Rails That Scales
A Distributed Database Library for Ruby on Rails

Abstract
A library that extends Ruby On Rails to spread its database workload over multiple servers

Motivation
Startups often run into problems scaling their databases when their traffic suddenly increases:

- Need more capacity?
- Slow to add/remove DBs
- Operations are spread across many servers
- All data is stored on one database server

Rationale:
- Reallocating the data: The challenge with range partitioning is that sometimes, the partitioning needs to change:
- Added a server? Have to include it in the system
- Removed a server? Have to move its data items
- The system determines how to move from the old partitioning to the new partitioning, with the goal:
  - Move as little data as possible
  - User-initiated, automatic process

Data distribution: how to split the data
- The system uses range partitioning:
  - Each data item ends up stored on one database server
  - Whenever an application server needs a data item, it only contacts the server that is responsible for that item
  - With a good function for converting data keys to random numbers, each server ends up with an approximately equal amount of data

Details
- Example of a simple range partitioning:
  - Note how each server is responsible for 25 values

System Limitations
- Some Rails methods, features unavailable
- Limitations on schema

Speed
- Slow to add/remove DBs
  - Rationale: Should only happen a few times

Reliability
- Redundancy must be provided by other means, like replication

Conclusion
- 2007
  - "At this point in time there’s no facility in Rails to talk to more than one database at a time."
  - Twitter Developer Alex Payne, in 2007

- 2010
  - Our project: Rails That Scales, provides this functionality in a library that integrates with Rails!