You Can't Move Forward Unless You Can Roll Back

By: Michael Black

The VP of Sales walks in

and tells you that your largest and oldest client wants to pay for a custom modification.

But here's the clincher, they haven't updated their software since before you were born, and the database today looks drastically different.

Briefly you consider hiding

but no, that won’t work because the VP is staring right at you waiting for your response.

It would be career suicide to tell him “No”, so you accept the challenge and start thinking how you’re going to make a modification on a database of which you can’t verify the structure.

In the storage closet

you just so happened to find some old floppy disks that are labeled:

The Application v1.0

The only problem is, good luck finding a computer that has a floppy drive in it. And who's to say that this version was the one actually released to the public?

Wait, what about the change log?

Every time a change is made to a stored procedure, function, or package; the architect is supposed to write a change log at the top of the script.

The procedures look like this:

```
CREATE OR REPLACE PROCEDURE my_proc IS
/******************************************************************************
   NAME:      my_proc
   PURPOSE:    This is a very important procedure in this application
   REVISIONS:
   Ver        Date          Author        Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        2/3/1980    Joe      Created this procedure.
   1.1        2/3/1985    Bob      There was huge error in the way we were calculating total debts payable which
caused huge problems. This change fixes this problem.
   1.2        2/3/1990    Adam     Changed lines 220-332
   1.3        2/3/1995    Steve    Changes lines 23-35
   1.4        2/3/2000    Joe      VP of Sales changed how he wants projected revenue to be calculated
   1.5        2/3/2005    Bob      New VP of Sales wanted to change how projected revenue is done
   2.0        2/3/2010    Adam     v2.0 of the application is released which changed how the tables are structured
and how this proc needs to load the data
   2.1        2/3/2015    Steve    Changed lines 75-98 with bug fixes related to the mobile app version of the
application
```

You come to the realization

that Change Logs don’t suffice when you need to be able to see line-by-line differences from one version of file to the next and leap between versions. As well as be able to make alternate paths of development.

You called up your DBA buddy

who works for the client and he's able to provide you an empty shell of the database. You somehow manage to pull everything off before the deadline and make the special modifications for the client.

The VP is so pleased he takes you golfing with him that weekend.

Determined not to let this happen

again, you decide to make a trip over to the application development department and learn how they were able to so easily handle this situation as they too had to write a modification to the GUI. You learn that they use a version control system to manage all their coding development.

Version Control Systems (VCS)

There are a lot of vendors, 40+ easily to choose from. Some of the more popular: Subversion (SVN), Git, Team Foundation.

Although many options exist, concepts and best practices can be used no matter which software solution you're using.

A good VCS provides

- a versioning methodology for each revision of an object
- isolation between working copies
- isolation between development tracks
- ability rollback to previous versions
- change/merge tracking
- the peace of mind to sleep at night

Basic Concepts

- Repository
- Working Copy / Checkouts
- Commits
- Branching
- Tagging
- Locks

Repository

The storage location of your files for safe keeping. There are centralized repositories and distributed models.

Centralized	Distributed
Client-Server	Peer-to-peer
SVN	Git
Perforce	Mercurail

Repository

The repository can be thought of as a library where you check files out. Much like a real library when you check out a book, developers can check out code, but with one big benefit; you don’t have to worry about losing a book or paying late fines if you don’t return it back in time.

Working Copy / Checkout

Your copy of the files on your computer. Any changes you make to these files are isolated from the copies your co-workers have checked out on their machines. Your changes don’t get shared with others until you commit them.

Committing / Checking-in

Sending changes from your local working copy to the repository. Often with the intent to be shared with everyone else.

Each commit is recognized as a new version of the file. If you ever need to you can compare any version to any other version.

Branching

Diverging a copy of the repository into its own development track with its own isolated series of revision numbers.

Branches can later be merged back into the mainstream (trunk) line of code, vice versa, or into other branches.

Branching - to do or not to do

Release Branching - Main development on the trunk. When a release is made a branch is also without the intention of ever being reintegrated.

Feature Branching - When difficult or long-term isolation is needed from the trunk, a branch is made with the intention of later integration.

Tagging

Marking and isolating a particular set of revisions (taking a snapshot) to an important point in time of how the database looked.

Development is not done on a tagged version.

Example Tags: v1.0 or v2.0

Locks

Locking a file prevents anyone else from committing changes.

In some VCS you have to lock/check-out an object before you can do work; pessimistic change control. Other VCS take an optimistic approach and you only lock when merging would be difficult.

Benefits of VCS

- Facilitates collaboration
- Complements Dev/Test/Prod practices
- Disaster Recovery
- Change Management / Change Logs
- Client Request Management

Collaboration

A VCS allows architects to work on same objects of database at the same time and be able to merge their changes together.

Or with a VCS an architect can lock an object while they are making edits so no one else makes changes that will be erased when you apply your changes.

Dev / Test / Prod

Branches provide the ability to have your dedicated lines of development for each environment, and with the merging capabilities, you can implement your changes from one environment to another.

Disaster Recovery

Yes, we all take backups of our databases but what about when that rogue disgruntled employee slowly over several months discreetly implements bugs and other changes over several backups? VCS tracking can easily show you every change a person makes.

Change Log / Change Management

VCS logs can be easily extracted to be surfaced to interested parties about changes going on.

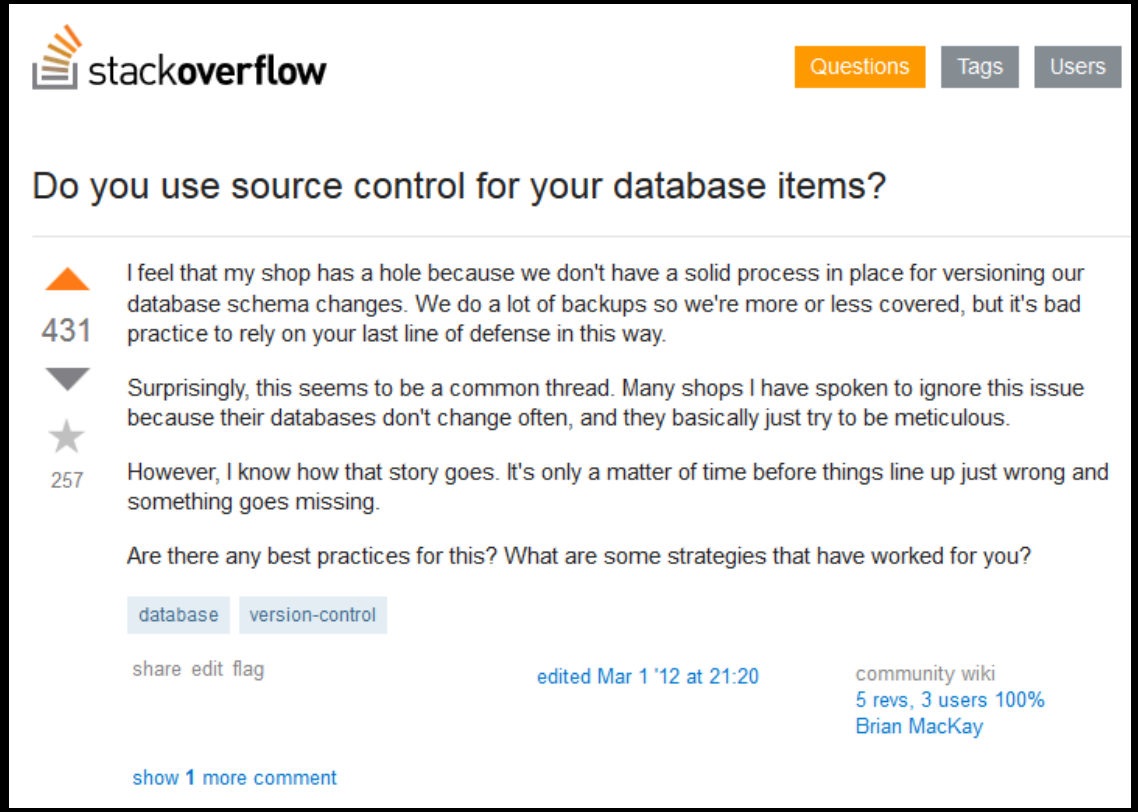
Architects can pick and choose which changes are merged when and where and not have to delay current development for release cycles.

Revisions can be executed in Release Cycles.

Client Request Management

If you have a client that frequently changes their mind, having well documented changes and logs why changes were made can aid in reminding a client why a change was made. The conversations and reasoning that were decided for that change to take place can be in code and in the commit logs.

You’re Not Alone



Why aren’t more DBs under VCS

- The databases are often a shared environment, developers rarely have their own private environment
- Detecting when changes occur
- DML is different if you’re creating vs. modifying

Shared Environments Are Tough

In typical application development, you can write, test, and run your application in complete isolation on your computer. Database development typically takes place on a shared server that everyone has their fingers in and are making changes all the time.

Detecting When Changes Occur

You know if a file of code changes on your personal computer, but how to do you tell if an object was changed and by whom?

There are 3rd party applications that can aid in this, or making the social/cultural change to diligently use the VCS can mitigate this.

DML Creating vs. Modifying

CREATE table vs. ALTER table

Which statement do you version control?

How do you keep track of sequence of events of changes to be made?

Is this really for me and my team?

YES!

Or you better be prepared for the next time the VP of Sales walks in with another demand from another ancient client.

Despair.com

CHANGE

WHEN THE WINDS OF CHANGE BLOW HARD ENOUGH,
THE MOST TRIVIAL OF THINGS CAN TURN INTO DEADLY PROJECTILES.

Michael Black