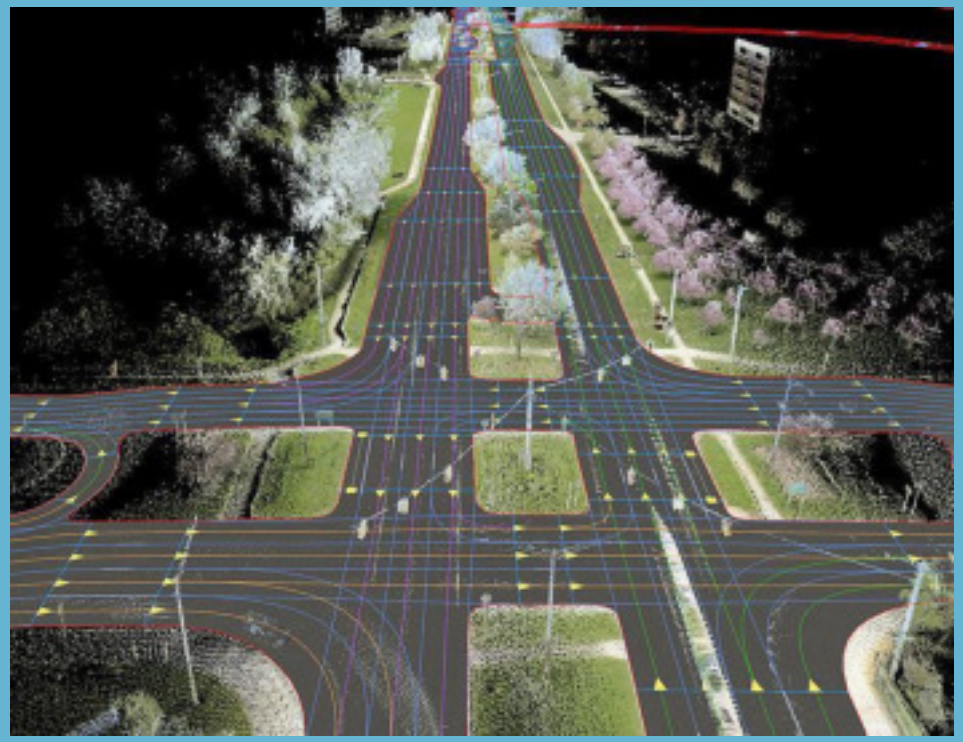
git as a NoSQL database

Kenneth Truyers
.NET Developer
Microsoft MVP / Pluralsight author
@kennethtruyers
www.kenneth-truyers.net

appyparking™

appyparking™



Mechanics of
git as a database

Why it’s a
fantastic idea

Why it’s a
terrible idea

Database

“a structured set of data held in a computer, especially one that is accessible in various ways”

NoSQL

“provides a mechanism for storage and retrieval of data in means other than the tabular relations used in relational databases”

Schema-less

Non relational

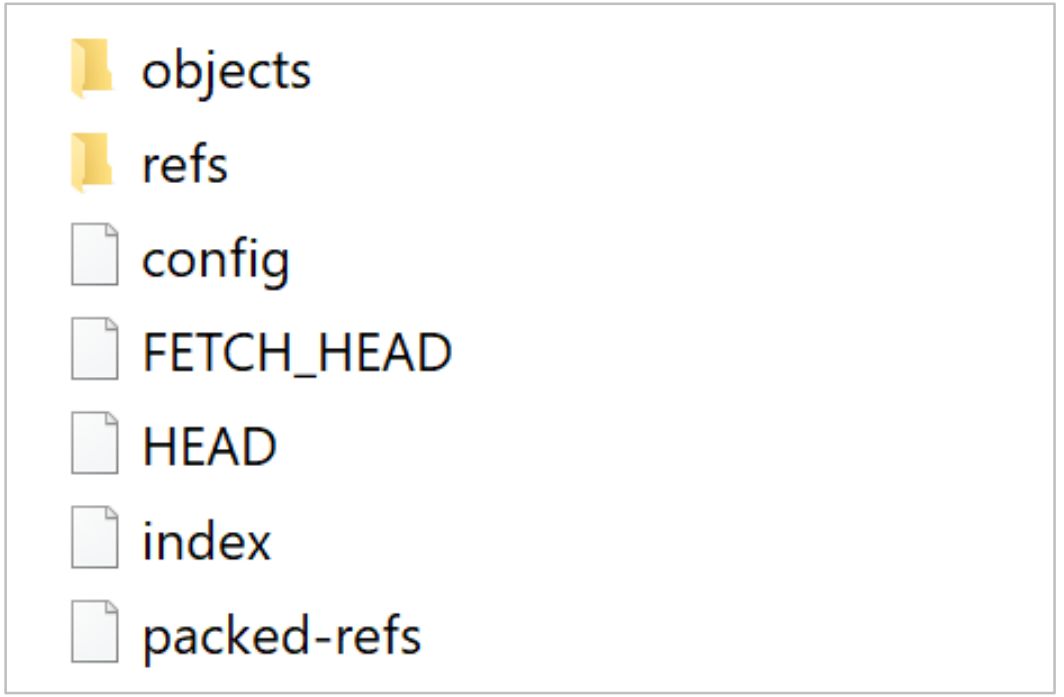
The naïve way

```
> git init myDatabase
Initialized empty Git repository in D:/myDatabase/.git/

> cd myDatabase

> echo {"id": 1, "name": "kenneth"} > 1.json

> git add 1.json

> git commit -m "Added person"
[master (root-commit) 6c6b907] Added person
 1 file changed, 1 insertion(+)
 create mode 100644 1.json

> git show master:1.json
{"id": 1, "name": "kenneth"}
```

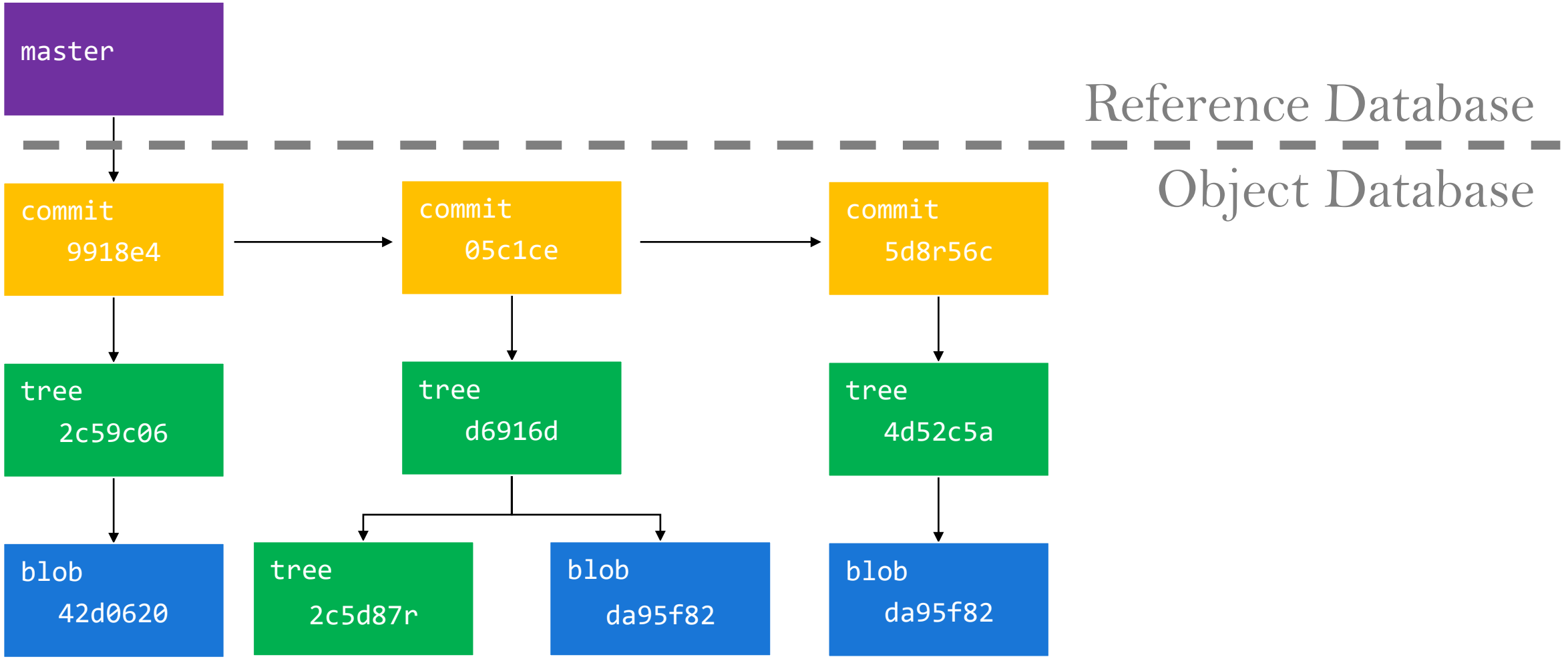
Git repositories

Client

Server (Bare repository)



Git data model



Git commands

Porcelain

add	archive	bisect	**branch**
checkout	**cherry-pick**	**clean**	**clone**
commit	diff	**fetch**	gc
init	**log**	**merge**	mv
pull	**push**	**rebase**	**reset**
revert	rm	show	shortlog
stash	status	submodule	tag

Git commands

Plumbing

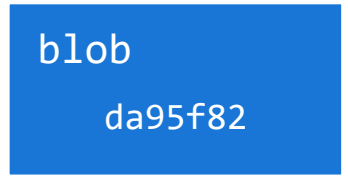
apply	commit-tree	hash-object	merge-file
mktag	mktree	for-each-ref	read-tree
update-ref	write-tree	cat-file	diff-files
diff-index	diff-tree	merge-base	rev-list
show-index	show-ref	unpack-tree	send-pack
http-fetch	http-push	shell	check-attr
mailinfo	mailsplit	sh-setup	stripspace

Plumbing

We need to get our hands dirty

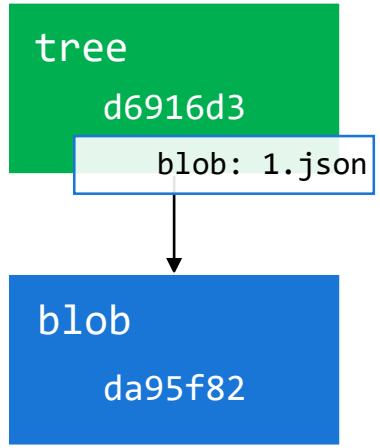
Blobs

```
> echo {"id":1, "name": "kenneth"} | git hash-object –w
--stdin
da95f8264a0ffe3df10e94eed6371ea83aee9a4d

> git cat-file -p da95f
{"id": 1, "name": "kenneth"}
```



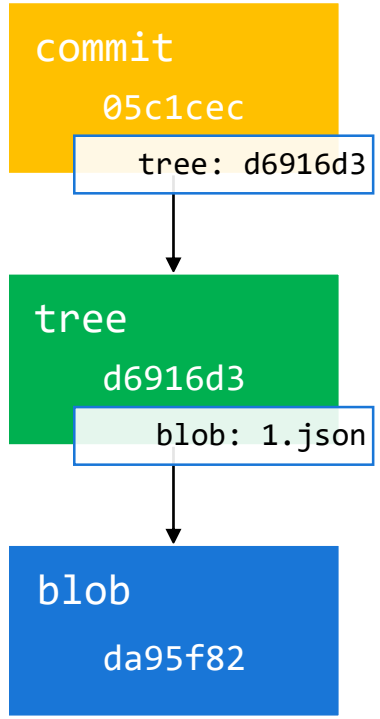
Trees

```
> git update-index --add --cacheinfo 100644
da95f8264a0ffe3df10e94eed6371ea83aee9a4d 1.json


> git write-tree
d6916d3e27baa9ef2742c2ba09696f22e41011a1

> git cat-file -p d6916d
100644 blob da95f8264a0ffe3df10e94eed6371ea83aee9a4d
1.json
```



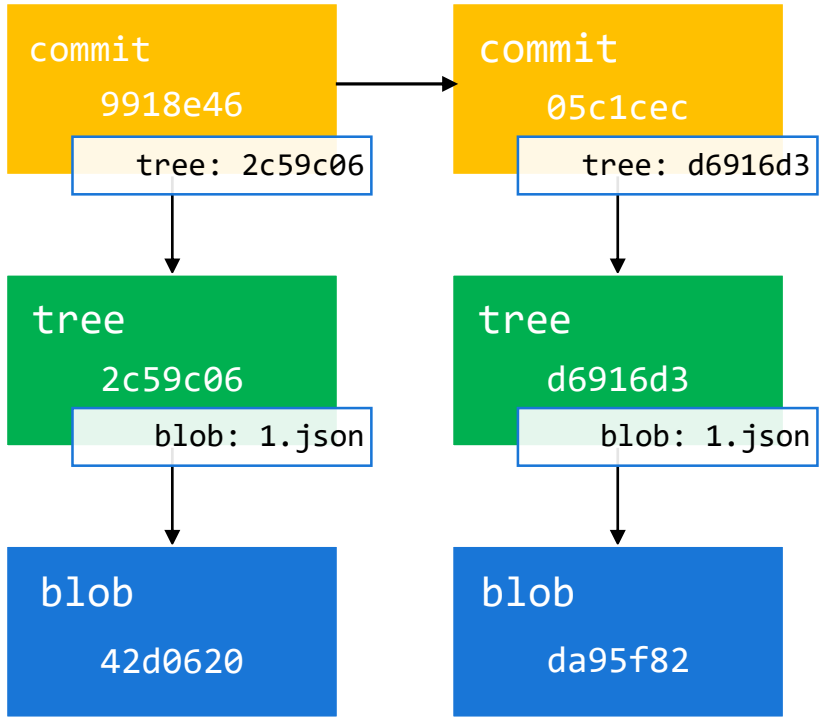
Commits

```
> echo "commit kenneth" | git commit-tree d6916d
05c1cec5685bbb84e806886dba0de5e2f120ab2a

> git log --stat 05c1ce
commit 05c1cec5685bbb84e806886dba0de5e2f120ab2a
Author: Kenneth Truyers <truyers.kenneth@gmail.com>
Date:   Sat Apr 29 10:15:23 2017 +0200

    "commit kenneth"

 1.json       | 1 +
 1 files changed, 1 insertions(+)

> git show 05c1cec:1.json
{"id": 1, "name": "kenneth"}
```



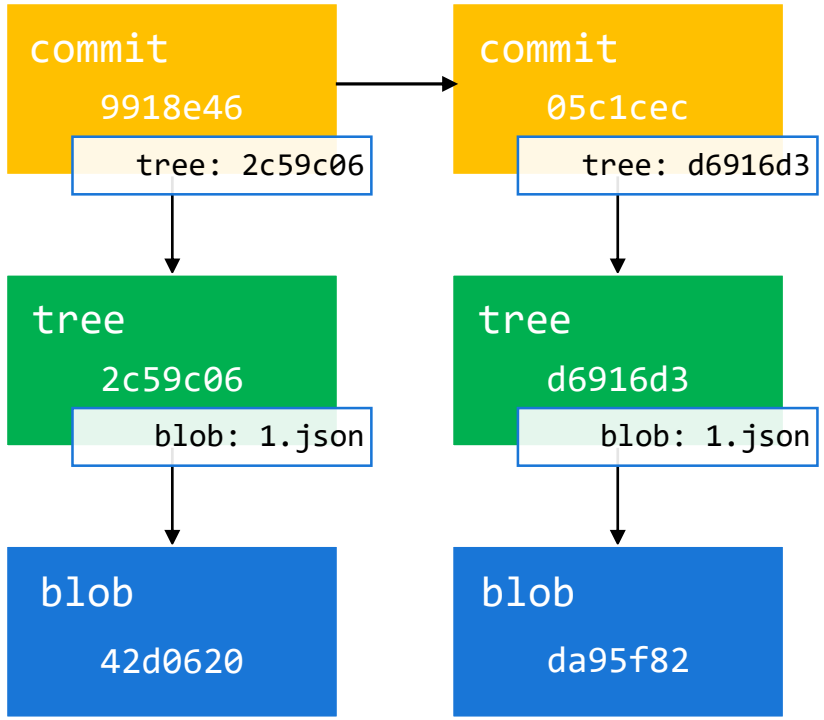
Updating the file

```
> echo {"id": 1, "name": "updated name"} | git hash-
object -w --stdin
42d0d209ecf70a96666f5a4c8ed97f3fd2b75dda

> git update-index --add --cacheinfo 100644
42d0d209ecf70a96666f5a4c8ed97f3fd2b75dda 1.json

> git write-tree
2c59068b29c38db26eda42def74b7142de392212

> echo "Commit Kenneth v2" | git commit-tree 2c59068 -p
05c1cec
9918e46dfc4241f0782265285970a7c16bf499e4
```



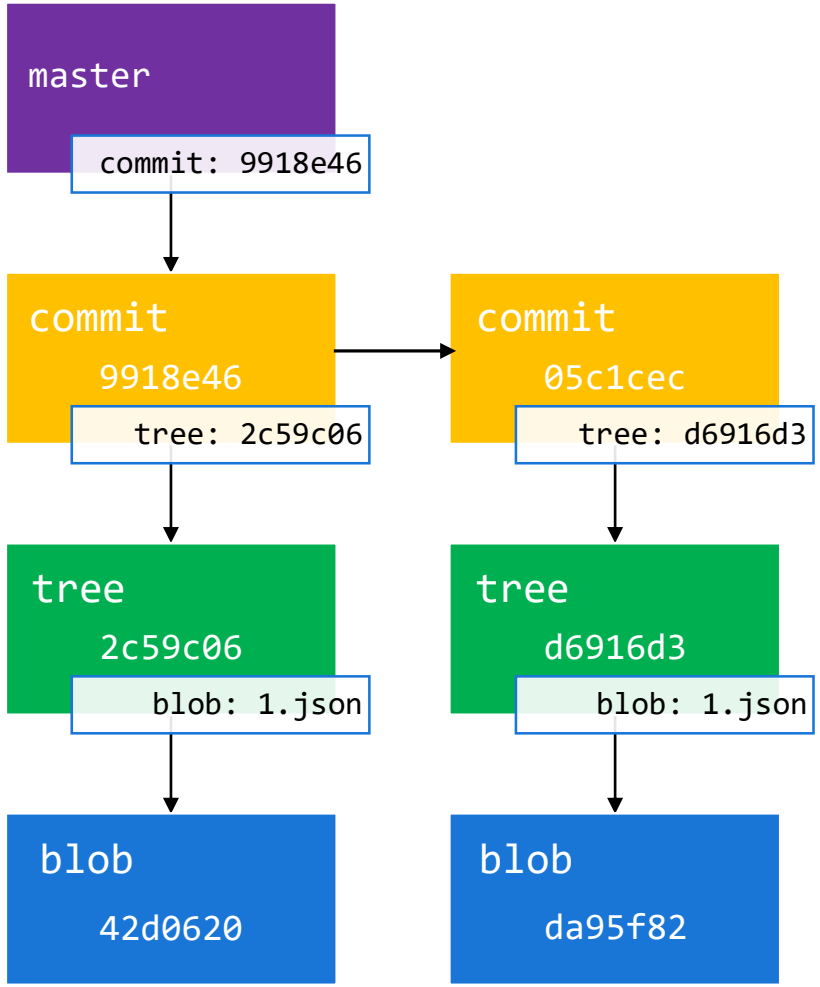
Updating the file

```
> git log --stat 9918e46
9918e46dfc4241f0782265285970a7c16bf4 "commit Kenneth v2"
 1.json     | 1 +
 1 file changed, 1 insertions(+)
05c1cec5685bbb84e806886dba0de5e2f120 "commit kenneth"
 1.json | 1 +
 1 file changed, 1 insertion(+)

> git show 9918e4:1.json
{"id": 1, "name": "updated name"}

> git show 05c1cec:1.json
{"id": 1, "name": "kenneth"}
```



Refs

```
> git update-ref refs/heads/master 9918e4

> git show master:1.json
{"id": 1, "name": "updated name"}
```



Cool, but… I'm a programmer

portable, pure C implementation of the core git methods

https://libgit2.github.com/

brings might and speed of libgit2 to the managed world of **.NET** and **Mono**

https://github.com/libgit2/libgit2sharp

HOW

Building a git db in C#

```
public class GitDb
{
    public string Save<T>(string branch, string key, T value)
    {

    }

    public T Get<T>(string branch, string key)
    {

    }
}
```

HOW

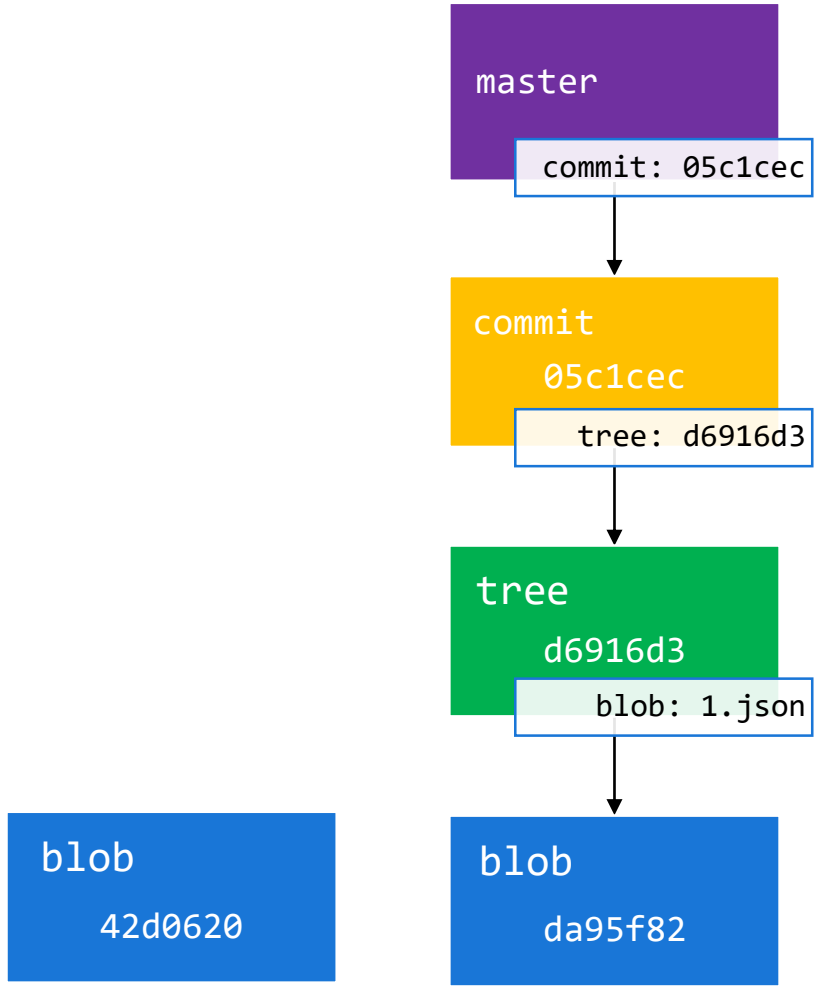
Building a git db in C#

```
> git init --bare
```

```
public class GitDb
{
    readonly Repository _repo;
    public GitDb(string path)
    {
        Repository.Init(path, isBare: true);
        _repo = new Repository(path);
    }

    public string Save<T>(string branch, string key, T value)
    {

    }

    public T Get<T>(string branch, string key)
    {

    }
}
```

Writing a blob

```
public string Save<T>(string branch, string key, T value)
{
    string val = JsonConvert.SerializeObject(value);
    byte[] bytes = Encoding.UTF8.GetBytes(val);
    Stream stream = new MemoryStream(bytes);
    Blob blob = _repo.ObjectDatabase.CreateBlob(stream);









}
```

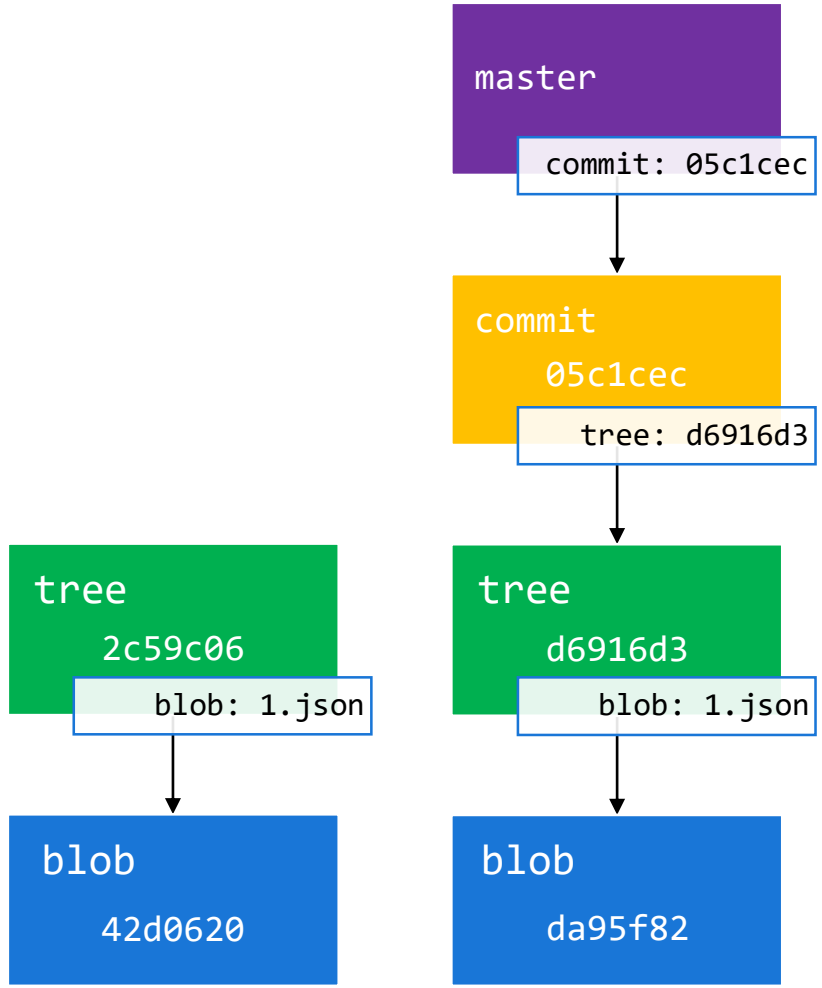


Writing a tree

```
public string Save<T>(string branch, string key, T value)
{
    string val = JsonConvert.SerializeObject(value);
    byte[] bytes = Encoding.UTF8.GetBytes(val);
    Stream stream = new MemoryStream(bytes);
    Blob blob = _repo.ObjectDatabase.CreateBlob(stream);
```

```
    Commit currentCommit = _repo.Branches[branch].Tip;
    TreeDefinition treeDef = TreeDefinition.From(currentCommit);
    treeDef.Add(key, blob, Mode.NonExecutableFile);
    Tree tree = _repo.ObjectDatabase.CreateTree(treeDef);
```

```
}
```

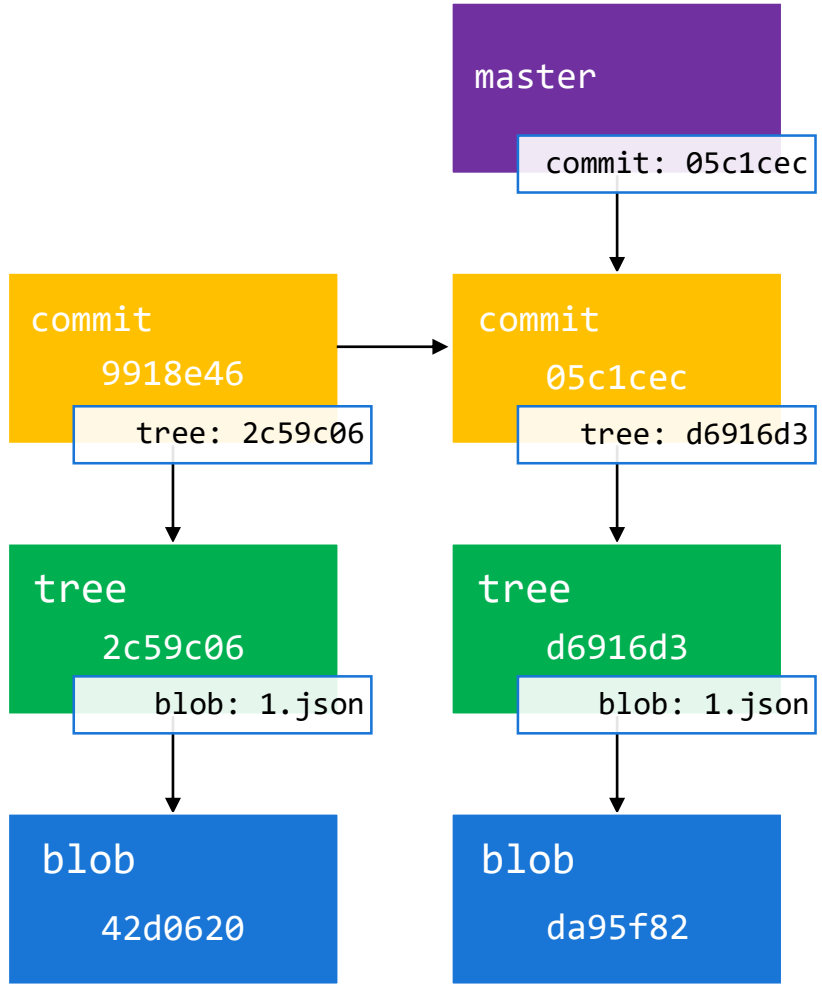


Committing the tree

```
public string Save<T>(string branch, string key, T value)
{
    string val = JsonConvert.SerializeObject(value);
    byte[] bytes = Encoding.UTF8.GetBytes(val);
    Stream stream = new MemoryStream(bytes);
    Blob blob = _repo.ObjectDatabase.CreateBlob(stream);

    Commit currentCommit = _repo.Branches[branch].Tip;
    TreeDefinition treeDef = TreeDefinition.From(currentCommit);
    treeDef.Add(key, blob, Mode.NonExecutableFile);
    var tree = _repo.ObjectDatabase.CreateTree(treeDef);
```

```
    var now = DateTimeOffset.Now;
    Commit commit = _repo.ObjectDatabase.CreateCommit(
        author: new Signature("author", "email", now),
        message: "commit message",
        tree: tree,
        parents: new List<Commit> {currentCommit});
```

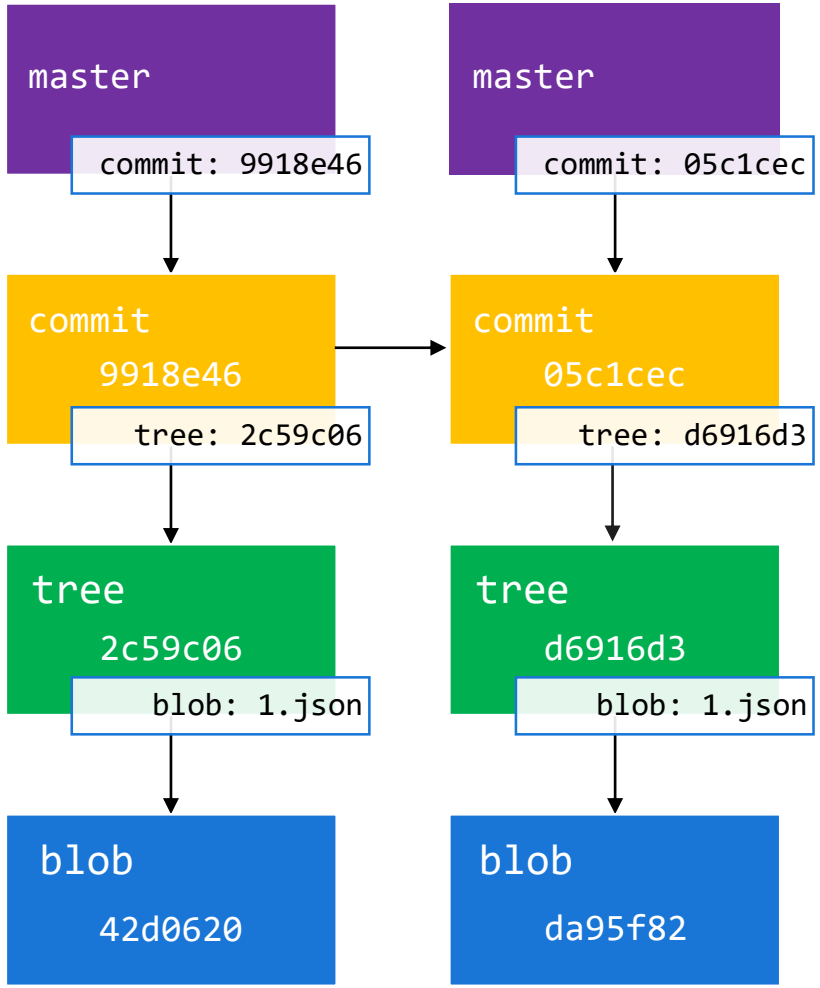
```
}
```



Updating the branch

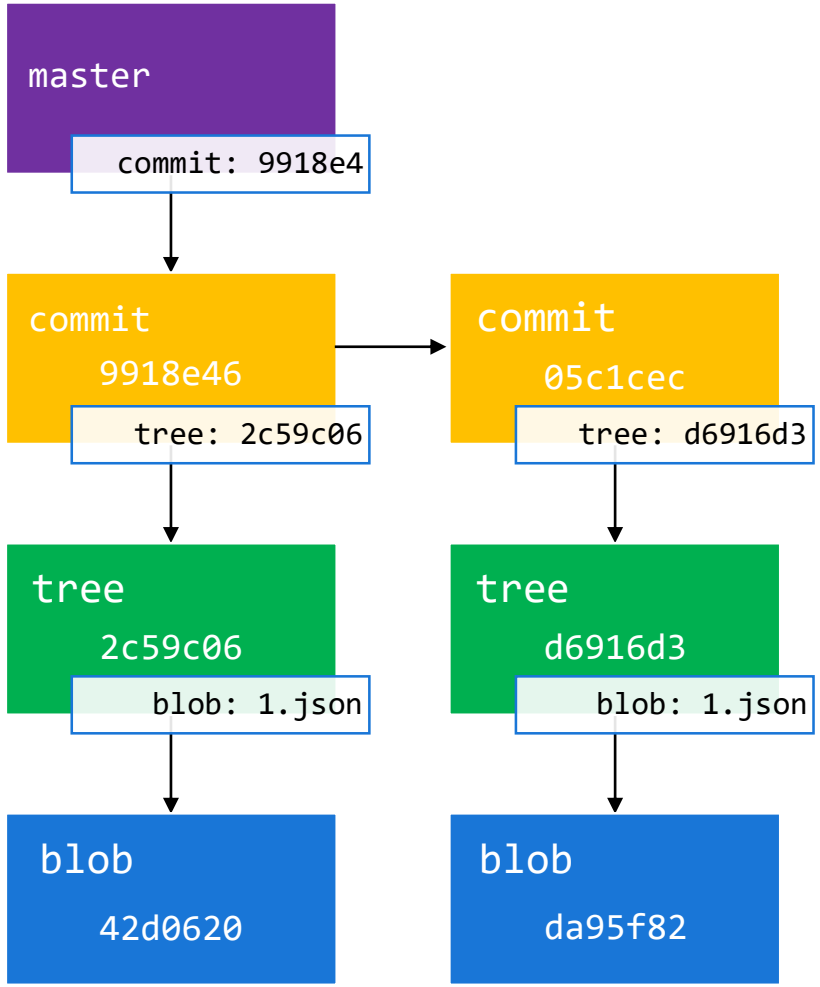
```
public string Save<T>(string branch, string key, T value)
{
    string val = JsonConvert.SerializeObject(value);
    byte[] bytes = Encoding.UTF8.GetBytes(val);
    Stream stream = new MemoryStream(bytes);
    Blob blob = _repo.ObjectDatabase.CreateBlob(stream);

    Commit currentCommit = _repo.Branches[branch].Tip;
    TreeDefinition treeDef = TreeDefinition.From(currentCommit);
    treeDef.Add(key, blob, Mode.NonExecutableFile);
    var tree = _repo.ObjectDatabase.CreateTree(treeDef);

    var now = DateTimeOffset.Now;
    Commit commit = _repo.ObjectDatabase.CreateCommit(
        author: new Signature("author", "email", now),
        message: "commit message",
        tree: tree,
        parents: new List<Commit> {currentCommit});
```

```
    String refName = _repo.Branches
                          .Single(b => b.FriendlyName == branch)
                          .CanonicalName;
    _repo.Refs.UpdateTarget(_repo.Refs[refName], commit.Id);

    return commit.Sha;
}
```

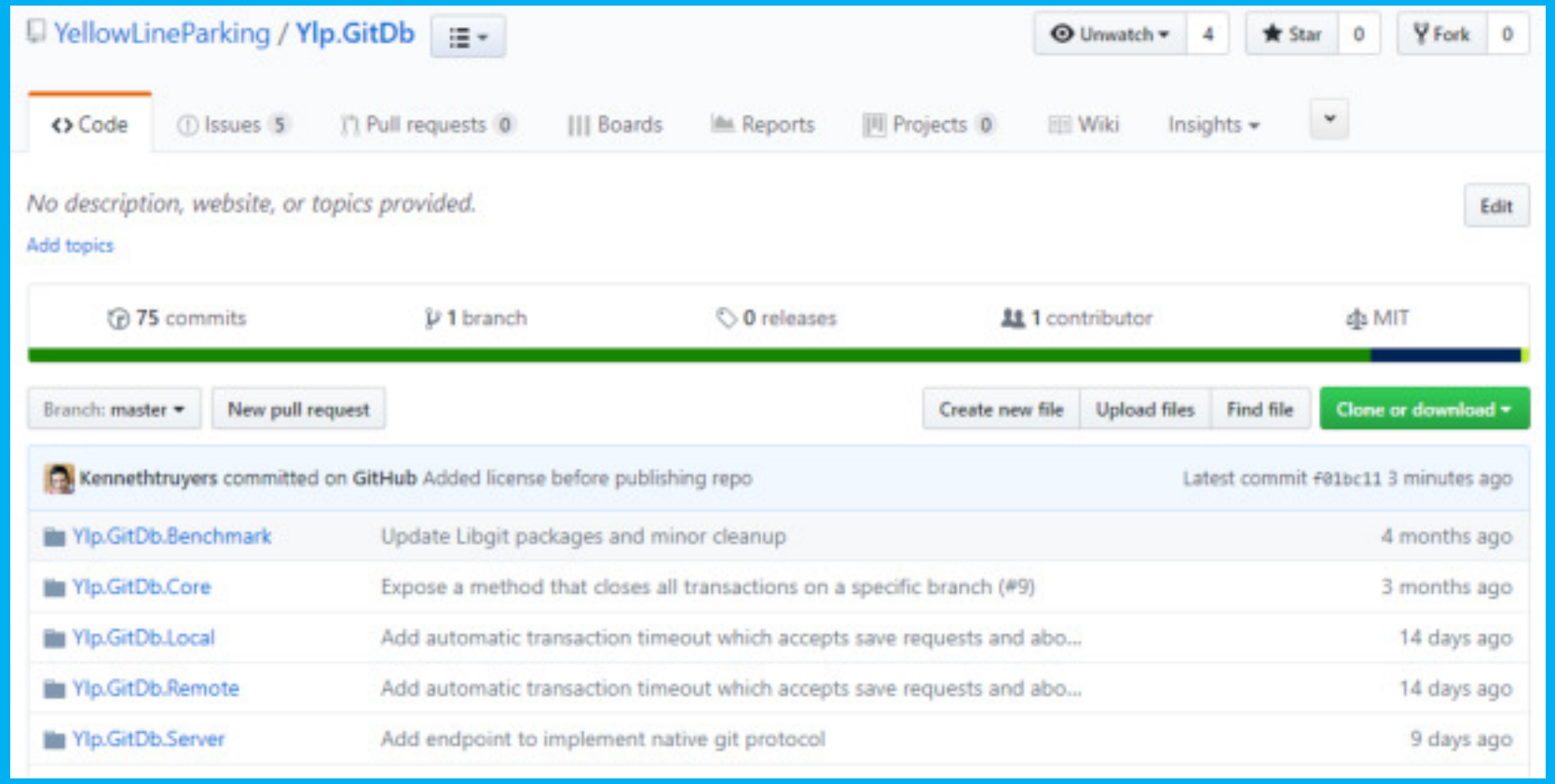


Reading

```
public T Get<T>(string branch, string key)
{
    Branch branchRef = _repo.Branches[branch];
    Commit commit = branchRef.Tip;

    TreeEntry entry = commit[key];
    GitObject gitObj = entry.Target;
    Blob blob = gitObj as Blob;
    string content = blob.GetContentText();
    return JsonConvert.DeserializeObject<T>(content);
}
```



Cool, but… where's my ORM?



github.com/YellowLineParking/Appy.GitDb

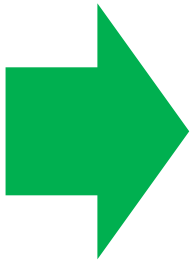
goo.gl/vQpxra

Why it’s a
fantastic idea

WHY

Schema-less

```
public class User
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

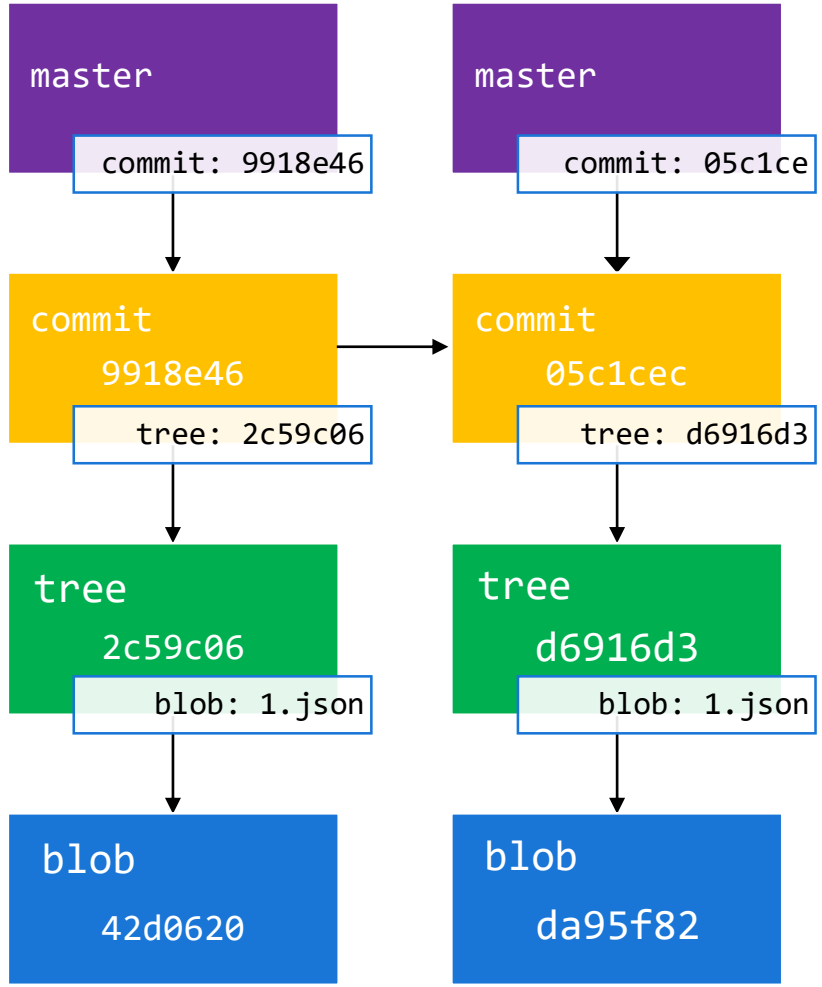


```
public class User
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Street { get; set; }
    public string Country { get; set; }
}
```

WHY

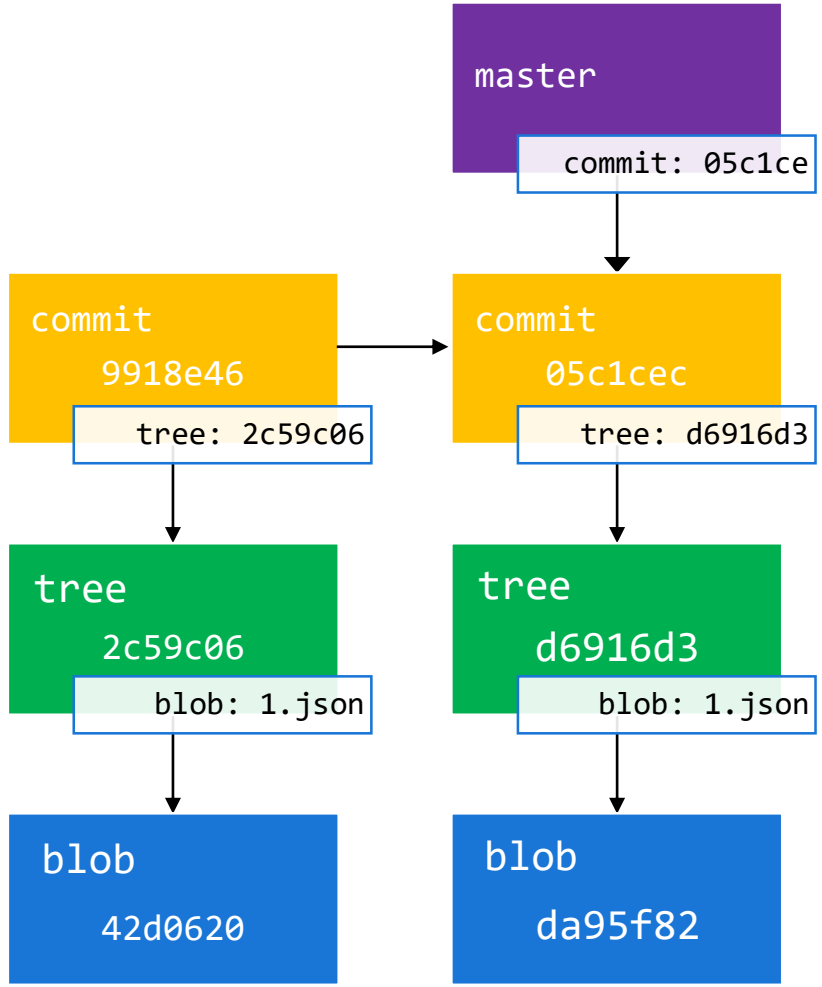
Versioning and roll-back

```
> git show master:1.json
{"id": 1, "name": "updated name"}

> git update-ref refs/heads/master 05c1ce

> git show master:1.json
{"id": 1, "name": "kenneth"}'
```



Diff

```
> git diff 9918e4 05c1ce -- 1.json

diff --git a/1.json b/1.json
--- a/1.json
+++ b/1.json
@@ -1 +1 @@
-{"id": 1, "name": "updated name"}
+{"id": 1, "name": "kenneth"}
```



WHY

Backup & Replication

```
> git remote add backup https://github.com/user/repo.git

> git push backup




> type .git/hooks/post-commit
#!/bin/sh
git push backup
```

WHY

Transactions

Short lived

```
> echo "file1" | git hash-object -w --stdin
42d0d20...
> echo "file2" | git hash-object -w --stdin
5d8f5r0...

> git update-index --add --cacheinfo 100644 42d0d20... 1.json
> git update-index --add --cacheinfo 100644 5d8f5r0... 2.json

> git write-tree
9c38db2...

> echo "commit transaction" | git commit-tree 9c38db2 -p
05c1cec
9918e46...
```

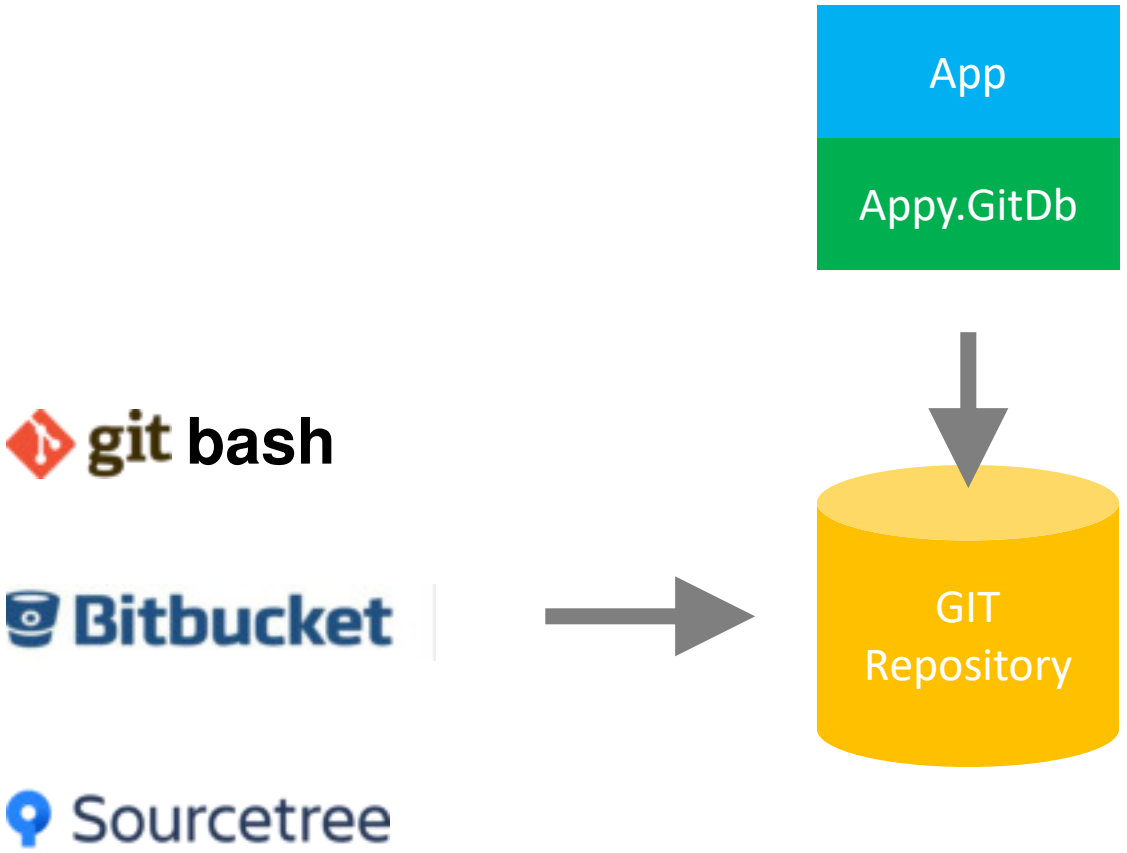
WHY

Transactions

Long lived

```
> git checkout -b transaction

...
...
...

> git checkout master

> git checkout merge transaction
```

Tooling


App
Appy.GitDb
git bash
Bitbucket
Sourcetree
GIT
Repository

Why it’s a
terrible idea

Queries

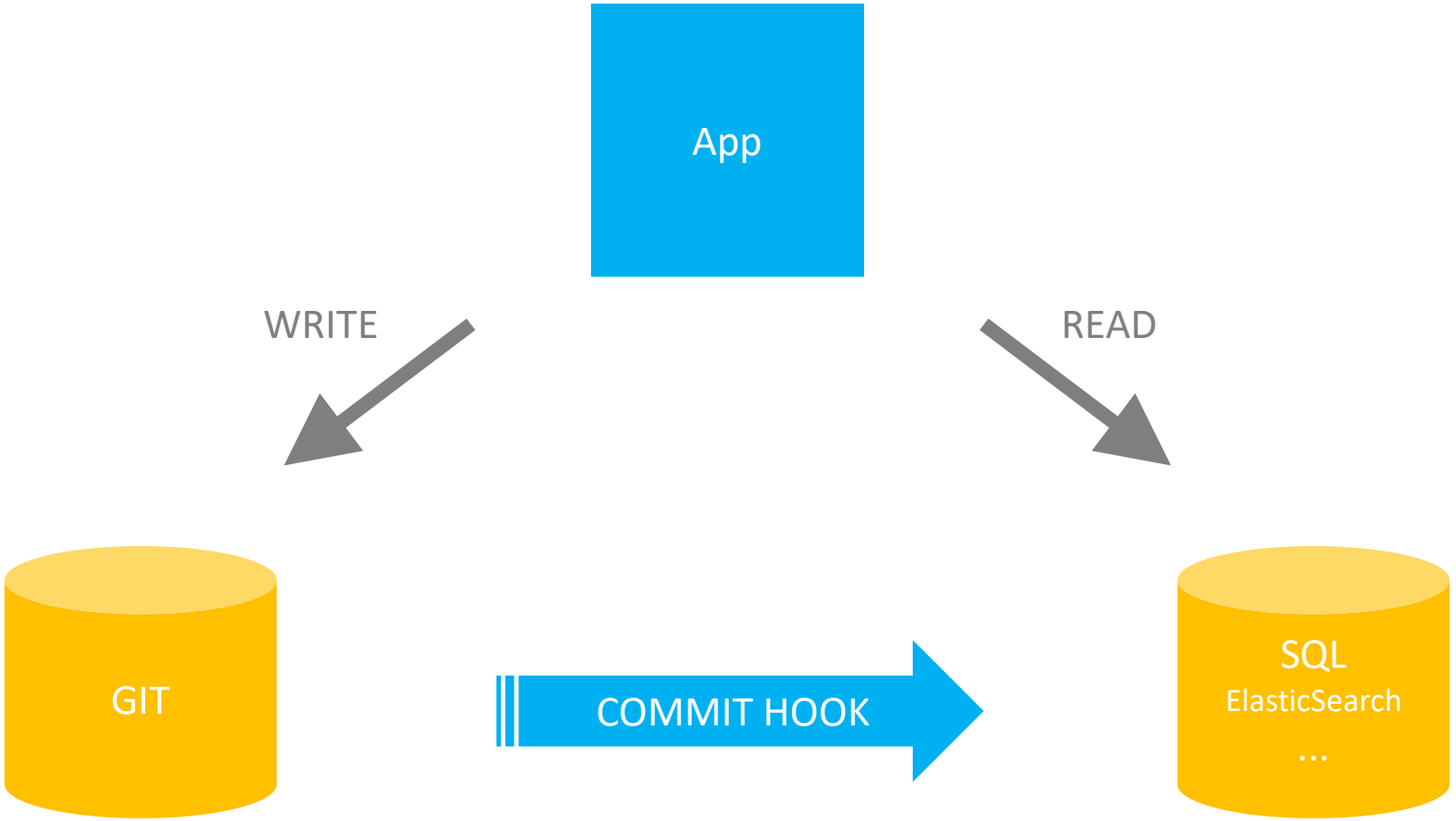
WHY NOT

KEY

KEY PREFIX (sort of*)

Solution

WHY NOT



Concurrency

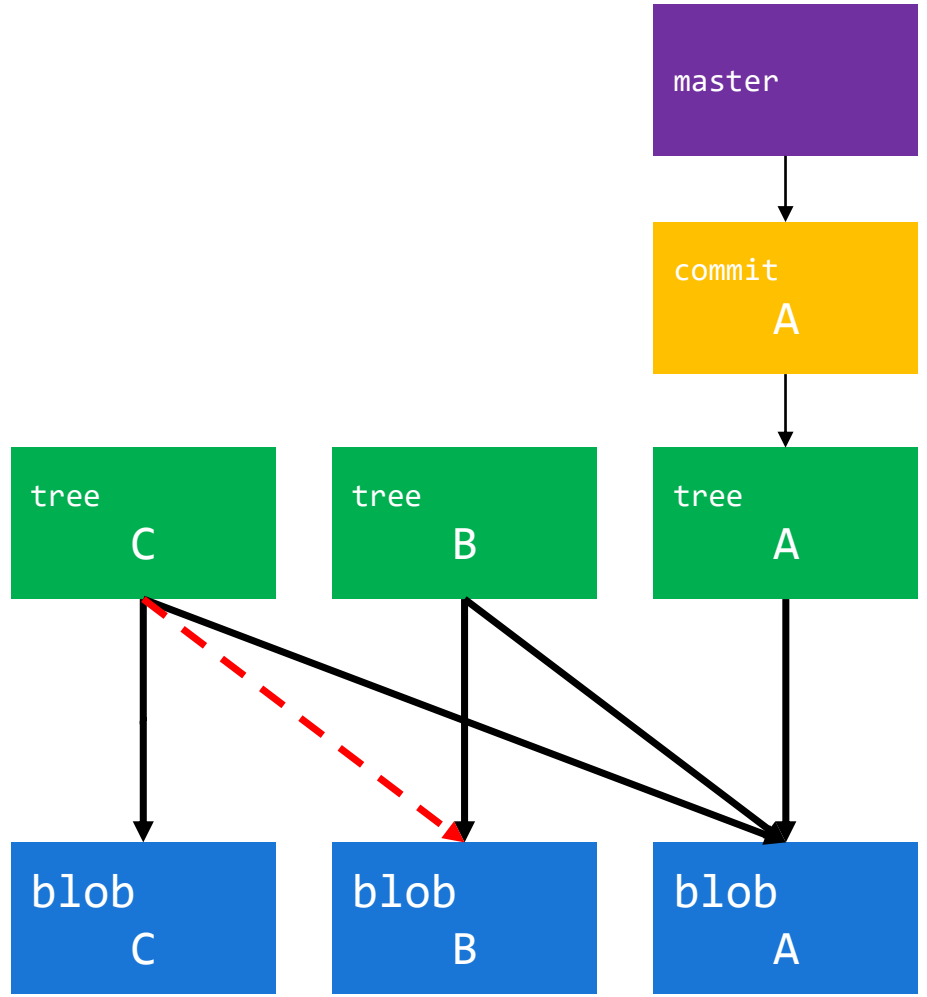
WHY NOT

```
public string Save<T>(string branch, string key, T value)
{
    Blob blob = _repo.ObjectDatabase.CreateBlob(...);


    TreeDefinition treeDef = TreeDefinition.From(...);

    treeDef.Add(key, blob, Mode.NonExecutableFile);
    Tree tree = _repo.ObjectDatabase.CreateTree(treeDef);

    Commit commit = _repo.ObjectDatabase.CreateCommit(...);

    _repo.Refs.UpdateTarget(_repo.Refs[refName], commit.Id);
}
```

Concurrency

WHY NOT

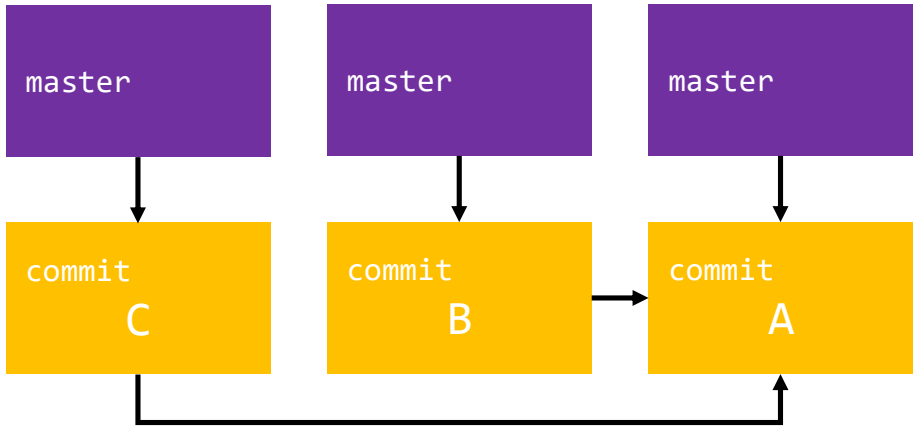
```
public string Save<T>(string branch, string key, T value)
{
    Blob blob = _repo.ObjectDatabase.CreateBlob(...);


    TreeDefinition treeDef = TreeDefinition.From(...);

    treeDef.Add(key, blob, Mode.NonExecutableFile);
    Tree tree = _repo.ObjectDatabase.CreateTree(treeDef);

    Commit commit = _repo.ObjectDatabase.CreateCommit(...);

    _repo.Refs.UpdateTarget(_repo.Refs[refName], commit.Id);
}
```



Concurrency

WHY NOT

```
public string Save<T>(string branch, string key, T value)
{
    Blob blob = _repo.ObjectDatabase.CreateBlob(...);


    TreeDefinition treeDef = TreeDefinition.From(...);

    treeDef.Add(key, blob, Mode.NonExecutableFile);
    Tree tree = _repo.ObjectDatabase.CreateTree(treeDef);

    Commit commit = _repo.ObjectDatabase.CreateCommit(...);

    _repo.Refs.UpdateTarget(_repo.Refs[refName], commit.Id);
}
```



Solution

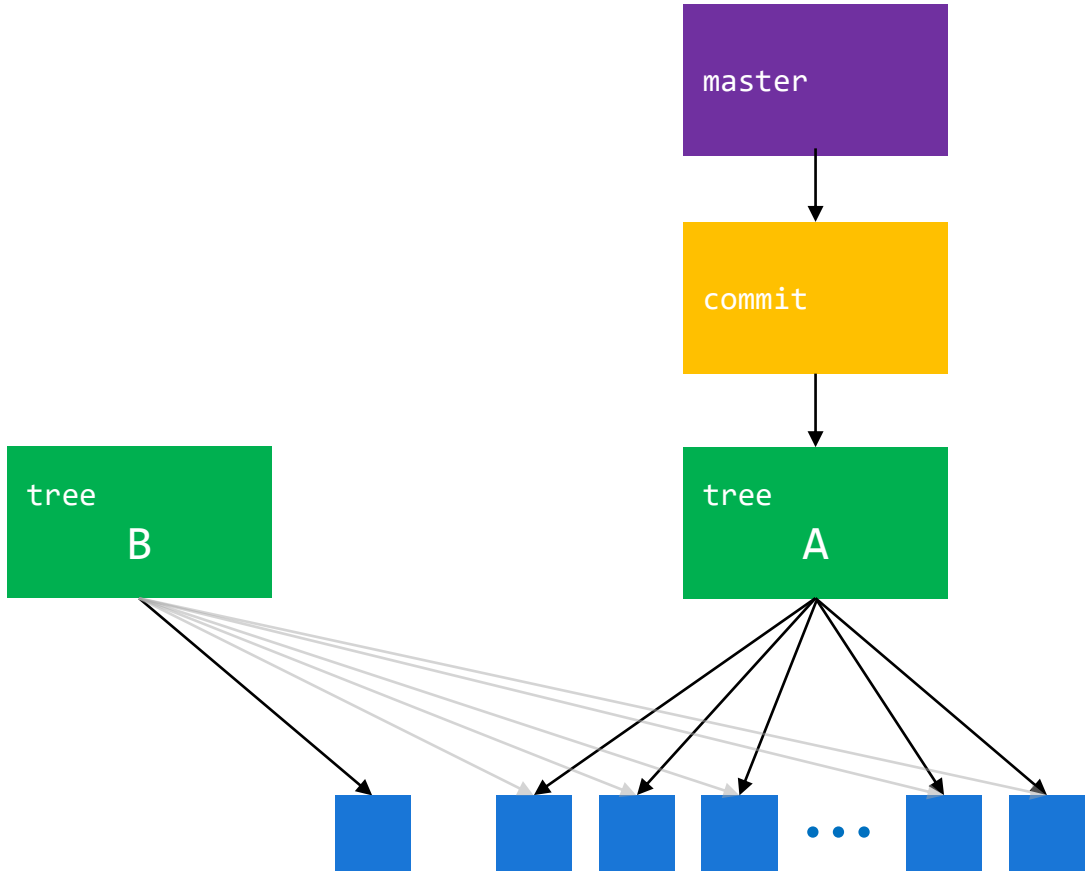
WHY NOT

```
var locks = new Dictionary<string, object>
{
    {"master", new object() },
    {"branch", new object() },
};

lock (locks[branch])
{
    // ...
}
```

WHY NOT

Performance

Writes per second	
125	(10,000 files)
18	(1,000,000 files)



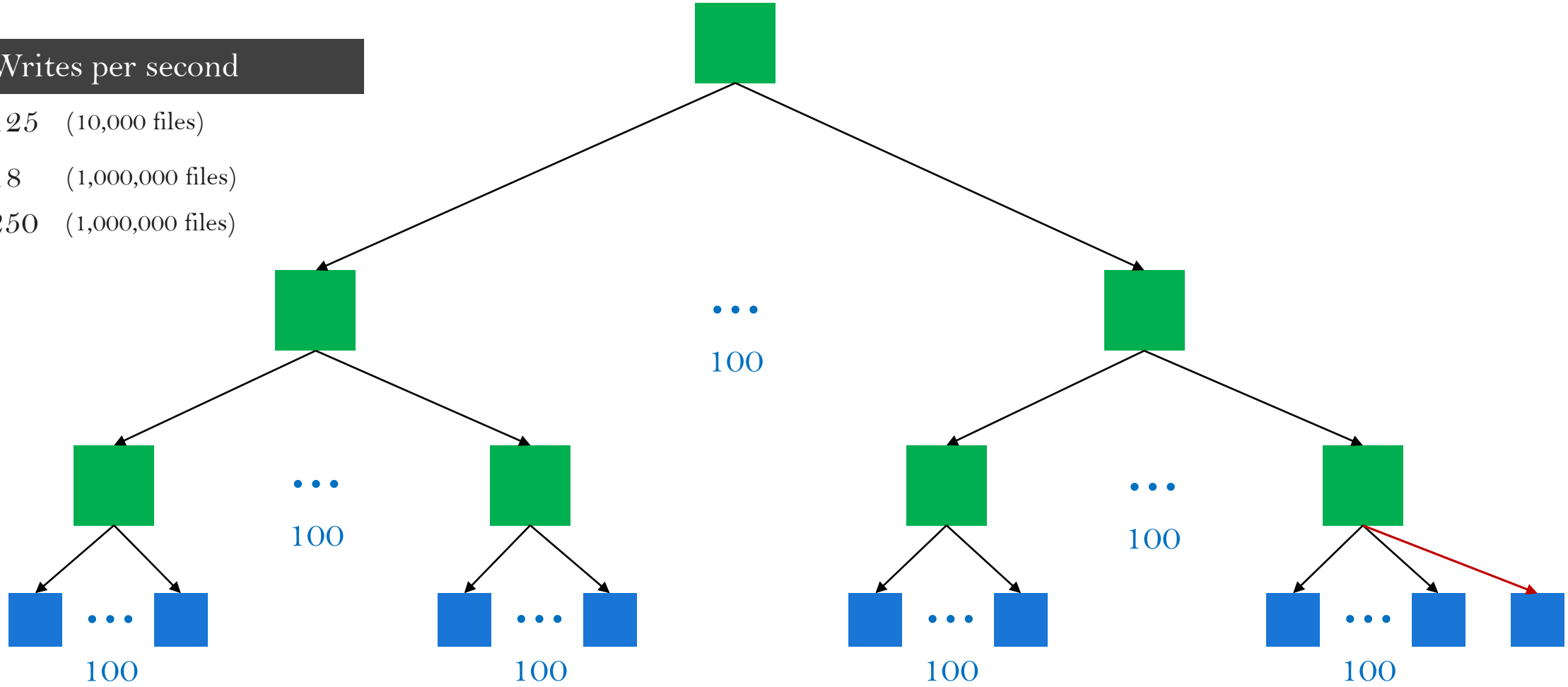

```
100644 blob b9e36d8ed060ebdad585e93cec84    1.json
100644 blob 5a947c79aae7b7f6c663f632bfcf    2.json
 .
 .
100644 blob 4b54f5a8e56cfbc90947e3071ef6    999998.json
100644 blob 06179b3af5668bf7aa1dcb205949    999999.json
100644 blob a3d0db85108eb3e0c3d2b989a2ce    1000000.json
100644 blob 5bc5e544f68a56f992ca2be2fd26    1000001.json
```

```
100644 blob b9e36d8ed060ebdad585e93cec84    1.json
100644 blob 5a947c79aae7b7f6c663f632bfcf    2.json
 .
 .
100644 blob 4b54f5a8e56cfbc90947e3071ef6    999998.json
100644 blob 06179b3af5668bf7aa1dcb205949    999999.json
100644 blob a3d0db85108eb3e0c3d2b989a2ce    1000000.json
```

Solution: tree nesting

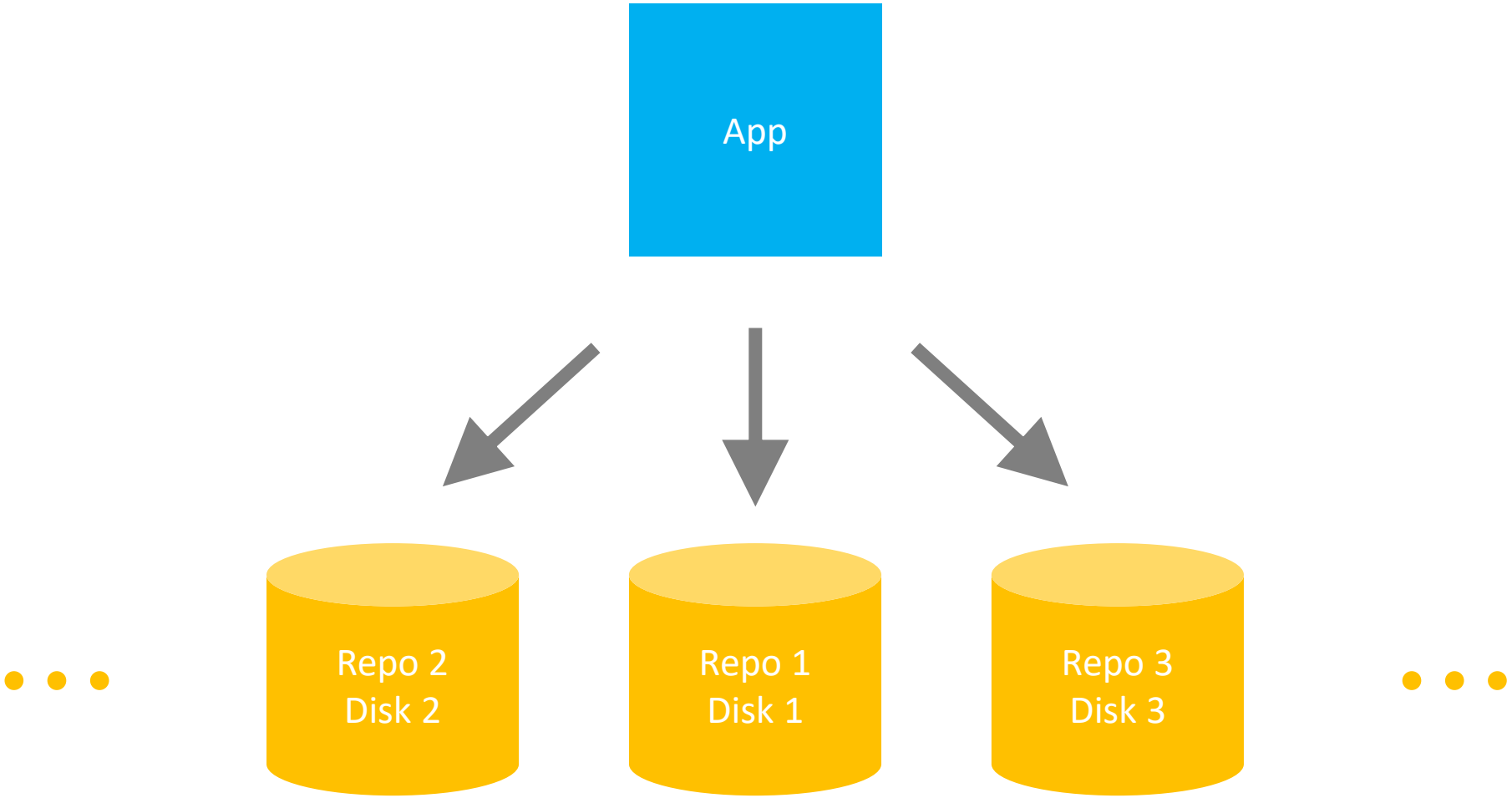
WHY NOT

Writes per second	
125	(10,000 files)
18	(1,000,000 files)
250	(1,000,000 files)



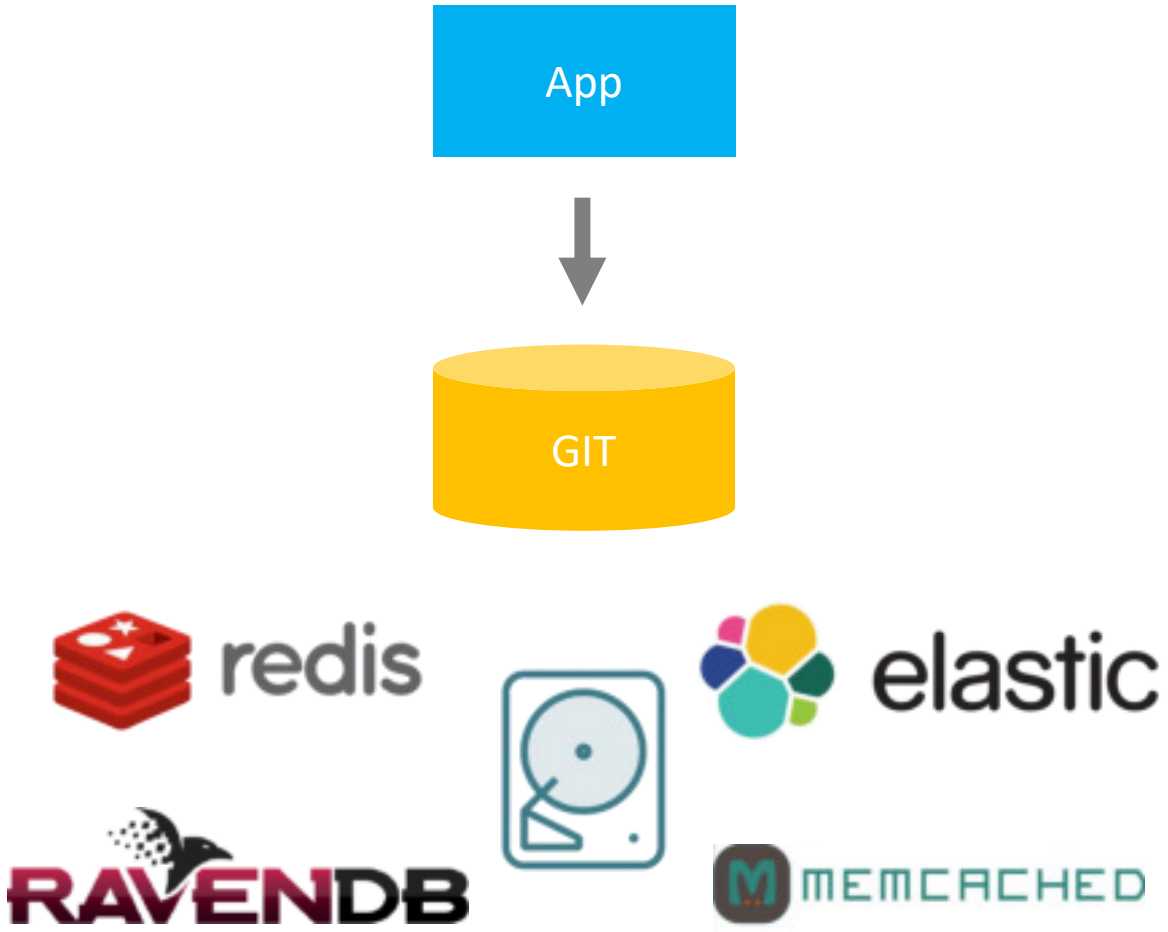
Solution: partitioning

WHY NOT



Solution: back-end

WHY NOT



WHY NOT

Merge conflicts

Solution?

WHY NOT

Good idea

Content heavy

CMS, Wiki, …

Partitionable

Country, customer, …

Bad idea

Fast writes

Immediate consistency

Demo

Kenneth Truyers
@kennethtruyers
www.kenneth-truyers.net

(ab)use your tools

bit.ly/git-db-code

bit.ly/git-nosql