FUNDAMENTOS E TÉCNICAS EM CIÊNCIAS DE DADOS

PROF. JOSENALDE OLIVEIRA

josenalde@eaj.ufrn.br
https://github.com/josenalde/datascience

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

NOSQL - DOCUMENTOS

1) Também armazenam um KVS, mas em conjuntos permitindo o armazenamento de estruturas como um JSON
   1) JavaScript Object Notation: *formato compacto, padrão aberto, para troca de dados simples entre sistemas*
      1) *É completamente independente de linguagem! – Douglas Crockford (1996, a partir de solução Netscape)*
   2) Cada objeto (**documento**) fica armazenado numa coleção específica, mas mesmo dentro de uma coleção não há um esquema fixo para os registros; não há **schema** (esquema)

```
{
    "país": "Brasil",
    "população": 206081432,
    "PIB total em trilhões de dólares":  3.101,
    "faz fronteira com": [
         "Argentina",
         "Bolívia",
         "Colômbia",
         "Guiana Francesa",
         "Guiana",
         "Paraguai",
         "Peru",
         "Suriname",
         "Uruguai",
         "Venezuela"
    ],
    "cidades": {
       "capital" : "Brasília",
       "mais populosa": "São Paulo"
    }
}
```

JSON

Relacional | NoSQL-Documento

Termos/conceitos do SQL	Termos/conceitos do MongoDB
Database	Database
Tabela	Coleção
Linha	Documento ou documento BSON
Coluna	Campo
Index	Index
Table join	Documentos aninhado (*embedded*) e vinculados
Chave primária — especifica qualquer coluna única ou uma combinação de colunas como chave primária	Chave primária — No MongoDB, a chave primária é automaticamente definida como campo *_id*
Agregação (*group by*)	Agregação de pipeline

NOSQL - DOCUMENTOS

1) Simplicidade para transferência de informações. Tipos da dados JSON
2) Elemento sempre começa com chaves, e conjunto de elementos com colchetes

Tipo	Descrição
Null	Valor vazio
Boolean	true ou false
Number	Número com sinal (inclui notação com E exponencial)
String	Sequência de caracteres Unicode
Object	Array não ordenado com itens chave-valor Chaves são strings distintas no mesmo objeto
Array	Lista ordenada de qualquer tipo, inteira entre colchetes e com cada elemento separado por vírgulas

```
[
{
    "país": "Brasil",
    "população": 206081432
},
{
    "país": "Argentina",
    "população": 41281631
},
{
    "país": "Bolívia",
    "população": 10426160
}
]
```

3) JSON não padroniza tipo data, nem dados binários
4) BSON (JSON binário – variante usada no mongoDB. Ao persistir dados no mongoDB, este é o formato interno. BSON adiciona os seguintes tipos:

1. MinKey, MaxKey, Timestamp — tipos utilizados internamente no MongoDB;
2. BinData — array de bytes para dados binários;
3. ObjectId — identificador único de um registro do MongoDB;
4. Date — representação de data;
5. Expressões regulares.

```
{
"_id" : ObjectId("57e08da696535fff4a345c67"),
"timestamp" : Timestamp(1474334118, 1),
"data" : ISODate("2016-09-20T01:16:41.720Z"
}
```

Timestamp: milissegundos desde 1.Jan.1970 (Unix Epoch) – 64 bits

<YYYY-mm-ddTHH:MM:ssZ>

UTC-3 (Brasília)

NOSQL - DOCUMENTOS

1) Uma nota fiscal em XML pode ser convertida para modelo relacional, mas seria decomposto em dezenas de tabelas, com chaves primárias para relacionar todas estas tabelas – custo para recuperar, processar, armazenar
2) Num banco de dados de documentos, um XML tem sua estrutura naturalmente armazenada

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
  </instrucoes>
</receita>
```

XML

Dados ESTRUTURADOS: esquema fixo, normalmente tabular (planilhas, tabelas de BD)

Não-ESTRUTURADOS: sem estrutura definida e mesmo metadados podem não ser úteis para análise (texto geral, páginas web, e-mails, postagens redes sociais, imagens, áudio, vídeo) – mais comum e que mais cresce

SEMI-ESTRUTURADOS: existe estrutura, mas não é fixa (XML, JSON, RDF, OWL)

```json
{
  "receita": {
    "-tempo_de_cozimento": "1 hora",
    "-tempo_de_preparo": "5 minutos",
    "-nome": "pão",
    "titulo": "Pão simples",
    "ingredientes": {
      "ingrediente": [
        {
          "-unidade": "xícaras",
          "-quantidade": "3",
          "#text": "Farinha"
        },
        {
          "-unidade": "gramas",
          "-quantidade": "7",
          "#text": "Fermento"
        },
        {
          "-unidade": "xícaras",
          "-quantidade": "1.5",
          "-estado": "morna",
          "#text": "Água"
        },
        {
          "-unidade": "colheres de chá",
          "-quantidade": "1",
          "#text": "Sal"
        }
      ]
    },
    "instrucoes": {
      "passo": [
        "Misture todos os ingredientes, e dissolva bem.",
        "Cubra com um pano e deixe por uma hora em um local morno.",
        "Misture novamente, coloque numa bandeja e asse num forno."
      ]
    }
  }
}
```

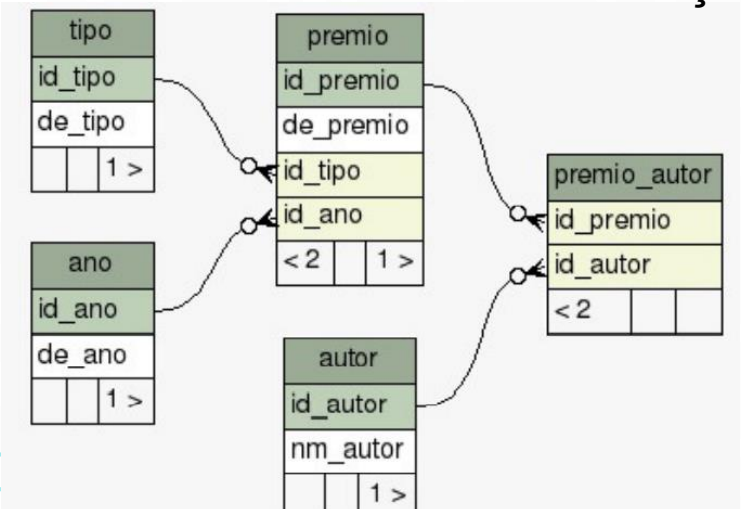
JSON

NOSQL - DOCUMENTOS

Mais um pouco de motivação

Suponha a página da WikiPedia do IgNobel:, onde são identificados o ano, tipo, autor/ganhador e descrição do prêmio

No relacional, NORMALIZADO, 5 tabelas – dados organizados, mas levar em conta a APLICAÇÃO – como ocorrem as consultas




```
select
p.de_premio, t.de_tipo, a.de_ano ,au.nm_autor
from premio p, tipo t, ano a, premio_autor pa, autor au
where p.id_premio = pa.id_premio
and p.id_tipo = t.id_tipo
and p.id_ano = a.id_ano
and pa.id_autor = au.id_autor
```

Concordamos que um acesso ao site, se o dado que lá estivesse fosse exibido de modo relacional, pela quantidade de acessos, teria problemas de performance!

Como a página exibe tudo de uma vez, faz sentido toda a informação estar concentrada e não distribuída em tabelas! Ou seja, é preciso que os dados estejam DESNORMALIZADOS!

Aplicação obedece regras do BD, ou o BD responde às demandas da aplicação?

Que tal esta solução?

```
{
    "ano" : 1992,
    "tipo" : "Medicina",
    "autores" : [
                 "F. Kanda",
                 "E. Yagi",
                 "M. Fukuda",
                 "K. Nakajima",
                 "T. Ohta",
                 "O. Nakata"],
    "premio" : "Elucidação dos Componentes Químicos Responsáveis
                pelo Chulé do Pé (Elucidation of Chemical
                Compounds Responsible for Foot Malodour),
                especialmente pela conclusão de que as pessoas
                que pensam que têm chulé, têm, e as que pensam
                que não têm, não têm."
}
```

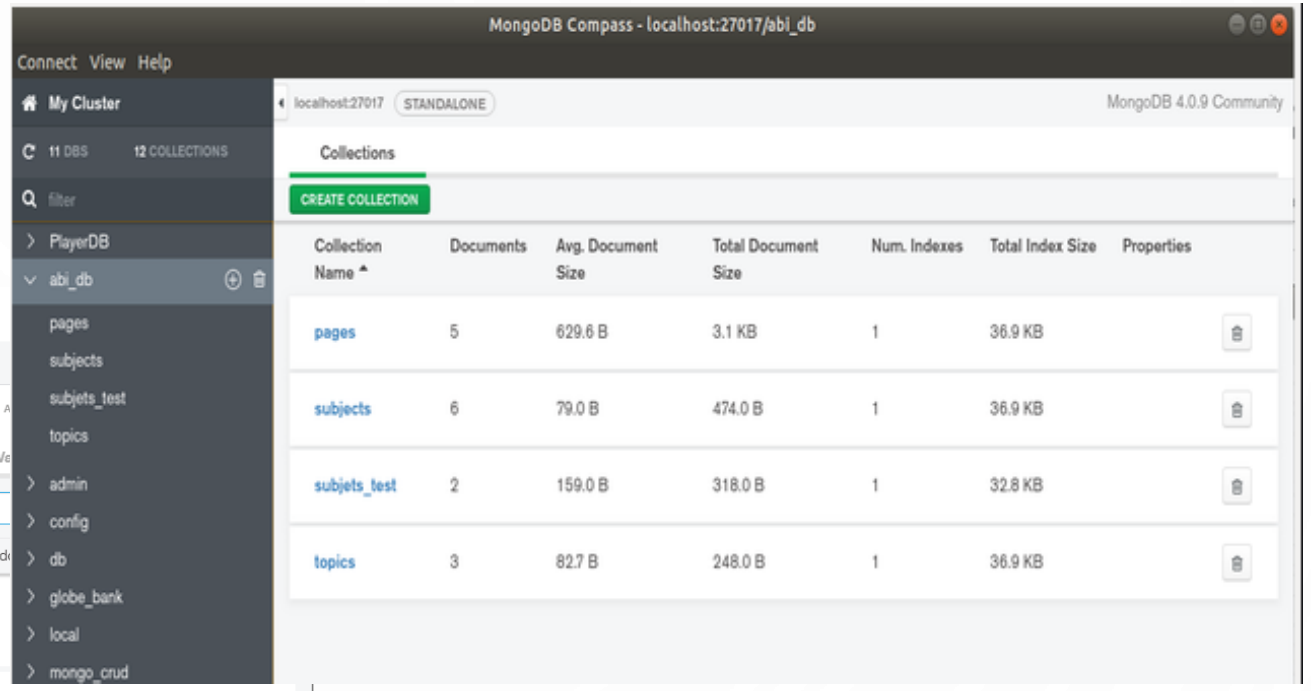
De 5 tabelas para 1 coleção

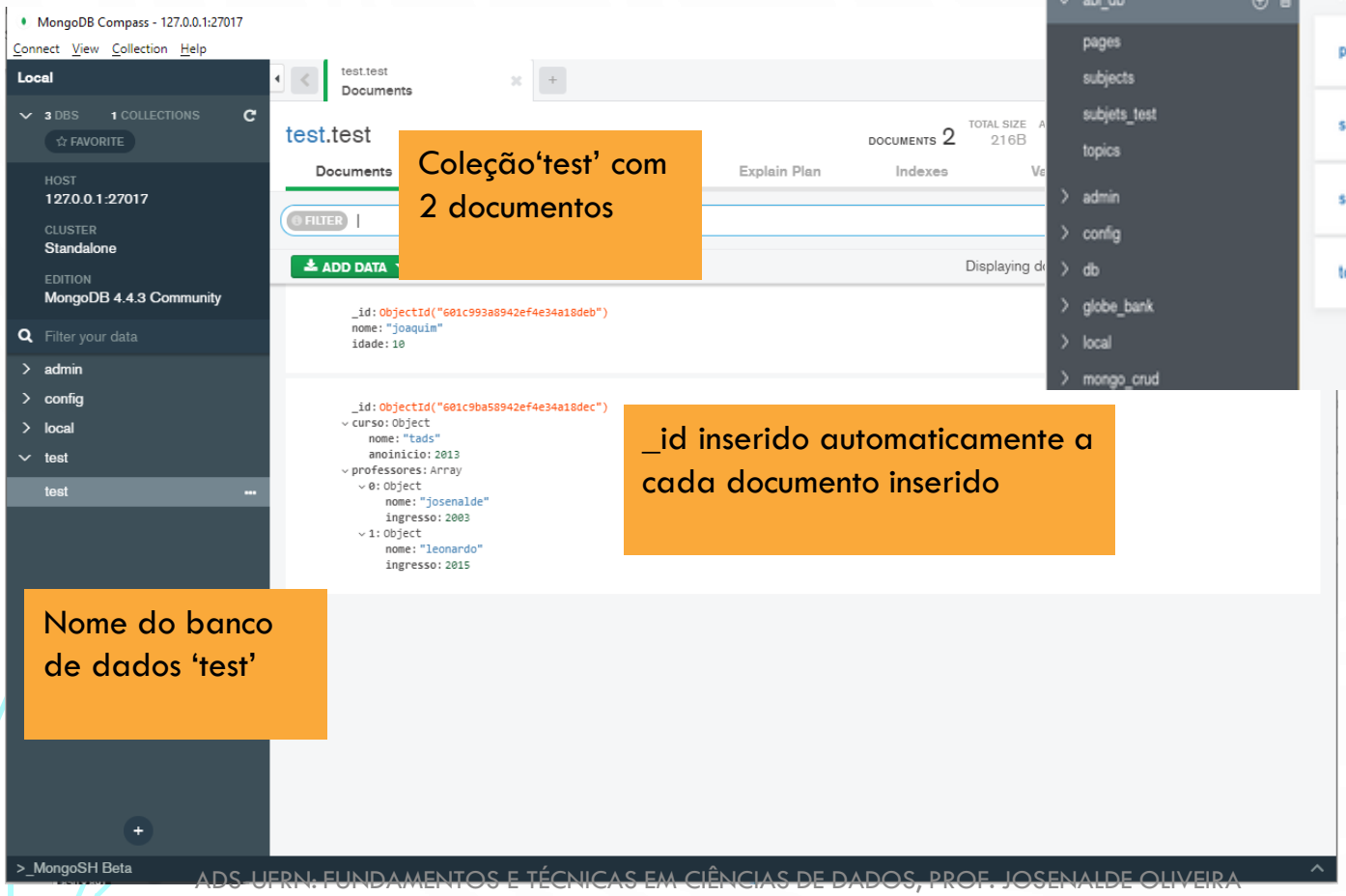
Exercício: MEGASENA

NOSQL - DOCUMENTOS

Compass

https://jsonformatter.curiousconcept.com/#





Coleção‘test’ com 2 documentos

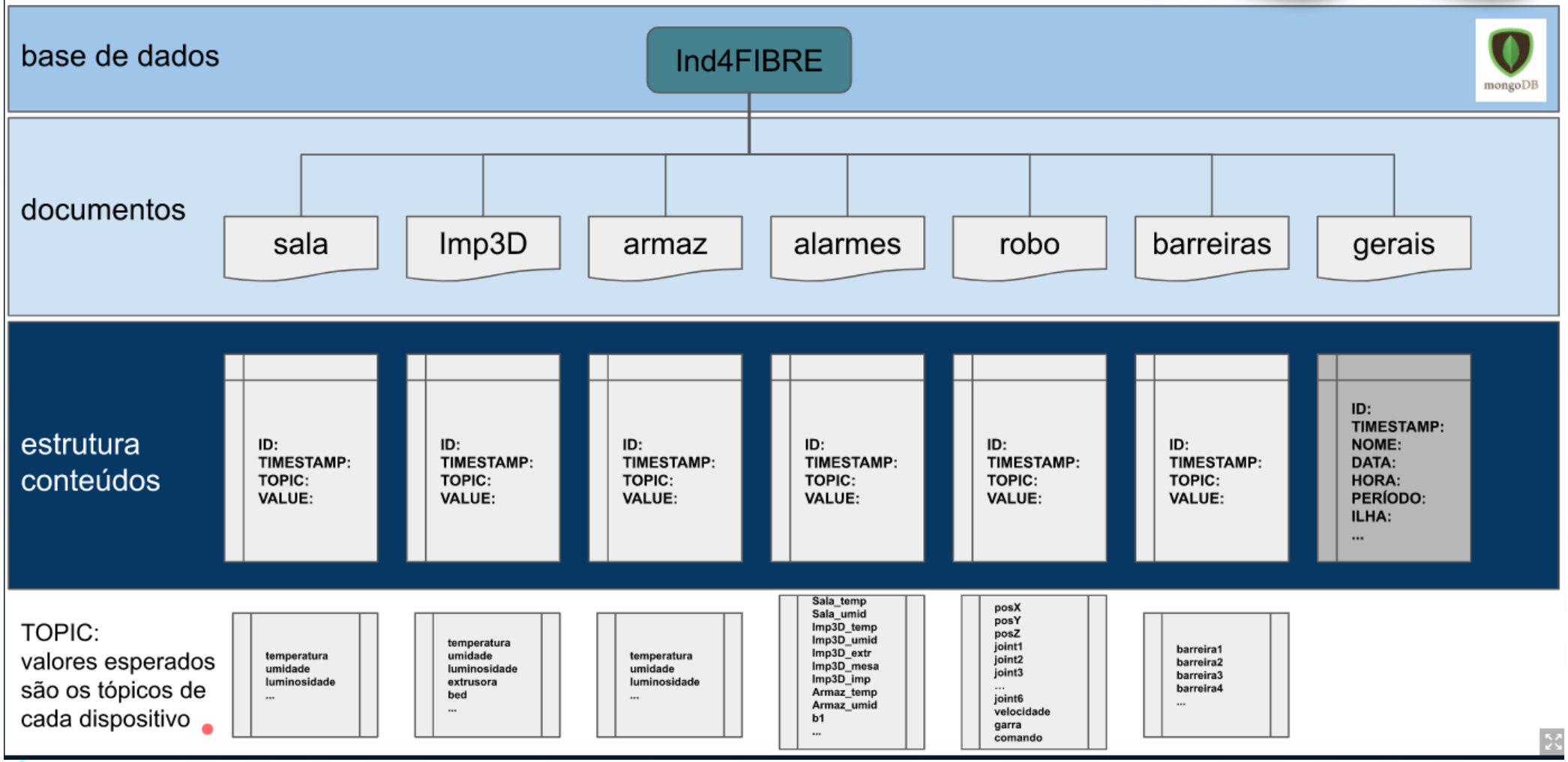
_id inserido automaticamente a cada documento inserido

Nome do banco de dados ‘test’

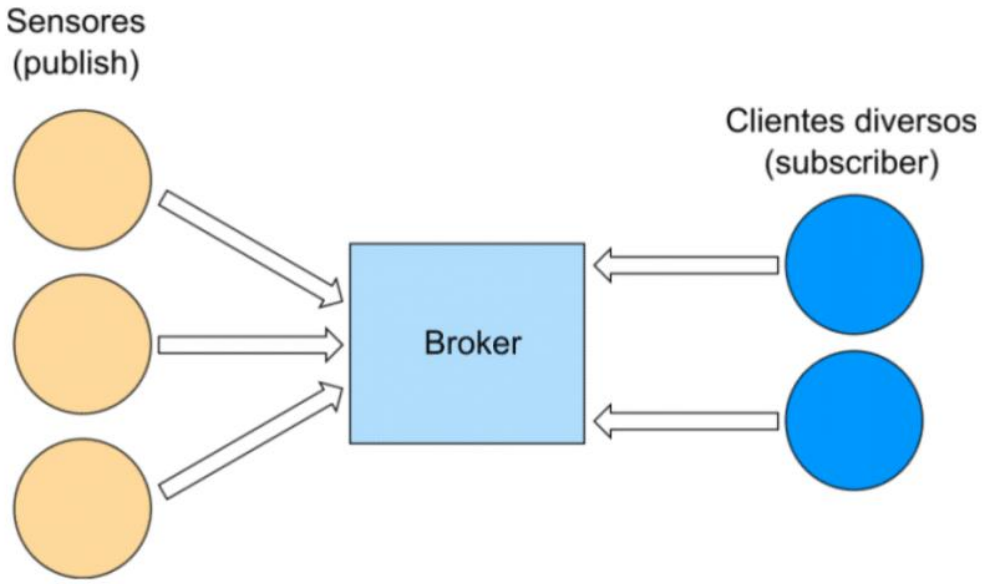
NOSQL – DOCUMENTOS - EXEMPLO

- Banco de dados local Mongodb
  - esquema

Modelo orientado a documentos

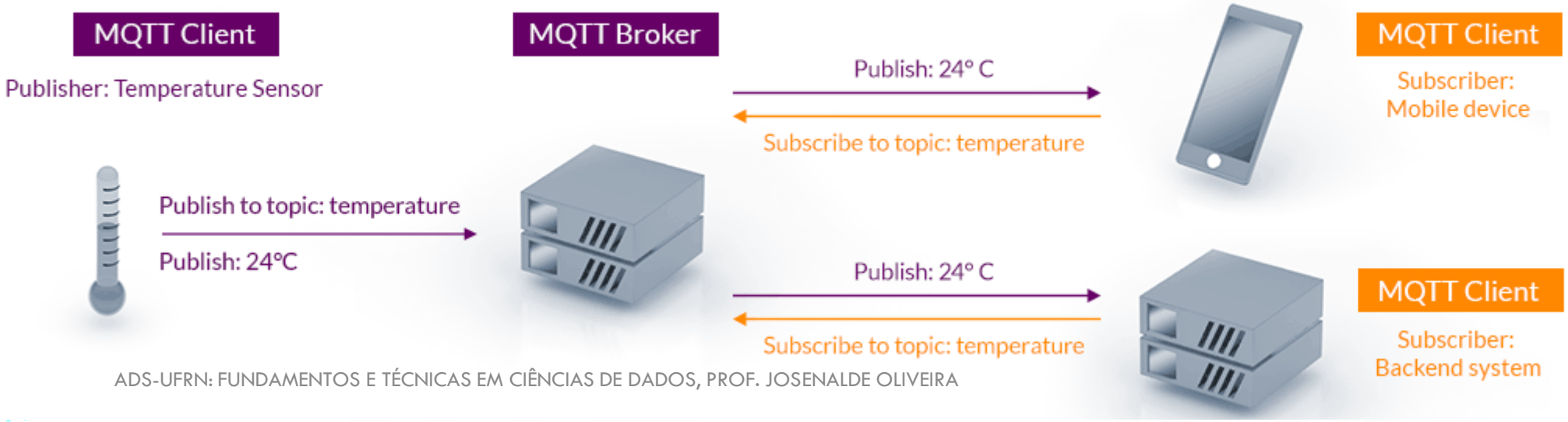


NOSQL - DOCUMENTOS

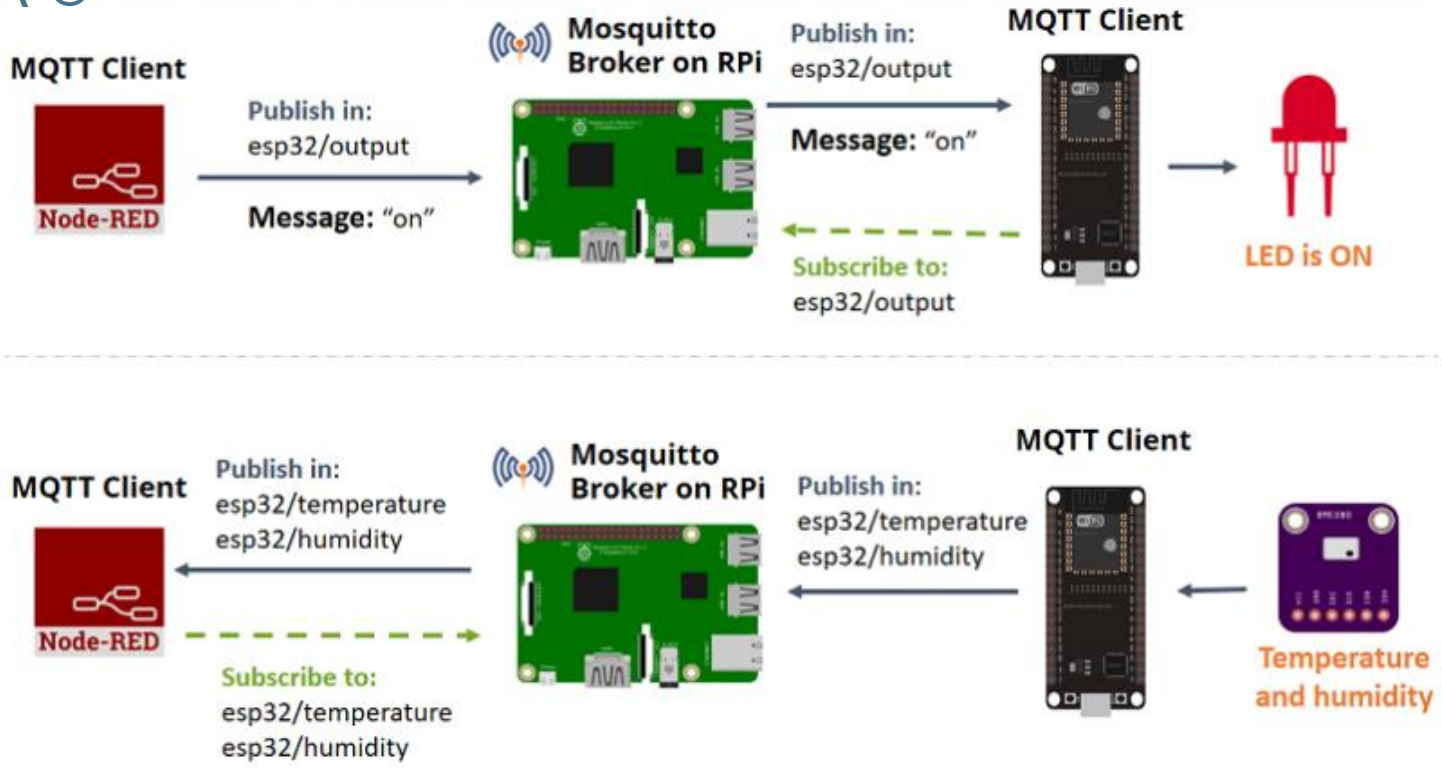


Exemplo de aplicação IoT: mensagens transmitidas via MQTT
Mensagens podem ser encapsuladas em estrutura JSON (serialização) para transmissão e para persistência em nosql e recuperadas para exibição: https://mqtt.org/
Broker: **mosquito (comum raspberry como broker)**

Normalmente envia msg para clientes
Pode persistir a sessão do cliente em memória ou em disco mosquitto.db



NOSQL – DOCUMENTOS - EXEMPLO



1) O LED está ligado ao uC em pino digital e seu estado (ON, OFF) é controlado por aplicação cliente que publica no tópico esp32/output no broker. O uC assina este tópico no broker, recebe a msg e liga o led

2) O uC possui sensor de temperatura e umidade conectado fisicamente e publica estas leituras nos tópicos esp32/temperatura e esp32/umidade. Estes tópicos são assinados Pela aplicação cliente, e o broker devolve ao solicitante para exibição em tela, dashboard, etc.

NOSQL – RESUMO

1) NoSQL é um termo técnico para denominar um banco de dados que não é relacional. Normalmente, ele é do tipo banco de dados de documento, orientado a objetos, chave-valor ou de grafos
2) De onde veio o termo mongoDB: O nome veio da palavra humongous, que significa enorme, gigantesco, para dar a ideia de grande gerenciamento de dados.
3) Quem usa mongoDB: Our Customers | MongoDB
4) Não substitui um banco relacional, pois não possui transação ou constraints de referência, que quase todo sistema possui, mas pode ser um complemento de uma base relacional, servindo como cache, por exemplo. Entretanto, se sua aplicação for desenhada adequadamente, ela pode usar inteiramente o NoSQL e não usar nenhuma base relacional.
5) O MongoDB cria um índice para cada collection, mas é esperado que validações nos campos aconteçam na aplicação. Existe um tipo bem simples de validação, que verifica se um campo é obrigatório ou se obedece a uma expressão regular. É possível criar índices simples e compostos (mais de um campo), inclusive para arrays. Existe também o poderoso índice de busca textual (full text search).
6) Opera em cluster
7) O limite é de 16Mb de tamanho máximo, permitindo ter até 100 níveis de documentos aninhados. Para ter um comparativo, existem algumas versões da Bíblia em formato texto na internet, que ocupam aproximandamente 4mb. Portanto, para ultrapassar o limite atual do MongoDB, um simples registro/ documento precisa ter mais texto do que quatro bíblias completas juntas. Os nomes de campos, collections e databases podem ter até 123 bytes. Uma collection pode ter até 64 índices, cada um deles contendo entre 1 e 31 campos. O tamanho máximo do banco de dados pode variar conforme o tipo de file system e o sistema operacional. Porém, grosso modo, são 4 terabytes para Windows e 54 para Linux. Consulte os limites restantes na documentação oficial http://docs.mongodb.org/manual/reference/limits/