Expanded Data Model

CAPSTONE PROJECT | ASSIGNMENT 2

In this assignment, you will enhance and complete your application data model using several more advanced field types, including timed relations and stored calculations.

As always, remember to save all C3 Type files in the `/src` folder.

1. USING TIMED INTERVAL RELATIONS

A. Creating a Timed Interval Relation for Smart Bulbs and Fixtures

We want to know which smart bulbs are in which fixtures over time. This is useful for two reasons:

1. It allows us to determine a smart bulb's location at any point in time.
2. It may provide predictive power for our failure risk score model, should we choose to use it as a feature.

We can accomplish this using a timed interval relation type. Timed interval relations represent the historical relationship between two entities.

To create our own timed interval relation between smart bulbs and fixtures, we must:

1. Create a type that mixes in `TimedIntervalRelation`.
2. Specify the parameters for TimedIntervalRelation<From, To>.
    a. In our case we are describing a `SmartBulb` to `Fixture` relationship.
3. Define the schema name as `SMRT_BLB_FXTR_RLTN`.

```
/**
 * SmartBulbToFixtureRelation.c3typ
 * Describes a timed relationship between a smart bulb and a fixture.
 * For more information, see {@link TimedIntervalRelation}.
 */
@db(index=['to', 'from'])
entity type SmartBulbToFixtureRelation mixes TimedIntervalRelation<SmartBulb,
Fixture> schema name "SMRT_BLB_FXTR_RLTN"
```

B. Modifying Fixture to Use the Timed Interval Relation

We must now connect the `Fixture` type with our new timed interval relation.

To do this, we must do the following:

1. Create a relationship history array sorted by recency.
2. Create a stored calculation field for the `id` of the `SmartBulb` currently in each `Fixture`.

Add the following code snippets to your `Fixture` type:

```
// NEW DB ANNOTATION: Include this on the line above the type declaration.
@db(order='descending(toUTC(start)), descending(toUTC(end))')

// NEW FIELD: The collection of relations that map to this fixture.
bulbHistory: [SmartBulbToFixtureRelation](to)

// NEW FIELD: The current {@link SmartBulb} attached to this fixture.
currentBulb: SmartBulb stored calc 'bulbHistory[0].(end == null).from'
```

As you can see, the `currentBulb` field takes the last `SmartBulb` in the `bulbHistory` array and selects its `id` via the `from` field. Additionally, we select `null` in case the `Fixture` is empty.

C. Modifying Smart Bulb to Use the Timed Interval Relation

To finish configuring our timed interval relation, we need to make several modifications to the `SmartBulb` type as well. These configurations are similar in nature to those made for `Fixture`.

Add the following code snippets to your `SmartBulb` type:

```
// NEW DB ANNOTATION: Include this on the line above the type declaration.
@db(order='descending(start), descending(end)')

// NEW FIELD: The collection of relations that pertain to this smart bulb.
fixtureHistory: [SmartBulbToFixtureRelation](from)

// NEW FIELD: The current {@link Fixture} to which this smart bulb is attached.
currentFixture: Fixture stored calc 'fixtureHistory[0].(end == null).to'
```

2. USING TIMED VALUE RELATIONS

Timed value relations are used to store status data – such as whether a `SmartBulb` is on or off – over time. They allow us to fetch status values for specific objects at specific points in time.

A. Creating a Timed Value Relation for Power Grid Status Data

We will now create a type to represent the status of the power grids connected to buildings.

To do this, we must do the following:

1. Declare the `PowerGridStatusSet` type to store our status information.

2. Specify the proper data store (we'll be uploading a lot of data into this type).
    a. In this case we will use Cassandra.
    b. We will configure this value using a database annotation.

```
/**
 * PowerGridStatusSet.c3typ
 * The status of the power grid for a {@link Building} at a specific time.
 */
@db(datastore='cassandra',
    partitionKeyField='parent',
    persistenceOrder='timestamp',
    persistDuplicates=false,
    compactType=true,
    shortId=true,
    shortIdReservationRange=100000)
entity type PowerGridStatusSet mixes TimedValueHistory<Building>, Integer schema
name 'PWR_GRD_STTS_ST'
```

B. Creating the Power Grid Status Type

We will now create a type that allows us to easily fetch the latest power grid status value.

```
/**
 * PowerGridStatus.c3typ
 * The status of the power grid for a {@link Building} at a specific time.
 */
type PowerGridStatus mixes TimedValue, Integer
```

C. Modifying Building to Use the Timed Value Relation

We must now connect `Building` to these timed value relation types.

Add the following code snippets:

```
// NEW DB ANNOTATION: Include this on the line above the type declaration.
@db(order='descending(timestamp)')

// NEW FIELD: The collection of statuses relevant to this {@link Building}.
gridStatusSet: [PowerGridStatusSet](parent)

// NEW FIELD: The current status of the power grid for this {@link Building}.
@db(timedValueHistoryField='gridStatusSet')
gridStatus: PowerGridStatus
```