Projet Informatique

2019

Steve Hostettler

Software Modeling and Verification Group

University of Geneva

Gestion du cycle de développement

Environnement de développement

Environnement de développement intégré:

- Edition du code
- Compilation
- Aide au débogage

Nous utiliserons Eclipse

Présentation d’Eclipse

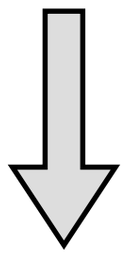
Gestion des versions

Partager les sources d’un projet

Archiver et sauvegarder les différentes versions

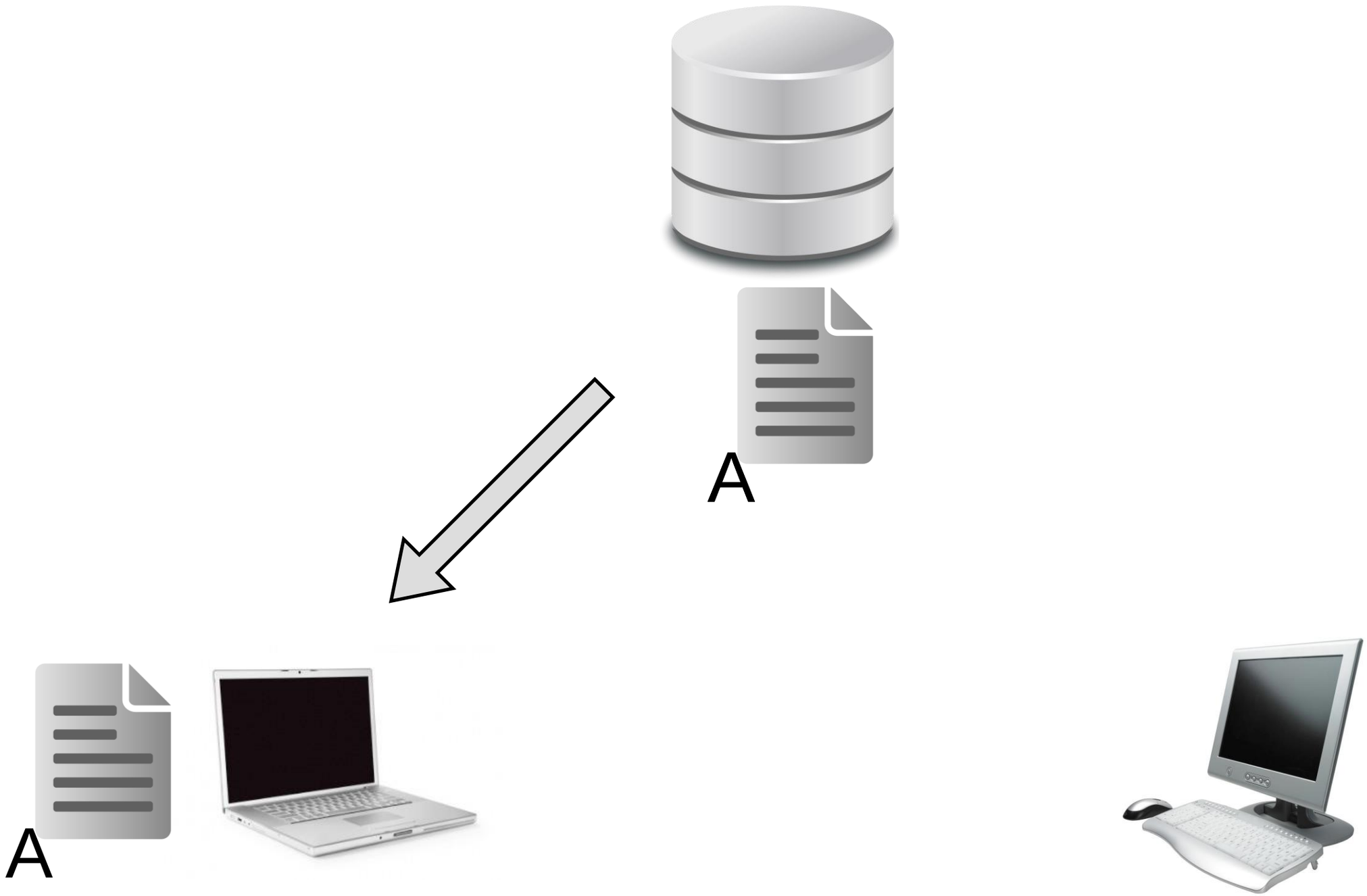
Revenir en arrière facilement

Pouvoir travailler hors-ligne

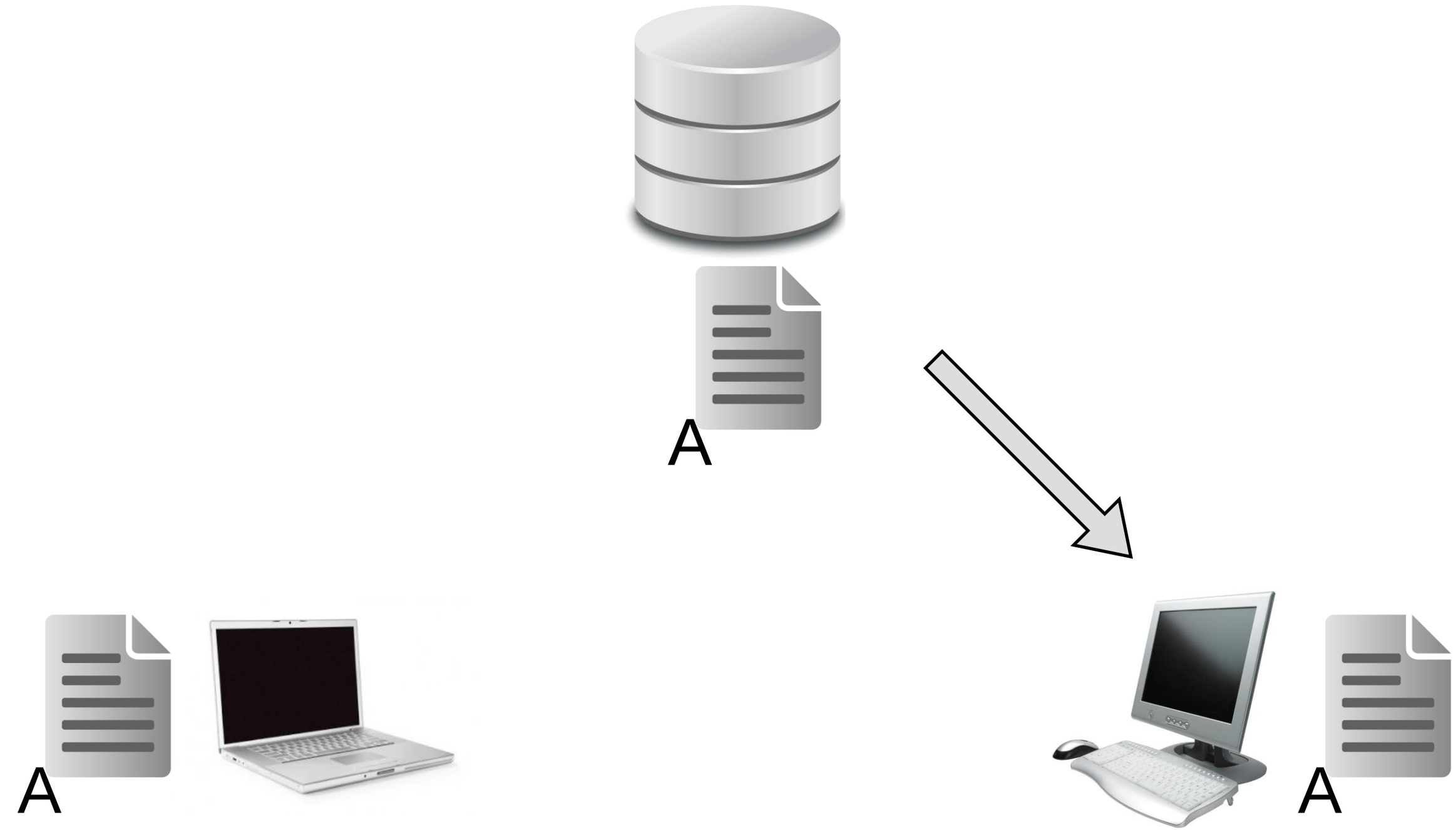


Système de gestion de versions

A



1. Lit A


A
A
A
2. Lit A

A

A’

A

3. Modifie A en A’

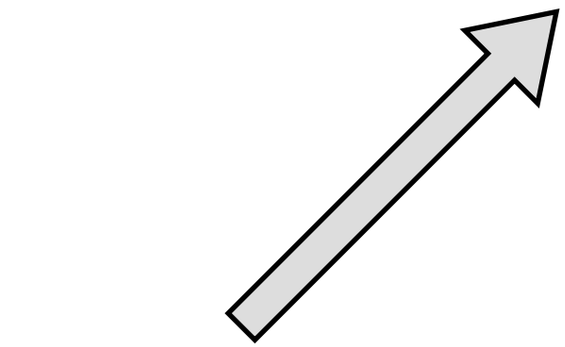
A

A’

A’’

4. Modifie A en A’’

A’

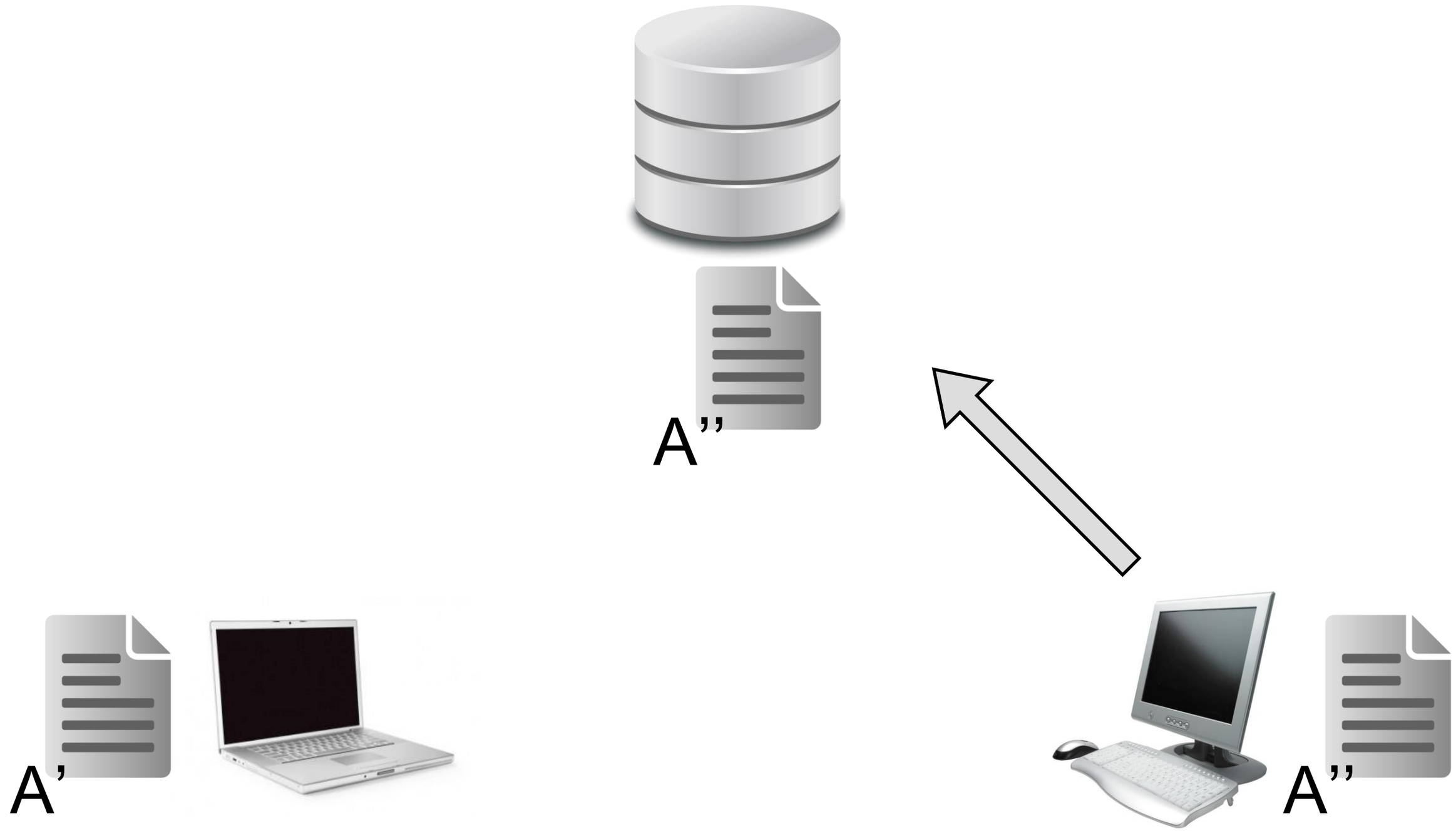


A

A’

5. Sauve A’

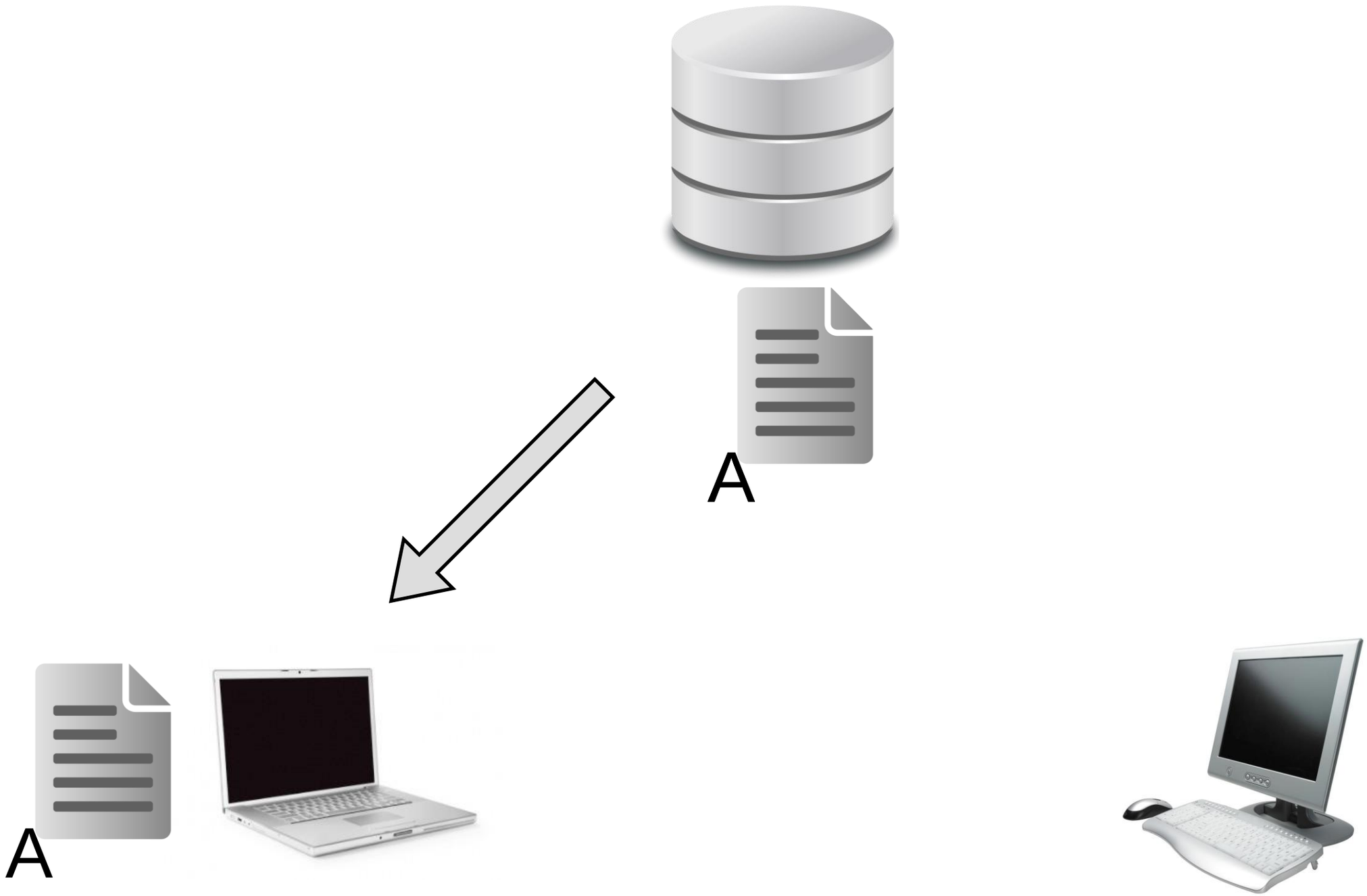
A’’


A’’
A’
A’’
6. Sauve A’’

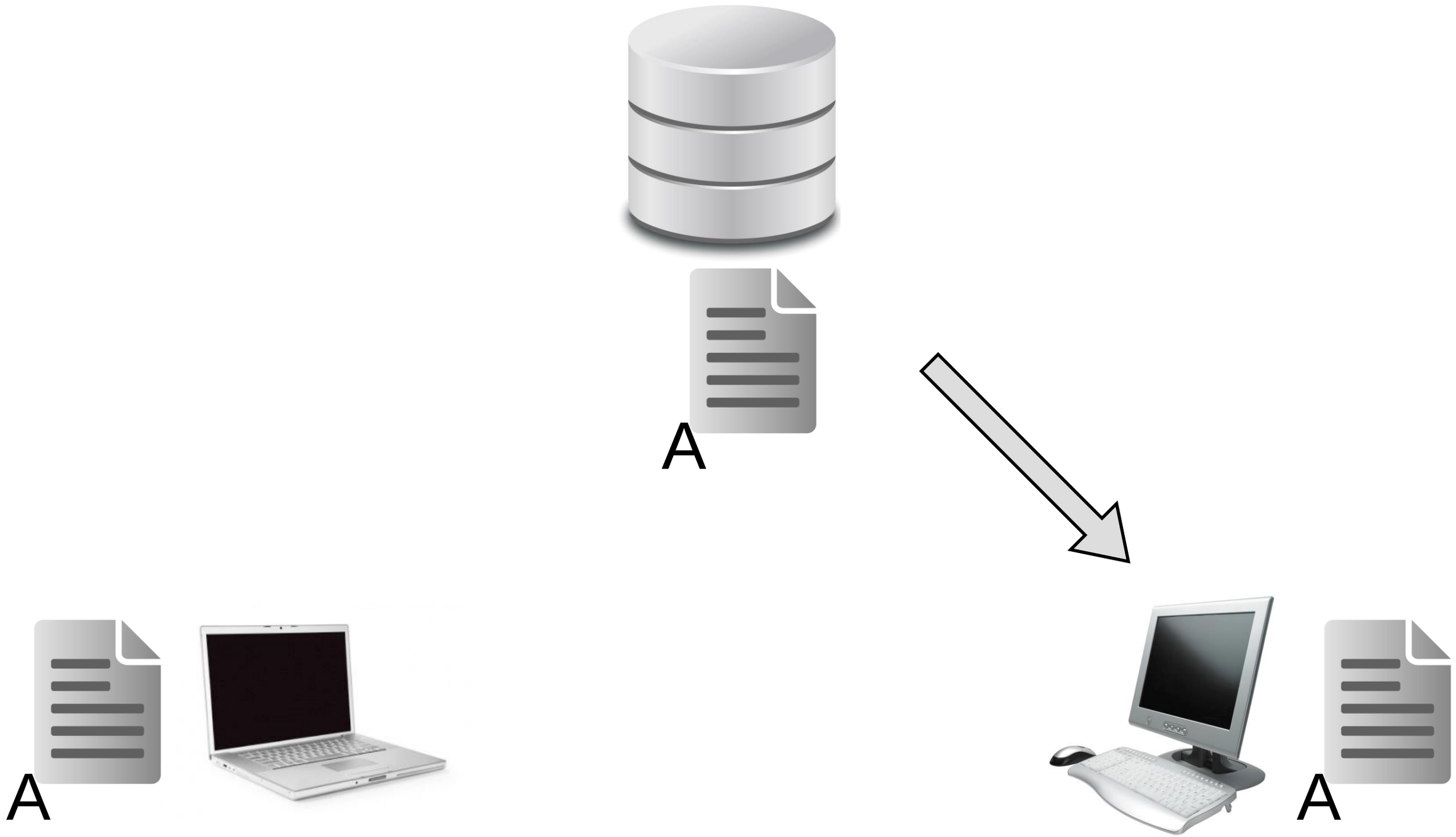

Conflit de version
A’
A’’

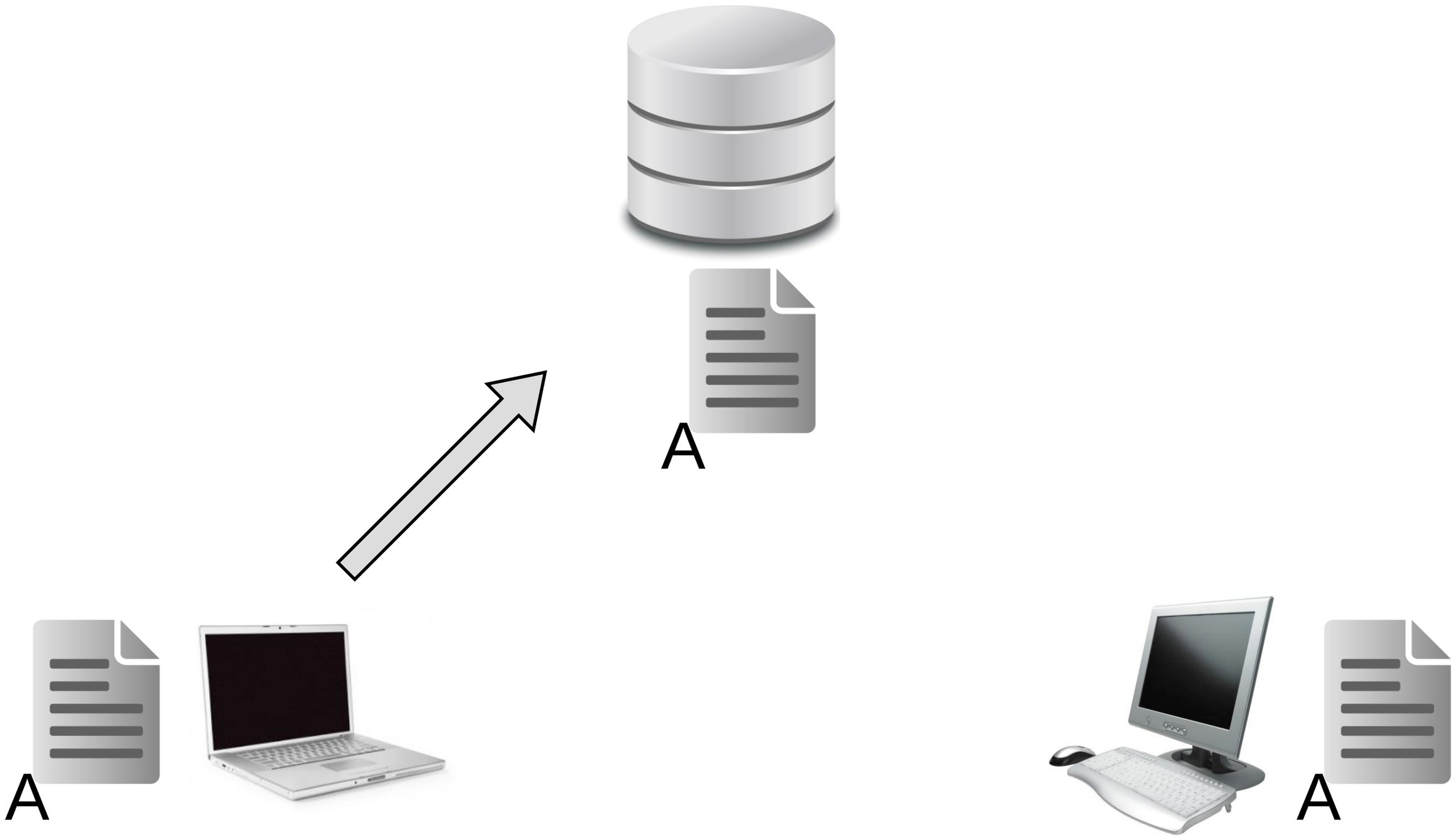
Solution 1: Protéger le fichier par un verrou

A



1. Lit A


A
A
A
2. Lit A

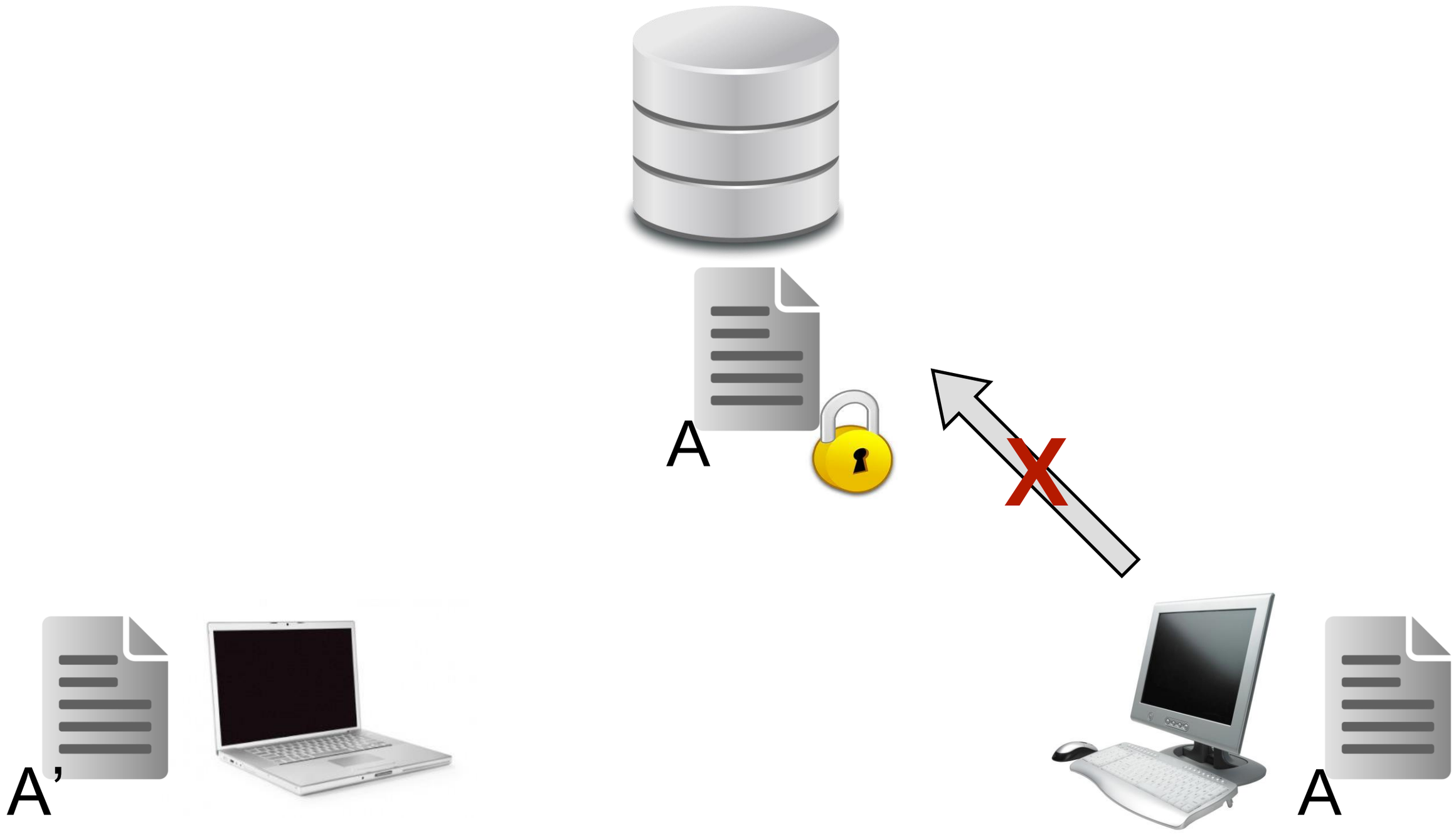


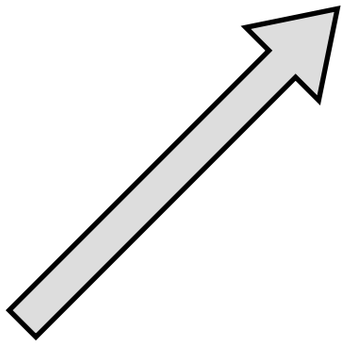
3. Verrouille A

A’

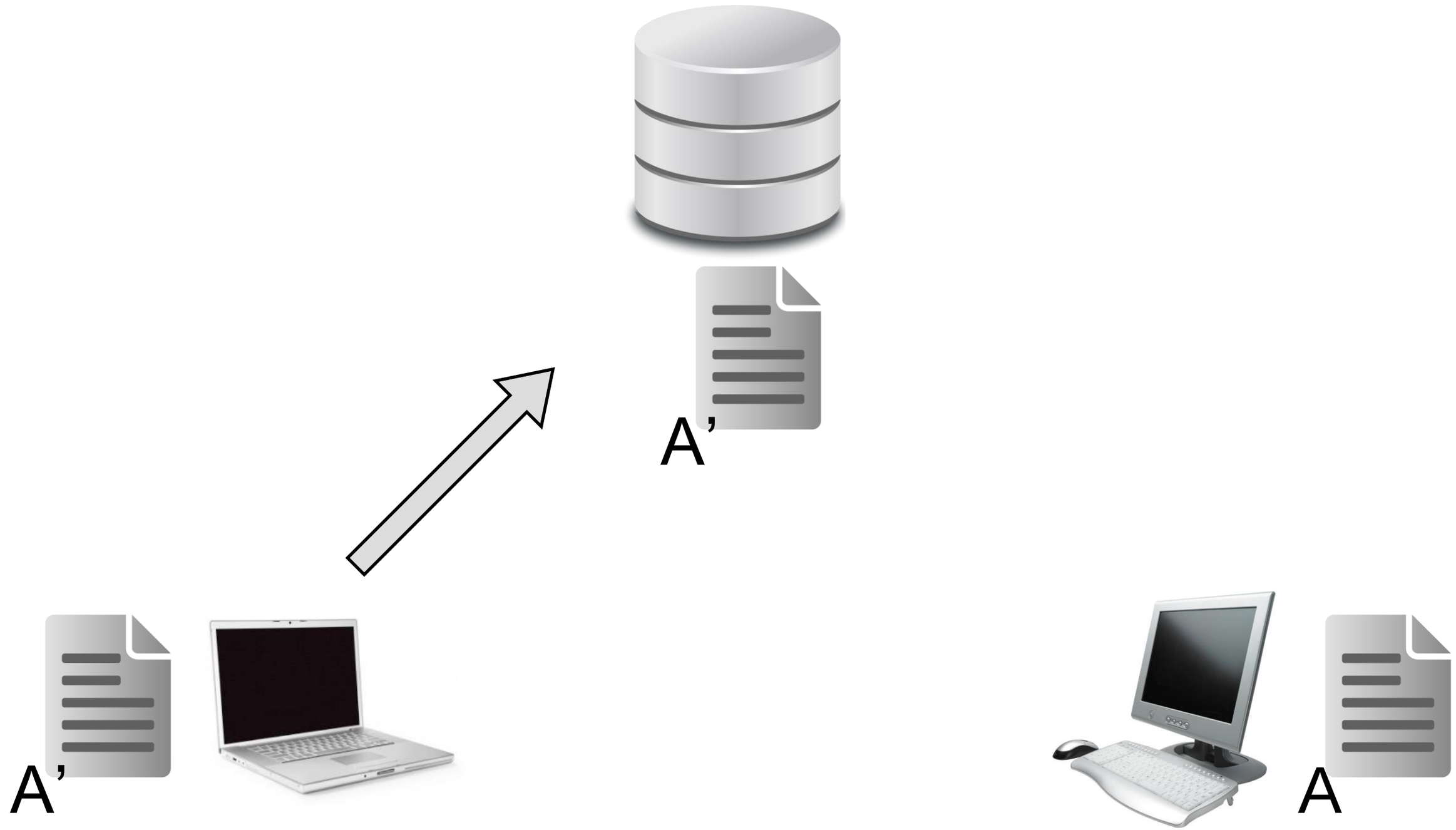
A

3. Modifie A en A’


A
A’
A
4. Verrouille A



5. Sauve A’



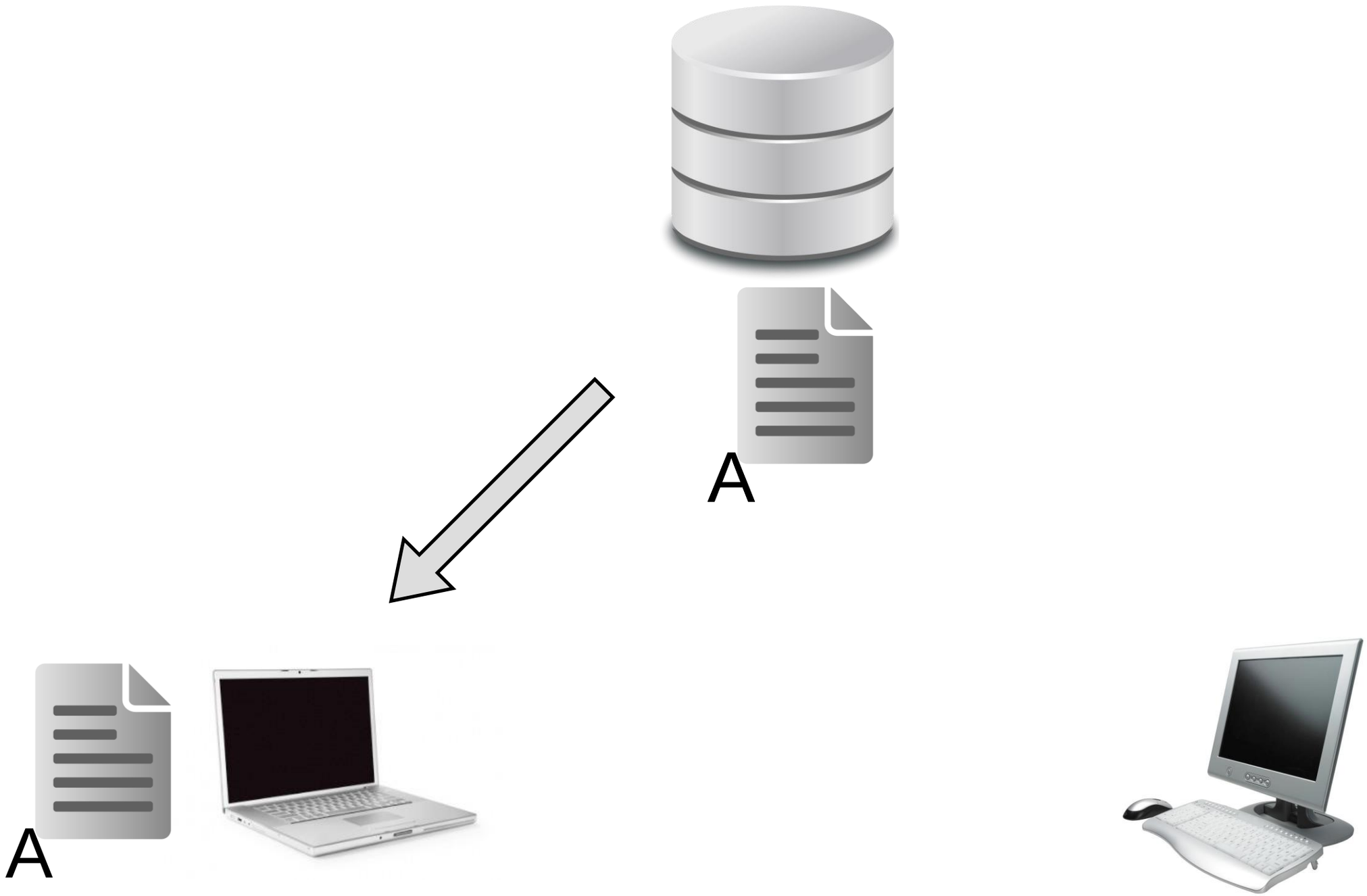
6. Déverouille A’

Verrou: Pas efficace pour travailler en groupe sur les même fichiers

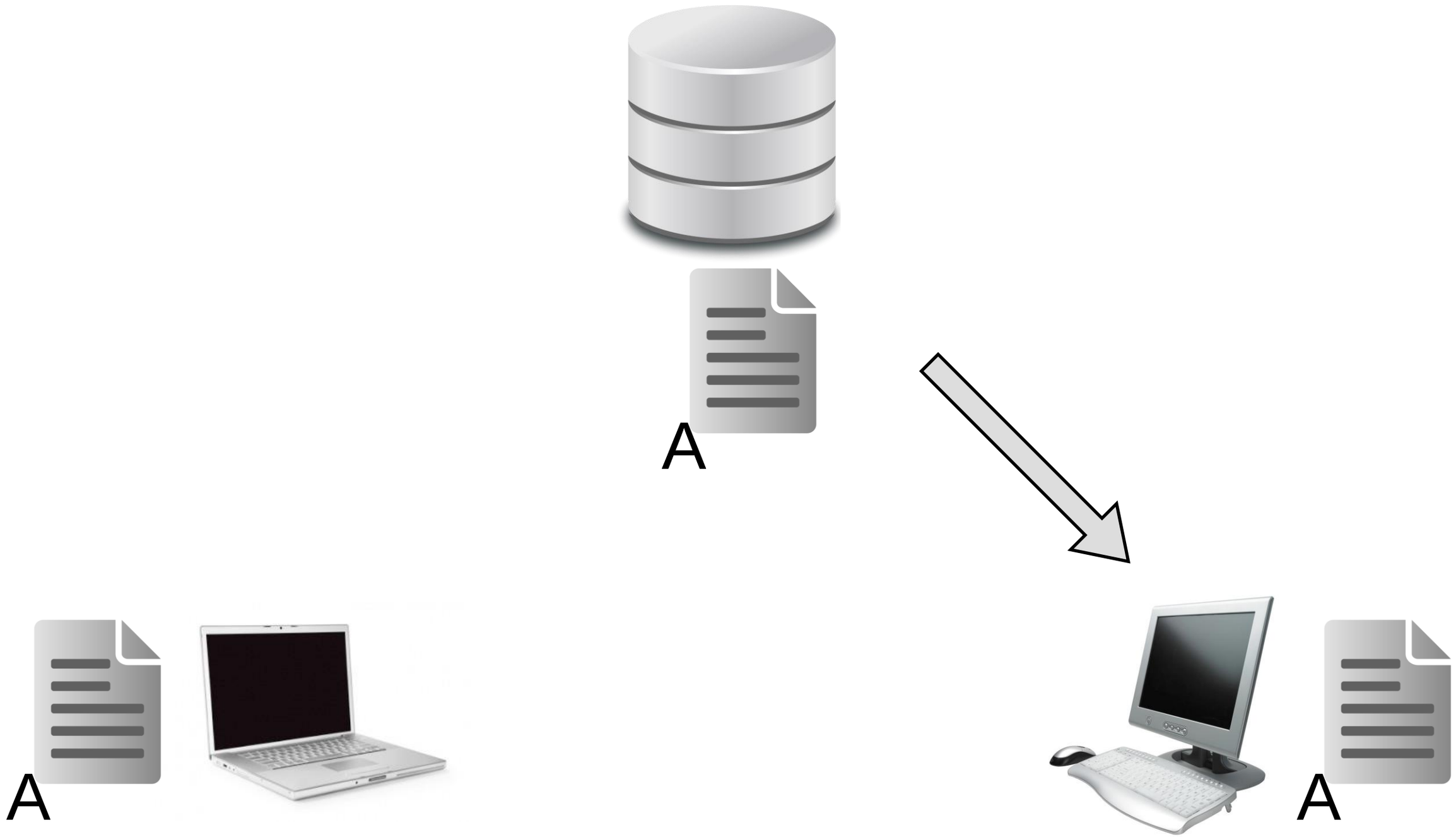
Solution 2:

Copier - Modifier - Fusionner

A



1. Lit A


A
A
A
2. Lit A

A

A’

A

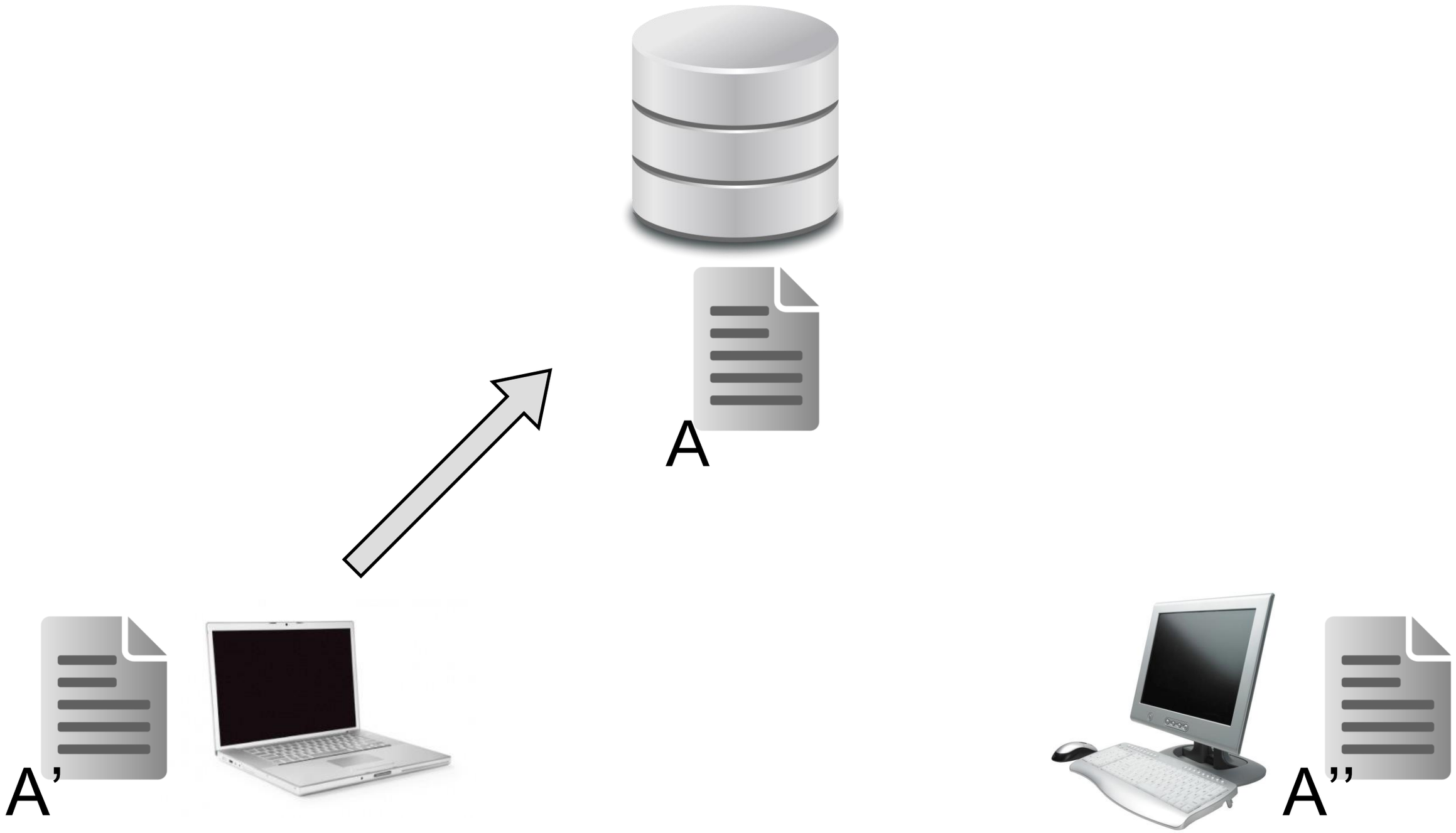
3. Modifie A en A’

A

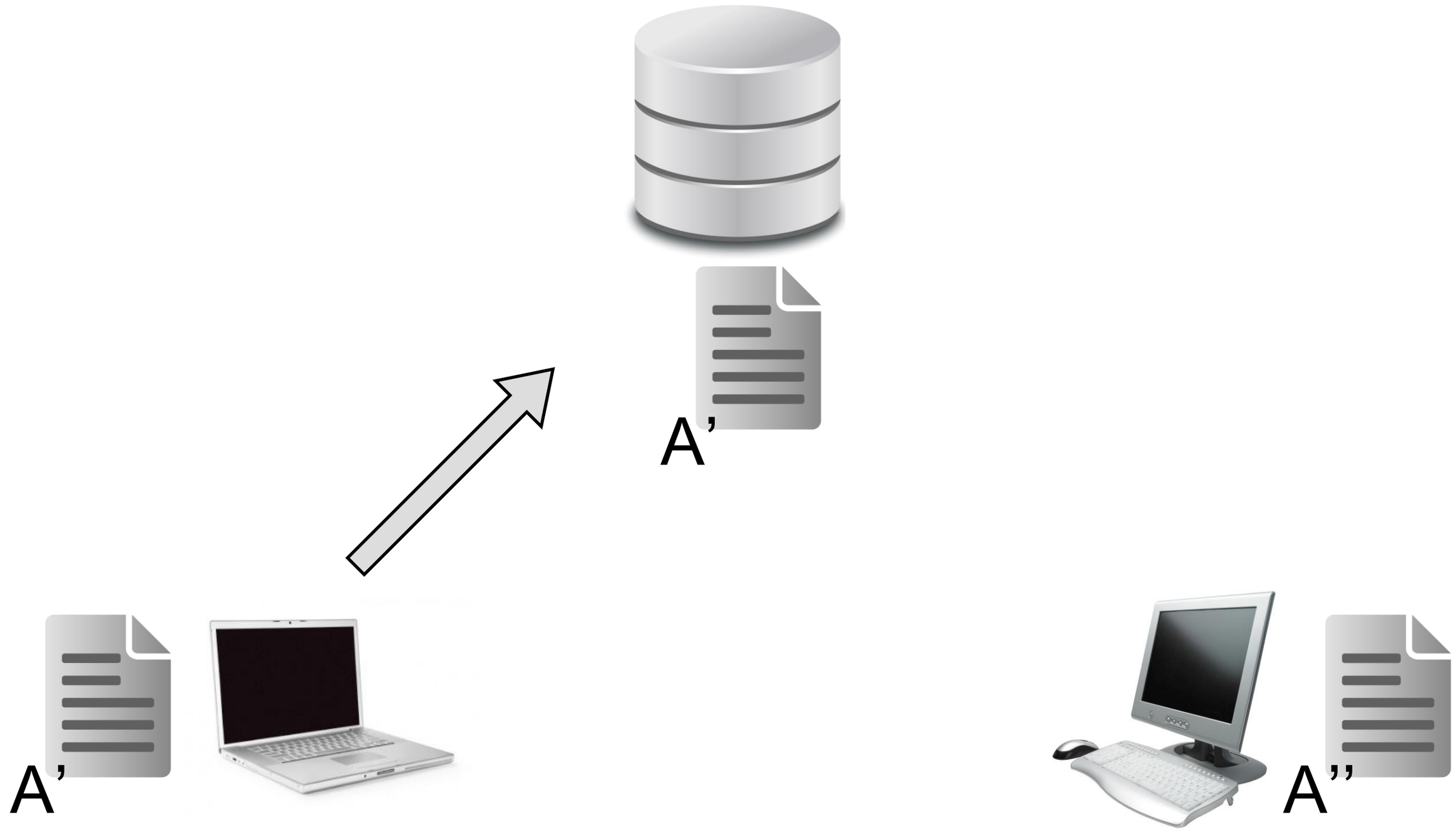
A’

A’’

4. Modifie A en A’’



5. Sauve A’ à la place de A


A’
A’
A’’

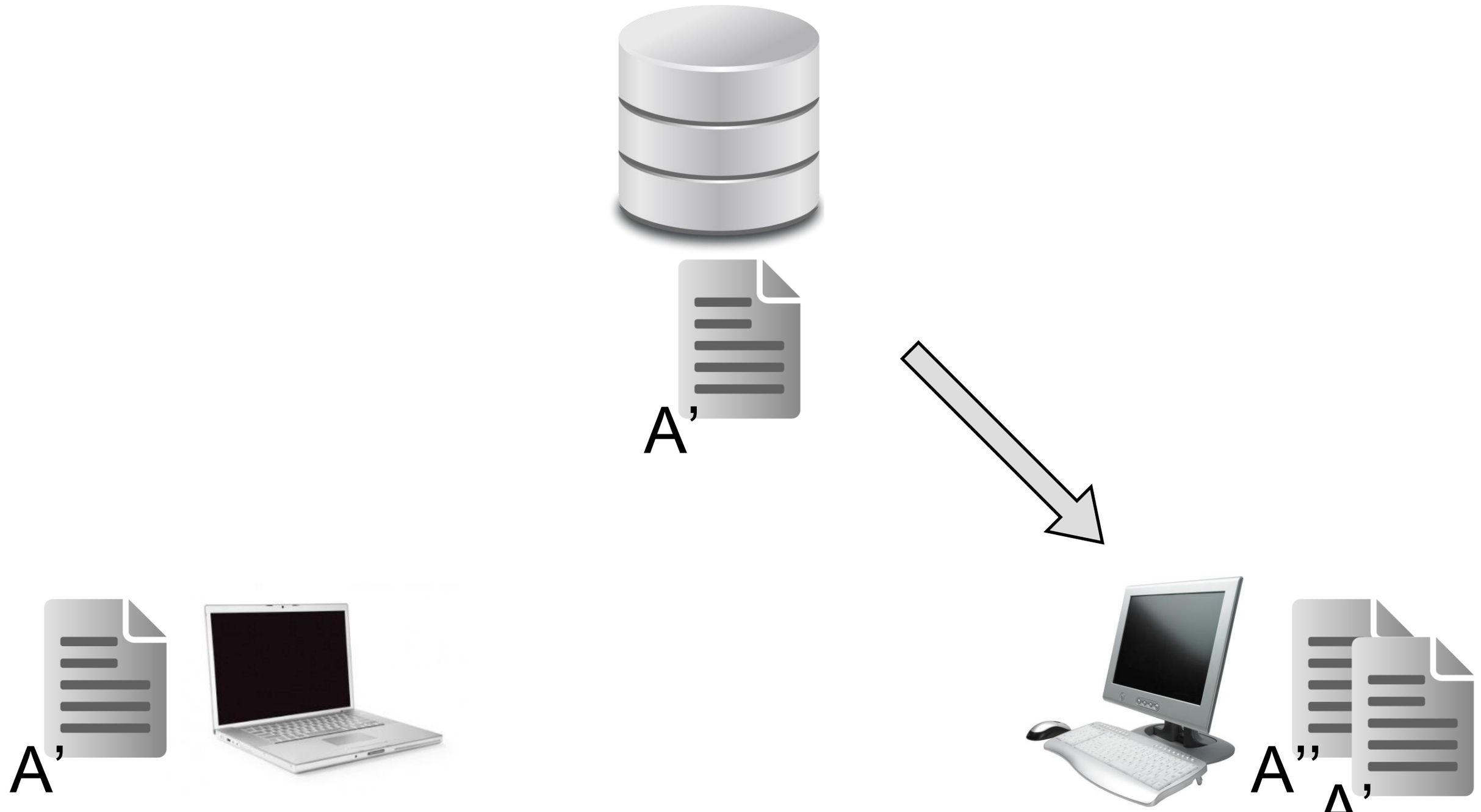
A’

A n’est plus la version de base

A’

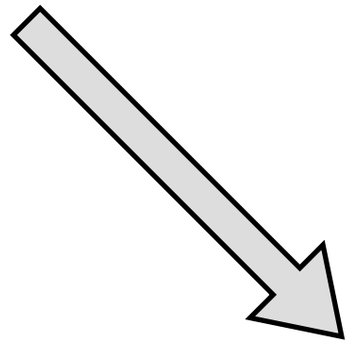
A’’

6. Sauve A’’ à la place de A



6. Lit la dernière version

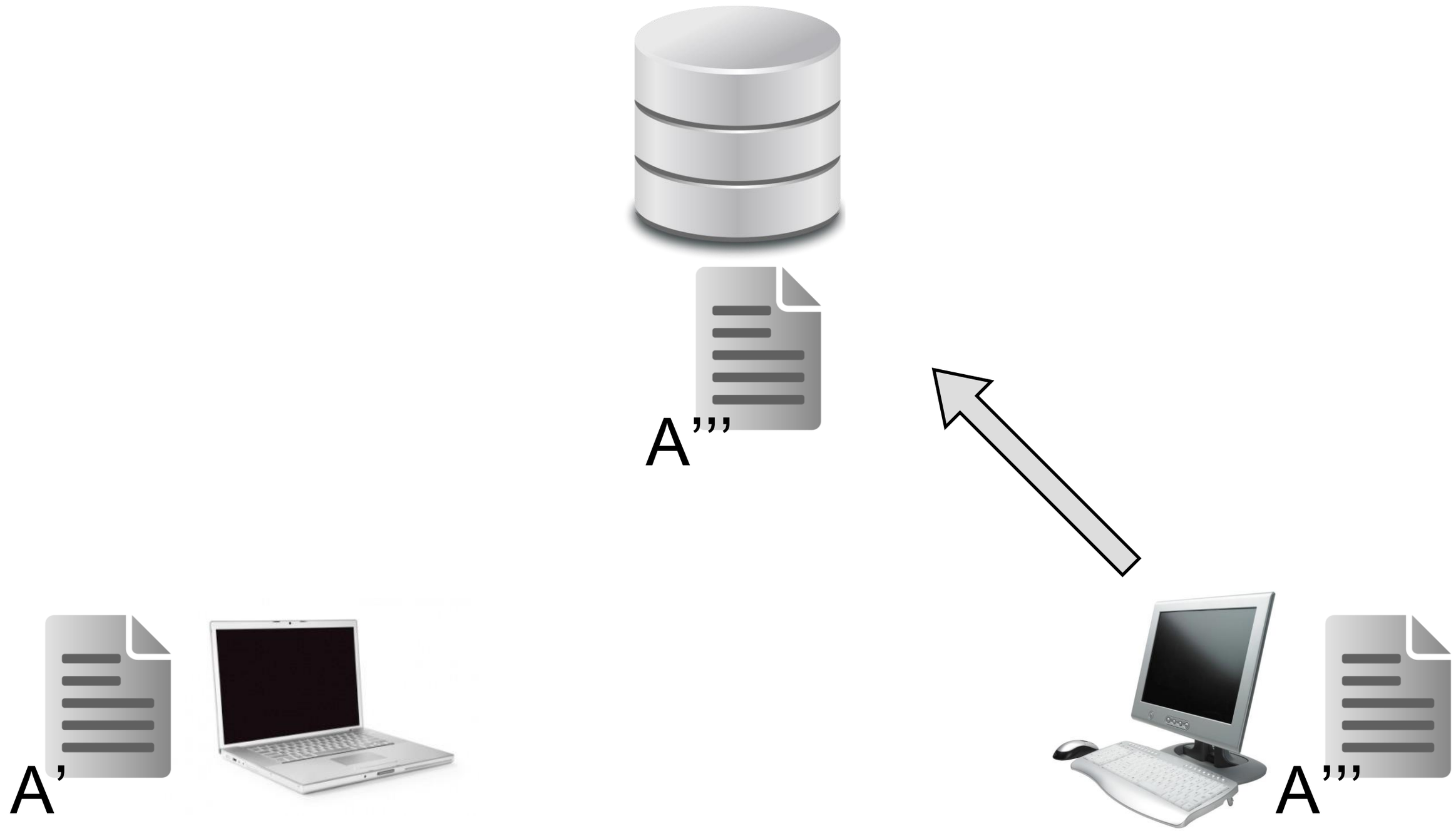
A’



A’

A’’’

7. Fusionne A’ et A’’



6. Sauve A’’’ à la place de A’

La solution utilisant copie-modifier-fusionner ne fonctionne efficacement que si les fichiers sont modularisés (méthode, classes) sinon il faut utiliser la technique du verrou

Git

- Transactions atomiques
- Chaque transaction crée une nouvelle version (UUID)
- Un numéro de révision est valable pour l’abre complet
- Open source
- Administration facile
- Indépendant de la plateforme

Git
(vocabulaire)

- Le “remote repository” est l’emplacement central où sont stockés les sources et leurs historiques
- “local repository” est la copie locale sur laquelle le développeur travaille
- “clone” ramène le projet pour la première fois. Le résultat est la copie de travail

Git
(vocabulaire)

- “commit” sauvegarde un ensemble de changements localement
- “push” pousse les derniers changements commités vers le « remote repository »
- “pull” ramène la dernière version du « remote repository » et la fusionne.
- Un “tag” copy l’état du projet à un moment donné sous un nom particulier

https://help.github.com/en/articles/github-glossary#upstream

Bonnes pratiques

- Dans eclipse, toujours faire une synchronisation avant de “committer” cela permet d’anticiper et de se tenir à jour de modifications
- “Committer” plusieurs fois par jour (2-3x)
- “Committer” des changements cohérents
- Pusher de façon régulière

Bonnes pratiques

- Avant de committer, toujours executer les tests unitaires
- Pas de commit si:
  - le code ne compile pas
  - les tests unitaires ne sont pas verts