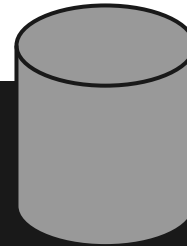


Bancos de dados NoSQL

Estudo de caso com CouchDB



O que garantiu a hegemonia dos bancos de dados relacionais?

- Persistência de dados;
- Concorrência;
- Padrão SQL;
- Integração entre aplicações;
- Relacionamento de dados.

Introdução

- 1988 - BD NoSQL;
- 2009 - Workshop NoSQL;
- Flexibilidade;
- Sem definição prévia de tabelas;
- Alternativa de BD para as aplicações.

Principais características

- Não há uma definição formal;
- Modelo não relacional;
- Sem esquema fixo;
- Sem interface SQL;
- Tendência para *clusters*;
- Normalmente *Open Source*.

Outras características

- Baixa latência;
- Tolerância a falhas;
- Escalabilidade;
- Alta disponibilidade;
- Esquemas flexíveis;
- Baixo custo;
- Busca com heurística.

Aplicações

- Alta quantidade de dados;
- Arquitetura distribuída;
- *Joins* não são suficientes;
- Esquema muda constantemente.

Prós e contras

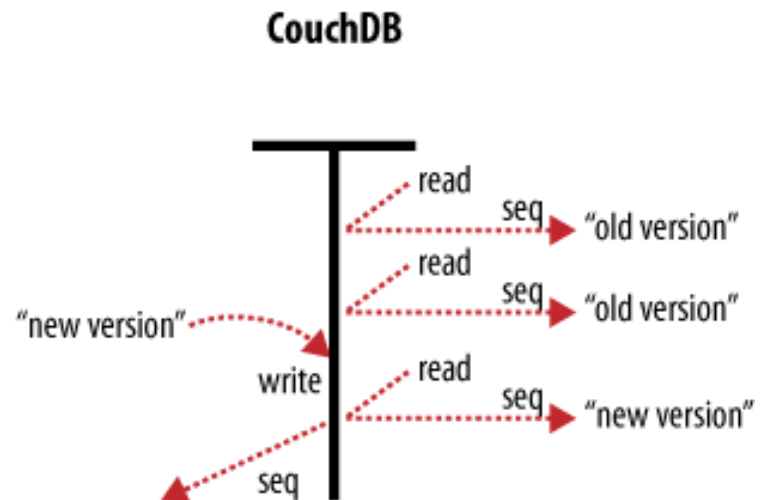
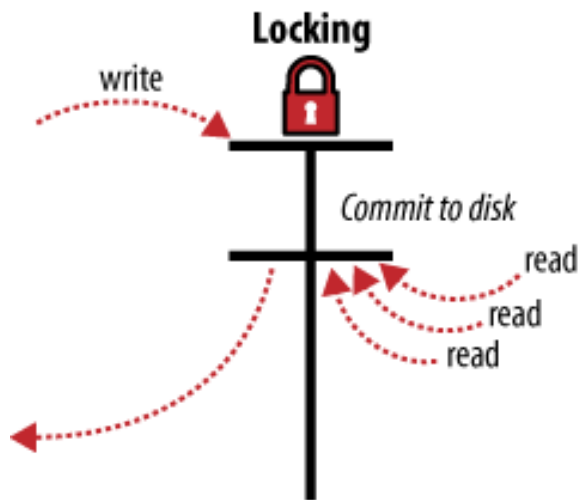
Prós

- Alta disponibilidade;
- Escalabilidade;
- Baixo custo;
- Esquemas flexíveis;
- Dados distribuídos.

Contras

- Buscas dinâmicas limitadas;
- Dificuldade por consistência;
- Sem padrão;
- Sem controle de acesso;
- Busca textual não tão completa como SQL.

NoSQL e banco de dados relacionais



Tipos de bancos de dados NoSQL

- Key Value;
- Wide Column Store;
- Document Store;
- Graph Store;
- Column Oriented Store.

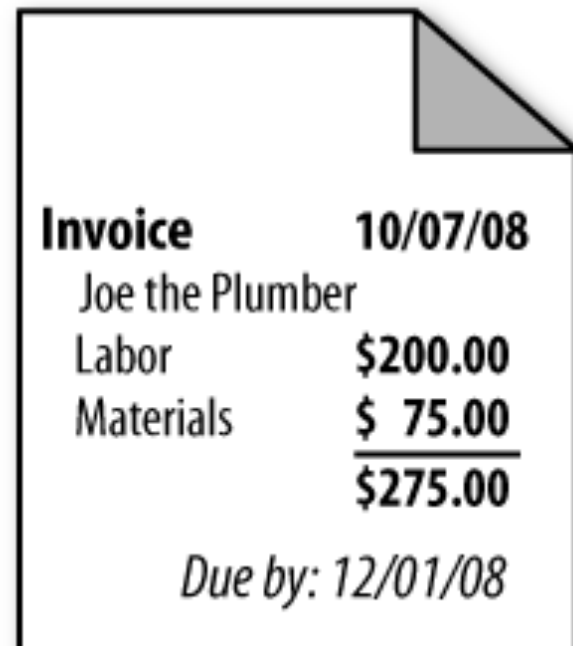
Registros em NoSQL

- Sem esquemas padrões de relacionamento;
- BD não gerência relacionamentos;
- Agrupamento da informação em um registro;
- Mundo real x Modelo.

Exemplo - Documentos autônomos

Conta

Real-world data is managed as real-world documents



Extracted from <http://guide.couchdb.org/draft/why.html>

Escalabilidad

Vertical x Horizontal

Relational databases are designed to run on a single machine, so to scale, you need buy a bigger machine



But it's cheaper and more effective to **scale horizontally** by buying lots of machines.



The machines in these large clusters are individually unreliable, but the overall cluster keeps working even as machines die - so the overall cluster is reliable.

The "cloud" is exactly this kind of cluster, which means relational databases don't play well with the cloud.

The rise of web services provides an effective alternative to shared databases for application integration, making it easier for different applications to choose their own data storage.

Google and Amazon were both early adopters of large clusters, and both eschewed relational databases.

Google → Bigtable

Amazon → Dynamo

Their efforts have been a large inspiration to the **NoSQL** community

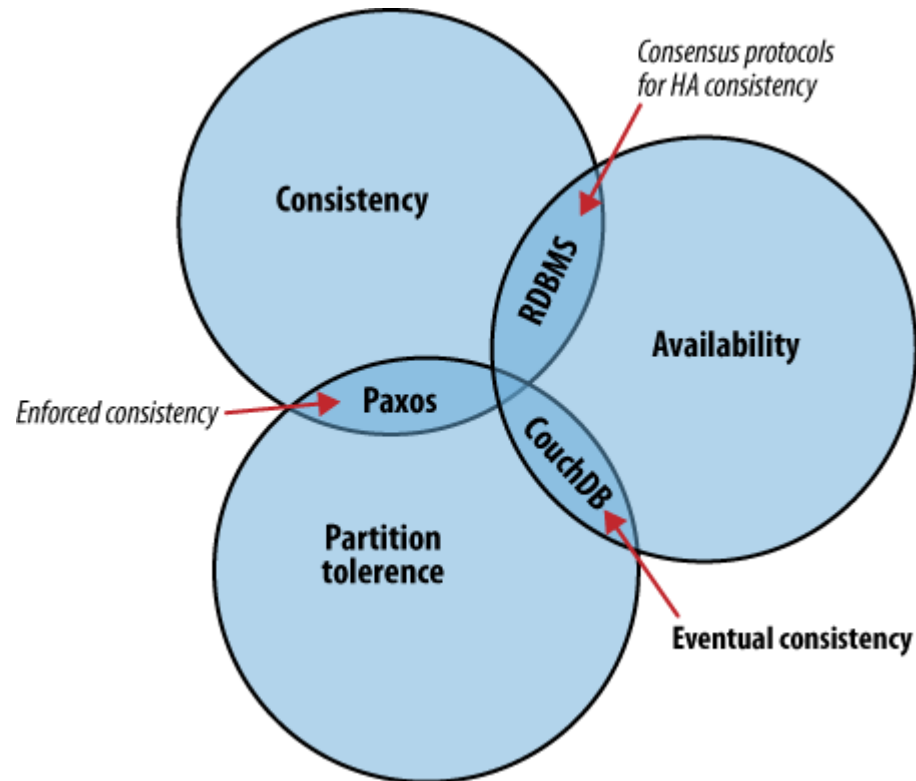
Sistemas Distribuídos

Características comuns

- Operam robustamente sobre uma rede;
- *Network links* podem desaparecer;
- "Dados se comportam diferente quando são utilizados simultaneamente".

Teorema CAP

- Como distribuir a lógica da aplicação?



CouchDB

- Cluster of Unreliable Commodity Hardware (Conjunto de Hardware não confiáveis);
- Código aberto;
- Facilidade na Replicação de dados;
- Dados armazenados como uma coleção de documentos no formato JSON;
- MapReduces (agregação e filtro);
- API REST para manipulação dos dados.

Futon



Organização física

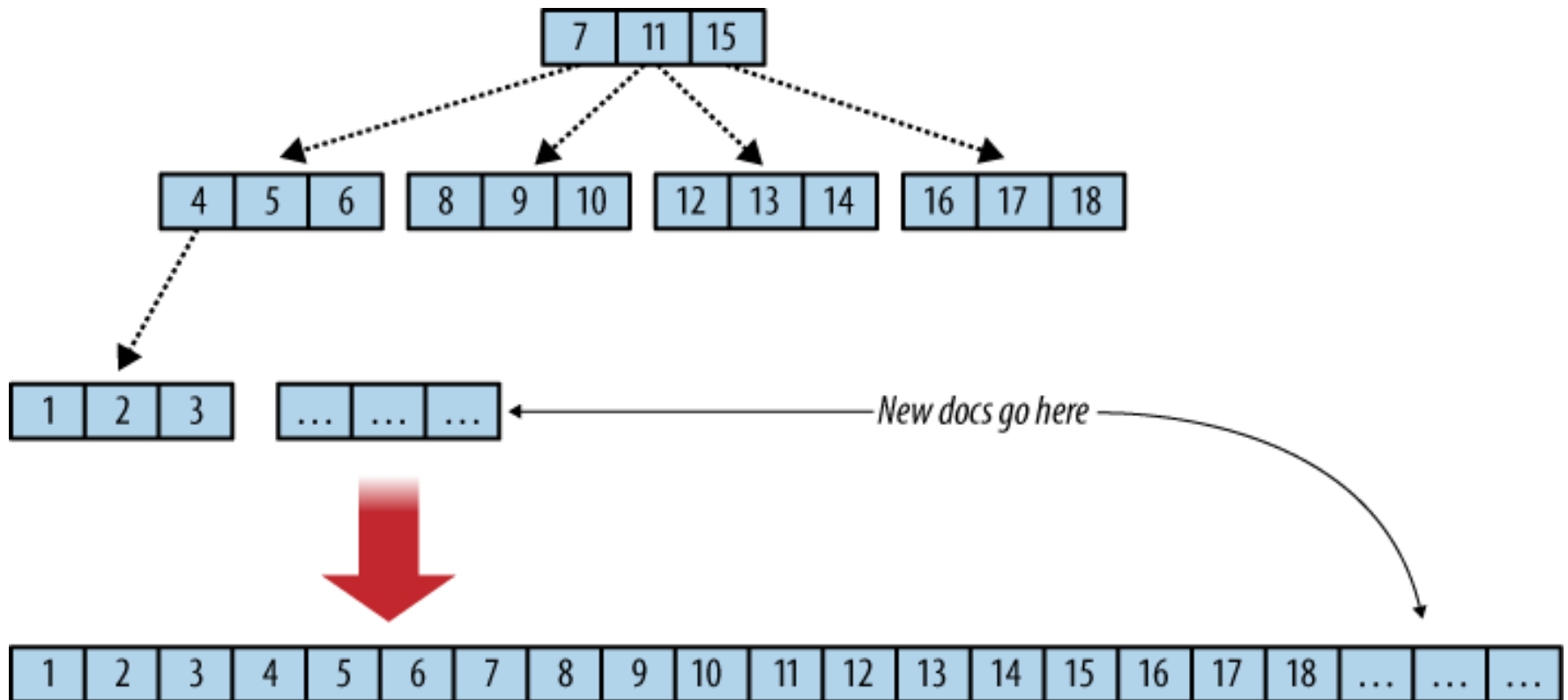
Estrutura: B+_tree

- Excelente para armazenar grandes quantidades de dados, e tem acesso rápido;
- CouchDB armazena as folhas em um dispositivo lento, como *hard disk*;
- Garantem um tempo de acesso inferior à 10ms [Dr. Rudolf Bayer].

Implementação

- *Multi-Version Concurrency Control (MVCC);*
- *Append-only design;*
- Leituras simultâneas de versões diferentes de um mesmo documento;
- Arquivo de banco de dados robusto.

Implementação



Commit - Exemplo

- 2 rodapés de 2k [*+checksum*].

Adicionar "2" ao arquivo ("texto").

- Inicialmente: textoxxx55
- Modificado: texto2xx66
- Corrompido I: texto0xx55 -
- Corrompido II: texto2xx45 -> texto2xx55
- Corrompido III: texto2xx64 -> texto2xx66

As leituras só são reconhecidas como 'bem sucedidas' quando os dois rodapés estão em disco.

API REST

- Utilização da Web;
- Arquitetura Orientada a Recursos;
- REST;
- Aplicações Web;
- GET, HEAD, POST, PUT, DELETE e COPY;
- URIs.

API REST

- /
- /{nomeDoBanco}
- /{nomeDoBanco}/{identificadorDoDocumento}
- /{nomeDoBanco}/{identificadorDoDocumento}/{nomeDoAnexo}
- /{nomeDoBanco}/_design/{nomeDoDesing}
- /{nomeDoBanco}/_design/{nomeDoDesing}/_view/
 {nomeDaView}
- /{nomeDoBanco}/_design/{nomeDoDesing}/_show/
 {nomeDaShowFunction}/{identificadorDoDocumento}
- /{nomeDoBanco}/_design/{nomeDoDesing}/_list/
 {nomeDaListFunction}/{nomeDaView}

JSON

- Formato para representação de objetos;
- Leve e rápido (em comparação com XML);
- Pode ser analisado pela função `eval()` (implementada em todos navegadores web).

```
var dados = {  
  
  "aluno" : [  
  
    {"nome": "João", "provas": [{"nota": 8}, {"nota": 6}, {"nota": 10}, {"nota": 2}]},  
  
    {"nome": "Maria", "provas": [{"nota": 3}, {"nota": 5}, {"nota": 8}, {"nota": 1}]},  
  
    {"nome": "Pedro", "provas": [{"nota": 7}, {"nota": 6}, {"nota": 6}, {"nota": 8}]},  
  
  ]  
}; var json = eval(dados); json.aluno[0].nome
```

JSON x XML

- Mais rápido, leve, e os dados são mais compactos que o XML (devido as sucessivas tags);
- Acesso e leitura dos dados:
 - JSON: Dados são acessíveis pelos objetos JSON no código JavaScript;
 - XML: Dados precisam de ser interpretados (*parsed*) e atribuídos a variáveis através de API's DOM. Possui cabeçalhos adicionais, referências a outros documentos.
- Tipos de dados dos objetos:
 - JSON: *strings, numbers, arrays, booleans, null*;
 - XML: *strings*.

Documento

- Formato JSON;
- `_id`
- `_rev`
- `_attachments`
- `/ {nomeDoBanco} / {identificadorDoDocumento}`
- `/ {nomeDoBanco} / {identificadorDoDocumento}`
`/ {nomeDoAnexo}`
- Modelado de diversas formas.

Documento



Documento

Modelagem 1: postagem

```
{  
  titulo: "Título da postagem",  
  texto: "...",  
  data: "11/09/2001",  
  tipo: "postagem"  
}
```

Documento

Modelagem 1: comentário

```
{  
  autor: "...",  
  comentario: "...",  
  identificadorDa Postagem:  
  "33a483fea6e8d3fae8fb338d9b00179d",  
  tipo: "comentario"  
}
```

Documento

Modelagem 2: postagem

```
{
  titulo: "Título da postagem",
  texto: "...",
  data: "11/09/2001",
  identificadoresDosComentarios: [
    "6beffd0c0c9bf04e72d0dcc029034f2c",
    "9eb51ac3d4df5eb1859ffb972488c7e4",
    "e7d760af60bd8ee418a81a8ed15dcd36"
  ],
  tipo: "postagem"
}
```

Documento

Modelagem 2: comentário

```
{  
  autor: "...",  
  comentario: "...",  
  tipo: "comentario"  
}
```

Documento

Modelagem 3: postagem e comentários

```
{
  titulo: "Título da postagem",
  texto: "...",
  data: "11/09/2001",
  comentarios: [
    {autor: "...", comentario: "..."},
    {autor: "...", comentario: "..."},
    {autor: "...", comentario: "..."}
  ],
  tipo: "postagem"
}
```

Design documents

Design documents

- Código de aplicação;
- *Validate functions, show functions, list functions e views;*
- `/ {nomeDoBanco} / {identificadorDoDocumento}`
- `/ {nomeDoBanco} / _design/ {nomeDoDesing}`
- São documentos;
- Múltiplos *design documents*.

Design documents

```
1 {  
2   "_id": "_design/sofa", ← Determines the app URL  
3   "_rev": "3157636749",  
4  
5  
6   "language": "javascript", (for the web)  
7  
8  
9   "validate_doc_update": "function (newDoc, oldDoc, userCtx) { ... }",  
10                                     Application is stored as JSON data  
11  
12   "views": { ← Views field stores incremental  
13     "comments": { map reduce functions  
14       "map": "function(doc) { ... };",  
15       "reduce": "function(keys, values, rereduce) { ... };"  
16     }  
17   },  
18  
19  
20   "shows": { ← Shows functions transform  
21     "post": "function(doc, req) { ... }"  
22   },  
23 }
```

Design documents

```
23
24
25  "_attachments": {
26    "jquery.couchapp.js": {
27      "stub": true,
28      "content_type": "text/javascript",
29      "length": 7539
30    }
31  },
32
33  "signatures": {
34    "jquery.couchapp.js": "80078849ad6ca281f6993bd012c708f5",
35  },
36
37
38  "lib": {
39    "templates": {
40      "post": "<!DOCTYPE html> ... </html>"
41    }
42  }
43 }
```

Attachments show up as stubs

CouchApp traces attachments here for faster deployments

CouchApp can include library code and data in your functions

Design documents

Validade function

- `validate_doc_update`
- Função JavaScript;
- Erro HTTP;
- Novo documento, documento antigo e contexto de usuário;
- `throw({forbidden: "Cai fora!"});`
- Apenas uma por *design document*.

Design documents

The Incoming Doc

Doc from disk, if any exist.

The user and their roles.

function (newDoc, savedDoc, userCtx) {

// ~~prefer the type as it appears on the saved Doc~~
var type = (savedDoc || newDoc)['type'];

// Domain Specific Language for validation.
function require(beTrue, message) {
 if (!beTrue) throw({forbidden : message});
};

A Doc is invalid if
the function throws
an error.

Design documents

```
// Ensure the timestamp hasn't changed.  
require((savedDoc && savedDoc.created_at === newDoc.created_at),  
  "You may not alter the created_at time.");
```

```
// Post-specific validations.
```

```
if (type === 'post') {
```

```
  validate([
```

```
    newDoc.author,
```

```
    newDoc.title,
```

```
    newDoc.body,
```

```
    newDoc.format,
```

```
    newDoc.html,
```

```
    newDoc.slug,
```

```
    newDoc.slug === newDoc._id,
```

```
    newDoc.created_at,
```

```
  ]);
```

```
}
```

**We'll cover authorization later.*

```
"Posts must have an author.",
```

```
"Posts must have a title.",
```

```
"Posts must have a body.",
```

```
"Posts require a format.",
```

```
"Posts must have an html field.",
```

```
"Posts must have a slug.",
```

```
"Post slugs must be used as the _id.",
```

```
"Posts must have a created_at date."
```

If a user or application were to save a doc with a mismatched timestamp, the function CouchDB would return a 403 Forbidden HTTP status.

The messages sent to client browsers when something is invalid.

Design documents

Show function

- Documentos em outros formatos;
- Documento e detalhes da requisição;
- `/ {nomeDoBanco} / _design / {nomeDoDesing}`
`/ _show / {nomeDaShowFunction} /`
`{identificadorDoDocumento}`
- Várias por *design document*.

Design documents

The document as stored in CouchDB

```
7  
8  
9 function(doc, req) {
```

Details about the HTTP request

```
10  
11  
12  
13  
14 return {
```

```
15   body : "The Blog Post Called: " + doc.title
```

Returns a response object.

The default Content-Type is text/html.

```
16  
17  
18  
19   };
```

```
20  
21  
22  
23  
24  
25 }
```

Design documents

List function

- Semelhante a *show function*, porém aplicadas a views;
- `/ {nomeDoBanco} / _design/
 {nomeDoDesing} / _list/
 {nomeDaListFunction} / {nomeDaView}`
- Várias por *design document*.

Design documents

Views

- Buscas;
- *MapReduce*;
- Map: filtragem e ordenação;
- Reduce: agregação.

Design documents

Views

- Executada uma vez para todos documentos;
- Executada ao atualizar um documento;
- Primeiro acesso demorado.
- `/ {nomeDoBanco} / _design/
{nomeDoDesing} / _view / {nomeDaView}`

Design documents

Map

- Documento;
- `emit`
- Chave e valor;
- Chave complexa.

Design documents

Map

- Seleção;
- Projeção;
- Múltiplos emits;
- Árvore B em arquivo separado.

Design documents

Map

- `/ {nomeDoBanco} / _design / {nomeDoDesing}`
 `/ _list / {nomeDaListFunction} / {nomeDaView}`
- `emit`
- `?key={chave}`
- `?startkey={chave} e ?endkey={chave}`
- `?descending=true`
- `?limit={limite}`

Design documents

SORTING Comments by
Post & date.

function(doc) {

run once for each doc - rev.

if (doc.type == "comment") {

not a join - restricted to one type.

emit([doc.post_id, doc.created_at], doc);

Array key

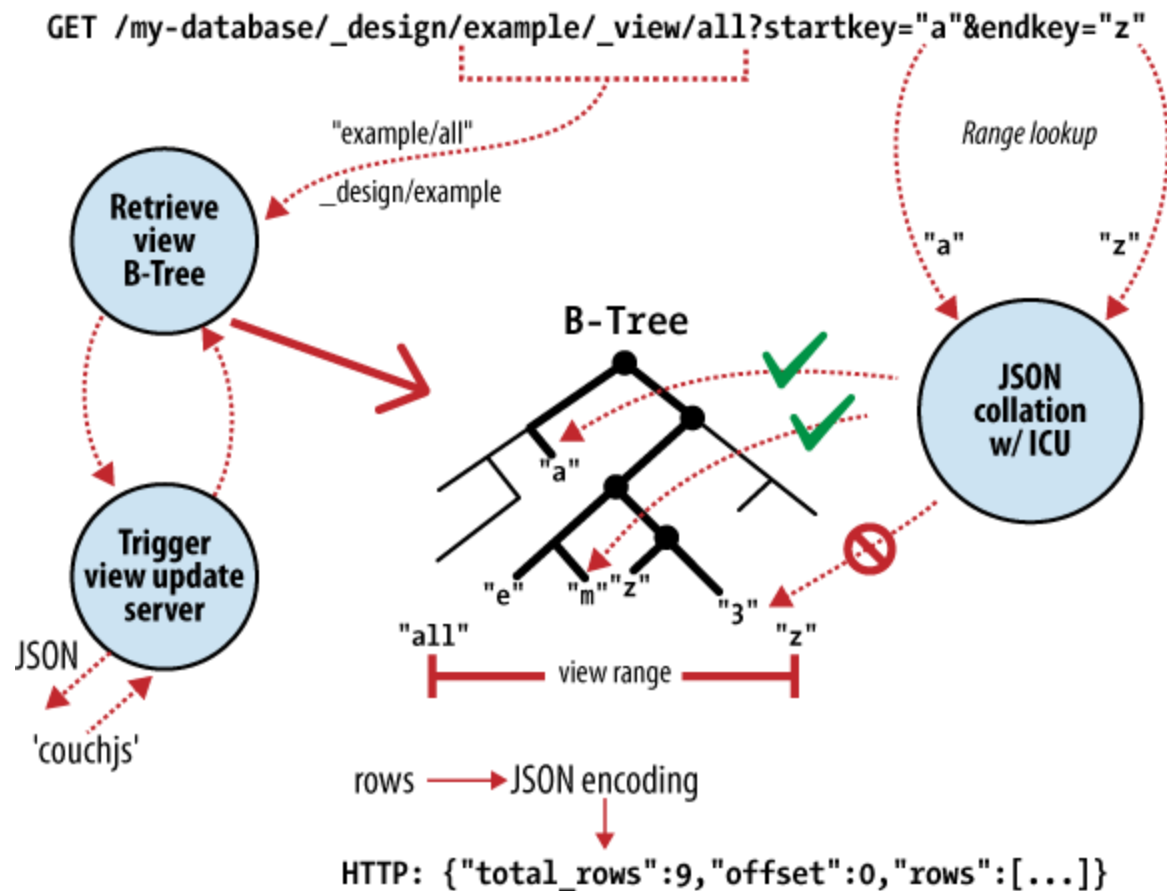
THE
COMMENT
(for display)

~~useful for group-by reduces~~

useful for group-level reduces,
eg. # of comments per post.

}
};
↑
funny mistake

Design documents

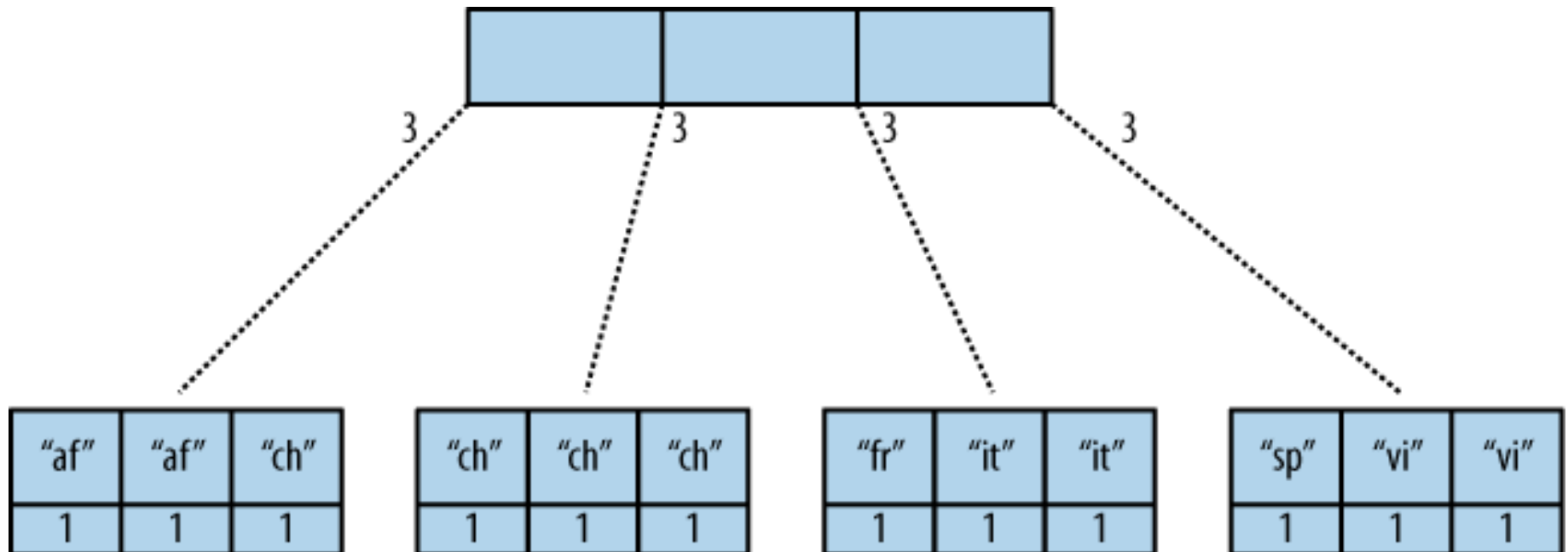


Design documents

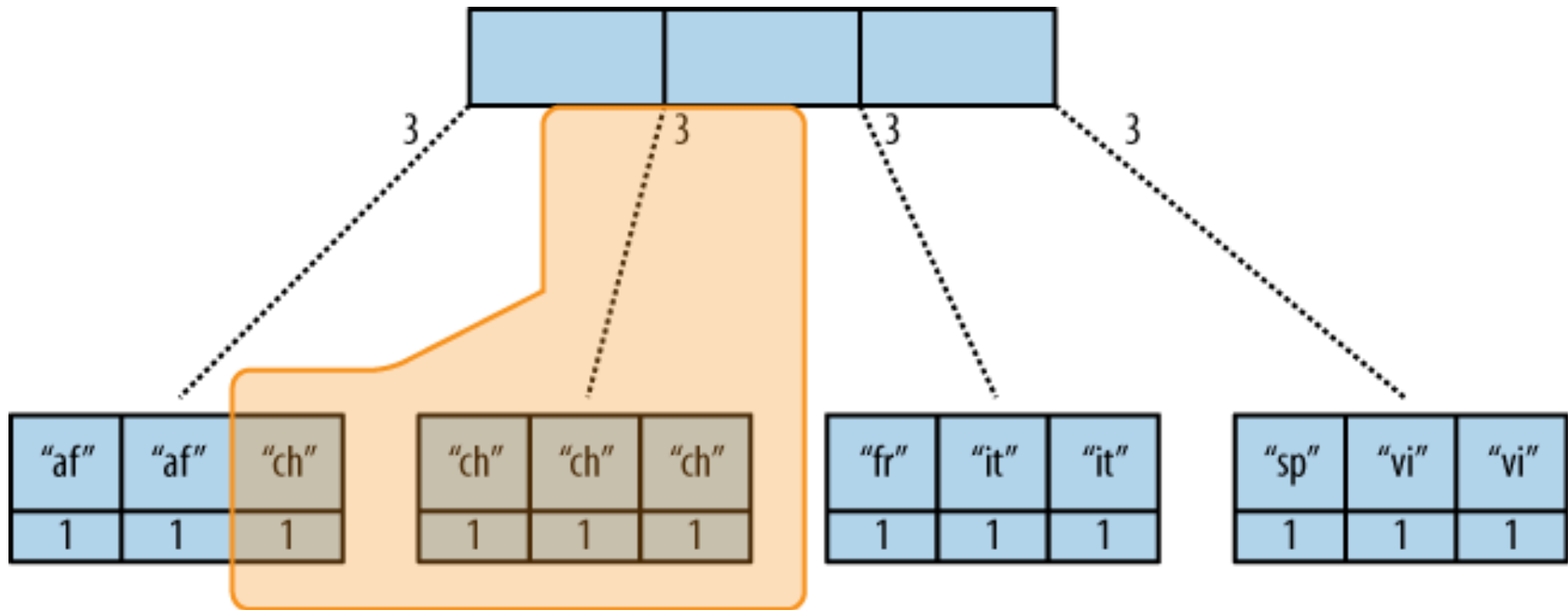
Reduce

- Contar registros, somar valores, achar o máximo e outros;
- Agrupamentos de nodos folhas da árvore B proveniente do *map* são aplicados ao *reduce*;
- `?group=true`
- `?group_level={nivel}`
- Chaves, valores e `rereduce = false`;
- Nulo, *reduces* anteriores e `rereduce = true`;
- Paralelismo.

Design documents



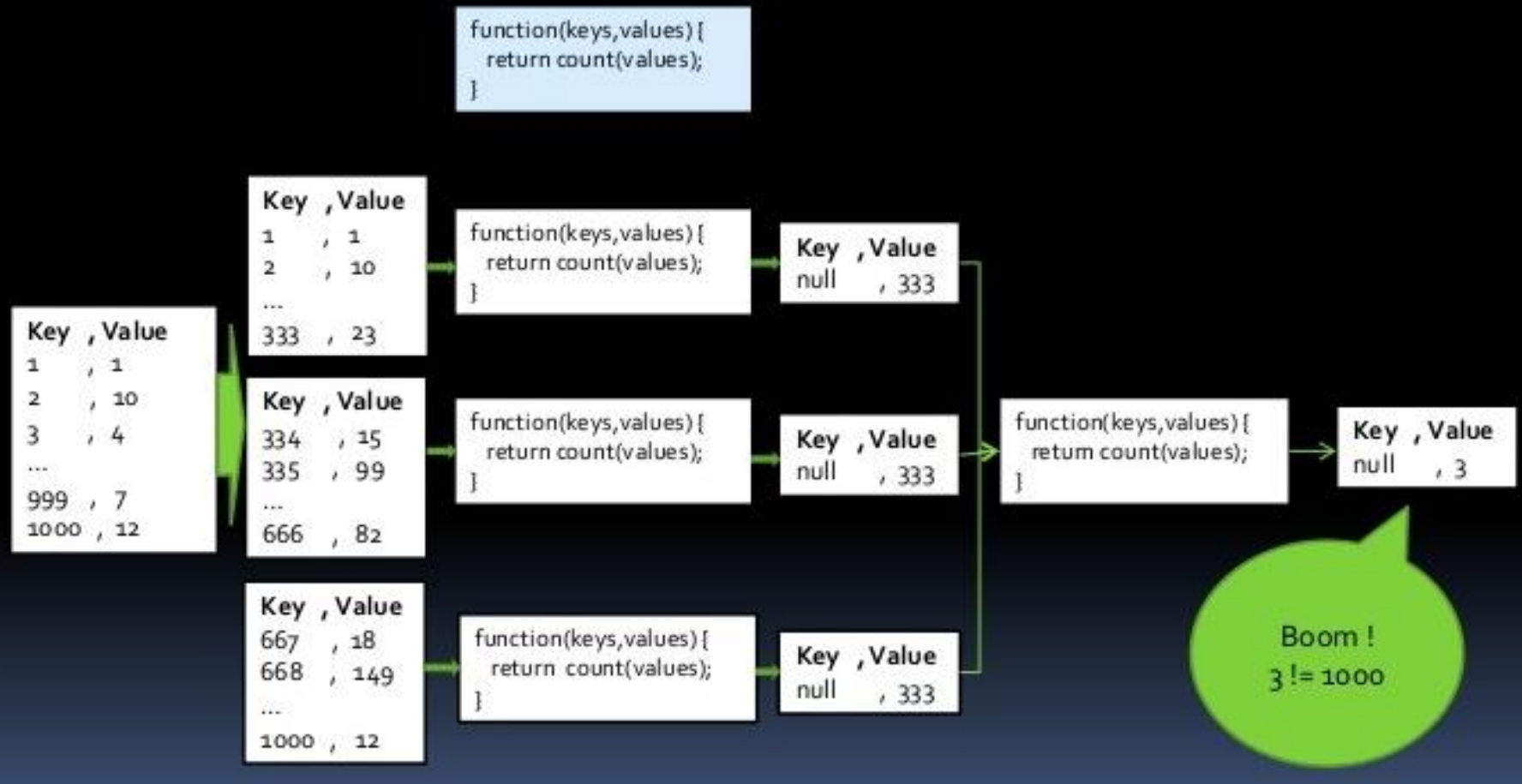
Design documents



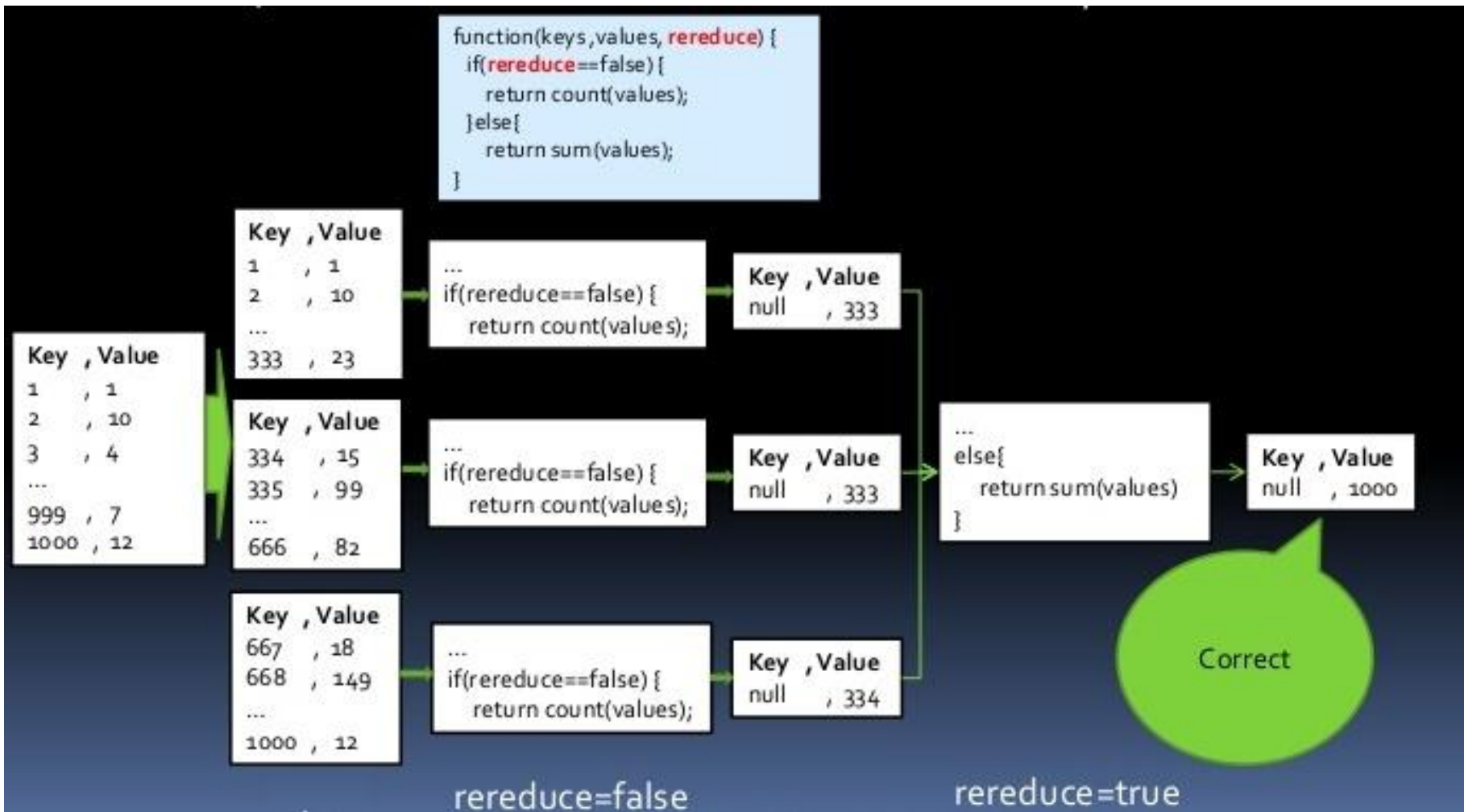
Design documents

```
function (chaves, valores, rereduce) {  
    if (rereduce) {  
        return sum(valores);  
    }  
    return count(valores);  
}
```

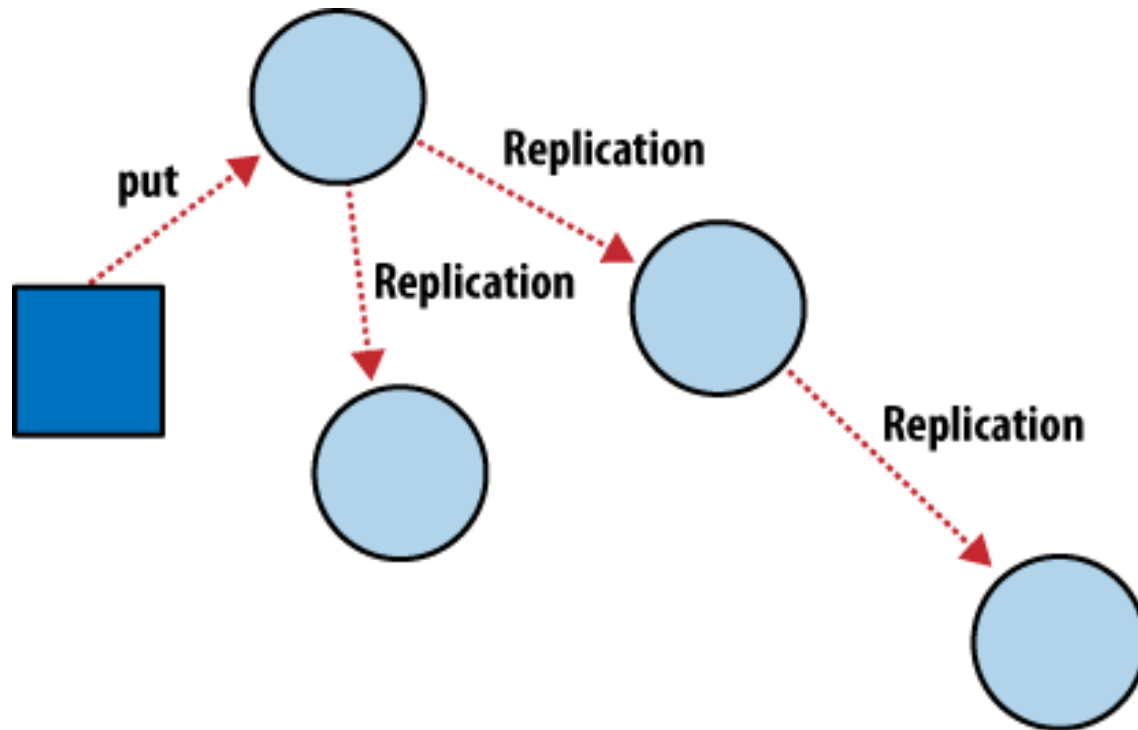
Design documents



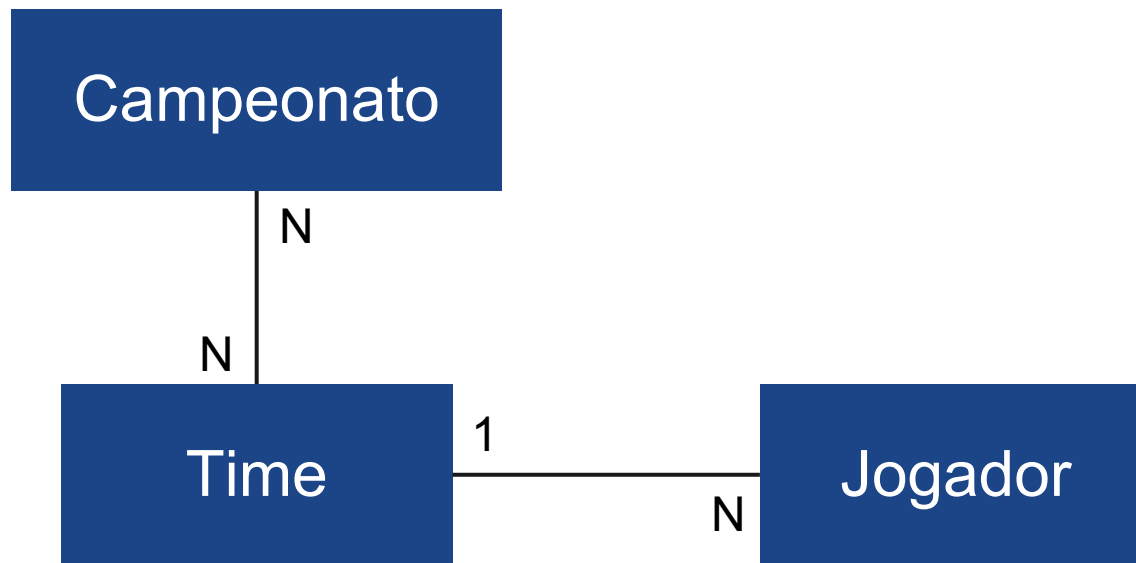
Design documents



Replicação



Exemplo SQL x CouchDB



Exemplo SQL x CouchDB

Time

```
{  
  "nome": "Avaí Futebol Clube",  
  "dataDeFundacao": "1923-09-01",  
  "estado": "SC",  
  "capacidadeDoEstadio": 17537,  
  "rankingNacional": 20,  
  "tipo": "time"  
}
```


Exemplo SQL x CouchDB

Exemplo 1

```
SELECT * FROM times
```

Exemplo SQL x CouchDB

```
SELECT * FROM times
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit(documento._id, documento);  
  }  
}
```

```
/futebol/_design/futebol/_view/times
```

Exemplo SQL x CouchDB

Exemplo 2

```
SELECT nome, dataDeFundacao  
FROM times
```

Exemplo SQL x CouchDB

```
SELECT nome, dataDeFundacao  
FROM times
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit(documento._id, [documento.nome, documento.dataDeFundacao]);  
  }  
}
```

/futebol/_design/futebol/_view/timesNomeDataDeFundacao

Exemplo SQL x CouchDB

Exemplo 3

```
SELECT *  
FROM times  
WHERE _id =  
"bb50e9d6ff29c2f1331c6e6fa101ba8f"
```

```
/futebol/bb50e9d6ff29c2f1331c6e6fa101a727
```

```
/futebol/_design/futebol/_view/times?key="  
bb50e9d6ff29c2f1331c6e6fa101a727"
```

Exemplo SQL x CouchDB

Exemplo 4

```
SELECT nome  
FROM times  
WHERE estado = "SC"
```

Exemplo SQL x CouchDB

```
SELECT nome  
FROM times  
WHERE estado = "SC"
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    if (documento.estado === "SC") {  
      emit(documento._id, [documento.nome]);  
    }  
  }  
}
```

/futebol/_design/futebol/_view/timesDoEstadoDeSantaCatarina

Exemplo SQL x CouchDB

```
SELECT nome  
FROM times  
WHERE estado = "SC"
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit(documento.estado, [documento.nome]);  
  }  
}
```

[/futebol/_design/futebol/_view/timesDoEstadoDeSantaCatarinaComIndiceEstado?key="SC"](/futebol/_design/futebol/_view/timesDoEstadoDeSantaCatarinaComIndiceEstado?key='SC')

Exemplo SQL x CouchDB

Exemplo 5

```
SELECT nome  
FROM times  
WHERE dataDeFundacao BETWEEN 1910 AND 1920
```

Exemplo SQL x CouchDB

```
SELECT nome  
FROM times  
WHERE dataDeFundacao BETWEEN 1910 AND 1920
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit(documento.dataDeFundacao, [documento.nome]);  
  }  
}
```

Exemplo SQL x CouchDB

Exemplo 6

```
SELECT nome, capacidadeDoEstadio  
FROM time  
ORDER BY capacidadeDoEstadio
```

Exemplo SQL x CouchDB

```
SELECT nome, capacidadeDoEstadio  
FROM time  
ORDER BY capacidadeDoEstadio
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    email(documento.capacidadeDoEstadio, [documento.nome, documento.capacidadeDoEstadio]);  
  }  
}
```

futebol/_design/futebol/_view/timesComIndiceCapacidadeDoEstadio

Exemplo SQL x CouchDB

Exemplo 7

```
SELECT nome, capacidadeDoEstadio  
FROM time  
ORDER BY capacidadeDoEstadio DESC
```

Exemplo SQL x CouchDB

```
SELECT nome, capacidadeDoEstadio  
FROM time  
ORDER BY capacidadeDoEstadio DESC
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    email(documento.capacidadeDoEstadio, [documento.nome, documento.capacidadeDoEstadio]);  
  }  
}
```

futebol/_design/futebol/_view/timesComIndiceCapacidadeDoEstadio?descending=true

Exemplo SQL x CouchDB

Exemplo 8

```
SELECT t.nome, j.nome  
FROM jogadores j JOIN times t  
ON j.identificadorDoTime = t._id
```

Exemplo SQL x CouchDB

Jogador

```
{  
  "nome": "Marcelo Toscano",  
  "posicao": "Atacante",  
  "identificadorDoTime":  
  "bb50e9d6ff29c2f1331c6e6fa1020373",  
  "tipo": "jogador"  
}
```


Exemplo SQL x CouchDB

```
SELECT t.nome, j.nome  
FROM jogadores j JOIN times t  
ON j.identificadorDoTime = t._id
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit([documento._id, 0], documento.nome);  
  } else if (documento.tipo === "jogador") {  
    emit([documento.identificadorDoTime, 1], documento.nome);  
  }  
}
```

futebol/_design/futebol/_view/timesJuncaoComJogadores

Exemplo SQL x CouchDB

Exemplo 9

```
SELECT t.nome, j.nome  
FROM jogadores j JOIN times t  
ON j.identificadorDoTime = t._id
```

Exemplo SQL x CouchDB

Time

```
{  
  "nome": "Fluminense Football Club",  
  ...  
  "identificadorDosJogadores": [  
    "4c19e4e54f081740444a3095ee003dd3",  
    "4c19e4e54f081740444a3095ee0049c9",  
    "4c19e4e54f081740444a3095ee003450"  
  ],  
  "tipo": "time"  
}
```

Exemplo SQL x CouchDB

```
SELECT t.nome, j.nome  
FROM jogadores j JOIN times t  
ON j.identificadorDoTime = t._id
```

```
function (documento) {  
  if (documento.tipo === "time" && documento.identificadorDosJogadores) {  
    var jogadores = documento.identificadorDosJogadores;  
    for (var indice in jogadores) {  
      emit([jogadores[indice], 0], documento.nome);  
    }  
  } else if (documento.tipo === "jogador") {  
    emit([documento._id, 1], documento.nome);  
  }  
}
```

Exemplo SQL x CouchDB

Exemplo 10

```
SELECT nome FROM times  
WHERE nome LIKE "%Atlético%"
```

Exemplo SQL x CouchDB

```
SELECT nome FROM times  
WHERE nome LIKE "%Atlético%"
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    var nomes = documento.nome.split(".");  
    for (var indice in nomes) {  
      emit(nomes[indice], documento.nome);  
    }  
  }  
}
```

futebol/_design/futebol/_view/timesComIndiceParteDoNome?key="Atlético"

Exemplo SQL x CouchDB

Exemplo 11

```
SELECT COUNT(*) FROM times
```

Exemplo SQL x CouchDB

SELECT COUNT(*) FROM times

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit(documento._id, null);  
  }  
}
```

```
function (chaves, valores, rereduce) {  
  if (rereduce) {  
    return sum(valores);  
  } else {  
    return valores.length;  
  }  
}
```

futebol/_design/futebol/_view/timesContagem

Exemplo SQL x CouchDB

Exemplo 12

```
SELECT COUNT(estado), estado  
FROM times GROUP BY estado
```

Exemplo SQL x CouchDB

```
SELECT COUNT(estado), estado  
FROM times GROUP BY estado
```

```
function (documento) {  
  if (documento.tipo === "time") {  
    emit(documento.estado, null);  
  }  
}  
  
function (chaves, valores, rereduce) {  
  return valores.length;  
}
```

futebol/_design/futebol/_view/timesContagemPorEstado?group=true

Exemplo SQL x CouchDB

Exemplo 13

```
SELECT MAX(capacidadeDoEstadio)  
FROM times
```

Exemplo SQL x CouchDB

```
SELECT MAX(capacidadeDoEstadio)
FROM times
```

```
function (documento) {
  if (documento.tipo === "time") {
    emit(documento._id, documento.capacidadeDoEstadio);
  }
}

function (chaves, valores, rereduce) {
  var maximo = -Infinity;
  for (var indice in valores) {
    var valor = valores[indice];
    if (valor > maximo) {
      maximo = valor;
    }
  }
  return maximo;
}
```

```
futebol/_design/futebol/_view/timesMaiorCapacidadeDoEstadio
```

Conclusão

BD Relacional

- Especificação bem definida;
- Consistência de dados.

BD NoSQL

- Flexibilidade;
- BDs distribuídos;
- Big Data.