Coexist: A Minimalist Data Access System

Anthony Naddeo
advisor: Sudarshan Chawathe

May 2, 2013

The Problem

- Redundant application development
- Single platform clients in popular frameworks
- Application development involves coding

Available Solutions

"It lets you write beautiful code by favoring convention over configuration"

-Ruby on Rails

"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design"

-Django

"With it you can create small web apps like an Address book, a To do list or even a Wine Cellar without writing any code"

-Evolutility

An Ideal Solution Involves...

- Codeless application development
- Clients for multiple platforms
- Minimal development times

Coexist: API based data access

Codeless application development

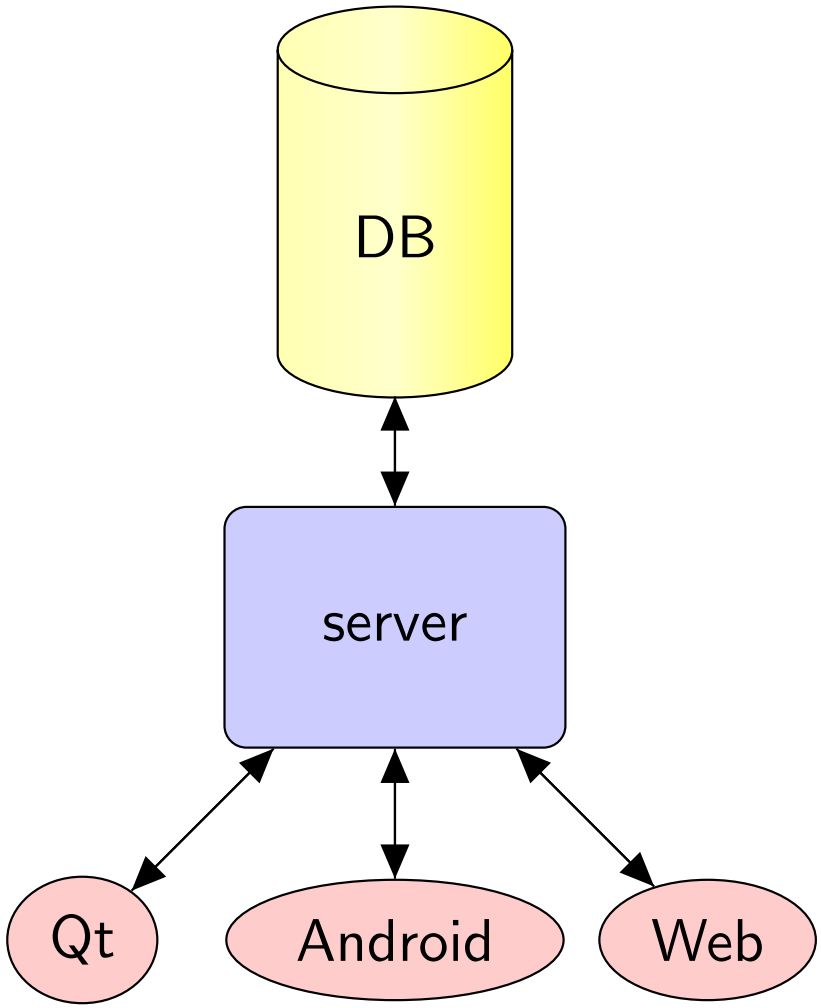
- Provide compatible clients

Clients for multiple platforms

- Decouple the server and clients

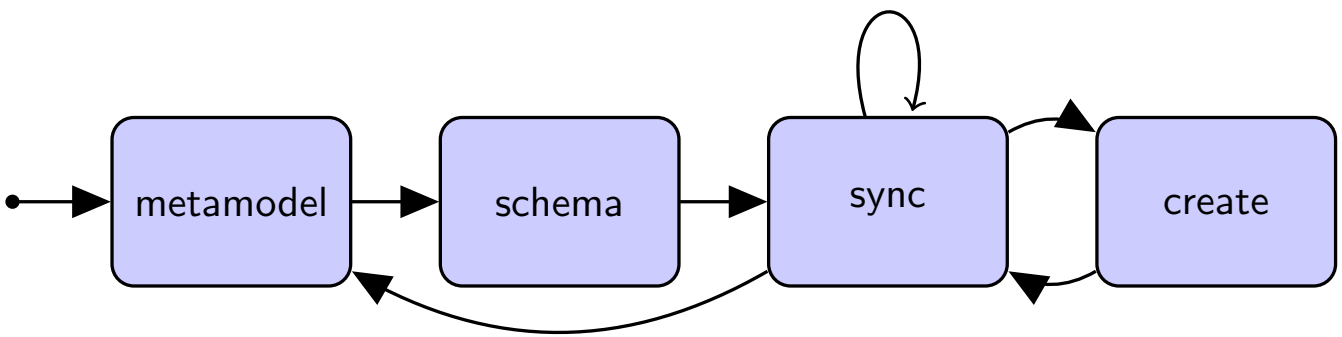
Minimal development times

- Require only configuration files



API Overview


```
http://domain.com/api/metamodel
http://domain.com/api/schema
http://domain.com/api/sync
http://domain.com/api/create
```



API Example


```
{
    status: 200,
    message: "OK",
    version: 2,
  - create: {
      + forms: […],
      + views: […]
    }
}
```


```
{
    status: 200,
    message: "OK",
    version: 2,
  + sql: […]
}
```

/api/sync: Database replication

We make use of *metacolumns*: typical SQL columns that are used for some application level purpose.

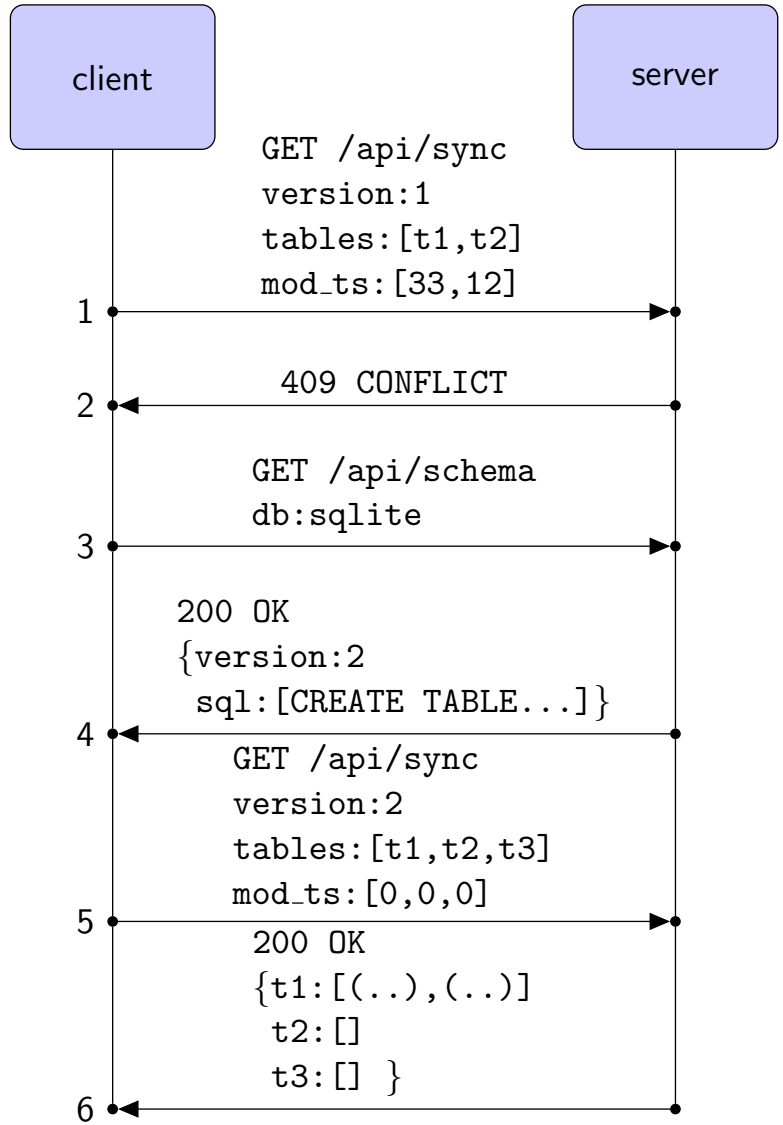
id	name	year	*mod_ts*
1	Alice	4	*0*
2	Bob	4	*10*
3	Carol	3	*0*
4	Dan	4	*10*

- The `mod_ts` column contains the most recent update on that tuple
- Allows the server to give clients missing information when supplied with most recent client side `mod_ts`

Sample client - server dialog

Here is a sample conversation between a server and a client who has an out of date schema while attempting to re-sync.

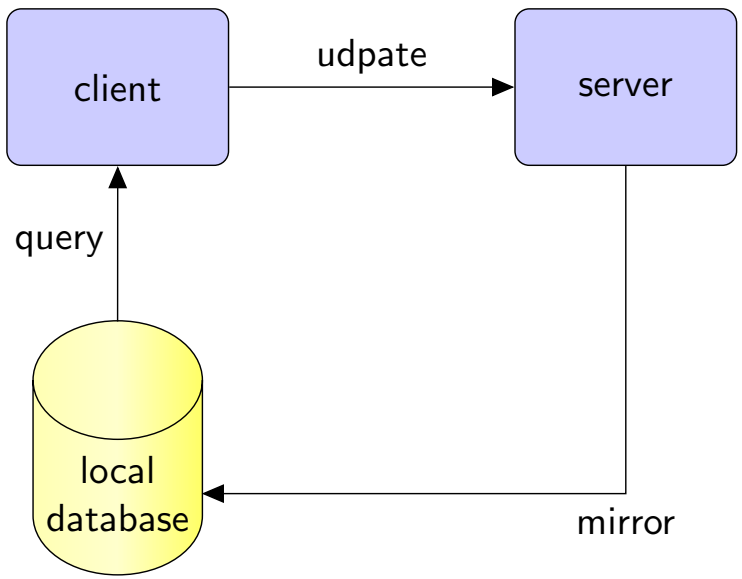
note: The application will not require updates for database changes.



How does the mobile client work.

The entire database is mirrored locally.

- Only dependent on network for posting changes
- Faster query times than Window or Cache could ever reach
- Simple to understand



How does the client know what to request?


```
[
{"tag":"CAPS05654",
"serial":"00020300105608 4",
"model":"TCH-17",
"contract number":"none",
"hostname":"USR-Orono1",
"description":"
    TotalControl17"
},
{"tag":"CAPS05407",
"serial":"72729663",
"model":"7206VXR",
"contract number":"none",
"hostname":"GW-UMF",
"description":"6.SlotChassis
    "
}
]
```

```
create table Exercises(
  exercise VARCHAR(30)
      PRIMARY KEY,
  mod_ts DATETIME,
  deleted INTEGER
);

create table Sets(
  name VARCHAR(30),
  exercise VARCHAR(30),
  set_num INTEGER ,
  reps_done INTEGER ,
  weight INTEGER ,
  date_done DATE,
  mod_ts DATETIME,
  deleted INTEGER
);
```

How does deployment work?

Tentatively, a conf file must be filled out. I have made a tool that will download and compile the latest version of Coexist and its clients according to this.

```
; Android stuff
name=Workoutlog
image=logo.png
notification=notification.png
package=com.domain
api=http://domain.com:/api/

; Server stuff
version=1
user=
pass=
db=
host=localhost
dbms=mysql

create_dir=ui
schema_dir=sql
```

What needs to be done?

- Add more client types (Qt, Desktop Java etc.)
- Support more client features (i.e. Barcode scanning on mobile clients.)
- Create tools to automate the configuration.

I would like to thank

My advisor, Sudarshan Chawathe

Authentication

- $sha1(version + T + M + username + sha1(password))$

Handling deletes