Using MapReduce to Retrieve Our News Feed

Databases Three

Otago Polytechnic
Dunedin, New Zealand

Last time

- We saw how to use MapReduce to perform complex queries.
- We supply a *map* function that gets data from documents.
- We supply a *reduce* function that aggregates the map results.
- We supply additional options as required.

Retrieving our feed

Map stage

- Iterate over the users followed by a given user.
- *Emit* the splatts for each user in the list.

Map code

```
var map = function() {
  if(this.splatts) {
    emit("feed", {"list": this.splatts})
  }
}
```

Retrieving our feed

Reduce stage

- Merge our lists of splatts together.
- At this stage, we leave them unsorted.

Reduce code

```
var reduce = function(key, values) {
    var myfeed = {"list": []};
    values.forEach(function(v) {
        myfeed.list = myfeed.list.concat(v.list);
    });
    return myfeed;
}
```

Retrieving our feed

Additional options

- Output
- Query
- Finalisation

Output

```
output:  {inline: 1}
```

Query

```
query: {_id:  {$in: db.users.findOne(
  {_id: ObjectId("541671756269625 9b8000000")}).follow_ids }
}
```

Finalisation

```
var finalise = function(key, val) {
    var mylist = val.list;
    if(mylist) {
        mylist.sort(function(a, b) {
          return b.created_at - a.created_at});
     }
    return {"list": mylist};
}
```

Putting it all together

```
db.users.mapReduce(map, reduce,
   {
   out: {inline: 1},
   finalize: finalise,
   query: {_id:  {$in: db.users.findOne(
      {_id: ObjectId("541671756269625 9b8000000")}).follow_ids }
       }
   })
```

In Ruby

Define strings containing the JavaScript code for map, reduce, and finalise. Then, you can do this:

```
user = User.find(params[:id])
result = User.in(id: user.follow_ids).map_reduce(map, reduce).
         out(inline: true).finalize(finalise)
render json: result.entries[0][:value][:list]
```