ITDevCon

European Delphi Conference

14, 15 november 2013 - VERONA (Italy)

No SQL, thanks!

Creare e accedere a database NoSQL in Delphi

Marco Breveglieri

Software and Web Developer

bit Time software

ITDevCon
European Delphi Conference

Introduzione

Introduzione ai database NoSQL

Definizione di «NoSQL»

“A NoSQL database provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational databases. […]

NoSQL systems are also referred to as "Not only SQL" to emphasize that they may in fact allow SQL-like query languages to be used”

Wikipedia

Caratteristiche

Si riferisce a un tipo di storage che

- non si basa sul modello relazionale (non è un RDBMS)
- non utilizza SQL come linguaggio di interrogazione
- favorisce scalabilità e disponibilità rispetto a consistenza e atomicità
- può basarsi su tipi diversi di strutture dati

Cosa si intende per «NoSQL» (1)

«NoSQL» is *not* RDBMS (ACID).

«ACID» significa

- Atomicity: ogni transazione va a buon fine o viene interamente annullata
- Consistency: una transazione non può lasciare il database in uno stato inconsistente
- Isolation: ogni transazione non deve interferire con le altre
- Durability: una transazione completata persiste

Cosa si intende per «NoSQL» (2)

«NoSQL» follows BASE approach.

«BASE» significa

- Basic availability: ad ogni richiesta viene fornita il prima possibile una risposta (buon fine o errore)
- Soft state: lo stato del sistema può variare nel tempo per mantenere la consistenza dei dati, senza che vi sia un input
- Eventual consistency: il database può trovarsi in uno stato di inconsistenza, ma diventerà consistente dopo un po' di tempo

NoSQL non rappresenta la soluzione ideale per tutti gli scenari.

Perché NoSQL?

Vantaggi

- **Rappresentazione «schema less»**: consente di definire la struttura del DB evolvendola nel tempo senza doverla fissare necessariamente a priori
- **Riduzione dei tempi di sviluppo**: evitare l'uso di SQL, di JOIN e altri costrutti complessi consente di scrivere più rapidamente il codice
- **Elevata velocità**: i dati possono essere trasferiti a velocità molto elevate, rispondendo a molti più utenti
- **Estrema scalabilità**: possono gestire picchi di utilizzo garantendo comunque il servizio agli utenti

Differenze di approccio

RDBMS vs NoSQL

Approccio con RDBMS (1)

- **Identificazione degli elementi del dominio**: dall'analisi del software si individuano gli elementi che appartengono al dominio del sistema
- **Creazione di un modello**: gli elementi del dominio vengono raffigurati all'interno di un diagramma ER
- **Definizione delle entità**: si definiscono le tabelle che corrispondono a ciascun elemento del modello, con le relative colonne e i tipi associati
- **Definizione delle relazioni**: si mettono in relazione le tabelle in base ai legami logici tra le entità
- **Programmazione del database e sviluppo dell'applicazione**: hanno inizio le iterazioni che ciclicamente modificano il codice dell'applicazione, la struttura del database e il codice memorizzato in trigger e stored procedure

Murphy è sempre in agguato…

- Il management cambia completamente il team di sviluppo: i nuovi arrivati non sanno nulla dell'architettura attuale ma hanno bisogno di stravolgerla per fare fronte a nuove richieste
- Tramite il proprio sito Web, l'azienda apre l'accesso a ordini provenienti da clienti finali sparsi sul territorio, o conduce una campagna che aumenta esponenzialmente agenti e utenti
- L'azienda acquisisce una società e ha bisogno di «fondere» le basi dati per ottenere un unico archivio omnicomprensivo
- Viene aperto un canale verso un servizio, o un sistema, che consente di veicolare in modo automatizzato una quantità estremamente superiore di ordini
- L'amministratore delegato (improbabile) lancia l'idea di aprire la piattaforma ad altri servizi tramite un'API dedicata
- Il sistema viene integrato con pagine, contributi e condivisione di risorse da e verso i social network

…ed ecco che le cose si complicano. ☹

Approccio con NoSQL

- **Flessibilità dello schema**: possiamo definire le entità e la loro struttura man mano che emergono dettagli al riguardo (con un minimo di accortezza)
- **Interrogazione dei dati**: non sono necessarie query complesse per interrogare i dati, né sono necessari «join» tra tabelle, mentre possiamo ottenere velocemente tutti i dati relativi alle entità richieste, beneficiando (ove presente) anche di una cache
- **Sincronizzazione dei dati**: molti database NoSQL offrono efficienti soluzioni per sincronizzare nell'ordine di millisecondi storage dislocati, senza l'uso di transazioni e tempi di attesa
- **Scalabilità**: l'assenza di transazioni ACID e la flessibilità dello schema, con la possibilità di trasferire atomicamente un'entità e i dati correlati, consentono di supportare un numero crescente ed esponenziale di operazioni

Tipi di storage

Le varie tipologie di database NoSQL

Categorie di database NoSQL

- I database sono catalogati in base alla tipologia di struttura adottata nella memorizzazione dei dati
- Le strutture dati più diffuse sono tendenzialmente semplici e «vagamente note» (es. dizionari, tabelle hash, collezioni, ecc.)
- Si tratta di strutture dati orizzontali
- Sono in larga parte privi di supporto alla programmazione lato server (es. non hanno stored procedure e funzioni, non supportano linguaggi SQL né dialetti)

Tipo di storage «Column Oriented»

- Memorizza i dati organizzandoli in colonne (a differenza delle righe prominenti nei sistemi RDBMS)

Ad esempio,

EMP_ID	FIRST_NAME	LAST_NAME	AGE	SALARY
001	Anuj	Sharma	45	99999999
002	Anand	Smith	34	5000000
003	Vikas	Gupta	39	7500000
004	Dinesh	Verma	32	2000000

diventa

```
001,002,003,004
Anuj,Anand,Vikas,Dinesh
Sharma,Smith,Gupta,Verma
45,34,39,32
99999999,5000000,7500000,2000000
```

Tipo di storage «Document Store» (1)

- Conosciuto anche come «Document Oriented Database», consente di inserire, estrarre e manipolare dati strutturati
- I formati di rappresentazione sono XML, JSON, BSON, YAML
- L’accesso avviene normalmente tramite protocollo HTTP utilizzando API RESTful
- I «documenti» corrispondono ai record RDBMS, ma sono strutture dati più complesse che includono informazioni correlate
- Generalmente, non è richiesta l’adesione del documento a un preciso Schema precostituito
- I database sono comunque in grado di generare indici per velocizzare le ricerche

Tipo di storage «Document Store» (2)

- Un esempio di documento (in formato JSON)

```
{
        "EmpID": "002",
        "FirstName": "Anand",
        "LastName": "Smith",
        "Age": "34",
        "Salary": "5000000",
        "Address": {
                "Line1" : "Via Monte Pastello, 1",
                "City"  : "Verona",
                "State" : "Italy"
        }
}
```

Tipo di storage «Key-Value Store»

- Simile al «Document Store», ma consente di ottenere un documento solo attraverso la sua chiave e il documento è «opaco»
- Mantiene una struttura Schema-less
- Velocizza al massimo le ricerche sulla chiave

Tipo di storage «Graph»

- Rappresenta una categoria speciale di database NoSQL basato sui grafi
- E’ indicato per archivi dati pesantemente basati su relazioni
- Può essere usato in combinazione con altri database

Database NoSQL più diffusi

Document	Key/Value	XML	Column	Graph
MongoDB	Redis	BaseX	BigTable	Neo4J
CouchDB	Membase	eXist	Hadoop / Hbase	FlockDB
RavenDB	Voldemort		Cassandra	InfiniteGraph
Terrastore	MemcacheDB		SimpleDB	
			Cloudera	

Alcune note:

- l'elenco non è definitivo e completo
- la maggior parte sono gratuiti e OpenSource

ITDevCon
European Delphi Conference

CouchDB e Delphi

Introduzione a CouchDB

- E’ un database NoSQL di tipo «Document Store»
- E’ disponibile per Windows, Linux, Mac OSX (scaricabile dal sito ufficiale: http://couchdb.apache.org)
- Consente l’accesso a client tramite una interfaccia RESTful
- Si amministra tramite una interfaccia Web incorporata: «Futon»

Demo