CRUD: CREATE, READ, UPDATE, DELETE

32

32

Insertion

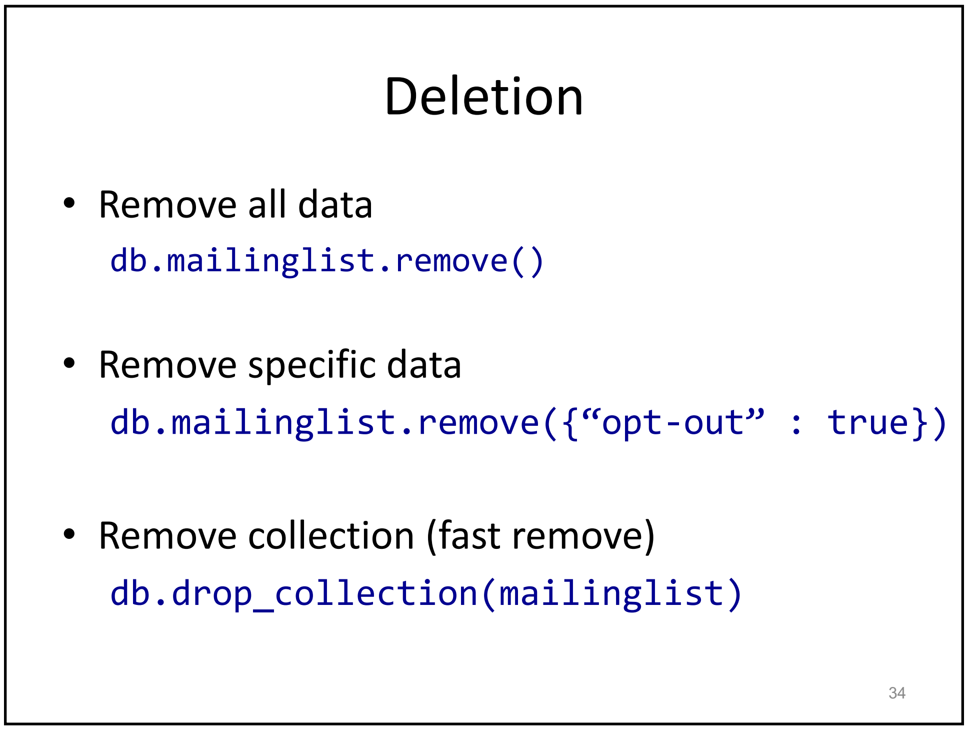
- Single insertion

```
db.foo.insert({“bar” : “baz”})
```

- Batch insertion

33

33



Deletion

- Remove all data

```
db.mailinglist.remove()
```

- Remove specific data

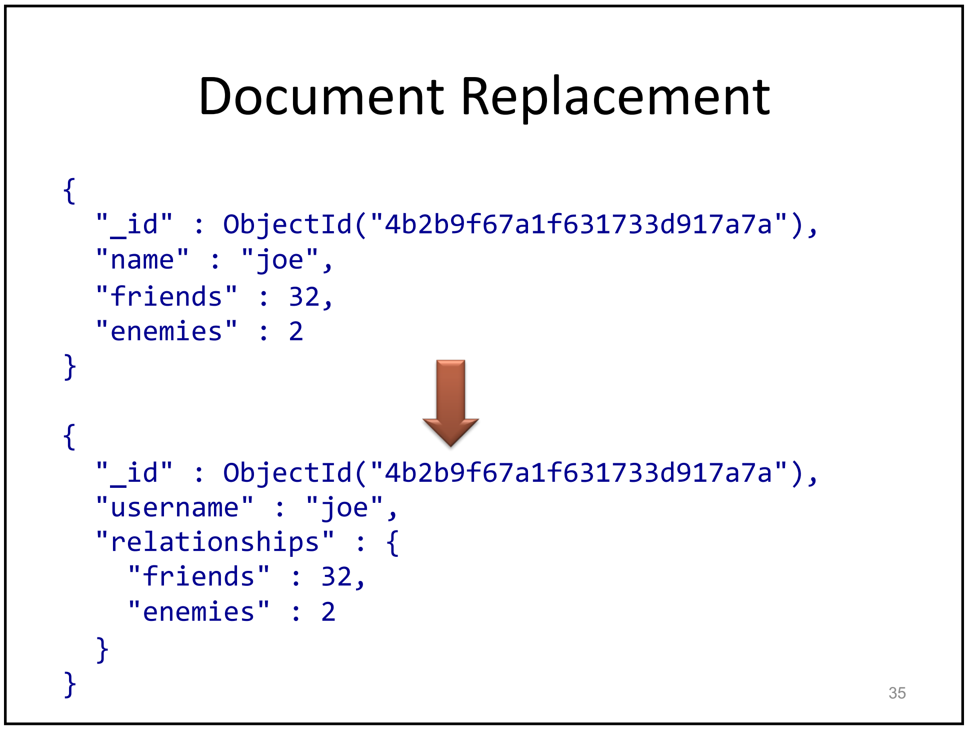
```
db.mailinglist.remove({“opt-out” : true})
```

- Remove collection (fast remove)

```
db.drop_collection(mailinglist)
```

34

34



Document Replacement

```
{
  "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
  "name" : "joe",
  "friends" : 32,
  "enemies" : 2
}

{
  "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
  "username" : "joe",
  "relationships" : {
    "friends" : 32,
    "enemies" : 2
  }
}
```

35

35

Document Replacement

```
var joe = db.users.findOne({"name" : "joe"});
```

```
{
  "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
  "name" : "joe",
  "friends" : 32,
  "enemies" : 2
}
```

36

36

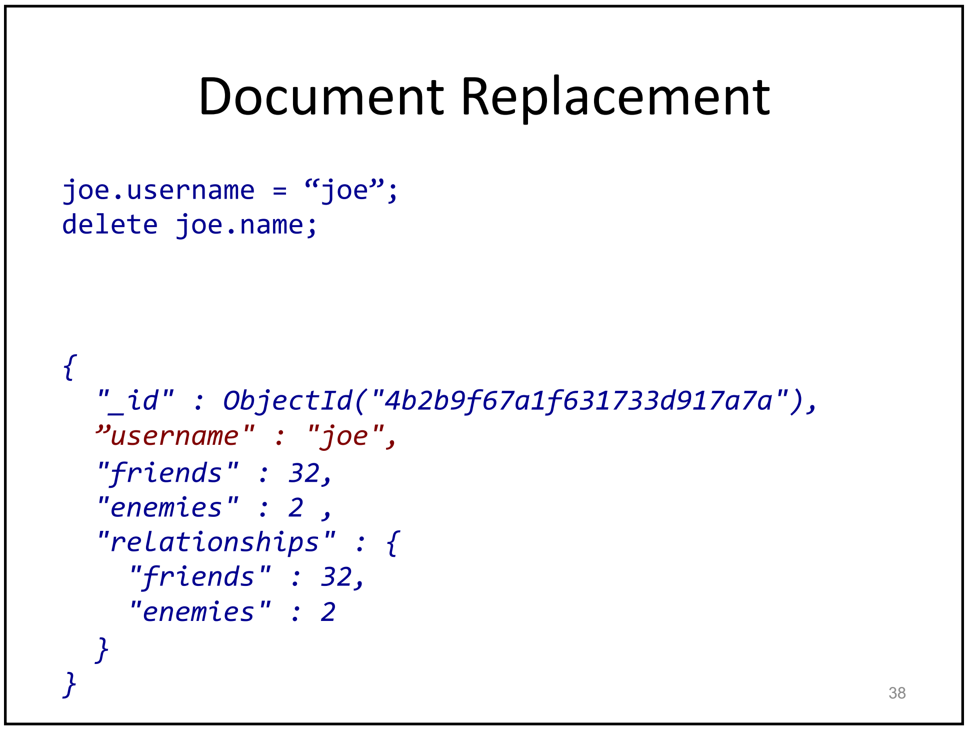
Document Replacement

```
joe.relationships = {
                      "friends" : joe.friends,
                      "enemies" : joe.enemies
                    };
```

```
{
  "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
  "name" : "joe",
  "friends" : 32,
  "enemies" : 2 ,
  "relationships" : {
    "friends" : 32,
    "enemies" : 2
  }
}
```

37

37



Document Replacement

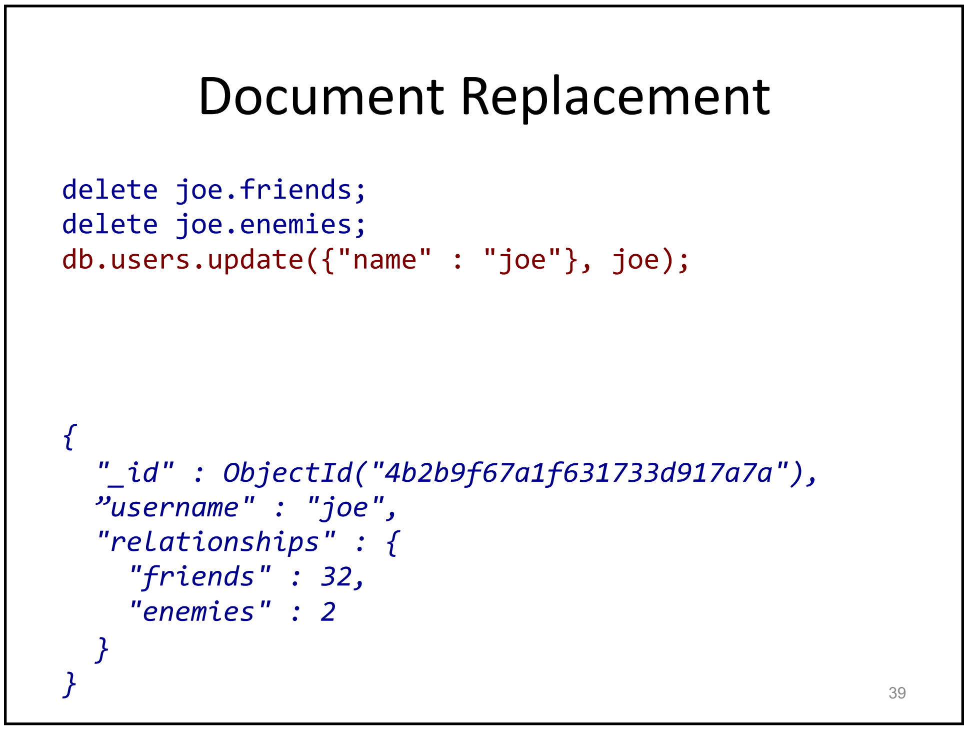
```
joe.username = “joe”;
delete joe.name;



{
  "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
  ”username" : "joe",
  "friends" : 32,
  "enemies" : 2 ,
  "relationships" : {
    "friends" : 32,
    "enemies" : 2
  }
}
```

38

38



Document Replacement

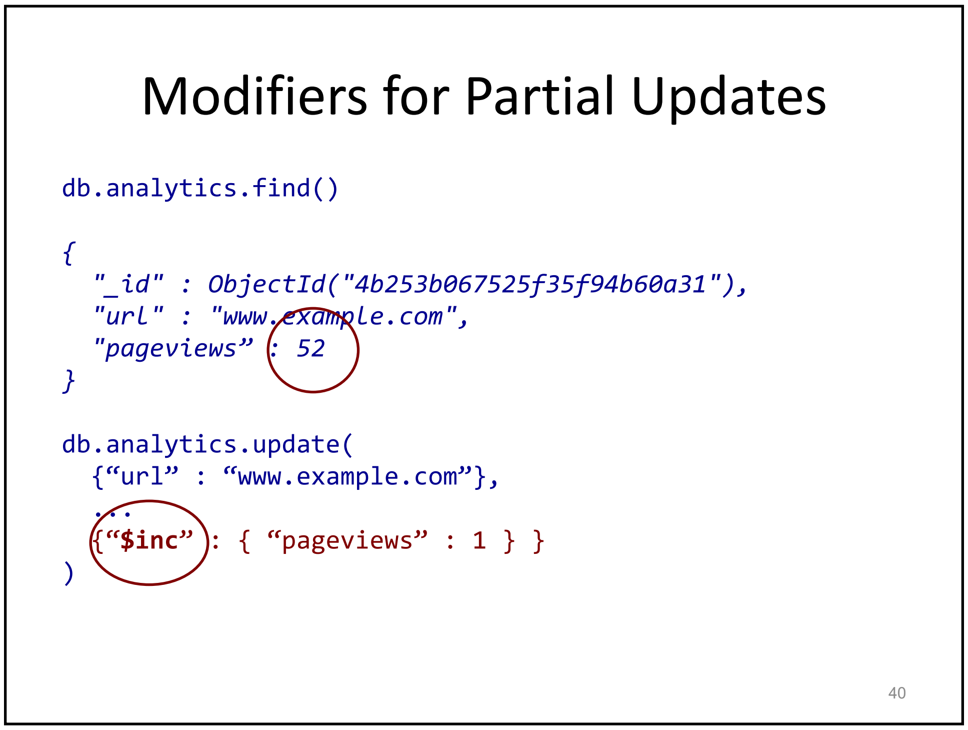
```
delete joe.friends;
delete joe.enemies;
db.users.update({"name" : "joe"}, joe);



{
  "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
  ”username" : "joe",
  "relationships" : {
    "friends" : 32,
    "enemies" : 2
  }
}
```

39

39



Modifiers for Partial Updates

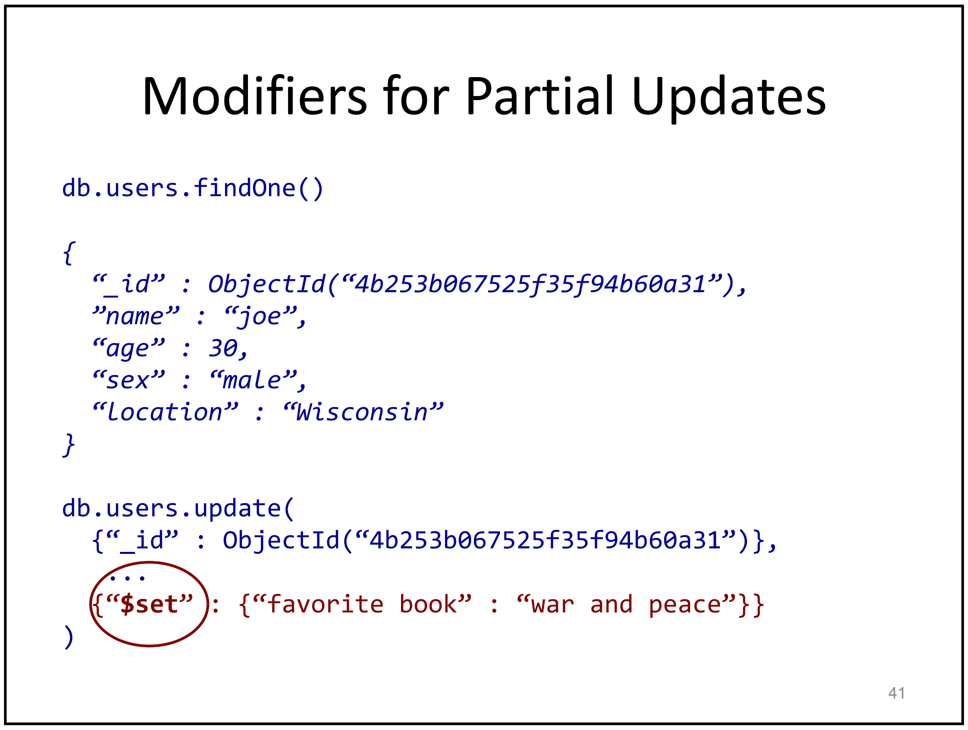
```
db.analytics.find()

{
  "_id" : ObjectId("4b253b067525f35f94b60a31"),
  "url" : "www.example.com",
  "pageviews” : 52
}

db.analytics.update(
  {“url” : “www.example.com”},
  ...
  {“$inc” : { “pageviews” : 1 } }
)
```

40

40



Modifiers for Partial Updates

```
db.users.findOne()

{
  “_id” : ObjectId(“4b253b067525f35f94b60a31”),
  ”name” : “joe”,
  “age” : 30,
  “sex” : “male”,
  “location” : “Wisconsin”
}

db.users.update(
  {“_id” : ObjectId(“4b253b067525f35f94b60a31”)},
  ...
  {“$set” : {“favorite book” : “war and peace”}}
)
```

41

41

Modifiers for Partial Updates

```
db.users.findOne()

{
  "_id" : ObjectId("4b253b067525f35f94b60a31"),
  "title" : "A Blog Post”,
  "content" : "...",
  "author" : {
    "name" : "joe”,
    "email" : "joe@example.com"
  }
}
db.blog.posts.update(
  {“author.name” : “joe”},
  ...
  {“$set” : {“author.name” : “joe schmoe”}}
)
```

42

Pushing onto a List

```
db.blog.posts.findOne()
{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "title" : "A blog post",
  "content" : "...”
}
db.blog.posts.update({"title" : "A blog post"},
 {$push : {"comments" : ... {"name" : "joe",
                             "email" : "joe@example.com",
                             "content" : "nice post."}}})
db.blog.posts.findOne()
{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "title" : "A blog post",
  "content" : "...",
  "comments" : [ {"name" : "joe”,
                  "email” : "joe@example.com",
                  "content" : "nice post.” } ]
}
```

43

Pushing onto a List

```
db.blog.posts.update({"title" : "A blog post"},
  {$push : {"comments" : ... {"name" : "bob",
                              "email" : "bob@example.com",
                              "content" : "good post."}}})

db.blog.posts.findOne()
{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "title" : "A blog post",
  "content" : "...",
  "comments" : [ {"name" : "joe”,
                  "email” : "joe@example.com",
                  "content" : "nice post.” },
                {"name" : ”bob”,
                  "email” : ”bob@example.com",
                  "content" : ”good post.” } ]
}
```

44

44

Pushing onto a List

```
db.papers.update(
  { },
  ... {$push : {"authors cited" : "Richie"}}
)


db.papers.update(
  {"authors cited" : {"$ne" : "Richie"}},
  ... {$push : {"authors cited" : "Richie"}}
)
```

45

45

Adding to a Set

```
db.users.findOne(
  {"_id" : ObjectId("4b2d75476cc613d5ee930164")}
)

{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "username" : "joe",
  "emails" : ["joe@example.com",
              "joe@gmail.com",
              "joe@yahoo.com”]
}
```

46

46

Adding to a Set

```
db.users.update(
  {"_id" : ObjectId("4b2d75476cc613d5ee930164")},
  ... {"$addToSet" : {"emails" : "joe@gmail.com"}})

db.users.update(
  {"_id" : ObjectId("4b2d75476cc613d5ee930164")},
  ... {"$addToSet" : {"emails" : "joe@hotmail.com"}})

db.users.findOne(
  {"_id" : ObjectId("4b2d75476cc613d5ee930164")}})
{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "username" : "joe",
  "emails" : ["joe@example.com", "joe@gmail.com",
              "joe@yahoo.com", "joe@hotmail.com”]
}
```

47

47

Adding Multiple Unique Elements

```
db.users.update(
  {"_id" : ObjectId("4b2d75476cc613d5ee930164")},
  {"$addToSet" : ... {"emails" : {"$each" : ["joe@php.net",
                                             "joe@example.com",
                                             "joe@python.org"]}}})


db.users.findOne({"_id" : ObjectId("4b2d75476cc613d5ee930164")})
{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "username" : "joe",
  "emails" : ["joe@example.com",
              "joe@gmail.com",
              "joe@yahoo.com",
              "joe@hotmail.com"
              "joe@php.net"
              "joe@python.org" ]
}
```

48

48

Removing List Elements

```
{$pop : {key : 1}}      // Remove from end
{$pop : {key : -1}}     // Remove from beginning

db.lists.insert(
  {"todo" : ["dishes", "laundry", "dry cleaning"]})

db.lists.update({}, {"$pull" : {"todo" : "laundry"}})

db.lists.find()
{
  "_id" : ObjectId("4b2d75476cc613d5ee930164"),
  "todo" : ["dishes", "dry cleaning"]
}
```

49

49

Positional Modifications

```
db.blog.posts.findOne()
{
  "_id" : ObjectId("4b329a216cc613d5ee930192"),
  "content" : "...",
  "comments" : [ ...
                 { "comment" : "good post",
                   "author" : "John",
                   "votes" : 0 },
                 ... ]
}

db.blog.update({"post" : post_id},
   ... {"$inc" : {"comments.0.votes" : 1}})

db.blog.update({"comments.author" : "John"},
   ... {"$set" : {"comments.$.author" : "Jim"}})
```

Non-Atomic Update

```
// check if we have an entry for this page
blog = db.analytics.findOne({url : "/blog"})

// if we do, add one to the number of views and save
if (blog) {
  blog.pageviews++;
  db.analytics.save(blog);
}
// otherwise, create a new document for this page
else {
  db.analytics.save({url : "/blog", pageviews : 1})
}
```

Atomic Update via Upsert Option

```
db.analytics.update({"url" : "/blog"},
                    {"$inc" : {”pageviews" : 1}},
                    true)

db.math.remove()
db.math.update({"count" : 25},
               {"$inc" : {"count" : 3}},
               true)
db.math.findOne()
{
  "_id" : ObjectId("4b3295f26cc613d5ee93018f"),
  "count" : 28
}
```

52

52

Multi-Updates

```
db.users.update(
  {birthday : "10/13/1978"},
  ... {$set : {gift : "Happy Birthday!"}},
  false,
  true)

db.runCommand({getLastError : 1})
{
  "err" : null,
  "updatedExisting" : true,
  "n" : 5,
  "ok" : true
}
```

53

53

Non-atomic Query & Update

```
{
  "_id" : ObjectId(),
  "status" : state,
  "priority" : N
}

ps = db.processes.find({"status" : "READY")
                 .sort({"priority" : -1})
                 .limit(1)
                 .next()
db.processes.update(
  {"_id" : ps._id},
  {"$set" : {"status" : "RUNNING"}})
do_something(ps);
db.processes.update(
  {"_id" : ps._id},
  {"$set" : {"status" : "DONE"}})
```

54

54

Atomic Query & Update

```
ps = db.runCommand(
  { "findAndModify" : "processes",
    ... "query" : {"status" : "READY"},
    ... "sort" : {"priority" : -1},
    ... "update" : {"$set" : {"status" : "RUNNING"}}
  }).value
{
  "ok" : 1,
  "value" : { "_id" :
ObjectId("4b3e7a18005cab32be6291f7"),
              "priority" : 1,
              "status" : "READY"
            }
}
do_something(ps);
db.processes.update(
  {"_id" : ps._id},
  {"$set" : {"status" : "DONE"}})
```

55

55

`FindAndModify` Parameters

Key	Parameter
`findAndModify`	Collection name
`query`	Query document
`sort`	Critera for sorting results
`update`	Modifier document
`remove`	Boolean specifying whether to remove document
`new`	Boolean: return updated document or preupdated document. Default: preupdate.

Executing Update Commands

- Fire-and-forget
- Safe updates
  - Run `getLastError` immediately after
  - Problem: latency