Vent

What is?

- Smart contract to SQL mapping layer
- Support for Burrow and now Ethereum

The idea

- Emit Solidity events (EVM opcodes `LOG0, ..., LOG4`)
- Define some SQL table projections
- Vent creates read-only tables representing the projections
- Vent maps the fields in a stream of events to rows in the table

Emit solidity events

```
event UpdateDescription(
   bytes32 indexed name,
   bytes32 indexed key,
   bytes32 indexed description);

event DeleteDescription(
    bytes32 indexed name,
    bytes32 indexed key,
    int __DELETE__);
```

Define projections

```
[
  {
    "TableName": "Descriptions",
    "Filter": "EventType = 'LogEvent' OR EventType = 'MultiSpoon' OR EventType = 'Splodge-Gniver' OR EventType = 'Splodge-Gniver' OR EventType = 'COLY-BLOPS'",
    "DeleteMarkerField": "__DELETE__",
    "FieldMappings": [
      {
        "Field": "key",
        "ColumnName": "testkey",
        "Type": "bytes32",
        "Primary": true,
        "Notify": ["keyed_meta"]
      },
      {
        "Field": "name",
        "ColumnName": "testname",
        "Type": "bytes32",
        "Primary": true,
        "BytesToString": true,
        "Notify": ["meta", "keyed_meta"]
      },
      {
        "Field": "description",
        "ColumnName": "testdescription",
        "Type": "bytes32",
        "Primary": false,
      }
    ]
  }
]
```

Get tables

```
| key | name           | description | chain metadata |
| 1   | Irrigatornator | hose        | ...            |
| 2   | Propignado     | flowerpot   | ...            |
| 3   | BothyMax       | shed        | ...            |
```

Modes

- View mode (upserts)
- Log mode (append only)
- Based on primary key allocation

Domain model

- Kind of solidity-to-relational mapping

Features

- Chain blockstamping
- Update projections and rebuild tables completely
- Rewind/replay state
- Emit events to delete
- Postgres and SQLite support
- Postgres notifications on columns

Ethereum support

- New in 0.31.0
- Uses web3 JSON-RPC `eth_getLogs`
- Based on compatibility of event serialisations

Synchronisation

- Via high watermark helper `vent.sync()`
- Waits for the height in return of last Tx to hit database

```
return this.contracts.do(async (c) => {
  await Promise.all(users.map((user) => c.addUserToOrganization(user, organizationAddress, user)));
  const permissions = new PermissionService(this.txr);
  await permissions.assignRoleToUsers(organizationAddress, RoleEnum.RESTRICTED_USER, users);
});
```

Future work

- Other chains?
- More complex SQL relations and types
- Views and generated columns