IoTQL - CREATE: Time and Triggers

This discusses CREATE and DO, which allow time-based / trigger-based actions to be defined - and persisted over multiple sessions.

Scenes

Basically, these are just macros

Simple form

```
CREATE SCENE
    lights_on_1
SET
    state.on = true
WHERE
    meta.facet & facets.lighting
;
DO lights_on
```

More complex

BEGIN and END bookend multiple statements

```
CREATE SCENE
    lights_on_2
BEGIN
    SET state.on = true
    WHERE id = "thing-01";
    SET state.on = true, color = #FF0000
    WHERE id = "thing-02";
END
```

Arguments

```
CREATE SCENE
```

```
    lights(value)
SET
    state.on = :value
WHERE
    meta.facet & lighting

;
DO lights(true);
DO lights(false)
```

Triggers

Triggers are code that are run when the expression in the WHEN clause *becomes* true.

State based

When the front door is opened, the code in the BEGIN / END clause is run.

```
CREATE TRIGGER
    front_door_light
WHEN
    state.open = true
WHERE
    meta.name = "Front Door Contact Switch"
BEGIN
    SET
        state.on = true
    WHERE
        meta.name = "Front Door Light"
    ;
    SLEEP(minutes=10)
    ;
    SET
        state.on = false
    WHERE
        meta.name = "Front Door Light"
    ;
END
```

Note:

- the main issue with this one is if the lights are already on, or somewhere turns them on, they'll turn off because of this rule. There almost has to be some sort of "incrementing" system.
- what happens if it is triggered again? we need modes, to allow multiple to run at a time, cancel the one running, cancel the one being triggered.
- these rules have to apply per-thing, right?
- note the pythonic named arguments (minutes=10) to a function

Ideas:

```
LOCK()
LOCK(id, cancel=true)
```

Time based

```
CREATE TRIGGER
    sunset_lights
WHEN
    SUNSET(minutes=30)
BEGIN
    SET
        state.on = true
    WHERE
        meta.facet & facets.lighting
    AND
        meta.zone & "Outdoors"
END

CREATE TRIGGER
    sunrise_lights
WHEN
    SUNSET(minutes=-30)
BEGIN
    SET
        state.on = false
    WHERE
        meta.facet & facet.lighting
    AND
        meta.zone & "Outdoors"
END
```

Sequences

This may be a hack - allow multiple WHENs in a row?

```
CREATE TRIGGER
    something
WHEN
    state.on = true
WHEN
    state.on = falde
BEGIN
    -- the lights were turned on and off
END
```

Views & Timeseries

Views are just like SQL VIEWS. Note that you have to query using `FROM view`, which normally you don't have to do.

average temperature

This gets the average temperature of the house

```
CREATE VIEW
    house_temperature
SELECT
    AVG(state.sensor.temperature) AS temperature
WHERE
    meta.facet & facets.sensor.temperature
;

SELECT
    temperature
FROM
    house_temperature
;
```

Notes:

- what if the units are incompatible? obviously you could do a UNITS conversion, but what if there's not?

Timeseries data

This is where it gets fun.

```
CREATE VIEW
    temperature
SELECT
    timeseries.average(state.sensor.temperature, minutes=5) AS avg,
    timeseries.minimum(state.sensor.temperature, minutes=5) AS min,
    timeseries.maximum(state.sensor.temperature, minutes=5) AS max,
WHERE
    meta.facet & facet.sensor.temperature
;

SELECT
    avg
FROM
    temperature
;
```

Notes:

- note the functions with "." in them
- it would be nice to tie this into data stores for super-long queries, e.g. highest temperature this month.