OpSim v3.3 - Database Agnostic Implementation

We would like to create a streamline process for executing a simulation, running standard MAF analyses on it, and locating output in an easily accessible repository.

There are four options for moving forward:

1. continue as now: run opsim on mysql,, post process to yield a sqlite file, run maf on sqlite
2. run opsim on mysql but directly generate sqlite file, run maf on sqlite
3. run opsim on sqlite solely
4. convert opsim to be ‘database agnostic’

We discussed and agreed on option 4.

- This means (maybe) we will have a separate database file for every sessionID (recorded in a tracking DB) whether it is MySQL or SQLite**.
- OpSim will most likely continue to use the MySQL option internally
- The conversion tool to produce an SQLite file for distribution when needed will be included in OpSim to phase out to keep things simple and less prone to bugs if schemas get out of sync.
- There will be both internally (v3.2) and externally facing repositories (externally facing will use SQLite).

These are the **tasks** and **time estimates (TBD)** to implement this plan:

1. Add code to choose whether to run on SQLite or MySQL (database agnostic)
    a. itemize the steps needed here
    b. continued
2. **Reassess for MySQL whether creating a single DB for all sessionIDs v. creating a single DB for each sessionID is a good decision
    a. this has ramifications for MAF which doesn't operate on the first but can operate on the second
    b. in planning to execute many runs, a single DB for all sessionIDs will get quite large and performing joins could get cumbersome.
3. Make changes in DB schema (make sure SQLite and MySQL match); moving conversion tool inside of OpSim instead of a stand alone tool
4. Add "universal" tag to Proposal table (already added to Config table in v3.2)
5. Eliminate need to manually adjust visitTime and visitExpTime
6. Create master script to run entire pipeline with no manual intervention needed (simulator, standard maf output, relocate files to repository)
7. Production and testing needed for translation layer (not sure what this means as far as tasks)
8. Internal repository was set up in v3.2; plan and implement external repository for v3.3
9. Update Sphinx documentation with differences in function as well as installation if applicable