2013/10/28

Managed designs – Mauro servienti

Mauro.servienti@Manageddesigns.it

Goals:

What is raven

How to use it

How to avoid screwing up

Comparison with other technologies

Problem: orm misuse

Polyglot persistence:

* several persistence technologies for different kinds of data
* need for it is a function of how many use cases a product implements (e.g. erp)

Example: Analysis services + oltp

* Uses ad-hoc data structure fit for a very specific purpose

Driver behind sql/e-r model adoption: hardware cost

Storage cost estimation should include backup costs

Examples of document-oriented db:

* Exchange
* Active directory
* Lotus notes

RavenDB exploits a **native windows** component: name?

Relational model problem: scaling (two examples)

* Performance (essential for cloud/saas)
* High availability (e.g. different timezones, slecht network prestaties) (essential for cloud/saas)

CAP Theorem

Consitency high performance high availability, you can only achieve TWO

(e.g. e-r model use case: high read/writes on same table)

Possible solution: eventual consistency (but how fast????)

e.g. you have inserted an invoice even though you can’t see it ☺ (i.e. ui-related problem)

PROBLEM: change orm isolation level depending on query, orm does not encourage you to think about it

**Missing in RDBMS:**

Failure mode (distributed data, one machine fails, what then?)

Write on a, read from b , data must be the same because we want consistency but what about performance???

**Sharding**: (feature-based data partitioning technique)

Example: geographically distributed data (2 branches: London, Milan, each one stores their orders in a geographically close location, distributed queries are transparent to the application layer)

Advantages: paged distributed queries

Application configures query behavior in case of failure e.g. a remote system is down:

* I want an exception
* I want partial results

Failed rdbms implementations

* Xml columns: kill rdbms
* Azure Table storage (limited by underlying SQL server)

(odata schema)

Document-based db are NOT hot-swappable

(e.g. nhibernate + user chooses backend)

Persistency-ignorance is not easily achieved, true for reading modules

Attributes: eav/vertical table

SQL Server solution: cte

Problem: how to get Nhibernate to understand it? 🡺 user type + dialect extension

Document-based db: you get it for nothing