DEVCON
EUROPEAN DELPHI CONFERENCE
Spring EDITION

"Sopra tutto, FerretDB"

Il potere NoSQL di MongoDB nelle mani degli sviluppatori Delphi

Marco Breveglieri

👨‍💻 Software Developer | 👩‍🏫 Trainer & Consultant | 🎥 Tech Content Creator

ROMA 2024

Marco Breveglieri

Software Developer | Trainer & Consultant | Tech Content Creator

@ABLS Team – Reggio Emilia (Italy)

Homepage
https://www.breveglieri.it

Blog tecnico
https://www.compilaquindiva.com

Delphi Podcast
https://www.delphipodcast.com

Twitch
https://www.twitch.tv/compilaquindiva

YouTube
https://www.youtube.com/@compilaquindiva

Marco Breveglieri

👨‍💻 Software Developer | 👩‍🏫 Trainer & Consultant | 🎥 Tech Content Creator

@ABLS Team – Reggio Emilia (Italy)

Collegati su

Homepage 👉 https://www.breveglieri.it
Blog 👉 https://www.compilaquindiva.com
Podcast 👉 https://www.delphipodcast.com
Twitch 👉 https://www.twitch.tv/compilaquindiva
YouTube 👉 https://www.youtube.com/@compilaquindiva
LinkedIn 👉 https://www.linkedin.com/in/marcobreveglieri
GitHub 👉 https://github.com/marcobreveglieri
Telegram 👉 https://t.me/compilaquindiva
Facebook 👉 https://www.facebook.com/compilaquindiva
Twitter 👉 https://x.com/mbreveglieri
Threads 👉 https://www.threads.net/@compilaquindiva
Mastodon 👉 https://mastodon.uno/@compilaquindiva

Supportami su

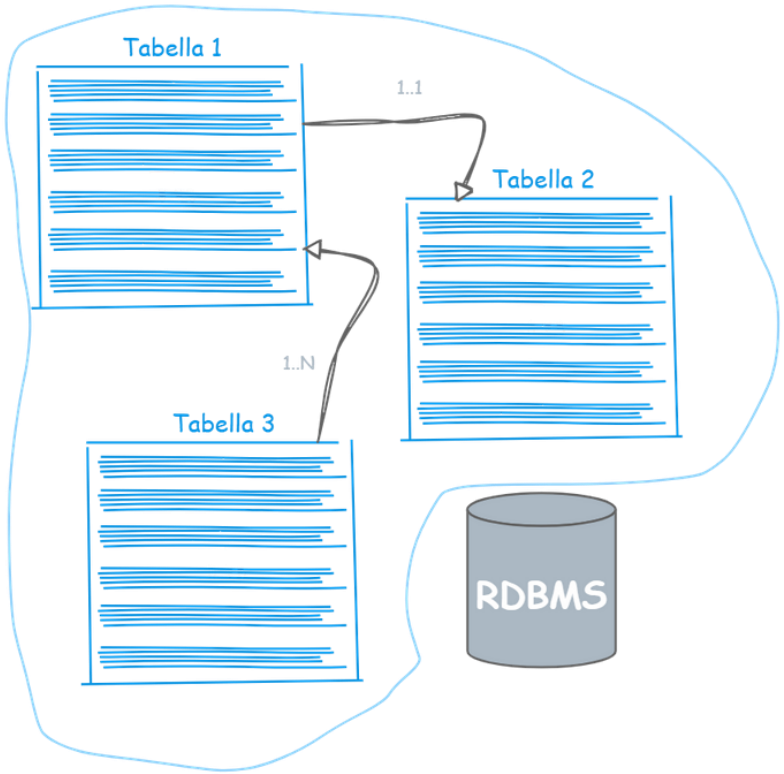
Buy Me a Coffee 👉 https://www.buymeacoffee.com/brevve
Ko-Fi 👉 https://ko-fi.com/compilaquindiva
Patreon 👉 https://www.patreon.com/compilaquindiva
PayPal 👉 https://paypal.me/marcobreveglieri

SQL… o NoSQL?

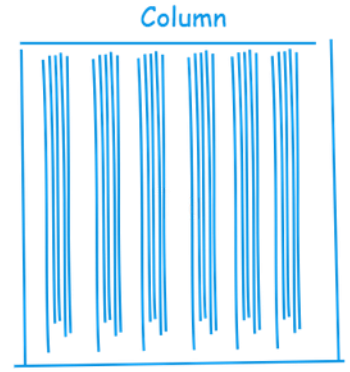
…questo è il dilemma!

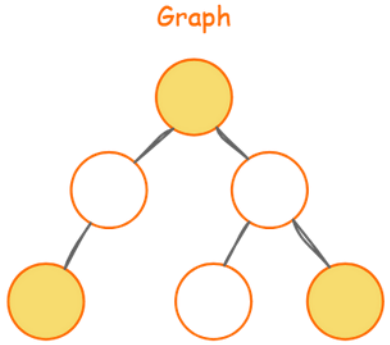
SQL vs NoSQL a confronto

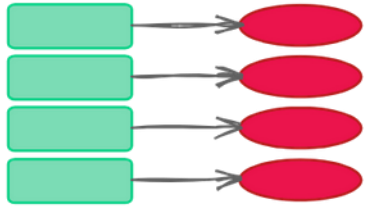
SQL



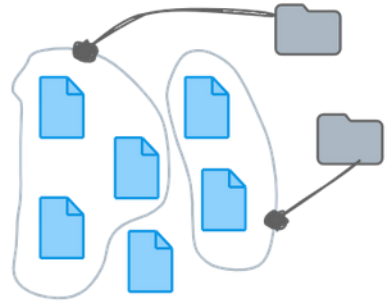
NoSQL







Key Value



Document

SQL vs NoSQL: le differenze

SQL

- Sfruttano il linguaggio SQL
- Basati su tabelle, colonne e righe
- Schema rigido e predeterminato
- Sfruttano le transazioni (ACID)
- Supportano query basate su join
- Devono essere normalizzati
- Scalano verticalmente

NoSQL

- Usano linguaggi «non solo» SQL
- Basati su collezioni e documenti
- Schema flessibile e destrutturato
- Uso di transazioni non richiesto
- Non richiedono join complessi
- Usano forme «denormalizzate»
- Scalano orizzontalmente

SQL vs NoSQL: scenari d'uso

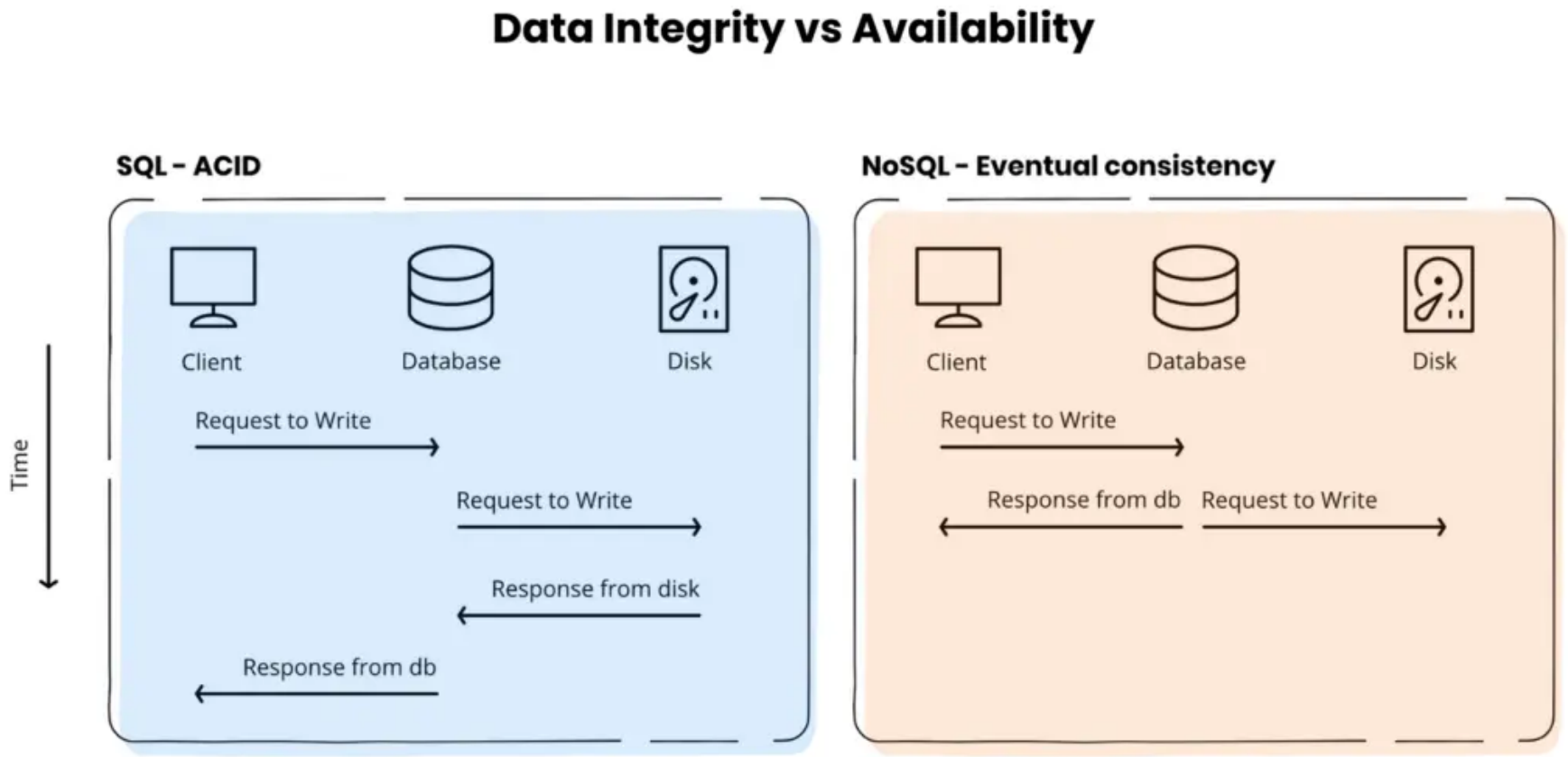
SQL

- Dati estremamente strutturati
- Integrità dei dati essenziale
- Riduzione di dati duplicati
- Accesso basato su ORM
- Schema rigido e creato up-front
- Quantità di dati limitata

NoSQL

- Dati in formato variabile
- Scalabilità dei dati essenziale
- Tolleranza a duplicazione dei dati
- Mappatura diretta su oggetti
- Schema flessibile e dinamico
- Scenario che coinvolge *Big Data*

SQL vs NoSQL: un esempio

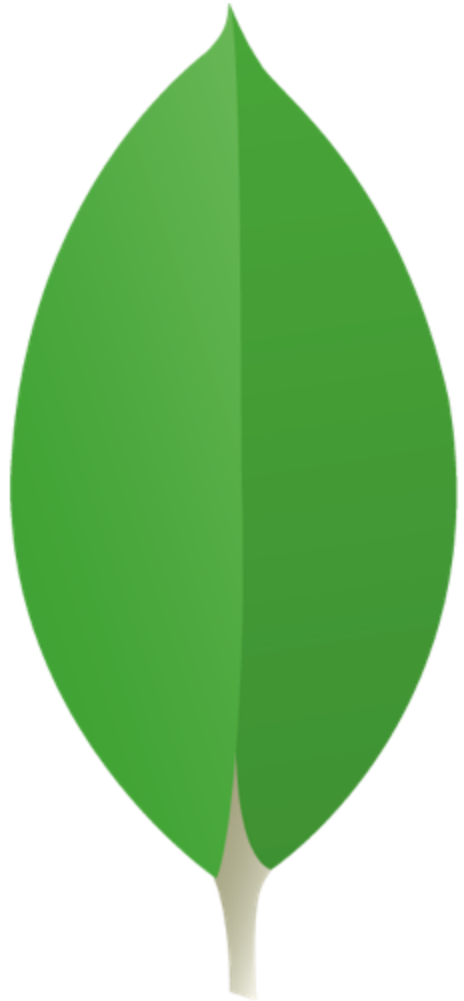


https://www.scalablepath.com/back-end/sql-vs-nosql

SQL vs NoSQL: qualche nome...

Conosciamo MongoDB

Un recap sul database NoSQL più popolare e diffuso



MongoDB in brevissimo

MongoDB è un database *«document-oriented»* di tipo NoSQL.

- **Document**: è il singolo documento che contiene i dati
  - *Struttura formata da coppie chiave/valor*
  - *E' rappresentato da un oggetto simil-JSON*
  - *Viene salvato su disco in formato BSON*
- **Collection**: è un contenitore di uno o più documenti
  - *Simile (in un certo senso) alla tabella di un classico RDBMS*
  - *Ha una struttura schema-less (contiene oggetti di qualsiasi tipo)*
- **Database**: ciascun database contiene più collection

~~Viene~~ distribuito con licenza *open-source*.

Esempi di documento

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
            level: 5,
            group: "dev"
          }
}
```

Embedded sub-document

Embedded sub-document

user document

```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

access document

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

Punti di forza 💪

- E’ un database di tipo «document-oriented» molto accessoriato
- Ha uno *Schema* aperto e flessibile (e pure validabile)
- Gestisce indici per velocizzare ricerche e aggiornamenti
- Consente interrogazioni tramite un *Query Language*
- Adotta un *Aggregation Framework* per le operazioni più complesse
- Sono disponibili tantissimi *Driver* per svariati linguaggi (e Delphi?)
- Supportato da tantissimi anni e molto apprezzato dalla community

Demo time!

Cambio di licensing

<u>Prima</u> del 2018

AGPL

GNU Affero General Public License

⇩

- **Se modifichi il codice di MongoDB**, devi rilasciare pubblicamente il codice modificato.

<u>Dopo</u> il 2018

SSPL

Server Side Public License

⇩

- **Se esegui MongoDB come servizio**, devi rilasciare la parte di codice che usa MongoDB come servizio.

Welcome FerretDB!

Esploriamo una valida alternativa a MongoDB compatibile con… MongoDB. 😁

FerretDB: una vera alternativa a MongoDB

- E' una soluzione completamente open-source!
  - Sfrutta come backend solo prodotti open-source
  - E' stato scritto in linguaggio Go
- Traduce le richieste effettuate con il protocollo di *MongoDB* in query SQL, usando PostgreSQL come motore di database
- E' compatibile con tutti i driver, tool e app per *MongoDB*
  - Consente di usare lo stesso linguaggio di MongoDB, rimanendo tuttavia open-source
  - Può essere usato come sostituto diretto di MongoDB 6.0+
- Il team aggiunge costantemente nuove feature per incrementarne la già elevata compatibilità dietro feedback degli sviluppatori
  - Sono in corso di sviluppo nuove estensioni per compatibilità con altri backend
- La prima release GA (v 1.0) è stata lanciata ad aprile 2023

Perché FerretDB?

- Consente di lavorare con i driver MongoDB già disponibili e collaudati, senza modifiche al codice esistente.
- Supporta framework e tool progettati per MongoDB.
- Offre una soluzione che è completamente open source.
- Apre la strada potenziale all'uso di altri backend (è un *layer*).
- Pur usando PostgreSQL (o SQLite), non richiede esperienza con questi database (seppur consigliata).

Le sfide da affrontare

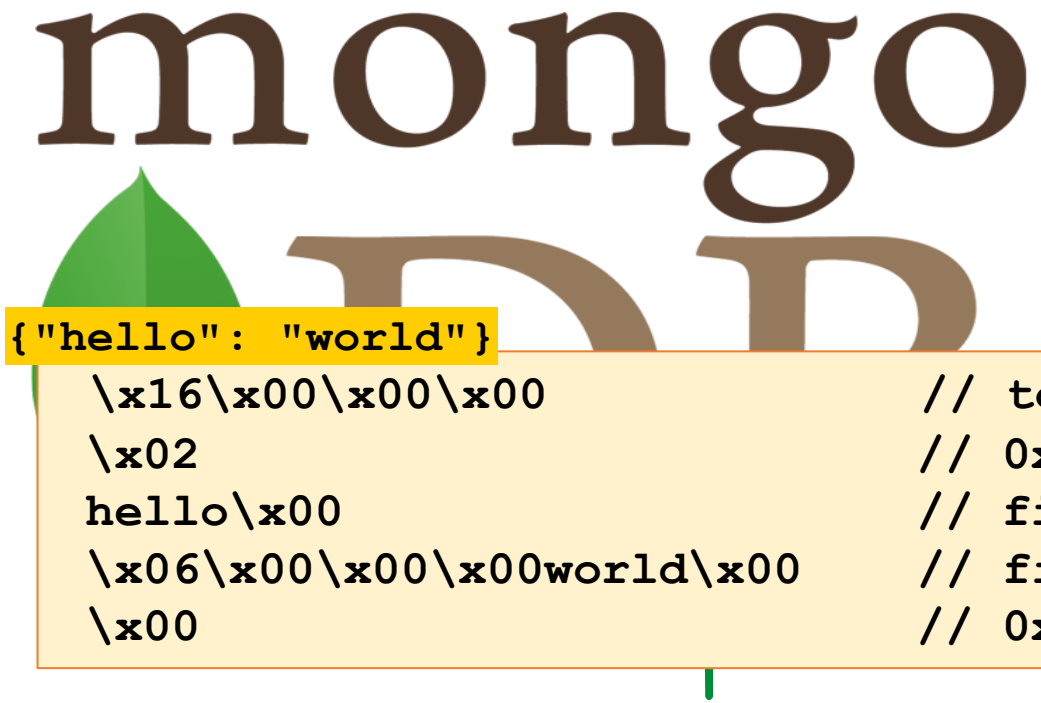
- MongoDB usa il formato BSON
  - E’ un JSON in formato binario
  - Supporta tipi aggiuntivi oltre a quelli standard
  - L’ordine delle proprietà (chiavi e valori) deve essere mantenuto
- Dialogo basato su formato JSON
  - E’ necessario leggere e scrivere documenti JSON in arrivo
  - Occorre trasformare BSON in JSON, e viceversa
- Necessità di persistenza dei dati
  - I documenti vanno letti e scritti su uno storage, o meglio un backend
  - Il backend è intercambiabile
- Interrogazione della base dati
  - Occorre implementare una business logic per query e altre operazioni (es. CRUD)
- Modello a oggetti
  - Servono DTO per memorizzare e convertire i dati in ingresso/uscita

Gli «strati» di FerretDB

Una panoramica dell’infrastruttura fondamentale di questo tool

Componenti del sistema

Application (Driver) Layer



- Legge e scrive documenti in formato BSON compatibili con MongoDB
  - BSON è un formato JSON binario
  - Supporta tipi aggiuntivi [illegible] quelli standard JSON
- Trasforma i documenti i[illegible] formato gestibile dal

```
{"hello": "world"}
  \x16\x00\x00\x00                  // total document size
  \x02                              // 0x02 = type String
  hello\x00                         // field name
  \x06\x00\x00\x00world\x00         // field value
  \x00                              // 0x00 = type EOO ('end of object')
```

Business Layer

- **Go** è un linguaggio open source by Google
- Semplice da programmare, efficiente, dalla compilazione veloce
- I dati vengono gestiti da un business layer sviluppato in linguaggio Go
  - Consente di tenere i dati in memoria
  - I tipi dei campi sono quelli nativi di Go
  - Go dispone di un *Garbage Collector*
- Include gli handler di comandi e la logica generale di processing
  - Le funzionalità sono implementate usando le API distribuite con Go
  - Per le funzionalità legate a JSON, BSON, ecc. si usano package open source
- Offre diversi punti di estensibilità (es. per il supporto a diversi backend)

Backend Layer

- Estensione del business model implementato in Go
  - Strutturato a interfacce intuitive (Backend, Collection, Database, …)
  - Gli oggetti del modello sono «as small as possible»
- Serializzazione in JSON e storage

```
{"hello": "world"}
```

```
{
  "$s": {
    "p": {
      "hello": {"t": "string"}
    },
    "$k": ["hello"]
  },
  "hello": "world"
}
```

es.

uto

Demo time!

MongoDB… ehm… FerretDB con FireDAC in Delphi

Accediamo a database MongoDB e/o FerretDB in Delphi con FireDAC

MongoDB FireDAC Driver

- Il driver FireDAC per MongoDB supporta le versioni dalla 3.0 in su
- Si interfaccia alla libreria del Mongo Driver per il linguaggio C
  - L'installazione di Delphi include già le librerie redistribuibili (a 32-64 bit)
- La connessione al database avviene tramite **`TFDConnection`**
  - Usa il componente ***`FireDAC.Comp.Client`*`.TFDConnection`**
  - Ricordati di «linkare» il driver per MongoDB
    - Usa il componente ***`FireDAC.Phys.MongoDB`*`.TFDPhysMongoDriverLink`**
    - Includi la unit ***`FireDAC.Phys.MongoDB`***

MongoConnection… e figli 

- Usa la proprietà CliObj del componente **TFDConnection**
  - Recupera l'oggetto «wrapper» della connessione FireDAC
  - Interagisce con il client DBMS sottostante per chiamate a basso livello
- Esegui il cast a **TMongoConnection** per accedere
  - Non creare una istanza di questo tipo: recuperala da TFDConnection!
  - La connessione deve essere aperta per utilizzare questo oggetto
- Usa l'oggetto *Env* (**TMongoEnv**) per creare nuovi documenti
- Le unit FireDAC per MongoDB contengono classi (che derivano da **TMongoObject**) per gli elementi principali dell'ecosistema MongoDB
  - ***TMongoConnection, TMongoDocument, TMongoCollection, TMongoIndex, TMongoCursor, TMongoOID, TMongoSession…***
  - Per il supporto JSON, gli oggetti si appoggiano a ***System.JSON***

Demo time!

Conclusioni

Risorse per approfondire e spazio alle vostre domande

E’ limitato?

- FerretDB è diverso da MongoDB su alcuni aspetti
  - *Comparazione non uniforme tra «jsonb» (PostgreSQL) e BSON (MongoDB)*
  - *Alcuni valori/tipi sono assenti o diversamente rappresentati: NaN, null, nested arrays, negative zero, Infinity, … (vedi documentazione)*
  - *Gestione delle sessioni*
  - *Richieste tradizionali e aggregation*
- Mancano alcune macro-feature
  - *Sharding e Clustering*
  - *Backup, restore, administration*
  - *Altre feature di classe «enterprise» (alcune sono implementate da PostgreSQL)*

“

L’obiettivo di *FerretDB* è **massimizzare l’accessibilità delle funzionalità di storage** e **la compatibilità con MongoDB** per costituire una valida alternativa open source all’uso di quest’ultimo come storage.

”

E’ stabile?

- FerretDB è sottoposto a rigorosi test legati allo sviluppo e alla solidità del tool
  - Sono presenti test specifici per «use case» particolari
  - Alcuni test sono scritti dai mantainer stessi, altri dagli sviluppatori di backend
  - La suite comprende test lunghi che vengono eseguiti su shard CI dedicati
- Non tutto l’ecosistema di FerretDB deve essere testato
  - I test dei driver non sono necessari (già eseguiti dai developer che li creano per MongoDB)
  - PostgreSQL in sé e i package open source sono già sottoposti ai loro test dedicati
- La stabilità è quella del runtime di Go
  - Il runtime è testato da Google
  - La pipeline e i tool di testing sono quelli disponibili in Go (non vanno creati ad hoc)

E’ ben supportato?

- E’ open source!
  - Si può contribuire al progetto se si conosce il linguaggio Go
  - Si possono segnalare issue direttamente su GitHub
- Il team adotta metodologie e tool globalmente noti e diffusi
  - Container (Docker), Task Runner (go-task), Linters (go-lint), versioning, regole nella gestione del ciclo di sviluppo
  - GitHub Actions per linting, test, security scan, action dedicate (anch’esse pubbliche e disponibili su GitHub!)
- FerretDB è il «core product» della startup che lo produce (FerretDB Inc.)

IMHO…

- Su Windows il supporto è limitato: ad esempio, si compila a mano
- Mancano feature interessanti di MongoDB (es. *Capped Collections*)
- L'amministrazione è diversa rispetto a MongoDB
- Adatto a scenari dove non abbiamo a che fare con grossi volumi di dati (inteso come «non Big Data»)
- Progetto recente e (al momento) soggetto a molte modifiche (molto pioneristico a mio avviso)
- Installato su sistema operativo Linux oppure in Docker è perfetto!
- Le feature più usate ci sono tutte e funzionano su diversi backend
- Per chi conosce PostgreSQL tutto risulta più semplice e familiare
- Può essere usato anche in «modalità Embedded» con SQLite e pure integrato nell'eseguibile!
- Progetto gestito bene, interessante, non banale, curato e molto promettente!

…e poi è open source… cosa volere di più? 😉

Risorse e approfondimenti

- FerretDB
  - Sito ufficiale 👉 https://www.ferretdb.com
  - Repository su GitHub 👉 https://github.com/FerretDB
  - Blog 👉 https://blog.ferretdb.io
  - Documentazione 👉 https://docs.ferretdb.io
  - Roadmap ufficiale: 👉 https://github.com/orgs/FerretDB/projects/2
  - Store 👉 https://store-ferretdb.myshopify.com
- MongoDB
  - Sito ufficiale di MongoDB 👉 https://www.mongodb.com
  - Webinar introduttivo 👉 https://www.youtube.com/watch?v=vk0WDErLkKU
- Delphi FireDAC + MongoDB/FerretDB
  - Tutorial Embarcadero 👉 http://docwiki.embarcadero.com/RADStudio/Athens/en/Connect_to_MongoDB_Database_(FireDAC)

Q & A

Grazie!