

# Re: Python Database Apps

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2007-09/msg01392.html>

---

- *From:* David <[wizzardx@xxxxxxxxx](mailto:wizzardx@xxxxxxxxx)>
  - *Date:* Tue, 11 Sep 2007 23:09:34 +0200
- 

It would help to get a feel of what is the most popular combination for people to develop their apps. It's gonna be a desktop app. The database engine is going to be the critical component. I like sqlite so that I don't need a database server on the client side. It would help though if there is a way to sync between multiple clients to a central server database. That's the tough part. To give you a better idea of what i'm trying to do, I am trying to write an app where multiple technicians have an issue tracker that is available both offline and online. I was looking to use sqlite as the local side that would sync new and updated issues to a central server when online. Any technician may work on any issue. Generally they will not be working on the same issue at the same time. The technicians are geographically diverse (anywhere in the southeast US, not in an office.)

If you have an offline mode then the most important thing to work out is how to handle conflicts in your data synchronization. You need a clear logic policy for this, and some means of conflict resolution. Also all parties need to be consistent and not go out of sync too badly. Maybe avoid conflicts altogether. eg define exactly what types of updates can take place offline (record insertions only). Or a policy where deletes+updates can only be made offline by the technician that "owns" the issue/has locked it. This approach has a few issues also.

If the database is relatively small and your data is simple then you could get away with simple table record comparisons to do syncing (only works properly for inserts, and you need 'natural keys' for comparison – updates and deletes are more complicated).

Another (really complicated) way is to use a series of log events, which your app can commit locally or to the server, and merge incoming events from the server. This is similar in principle to a SCM like CVS, subversion, git, bazaar, etc. SVK is probably the closest to your model (centralised scm, with offline commit mode). This gets complicated very quickly.

.