We also unit test every database update. Each test case makes some assertions about the state of the database prior to the update. It applies the update and then makes some assertions about the resulting state.

Still, these tests all work with "well-behaved" data. Weird things happen out in the field, though, and real data is always messier than any test data set. Our updates create tables, add indices, populate rows, and create new columns. Some of these changes can break badly if the data isn't what we expect. We worried about the risky time during the updates and looked for ways to make the process more resilient.

Safety features

Suppose something goes wrong with one of the updates. A studio could be shut down until Operations found a way to restore the database, and if the update really goes wrong, it might leave the database corrupted or in some intermediate state. Then the studio wouldn't even be able to roll back to the previous version of the application. To avoid that disaster scenario, the database updater makes a backup copy of the database before it starts applying the updates. If it can't make the backup copy, then it halts the update process.

If errors occur during the updates, the updater automatically attempts to reload from that backup copy. If even that step fails, well, at least there's a copy onsite so a support technician can talk the studio manager through a manual restore.

In fact, in the absolute worst case, the printing facility always has a copy of the database that's no more than one day old. We used some of the extra space on the daily DVD to send a complete copy of the database every day. There's something to be said for a small database and a lot of storage space.

Field results

The time we invested in automated database updates paid off in several ways. First, we improved performance and reliability through some early updates. Feedback from the user community was immediate and positive after that release. Second, the operations group greatly appreciated the easy deployment of new releases. Previous systems had required the studios to ship removable hard drives back and forth, with all the attendant logistics problems. Finally, having the update mechanism allowed us to focus on "just sufficient" database design. We did not peer into the crystal ball or overengineer the database schema. Instead, we just designed enough of the schema to support the current iteration.

Immutable Data and Ubiquitous GUIDs

In working with customers, the studio associate creates some compositions that use multiple photographs, inset into a design. These designs come from a design group at company headquarters. Some designs are perennial, others are seasonal. Christmas cards in a wide