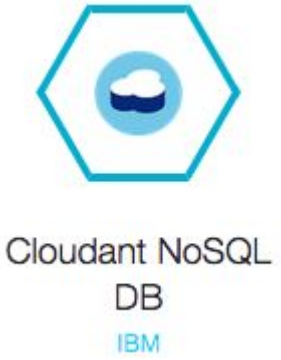
# Cloudant Hands On

Fernando Cardoso Durier da Silva

# Tópicos:




1- NoSQL(o que é?, NoSQL x SQL, exemplos);

2- Criando um banco no Cloudant;

3- A query no Cloudant;

4- Construindo search indexes;

5- Como se conectar ao banco e manipulação de dados no backend;

6- Privilégios no Cloudant;

# 1-SQL e Bancos de Dados Relacionais:

- Características:Os bancos armazenam seus dados de forma interligada entre eles. Em tabelas, formalmente em relações. Todas operações entre os dados é fechada, ou seja, o resultado de toda e qualquer operação é um relacionamento, ou não gera um retorno sendo assim um relacionamento vazio (conjunto vazio, como em procedures comuns);
- Tipos: Orientados a objetos, Organizados por Colunas, Organizados por Linhas, Super enxutos e etc...
- Propriedades:
  Atomicidade (a manipulação dos dados não pode ser pausada muito menos perdida, ou seja, ou todas as manipulações são executadas ou serão descartadas),
  Consistência (antes de qualquer transação o banco estará consistente para todos os usuários ao mesmo tempo e depois da transação assim o ficará, estruturalmente),
  Isolamento (nenhuma transação interfere no funcionamento de outra antes desta estar completa)
  e Durabilidade (os dados que estão no banco ficarão imutáveis até que uma transação aconteça e os modifique);


dashDB

# 1-SQL e Bancos de Dados Relacionais:



- Vantagens: Corretude rígida de dados, Fácil de montar análises uma vez que o banco tem operações pré-prontas para tal e os dados estão em estruturas relacionais, operações de CRUD “nativas”,  armazena relacionamentos, Robusto, Várias camadas de segurança e proteção de dados;
- Desvantagens: Difícil de escalar, é necessário um ambiente monitorado, seguro e disponível para hospedar o banco e inflexível a mudanças bruscas;
- Exemplos do uso de bancos relacionais, atualmente:
- Sistemas bancários;
- Repositórios de artigos científicos;
- Sistemas de vendas;
- Sistemas ERP;
- Aplicações que consomem grande quantidades de dados (BIG DATA related);

# 1-NoSQL e seus bancos:


- Características: São uma nova proposta de armazenar dados. Não devem jamais ser encarados como os substitutos dos bancos relacionais convencionais. São bancos que armazenam dados de forma não relacional de forma flexível e escalável.
- Tipos: Orientados a documentos, Tabelão, Orientados a Grafos, Orientados a colunas, Orientados a chave-valor, gráficos, hiperdimensionais e etc...
- Propriedades: Alta tolerância ao particionamento, consistência e durabilidade altamente questionáveis, muito práticos, não requerem um servidor muito elaborado para serem usados, normalmente, estando na nuvem, flexíveis aos tipos de dados armazenados por eles, não fazem bloqueio de recursos, ou seja, não importa quantas pessoas mexam no mesmo dado ele estará sempre disponível e com valores diferentes e versionados e etc ...

# 1-NoSQL e seus Bancos:


- Vantagens: Mais flexíveis aos tipos de dados armazenados, mais “disponíveis”, altamente escaláveis, conectáveis a quase tudo eletrônico, dependendo do banco muito bons para análises;
- Desvantagens: Consistência frágil, Necessidade de implementar funções e operações já nativas dos bancos relacionais convencionais, a aplicação é responsável por garantir consistência, durabilidade questionável.
- Exemplos do uso de bancos não relacionais, atualmente:
- A maioria das redes sociais, blogs e chats usam bancos NoSQL para armazenar posts, conversas e etc...
- Sensoriamento e IoT usufruem do poder de escalonamento e flexibilidade de dados;
- Muito bom para armazenar dados não estruturados e em grandes quantidades (BIG DATA);

# 2-Cloudant DBAAS:


- É um serviço de banco de dados não relacional, que administra JSONs de forma flexível garantindo o fluxo contínuo e ininterrupto de dados entre a aplicação e a base de dados;
- Hospedado na nuvem, mas, disponível também no Bluemix;
- Maior foco no desenvolvimento de aplicações que manipulem seus dados e não na administração do banco propriamente;
- Link: https://www.ibm.com/us-en/marketplace/database-management?&S_TACT=000000VQ&S_OFF_CD=10000738
- Funcionalidades:

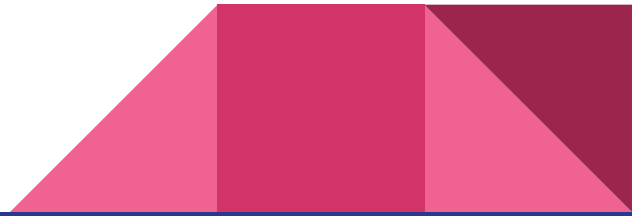
{

Disponibilidade global;
Flexibilidade de dados com JSON;
Bibliotecas de sincronismo mobile para armazenamento local mobile;
APIs de manipulação e criação de indexes geo espaciais;
Integração com outros bancos de dados até mesmo os relacionais;

}

# 2-Criando Primeiro Banco no Cloudant:

- Efetuar Login no dashboard online do Cloudant;
- No canto superior direito clicar em create database;
- Digitar o nome da database. O Cloudant só aceita como nome válido caracteres minúsculos não acentuados, números, ponto e underline;
- Depois de um tempinho a database é criada, por default, com 3 shards por doc e com privilégios para outros como _read only;



# 2-Dados do Cloudant:

- Assim como o couchDB (“matriz/pai”) o Cloudant é um banco de dados não relacional orientado a documentos;
- Armazenamento em JSONs facilitando manipulação pela rede e flexibilidade com relação aos dados;
- ex.:
- {“_id”:<valor alpha numérico auto incremental controlado pelo motor de database do cloudant>,
  “_rev”:<versão do doc, por default o cloudant armazena até 1000 revs por doc>,
  “attr1”:<valor>,
  “attr2”:{JSON},...,
  “attrN”:[<valor>,...,<valor>]}

# 3-Query no cloudant:

- As queries no cloudant são normalmente RESTful APIs de POST, fornecidas pelo próprio Cloudant, que recebem JSONs para filtrar resultados;

ex.:{

"selector": { "<doc.Attribute>": <possible Attribute value>/{ <logical operator>: <adequate value for the operator>} },

"fields": [<doc.Attribute1>, … , <doc.AttributeN> ],

"sort": [ { "<pre indexed doc.Attribute>": <asc or desc> } ]

}

# 4-Search Index:

- Os docs do cloudant são sempre indexados para serem pesquisados/requisitados. Por default, eles são indexados por seus atributos, mas, podemos criar novos indexes para agilizar consultas, adaptar resultados e montar counts ou grupamentos;
- Esses indexes customizáveis são armazenados em um tipo de documento chamado _design_document. Este é uma função javascript interpretada pelo motor de busca do Cloudant que faz uma indexação no seu dataset;
- ex.:

```
function(doc){
        if(doc.CITY){
            index(“nome do index”,doc.CITY,{store:true,facet:true});
        } //store:true -> the doc will be fully indexed
}
//facet:true -> the indexed attribute will be possible to count
```

# 4-Consultando o seu Search Index:

- https://<url das credentials do cloudant>/<database>/_design/<nome do design doc>/_search/<nome do searchIndex>?q=<campo indexado>:<possível valor>&include_docs=true;
- ex.1: https://efa69e26-50bf-4031-9047-335b836c1494-bluemix.cloudant.com/handson/_design/math/_search/math?q=*:*&integer1$gt:0&include_docs=true
- ex.2: https://efa69e26-50bf-4031-9047-335b836c1494-bluemix.cloudant.com/handson/_design/handson/_search/handson?q=city:%22RIO%20DE%20JANEIRO%22

# 4-Indexações Especiais:

- Podemos entender como uma especialização da função de searchIndex;
  - ex.:

```
function(doc){
    if(doc.Atrribute1 >= doc.Atrribute2 && doc.Attribute1%2 == 0){
        index(“specialIndex”,”{
            Attribute1:“+doc.Attribute1+
            “Attribute2:”+doc.Attribute2+
            “Calculation”: +(doc.Attribute1+doc.Attribute2)+
        ”}”,{store:true,facet:true});
    }
}
```

# 5-Node.js e Cloudant:

1. Node.js + Express.js + npm request + npm q = bom consumo e oferecimento de APIs RESTful;
2. Cloudant = Tudo acessível por RESTful APIs + Indexação com funções em Javascript + Um banco de JSONs praticamente;
3. Bluemix = contém Node.js e Cloudant para hospedar e vincular o service, respectivamente;
4. 1,2,3 = Um excelente backend para se trabalhar;

# 5-Requests:

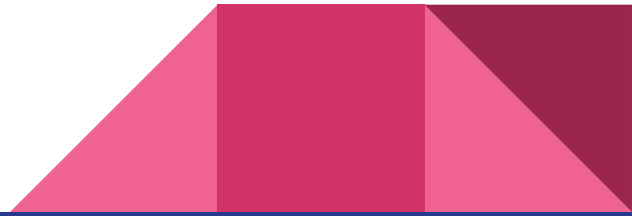
1. npm install request --save;
2. Usando requests para manipular docs e acessar o cloudant ;

```
var getFull = function(){  //exemplo de select all docs from database
    var getDefer = Q.defer();
    request({
            url:  <cloudantURL> + "/" + <databaseName> + "/_all_docs?include_docs=true",
            method: 'GET'
            }, function(error, response, body){
                    if(error) {
                            console.log(error);
                            getDefer.reject(error);
                    }
                    else {
                            getDefer.resolve(JSON.parse(body).rows );
                    }
            });
            return getDefer.promise;
};
```

3. Mais Exemplos no github: https://github.com/FernandoDurier/cloudantHandsOn.git

# 5-CRUD:

- Creating docs(POST);
- Reading docs(GET ou em casos raros POST);
- Updating docs(PUT ou em casos muito raros GET->POST->DELETE);
- Destroying docs(Delete);
- Mass insert (POST);
- Mass delete (POST);



# 6-Privilégios do Cloudant:

- Privilégios pela U.I. do site;
- Privilégio por API key;
- Tipos de privilégios:[_read,_write,_admin,_create];

| Permissões e Métodos | _read | _write | _admin | _create |
|---|---|---|---|---|
| GET | x | - | x | - |
| POST | - | x | x | x |
| PUT | - | - | x | x |
| DELETE | - | - | x | - |

# Referências:


- https://docs.cloudant.com/advanced.html
- https://docs.cloudant.com/index.html
- http://nosql-database.org/
- http://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044
- https://aws.amazon.com/pt/nosql/
- https://pt.wikipedia.org/wiki/NoSQL
- https://www.npmjs.com/package/request
- https://github.com/FernandoDurier/cloudantHandsOn.git
- https://www.npmjs.com/package/q