Database, Client, Engine

What is a Relational Database?

- Table
  - Rows, columns
    - Columns have names, data types
    - Each row is a single record
    - Operate on rows based on data in the columns

```
SELECT workbook_name FROM workbook WHERE
workbook_id = 1005;
```

table name

column name

which rows

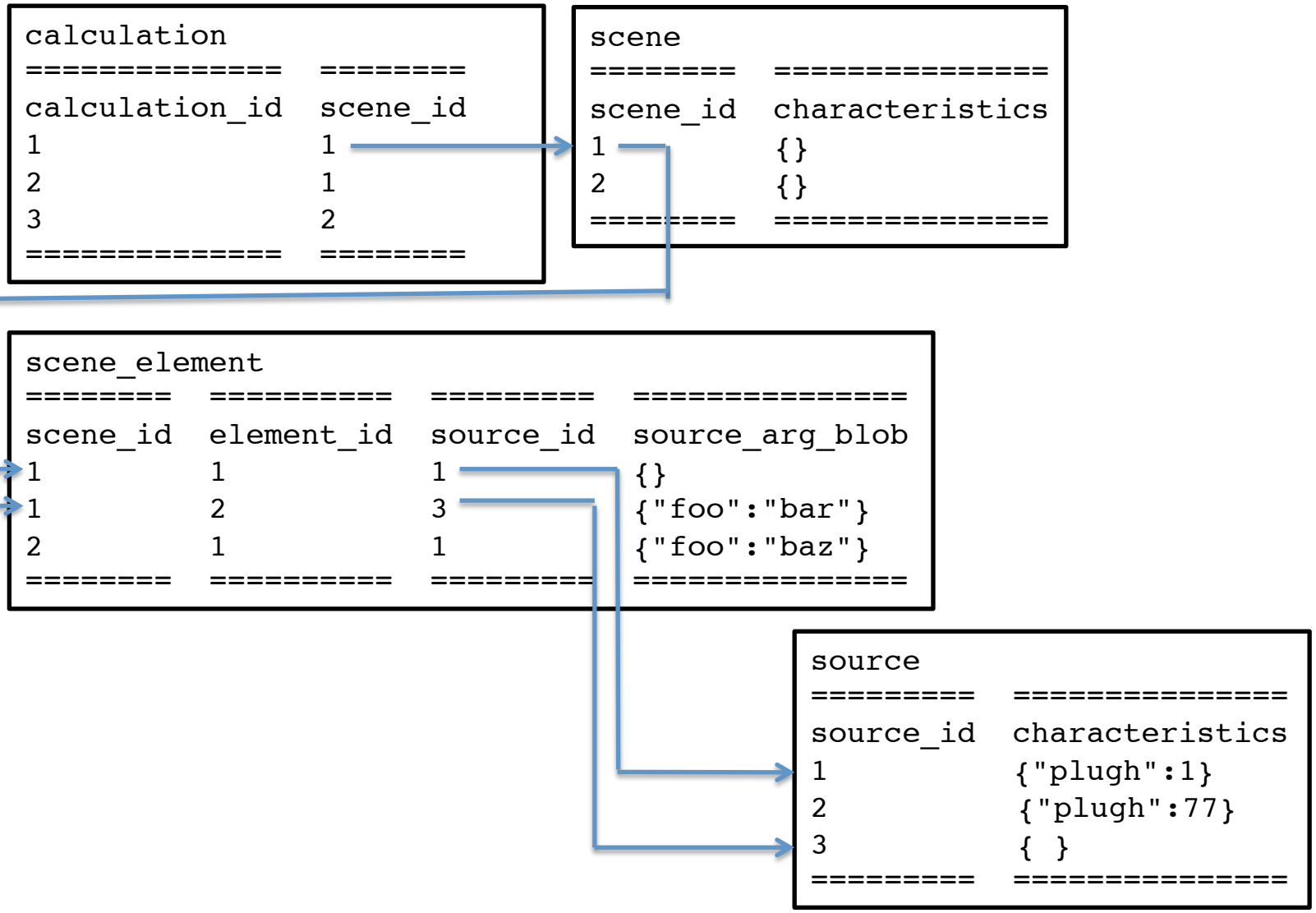
- Index
  - A way to find rows faster, based on content of one or more columns

Relations

- When a row in one table is related to a row in another
  - Usually by equality, but not necessarily
  - Can find data from multiple tables based on relation
  - 1 to N
  - 1 to 1
  - N to 1

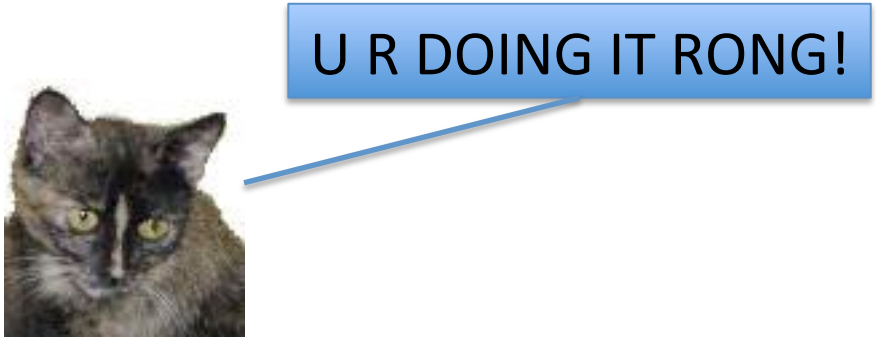
  - workbook_id, calculation_id are important columns to us – we look up almost everything by one or both of these

Relations


calculation
calculation_id scene_id
1 1
2 1
3 2
scene
scene_id characteristics
1 {}
2 {}
scene_element
scene_id element_id source_id source_arg_blob
1 1 1 {}
1 2 3 {"foo":"bar"}
2 1 1 {"foo":"baz"}
source
source_id characteristics
1 {"plugh":1}
2 {"plugh":77}
3 { }

Obligatory Database Rant

- You get data OUT of a database; you don't put data IN to a database
  - Choose your tables and columns according to how you need to access them
  - If you start defining columns without knowing how you will use them, then



Choice of Database

- Pandokia portable database interface
  - I don’t have time to take a shortcut; I’m in a hurry
  - No unnecessary complexity
  - wraps Python DBAPI 2.0 (not quite portable)
  - Sqlite (for developers)
  - MySQL (for production)
  - MS SQL Server (can talk to it later)

Sample Table

comment

pandokia extension for portability

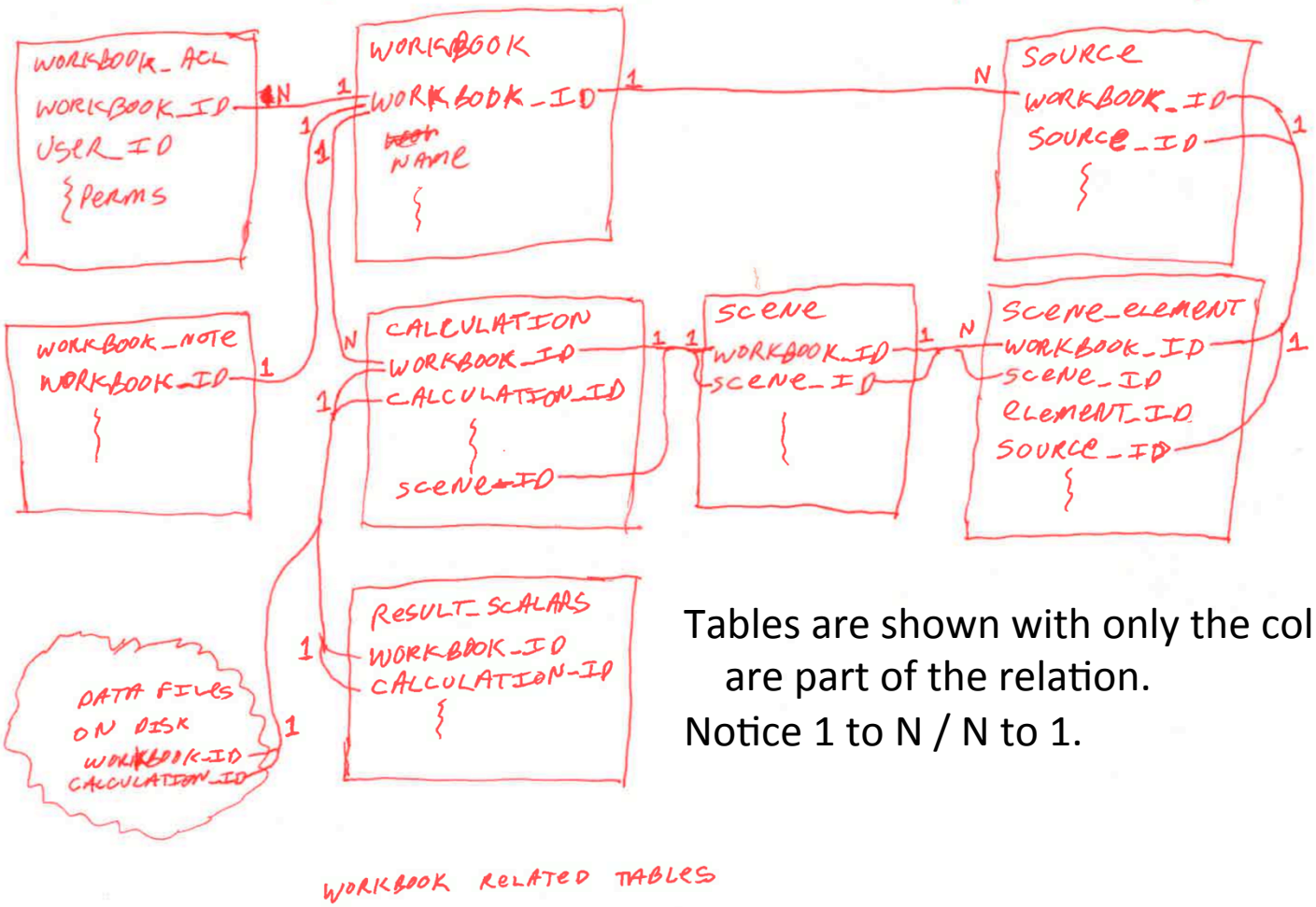
column type

column name

```
-- This is the scalar data of a workbook.
create table workbook (
    ++ sqlite
        workbook_id     INTEGER PRIMARY KEY,
    ++ mysqldb
        workbook_id     INTEGER AUTO_INCREMENT,
            PRIMARY KEY ( workbook_id ),
    ++
        workbook_name   VARCHAR(100) DEFAULT '',
        description     VARCHAR(140) DEFAULT '',
        proposal_id     VARCHAR(20) DEFAULT '',
                        -- set only by STScI
        proposal_state  CHAR(1) NOT NULL DEFAULT ' ',
                        -- set only by STScI
        test_mode       CHAR(1) NOT NULL DEFAULT '0',
                        -- if this workbook was touched in test mode
        deleted         INTEGER NOT NULL DEFAULT 0,
                        -- time of deletion; eligible to expire
                        -- after N hours/days/whatever
        created         INTEGER DEFAULT 0
                        -- time of creation
        );
```

color by vim

Relation Diagram of Pandeia



Tables are shown with only the columns that are part of the relation.

Notice 1 to N / N to 1.

WORKBOOK RELATED TABLES

Finding data

- Find a workbook by
  - workbook_id
- Find a calculation by
  - workbook_id
  - calculation_id
- Find a result by
  - workbook_id
  - calculation_id
- Find a scene by
  - workbook_id
  - scene_id

Finding data - fancy

- Find sources in a scene by
  - scene_element.workbook_id
  - scene_element.scene_id

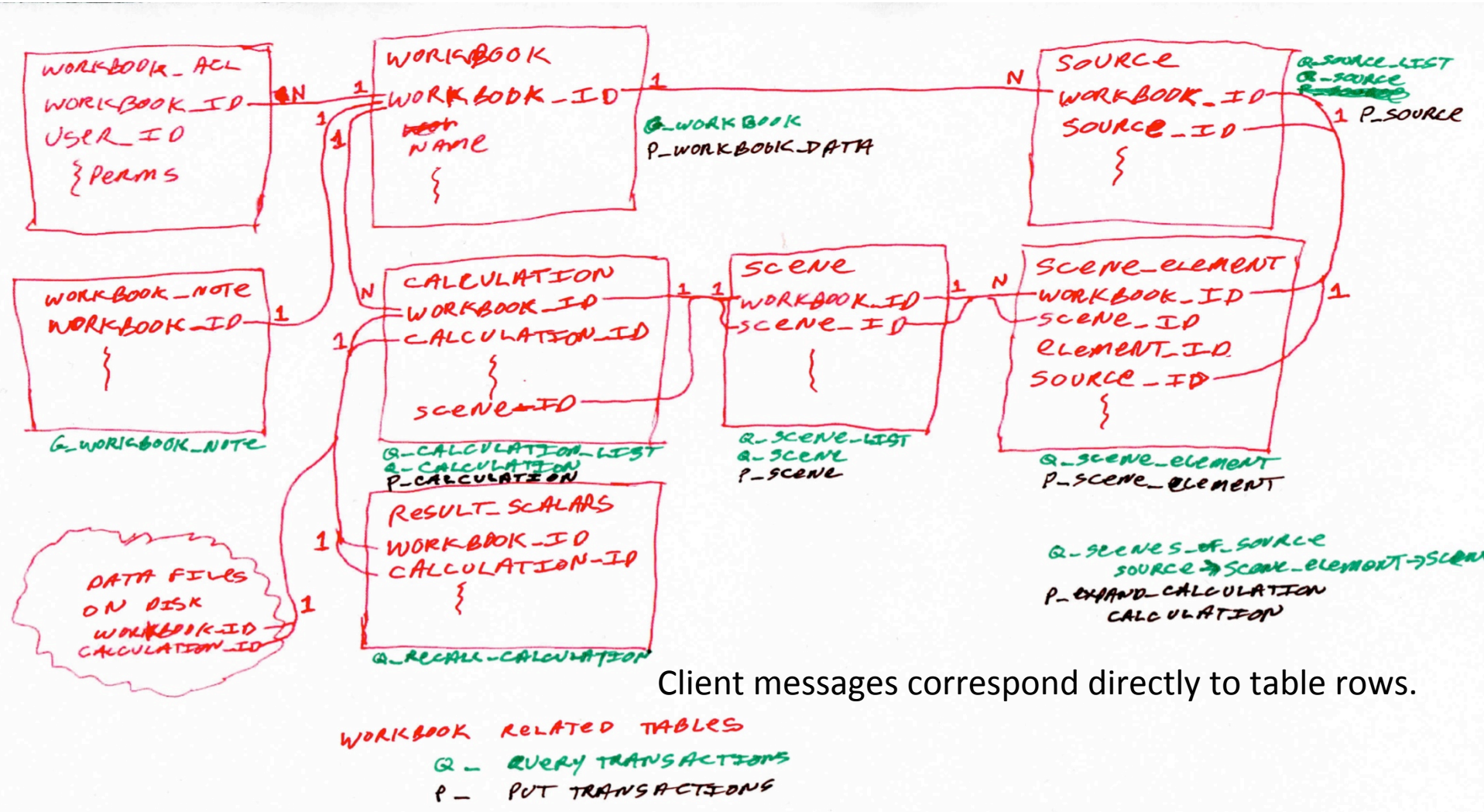
  Yields source_id, and then
  - source.workbook_id
  - source.source_id

- SQL lessons later if you like

Why is it like this?

- The scene_element table is how you do lists in SQL
  - scene table has a scene_id
  - many rows in scene_element table have that scene_id
  - those rows each identify a row in the source table

Mapping to Client



Client messages correspond directly to table rows.

WORKBOOK RELATED TABLES

Q_ QUERY TRANSACTIONS

P_ PUT TRANSACTIONS

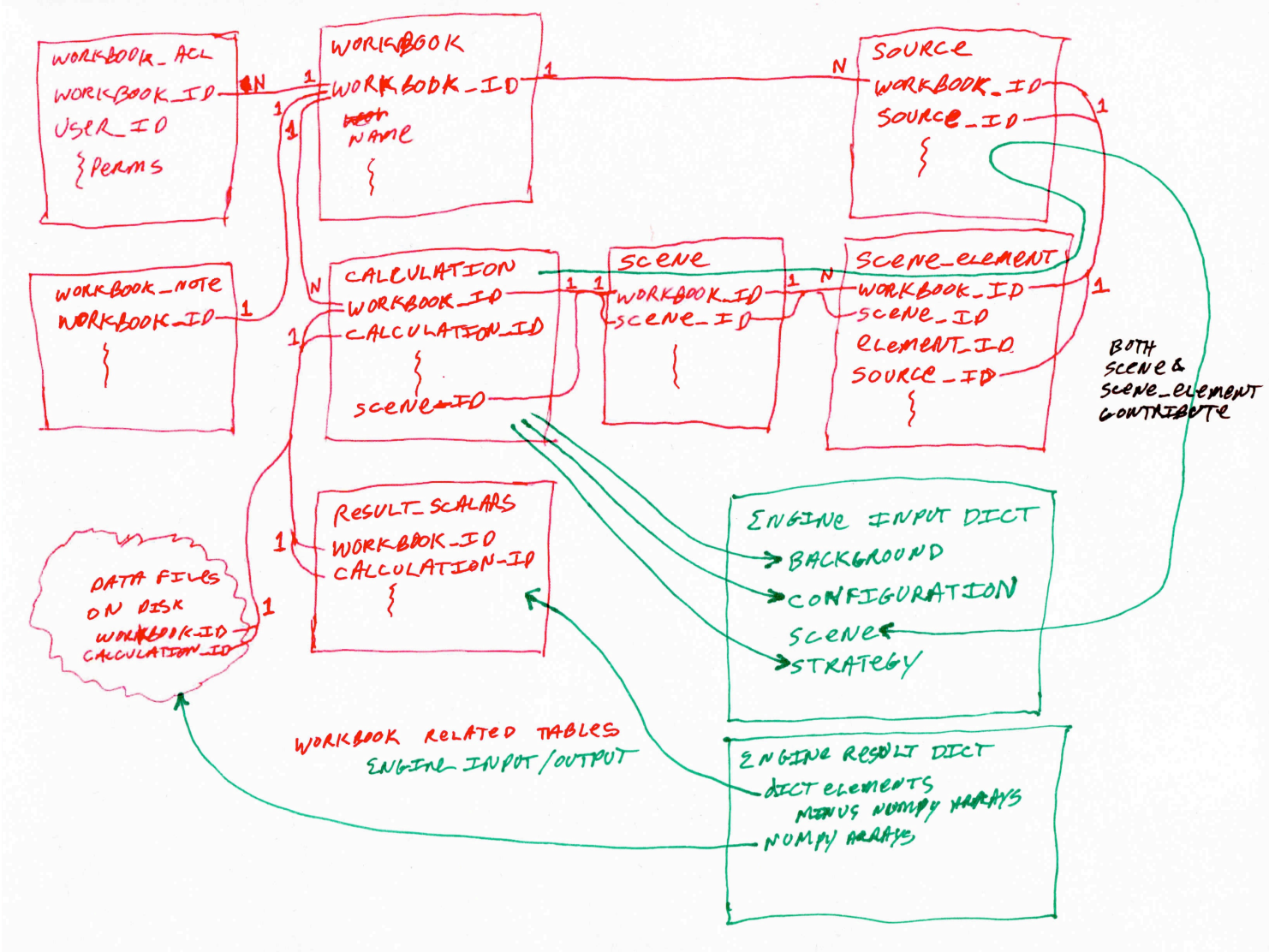
Mapping to Server

- Engine does not have same data model as rest of system
- Single dict:
  - background
    - Is a string from the calculation table; more complex later
  - configuration
    - Is the instrument configuration, from calculation table
  - scene
    - Is a list of sources
  - Strategy
    - Is the strategy configuration, from calculation table

Mapping to Server

Calculation table has data for different parts of engine input dict.

Engine has only one scene, constructed from scene, scene_element, source tables.



json “blobs” to engine input

- calculation.camera_config
  - x[‘configuration’]
- calculation.strategy_args
  - x[‘strategy’]
- scene.characteristics
  - nowhere
- source.characteristics
  - x[‘scene’][n]
- scene_element.source_arg_blob
  - x[‘scene’][n]

More Detail

- pandeia/doc/api_overview/README.txt
  - Currently lists 12 files with more detail
  - Missing client->engine API
    - whatever is in the json blobs

Groundhog eating a Tomato