename-opt            = :rename {(attr sql-name)+}}
rename-many-opt       = :rename-many-table {(card-many-attr sql-name)+}
```

The rename options maps map attribute names to SQL table names.

By default, columns in the SQL tables are named based on Datomic attributes, but they can be renamed using the `:rename` option. Each entry in the `:rename` option map must specify a Datomic attribute and the SQL name to which it will be renamed (alphanumeric and _ only).

Cardinality-many Datomic attributes are populated in a separate card-many table, named `membership-attr-namespace_x_card-many-attr-name` by default.

This card-many-table can be renamed with the `:rename-many-table` option. Each entry in the `:rename-many-table` option map must specify a cardinality-manyt Datomic attribute and the sql name to which the card-many table will be renamed (alphanumeric and _ only).

Scale

```
scale-opt             = :scale {(bigdec-attr integer)+}
```

The *:scale* value specifies the number of digits in the fractional part of a bigdec column. If not supplied it defaults to 2. If your data is not at scale 2 you must supply an explicit scale.

Column Joins

```
joins-map             = {(ref-attr table-name)+}
table-name            = sql-name of table generated by tables-map
```

The `:joins` map maps ref attrs to tables defined by the metaschema (think 'foreign keys'). See column joins in the overview for more details.