**What is the difference between a database and a repository?**

A Repository is a database that stores the metadata for Designer objects.

In other words, the data associated with objects like Entities, Modules, Table Definitions, etc. is stored as rows in tables that are part of the Repository.

A Repository would essentially be linked to at least one database, but in general, it is linked to many databases.

A database is just a place to store data, or an application database is a place to store the data for a particular computer application.

**What is version control?**

Benefits of version control

Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

For almost all software projects, the source code is like the crown jewels - a precious asset whose value must be protected. For most software teams, ***the source code is a repository*** of the invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort. Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.

Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree". One developer on the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree.

Good version control software supports a developer's preferred workflow without imposing one particular way of working. Ideally it also works on any platform, rather than dictate what operating system or tool chain developers must use. Great version control systems facilitate a smooth and continuous flow of changes to the code rather than the frustrating and clumsy mechanism of file locking - giving the green light to one developer at the expense of blocking the progress of others.

**1.A complete long-term change history of every file.**

**2.Branching and merging.**

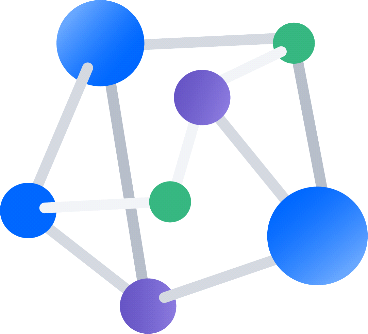
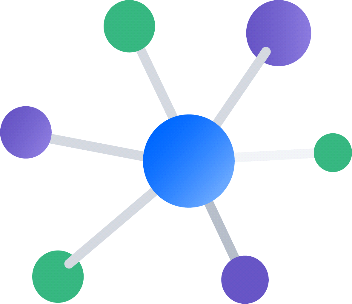
**3**.**Traceability**. Being able to trace each change made to the software and connect it to project management and **bug tracking software such as Jira,**

* **Conflict resolution**
* **Rollback and undo changes to source code**
* **Offsite source code backup:** When using a VCS collaboratively, a remote instance of the VCS needs to be created to share changes between developers. This remote VCS instance could be hosted off-site with a trusted third-party like Bitbucket. It then becomes a secure offsite backup. In an unfortunate scenario like a stolen laptop, the remote VCS instance will still retain a copy of the source code.

**Types of VCS (Version Control System)**

A VCS will track additions, deletions, modifications of the lines of text within that file. Popular software industry VCS options include **Git, Mercurial, SVN (Apache SubVersioN), CVS (Concurrent Versions Sys) and preforce.**

1. Centralized VCS
2. Distributed VCS

DISTRIBUTED CENTRALIZED 

In centralized VCS if the off-site source code back-up is lost it can cause productivity and data loss, and it will need to be replaced with another copy of the source code. If it temporarily becomes unavailable, it will prevent developers from pushing, merging or rolling-back code.

A distributed model architecture avoids these pitfalls by keeping a full copy of the source code at each VCS instance. If any of the previously mentioned centralized failure scenarios happen within the distributed model, a new VCS instance can be swapped in to lead development mitigating any serious drop in productivity.

**GIT** - very hot since Linus switched to it. this is distributed type and Open Source

**mercurial** - some smart people I know swear by it, is distributed type and open source and simpler compared to GitHub.

***Bazaar:*** is yet another distributed version control system, like Mercurial and Git, that offers a very friendly user experience. It calls itself “Version control for human beings.” It supports many different types of workflows, from solo to centralized to decentralized, with many variations in between.

**SVN** - currently the most popular open source. This is centralized type.

with widest adoption. Most open-source projects use Subversion as a repository because other larger projects, such as Source Forge, Apache, Python, Ruby and many others, use it as well. Google Code uses Subversion exclusively to distribute code.

If you’re a Windows user, Tortoise SVN is a great file browser for viewing, editing and modifying your Subversion code base. If you’re on a Mac, Versions is an elegant client that provides a “pleasant way to work with Subversion.” XCode is Apple’s developer environment and Subversion client that ships with Leopard on a Mac.

**CVS** - the one everybody is switching from. This is centralized type.

Tortoise CVS is a great client for CVS on Windows, and there are many different IDEs, such as XCode (Mac), Eclipse, NetBeans and Emacs, that use CVS.

**perforce** - imho, the best features, but it's not open source. The two-user license is free, though.

**Visual SourceSafe** - I'm not much in the Microsoft world, so I have no idea about this one, other than people like to rag on it as they rag on everything from Microsoft.

**SCCS** - for historical interest we mention this, the great-granddaddy of many of the above

**RCS (Revision Control Sys)** - and the granddaddy of many of the above

Acct in git and learn commands and try

https://btulasiramreddy@bitbucket.org/btulasiramreddy/training.git