Version Control

Lecture 4

Administrivia

- In order to do the homework and turn in the project, you *must* have run set-keys. Do so today!
- HW assignment will be posted Friday (for next Friday)
- Reading for Tuesday: Notes §2.1-2.7

The Problem

- Software projects can be large and complex.
- May involve many people, geographically distributed
- May require maintenance of several related versions
  - MacOS vs. Windows vs. GNU Linux
  - Stable release vs. beta release of next version
  - Commericial vs. non-commercial
- May require prototyping potential features while still maintaining existing ones.

Version-Control Systems

- Version-control systems attempt to address these and related problems.
- Allow maintenance and archiving of multiple versions of a piece of software:
  - Saving complete copies of source code
  - Comparing versions
  - Merging changes in several versions
  - Tracking changes

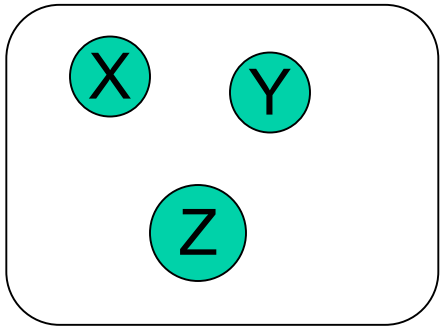
Subversion

- Subversion is an open-source version-control system.
- Successor to CVS
- Provides a simple model: numbered snapshots of directory structures
- Handles local or remote repositories

Subversion's Model

Repository

User 1



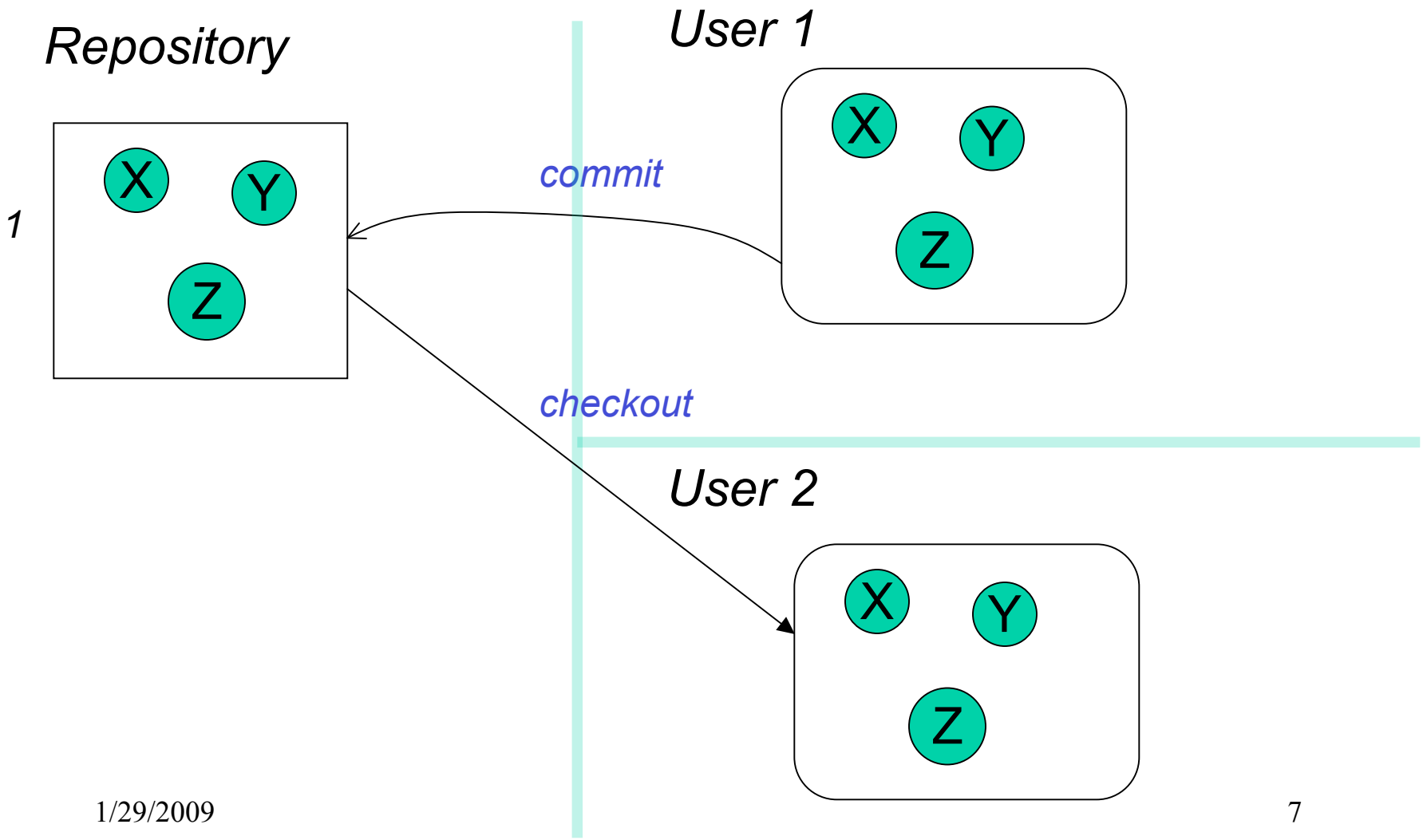
add X

add Y

add Z

User 2

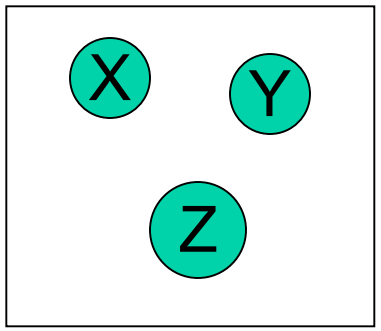
Subversion's Model



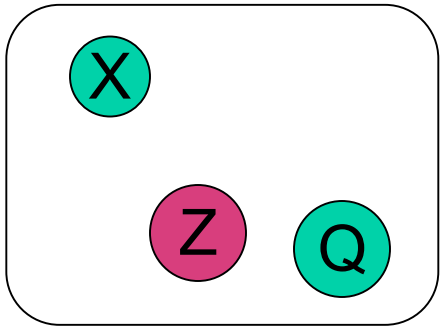
Subversion's Model

Repository

1



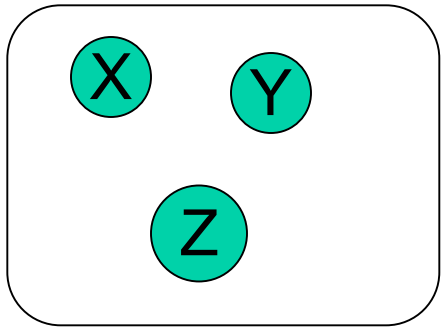
User 1



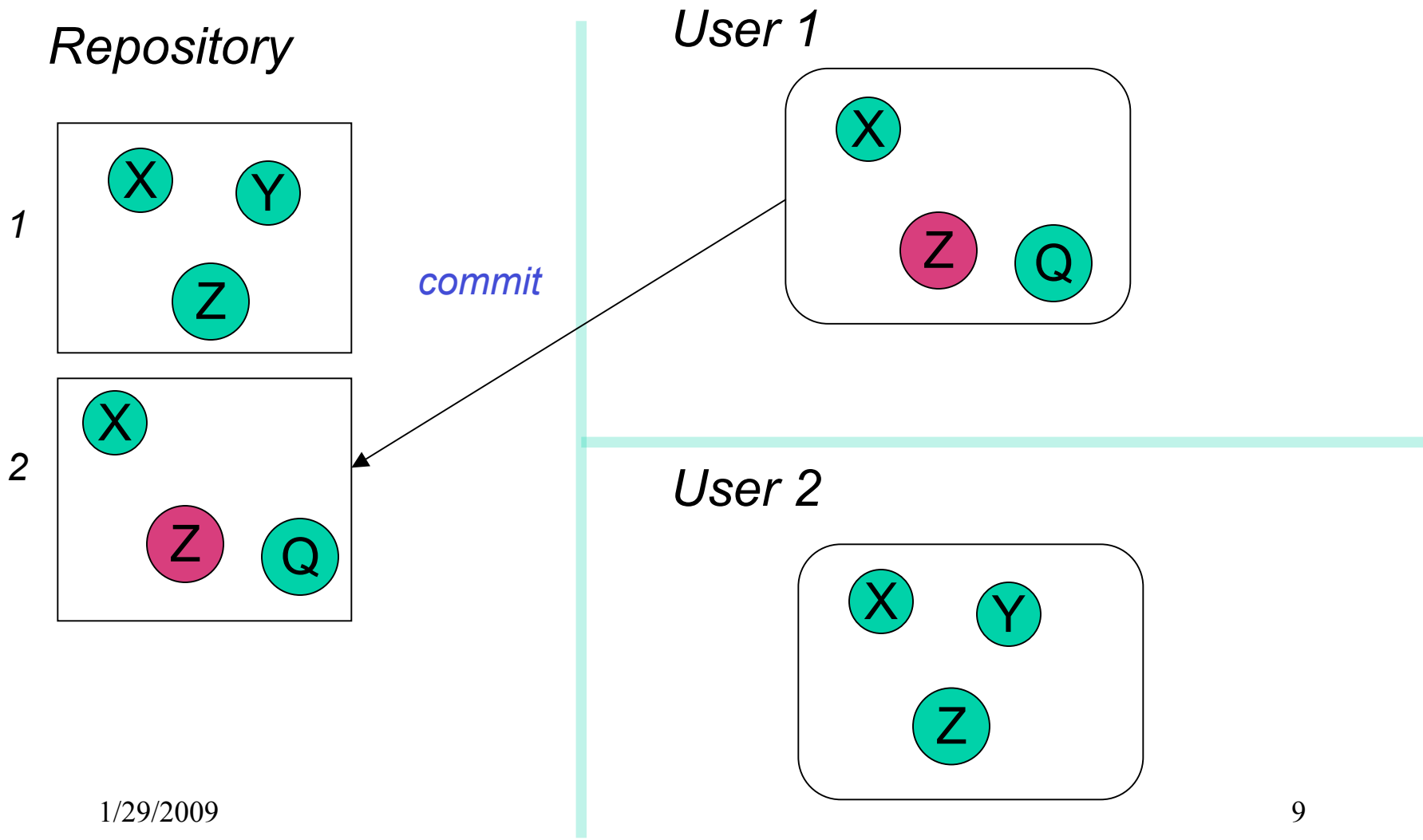
Delete Y

Add Q
Modify Z

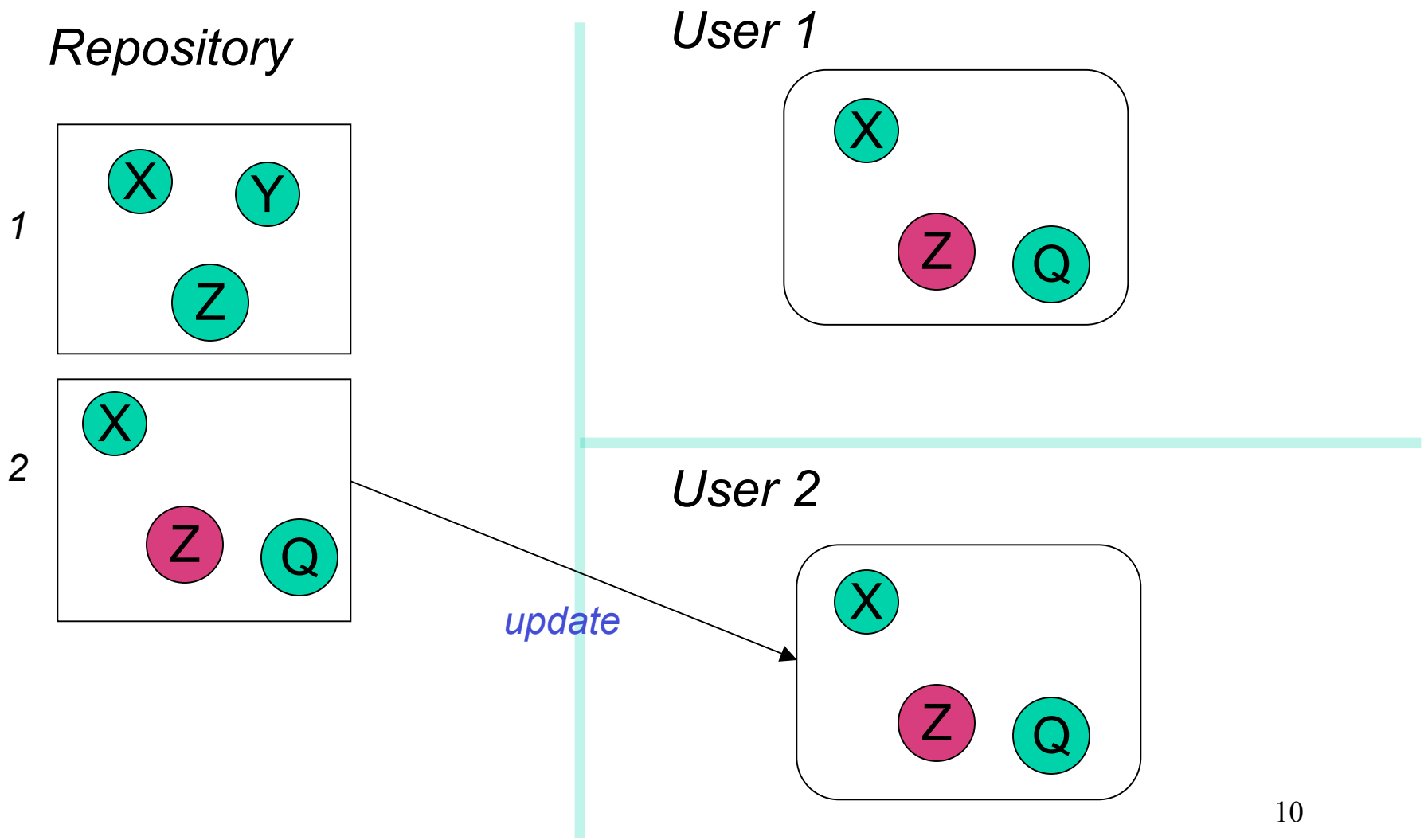
User 2



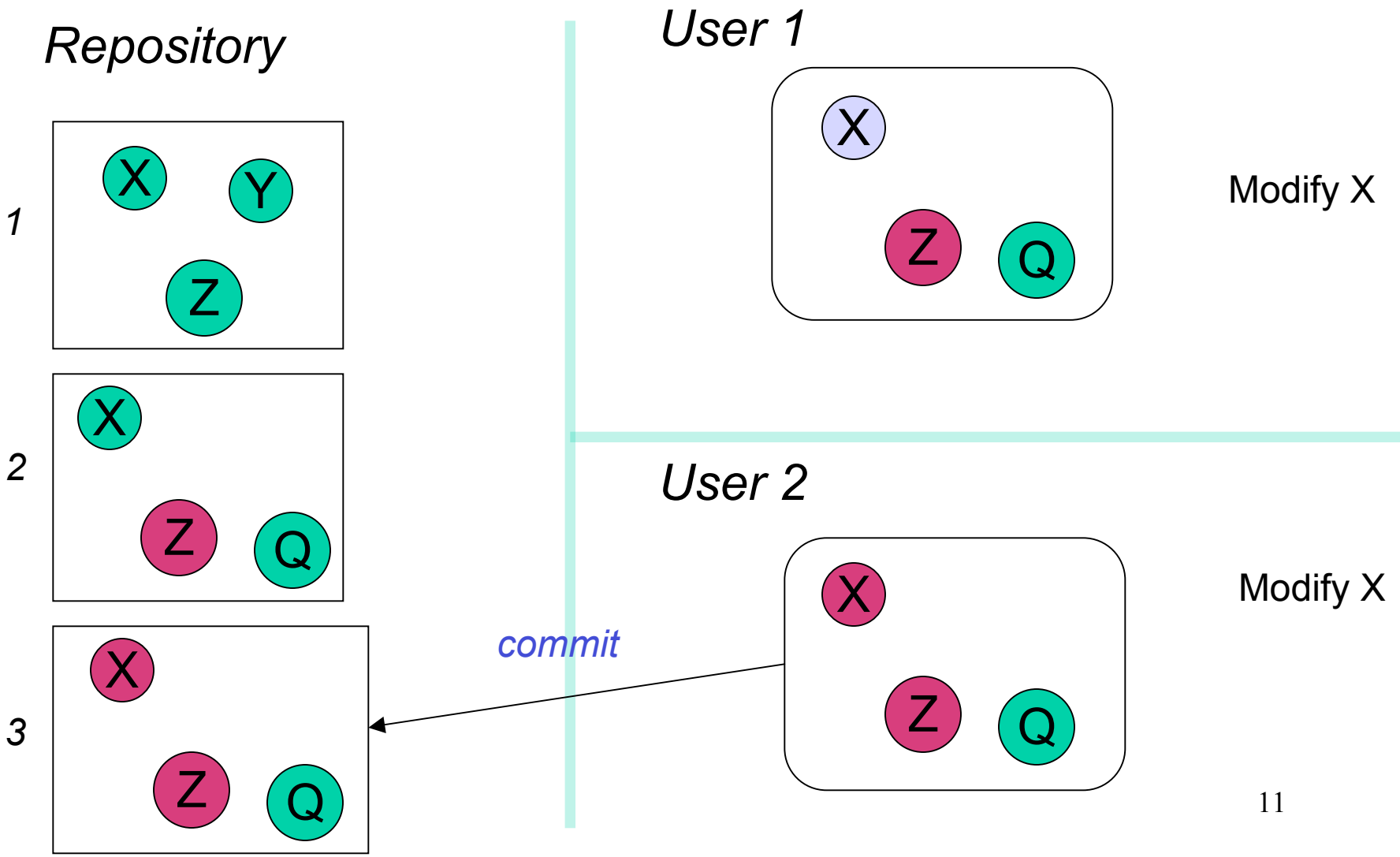
Subversion's Model



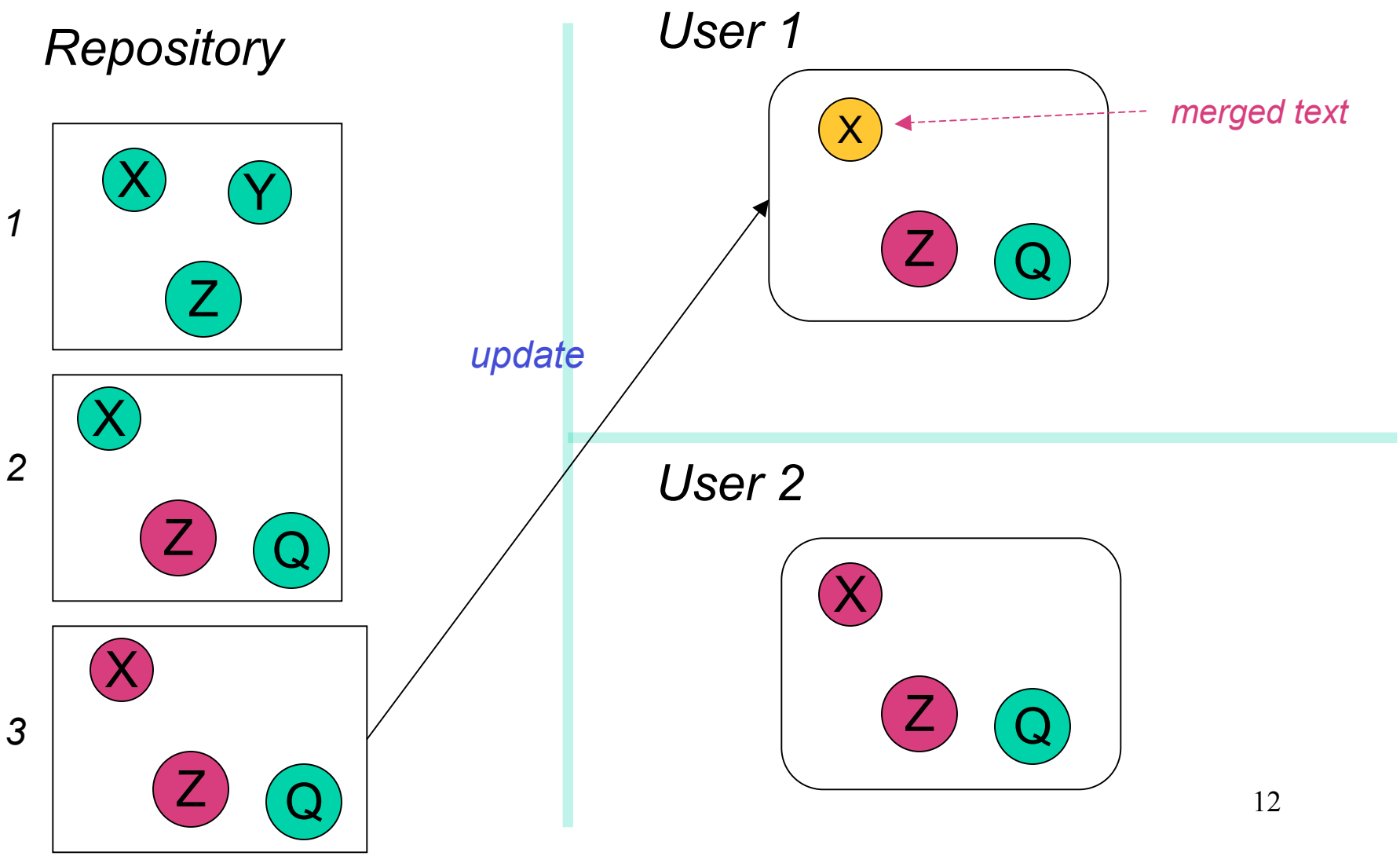
Subversion's Model



Subversion's Model



Subversion's Model



Terminology

- Repository: Set of versions
- Revision: A snapshot of a particular directory of files
- Revision number: A sequence number denoting a particular revision
- Working copy: A directory or file initially copied from a revision + administrative data

A Useful Property

- In the previous example, Subversion does not really keep 3 complete copies of the files.
- Instead, it maintains *differences* between versions: if you change little, your revision takes up little space.
- Copying an entire file or directory in the repository is very cheap
  - "Directory foo in revision 110 is the same as directory bar in revision 109"

Some Basic Commands

- We'll be using "ssh tunnels" to access our Subversion repositories.
- We created an ssh key pair for you when you first logged in.
- In the following, we consider login cs164-xx and team Ursa; we'll use nova as a convenient host.

Creating a working copy of a repository

- To get the latest revision of projects:

```
svn co svn+ssh://cs61b-ta@nova/Ursa/trunk  mydir
```

- Or just one directory:

```
svn co svn+ssh://cs61b-ta@nova/Ursa/trunk/proj1 \
    mydir
```

- A particular revision:

```
svn co -r100 \
    svn+ssh://cs61b-ta@nova/Ursa/trunk/proj1  old1
```

Some useful (local) abbreviations

- On instructional accounts, I have defined a few shortcuts:
  - $REPOS =
    svn+ssh://cs164-ta@quasar.cs.berkeley/cs164-xx
  - $STAFF =
    svn+ssh://cs164-ta@quasar.cs.berkeley/staff
  - $TEAMREPOS =
    svn+ssh://cs164-ta@quasar.cs.berkeley/URSA

  I'll use these from now on.

Abbreviated commands

- To get the latest revision of projects:

  svn co $TEAMREPOS/trunk mydir

- Or just one directory:

  svn co $TEAMREPOS/trunk/proj1 mydir

- A particular revision:

  svn co -r100 $TEAMREPOS/trunk/proj1 old1

Add, Delete, Rename Files, Directories

- When you add or remove a file or directory in a working copy, must inform Subversion of the fact:
  - svn add *NEW-FILE*
  - svn delete OLD-FILE-OR-DIR
  - svn move OLD-PLACE NEW-PLACE
- These forms *don't* change the repository, just your personal working directory.
- Must commit changes to change repository.

Reverting

- Before committing, can undo adds, removes, modifications.
- The command

  % svn revert *FILE*

  undoes changes to FILE.
- Reverting a modification or delete restores file.
- Reverting an add removes *FILE* from version control.

Committing Changes

- The command

  svn commit -m "Log message"

  in a working directory will create a new revision in the repository
- New revision differs from previous in the contents of the current directory, which may only be part of the whole tree.
- Message should be informative. If you leave off the -m, will call your favorite editor.

Updating

- To get versions of files from most recent revision, do this in directory you want updated

  svn update

- This will report files Subversion changes, adds, deletes, or *merges*
- Merged files are those modified both by you and (independently) in the repository since you updated/checked out.

Merges and Conflicts

- Reports of changes look like this:

```
U  foo1    foo1 is updated
A  foo2    foo2 is new
D  foo3    foo3 was deleted
R  foo4    foo4 was deleted, then re-add
G  foo5    foo5 had mods from you and in
           repository that did not overlap
C  foo6    Conflicts: overlapping changes
```

Notating Conflicts

- When you have a conflict, you'll find that the resulting file contains *both* overlapping changes:

```
<<<<<<<<< .mine
My change
========
Repository change
>>>>>>>>>> .r 99
```

(gives revision #)

Resolving Conflicts

- You can either choose to go with the repository version of conflicted file, or yours, or do a custom edit.
- Subversion keeps around your version and the repository version in foo6.mine, foo6.99
- Personally, I usually just edit the file.
- When conflicts are resolved, use

  svn resolved foo6

  to indicate resolution; then commit.

Branches and Tags

- Suppose Bob wants to make some changes to his project, checking in intermediate steps, but without interfering with partner Mary.
- Good practice is to create a *branch*, a copy of the project files independent of the trunk.
- Copy command does it:

```
svn copy $TEAMREPOS/trunk  \
         $TEAMREPOS/branches/Bobs-branch
svn co $TEAMREPOS/branched/Bobs-branch mydir
```

and go to work (be sure trunk is committed first).

Branches and Tags

- The use of the branches directory is convention; could put it anywhere.
- Again, this copy is cheap in the repository.
- Bob's changes in branches/Bobs-branch are completely independent of the trunk.
- Rather elegant idea: no new mechanism!

Tags

- A *tag* is the same as a branch, except that (by convention) we don't usually modify it once it is created.
- Conventional to put it in the tags subdirectory, as in the instructions for turning in your project.
- Tags are usually intended as names of particular snapshots of the trunk or some branch (e.g., a release).

Typical examples of turning in work

- You've completed hw1 and want to hand it in. Currently, it's in $REPOS/trunk/hw1 (and, of course, committed). Use

  % svn copy $REPOS/trunk/hw1 $REPOS/tags/hw1-N

  Or (using UNIX shorthand):

  % svn copy $REPOS/{trunk/hw1,tags/hw1-N}

  where N is unique number.

- Team submission:

  % svn copy $TEAMREPOS/{trunk/proj1,tags/proj1-N}

Comparing Revisions

- One great feature: ability to compare versions, branches.
- Simple case: what local changes have I made to this working directory?

  svn diff
- How does this working directory compare to revision 9?

  svn diff -r 9
- How do revisions 9 and 10 of directory differ?

  svn diff -r 9:10

More Comparisons

- How does Bobs-branch compare to revision 100 of the trunk?

```
svn diff  $TEAMREPOS/branches/Bobs-branch \
          $TEAMREPOS/trunk@100
```

Merging

- To *merge* changes between two revisions, *R1* and *R2*, of a file or directory into a working copy means to get the changes that occurred between *R1* and *R2* and make the same changes to the the working copy.
- To merge changes into current working copy:

  svn merge SOURCE1@REV1 SOURCE2@REV2

  where SOURCE1 and SOURCE2 are URLs (svn+ssh:...) or working directories and REV1, REV2 are revision numbers.

More Merging

- For short, when sources the same:

```
svn -r REV1:REV2 SOURCE
```

- To merge in changes that happened between two tagged revisions:

```
svn $TEAMREPOS/tags/{v1,v2} \
        branches/Bobs-branch
```

After Merging

- After merging, as for update, must resolve any conflicts.
- Then we commit the merged version.

Getting Information

- The command svn status is your friend. Identifies
  - changes, additions, deletions that have not been committed;
  - files, directories that have not been added
  - things you've messed up.
- To list what's in a repository directory:
  - svn ls $TEAMREPOS/tags
- To list revisions of a file or directory:
  - svn log FILEORDIR
  - svn log -v FILEORDIR # For details

Final thought

- If you commit early and often, system is quite forgiving. You can reconstruct previous states. You can freely clean things up by deleting and checking out again.
- BUT for this to work, you must commit regularly and must make sure that everything you want is under version control (svn status)