Databases & ORMs

Database Management Sys. (DBMS)

- How to define records (Data Definition)
- Creating/Updating/Deleting Records
- Querying/Retrieval of Records
- Administering the database system

Progression of Databases

- Navigational (< 1970s)
- Relational (> 1970s)
- NoSQL (> 2000s)

Postgres History

- 1970s (UC Berkley): INteractive Graphics REtrieval System
- 1980s: Postgres (“Ingres++”)
- 1995: POSTQUEL and Postgres95
- 1996: open source community adopts it
- Since: stability, testing, documentation, new features
- PostgreSQL

ACID

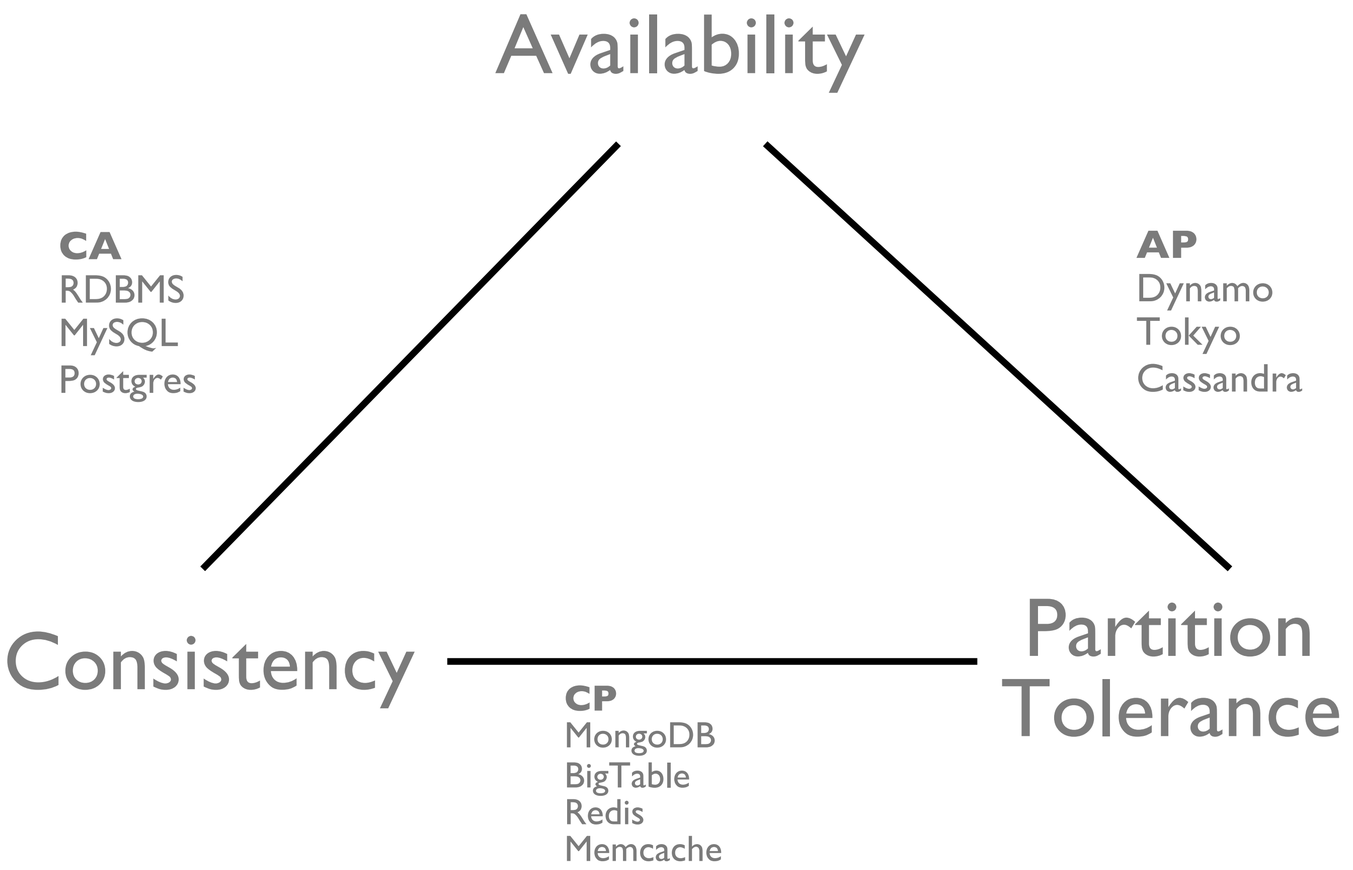
- All about transactions
- Atomicity
- Consistency
- Isolation
- Durability

CAP Theorem

- Consistency: all clients see the same data
- Availability: each client can always read or write
- Partition Tolerance: system works even if network splits

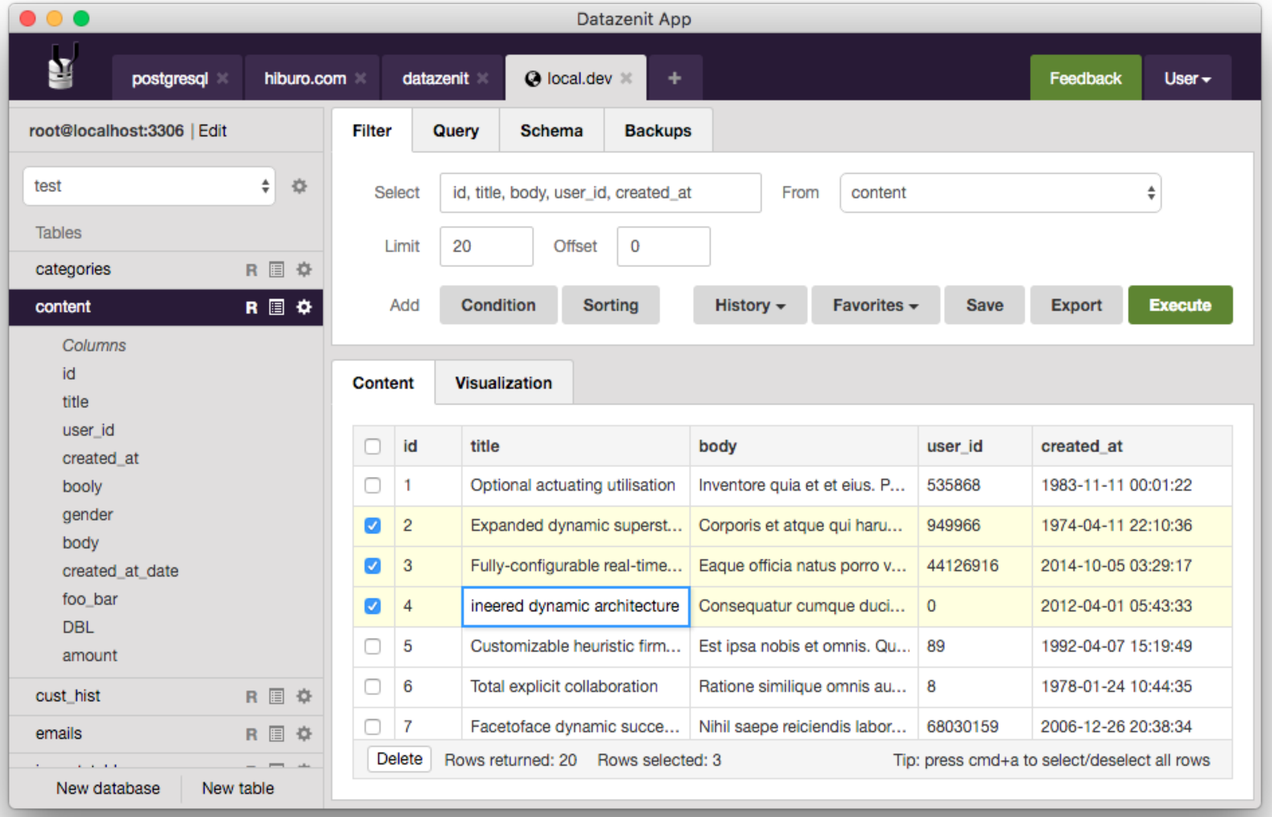
- "Pick 2"
  - (Reality is more complex…)

CAP, metaphorically speaking


Availability
CA
RDBMS
MySQL
Postgres
AP
Dynamo
Tokyo
Cassandra
Consistency
Partition
Tolerance
CP
MongoDB
BigTable
Redis
Memcache

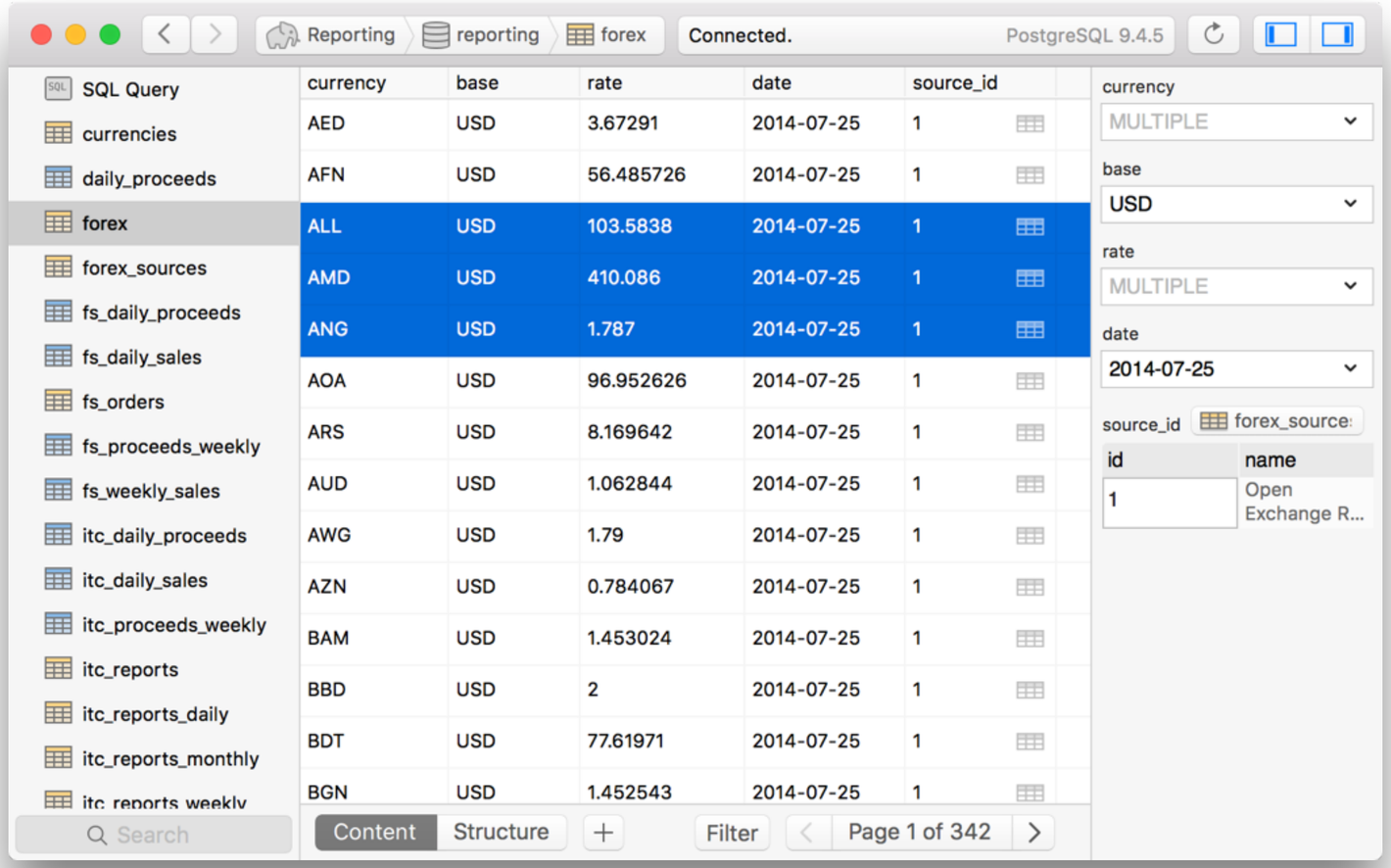
Tooling

Datazenit



Tooling

Postico



Sequelize

- **Sequelize is an Object-Relational Mapper (ORM)**
- **Access SQL databases from Node.js**
- **Sequelize features:**
  - Schema modeling/validation
  - Data casting (convert SQL types to JS types)
  - Query building
  - Hooks (code that runs pre/post save/delete/update)
  - Class and instance methods of models
  - Getters, setters, and virtual fields

Tables		Models
+	=	+
Rows		Instances

Sequelize Basics

- Make a Model (interactive blueprint object)
- Extend the Model with Hooks, Class & Instance Methods, Virtuals, etc.
- Connect/sync the completed Model to an *actual* table in an *actual* SQL database
- Use the Model (Table) to create/find Instances (Table)
- Use the Instances to save/update/delete

Create a Model

```
var Sequelize = require('sequelize');
var db = new Sequelize('postgres://localhost:5432/twitter');
var User = db.define('User', {
  name: Sequelize.String,
  pictureUrl: Sequelize.String
});
```

Sync Model to Table

```
User.sync().then(…);
```

Model & Instance Usage

```
var aPerson = User.build({
  name: "David",
  pictureUrl: "http://fillmurrary.com/10/10"
});

aPerson.save()
.then(…);

User.findAll()
.then(…);
```

Sequelize

- Lives inside Node.js process
- Knows how to communicate to various SQL DBMSs, including PostgreSQL

Wikistack

- Build a Wikipedia clone
- Walk you through installing and using sequelize
- Application of everything we've learned so far