Architetture Dati complesse

e dove trovarle

Danilo Dominici

Sponsor

altitudo

beanTech
IT moves your business

OVERNET.
upgrade your digital skills

[stesi]
Powered by Innovation

Chi è Danilo Dominici

Consulente senior in Altitudo, docente, speaker ed autore.

Che cosa faccio: progettazione, implementazione ed ottimizzazione architetturale, monitoring e performance tuning di soluzioni basate su SQL Server, on-premise e in cloud, PostgreSQL e Redis

Speaker ai principali eventi sulle tecnologie Microsoft (PASS Community Summit, WPC, Data Saturday, e… SQL Start!)

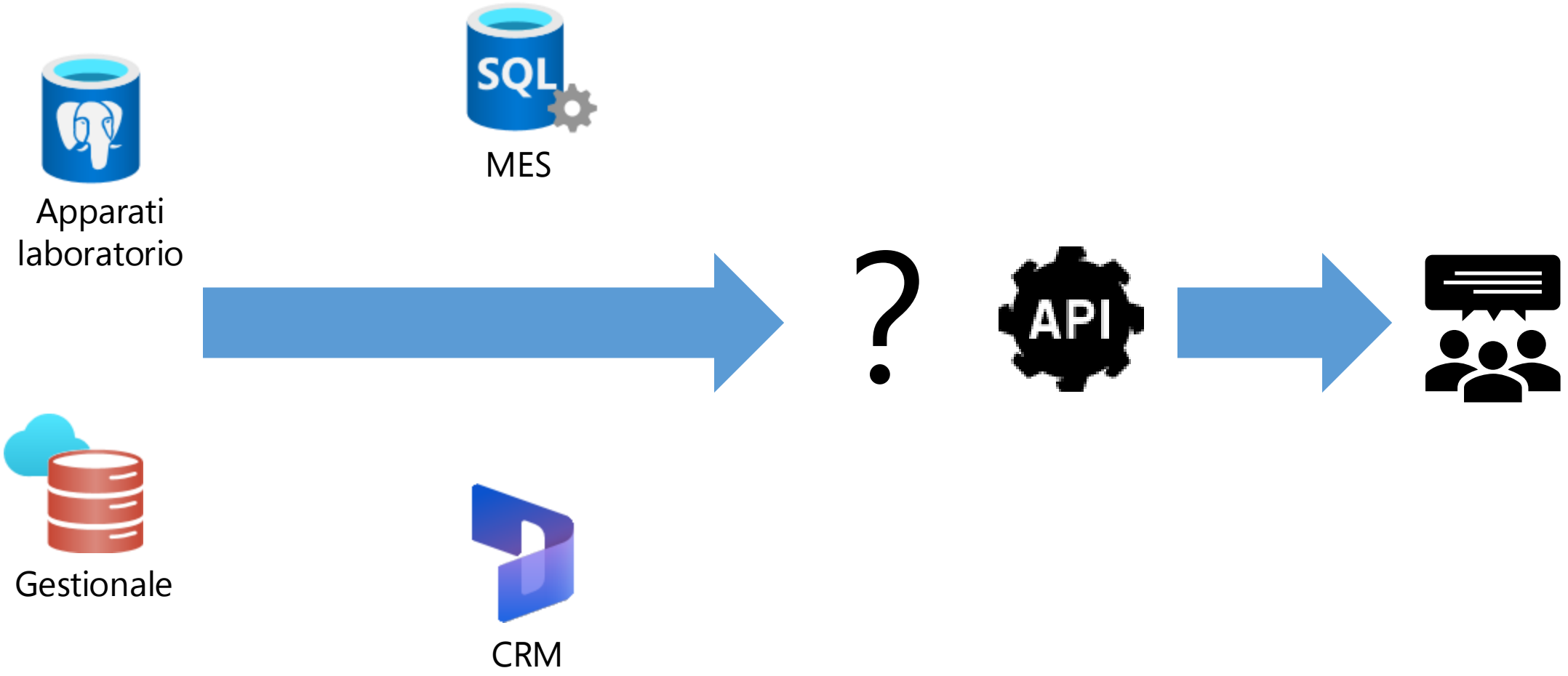
Microsoft Certified Trainer dal 2000, Data Platform MVP (2014 → 2020)

Autore di alcuni capitoli delle guide Microsoft sull'aggiornamento di SQL Server 2012 e 2014
Co-autore del libro "Azure Data Solutions: an overview"

Scenario: integrazione di diversi database

- Esistono diversi database che contengono i dati dei diversi applicativi aziendali (gestionale, CRM, MES, posta elettronica, motori di ricerca, etc)
- L'azienda deve centralizzare parte di queste informazioni per renderle disponibili ai propri clienti e fornitori
- Le modifiche ai dati devono essere replicate nel minor tempo possibile
- L'accesso ai dati da parte dei partners deve avvenire attraverso API REST
- Dove possibile, utilizzare tecnologia Open Source per ridurre i costi

Scenario: integrazione di diversi database



Le attività richieste

Step 1: replicare i dati

- I dati devono essere replicati dal database sorgente nel minor tempo possibile ed in modo affidabile
- Che cosa utilizziamo ?
  - Motori di ETL (SSIS, Talend, Apache NiFi, etc)
  - Programmi ad-hoc
  - Change Data Capture

Change Data Capture

- Tecnologia supportata da diversi database
  - SQL Server, PostgreSQL, MySQL, Oracle, DB2, etc
- Tutte le modifiche intervenute nel database vengono intercettate
- Veloce: legge direttamente dal log delle transazioni/redo log/WAL
- Posso restringere il campo alle sole tabelle che mi interessano
- Posso sfruttare CDC per inviare via *streaming* le modifiche ad una differente destinazione (es. cache, motore di ricerca, database centralizzato, etc)

Esempi di utilizzo del CDC ?

- Integrazione con applicazioni legacy
- Invalidazione della cache in maniera intelligente
- Monitorare le modifiche sui dati
- Data Warehousing
- Event sourcing (CQRS)

Debezium

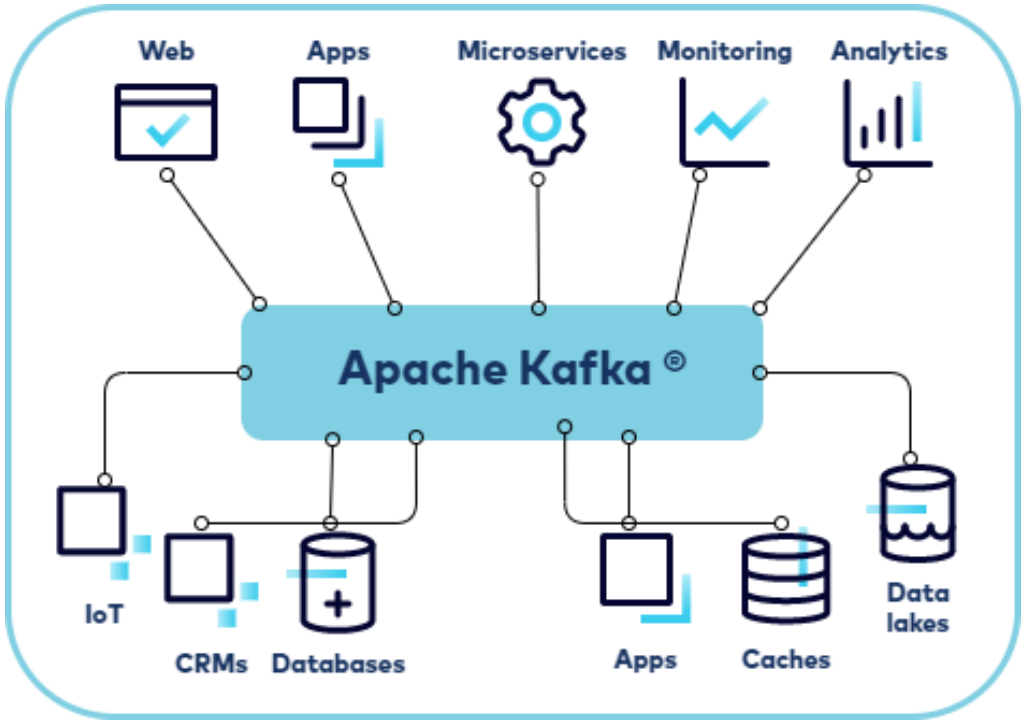
- Serie di connettori open source per Apache Kafka/Kafka Connect
- Consente l'uso di CDC su diversi database
  - SQL Server, PostgreSQL, MySQL, Oracle, MongoDB, DB2, Cassandra, Vitess, Spanner, JDBC, Informix
- Si basa sul log delle transazioni/wal/redo log
- Consente di effettuare snapshots dei dati, filtrare, etc
- Tra le caratteristiche principali:
  - Assicurazione che tutte le modifiche siano catturate
  - Ritardo di pochi millisecondi
  - Nessuna modifica alla struttura dati
  - Cattura anche le cancellazioni
  - Cattura dello stato precedente del record

Apache Kafka

Kafka è un sistema distribuito, open source, per lo streaming di eventi attraverso la logica *publish-subscribe,* disegnato per essere veloce, scalabile e durevole

Ha molteplici utilizzi:

- Sistema di messaggistica
- Tracciamento attività
- Acquisizione metriche
- Processamento flussi di dati
- Disaccoppiamento di sistemi
- Integrazione dati



Kafka Connect

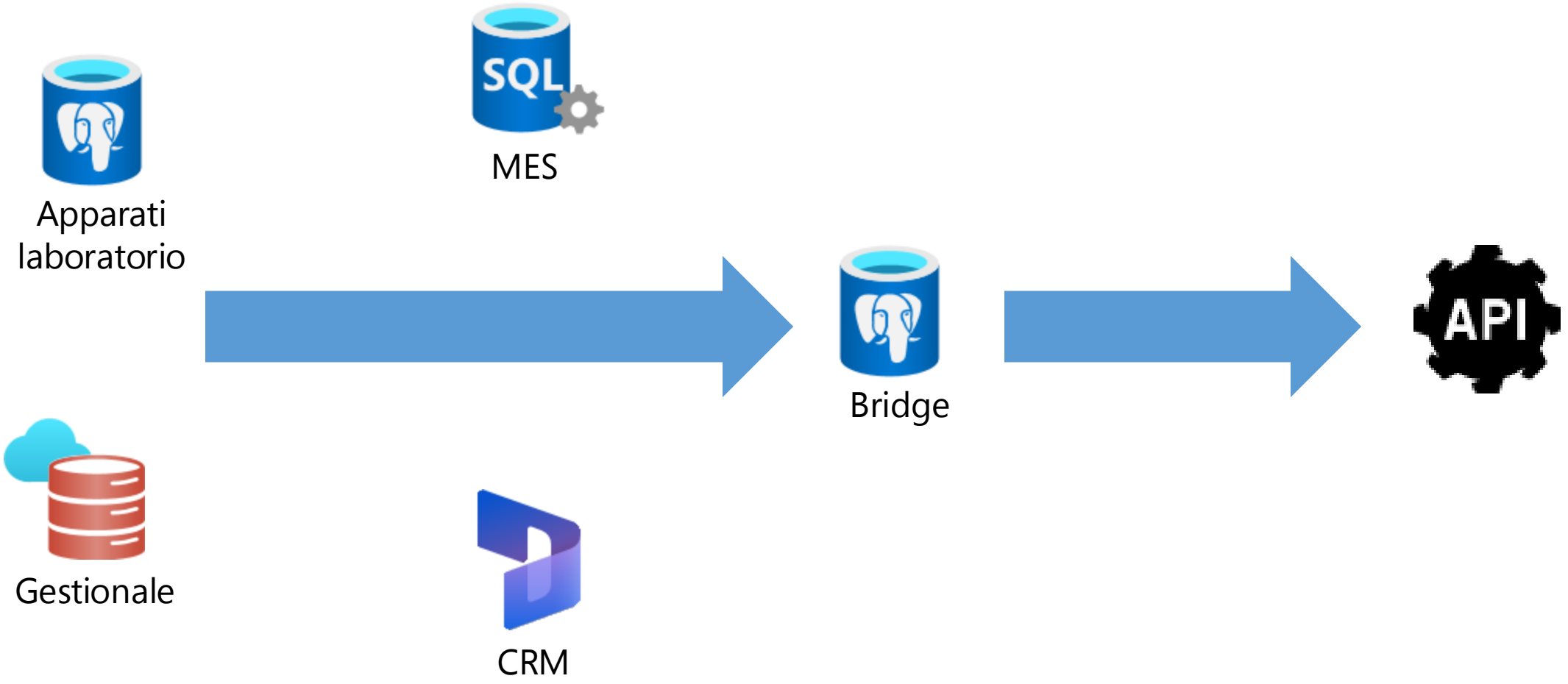
E’ un framework incluso in Apache Kafka per l’integrazione con altri sistemi

Si basa su *connettori* che consentono di spostare grandi quantità di dati da e verso Kafka

Può funzionare in modalità standalone o distribuita, nel caso in cui sia necessario scalare verso l’alto

Gestione automatica dell’offset (per leggere solo i dati non ancora processati in modo completamente trasparente)

Scenario: integrazione di diversi database



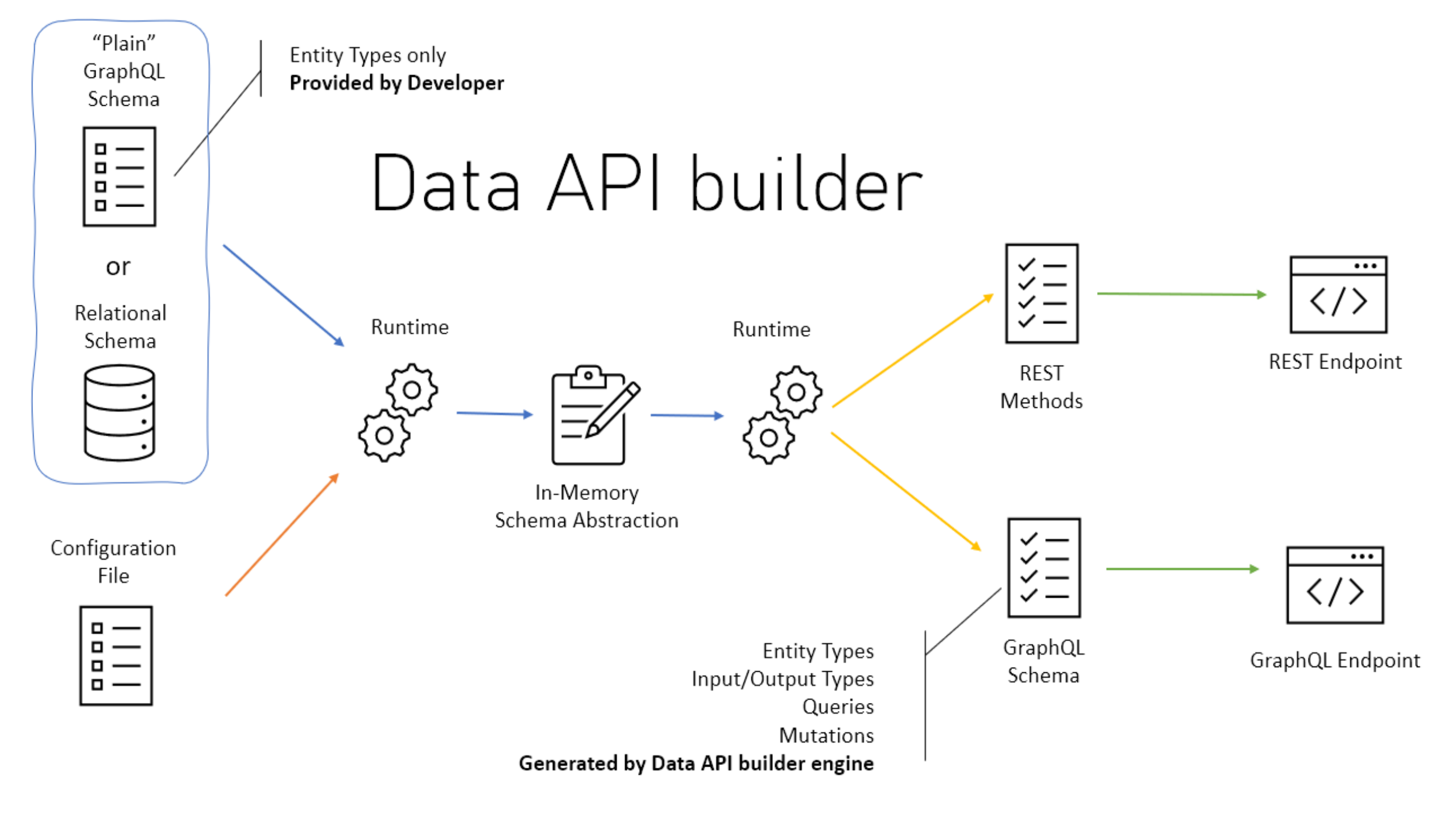
DEMO

Step 2: trasformazione dati (opzionale)

- Ripulire i dati
- Conversione di tipo/unità di misura
- Integrazione con dati esterni
- Controlli di qualità/sicurezza
- Anonimizzazione

Step 3: esporre i dati via API REST

- Che cosa utilizziamo ?
  - Codice custom
  - Data API Builder


"Plain"
GraphQL
Schema
Entity Types only
Provided by Developer
Data API builder
or
Relational
Schema
Runtime
Runtime
REST
Methods
REST Endpoint
In-Memory
Schema Abstraction
Configuration
File
Entity Types
Input/Output Types
Queries
Mutations
Generated by Data API builder engine
GraphQL
Schema
GraphQL Endpoint

Le caratteristiche

- Open source
- Consente di esporre collezioni, tabelle, viste e stored procedures come API REST e GraphQL
- Supporta l'autenticazione via OAuth2/JWT
- Supporta EasyAuth se eseguito in Azure
- Role-based authorization utilizzando i claims ricevuti
- Item-level security via policy expressions
- REST
  - Operazioni CRUD via POST, GET, PUT, PATCH, DELETE
  - filtering, sorting e pagination

Le caratteristiche

- GraphQL
  - queries e mutations
  - filtering, sorting e pagination
  - relationship navigation
- Sviluppo semplificato via CLI
- Può funzionare ovunque:
  - Server locale
  - Docker/Kubernetes
  - Azure Web App
  - Azure Static Web Apps

Installare Data API Builder

Verificare che sia installato il .NET Core (6.0 o 8.0)
dotnet --list-sdks

Se necessario, installare il .NET Core SDK (6.0 o 8.0)

Installare Data Api Builder
dotnet tool install -g --add-source 'https://api.nuget.org/v3/index.json' --ignore-failed-sources Microsoft.DataApiBuilder

In caso di problemi:
https://learn.microsoft.com/en-us/dotnet/core/tools/troubleshoot-usage-issues#nuget-feed-cant-be-accessed

Usare DAB CLI

Inizializzare il file di configurazione
dab init [opzioni]

Aggiungere le entità del database
dab add [entità] [opzioni]

Aggiornare le proprietà delle entità importate
dab update [entità] [opzioni]

Avviare il web server DAB
dab start

Il file di configurazione

Data source

Database type and connection string

Runtime settings

Runtime behavior (REST/GraphQL configuration)

Entity definitions

Database object mappings

```
{
  "$schema": "https://github.com/Azure/data-api-builder/releases/download/v0.8.51/dab.draft.schema.json",
  "data-source": {
  "database-type": "mssql",
  "connection-string": "Server=localhost;Database=Northwind;User ID=demouser;Password=D3moP@ssw0rd!;TrustServerCertificate=true",
  "options": {
  "set-session-context": false
  }
},
  "runtime": {
    "rest": {
      "enabled": true,
      "path": "/api"
    },
    "graphql": {
      "enabled": true,
      "path": "/graphql",
      "allow-introspection": true
    },
    "host": {
      "cors": {
      "origins": [],
      "allow-credentials": false
    },
    "authentication": {
      "provider": "StaticWebApps"
    },
    "mode": "development"
  }
},
 "entities": {}
}
```

DEMO

Ricapitoliamo

Problema: replicare i dati di diversi database in un unico repository centralizzato, dal quale mettere a disposizione delle API REST verso i propri partner

Soluzione proposta:

- Debezium
  - Basato su Apache Kafka
  - Utilizza Change Data Capture per replicare solo i dati cambiati nel tempo
- Data API Builder
  - Sviluppato e mantenuto da Microsoft + community
  - Open Source
  - Consente di connettersi a SQL Server (on-prem o Azure), PostgreSQL, MySQL, Cosmos DB

Ricapitoliamo (continua)

- PostgreSQL
  - Database relazionale open source
  - Estensioni «specializzate»
    - postGIS per mantenere i dati geospaziali
    - Pgvector per mantenere i vettori generati da IA
    - Tanto altro…

Domande ?

Grazie!

Danilo Dominici

ddominici@gmail.com

xxx

xxx

/danilodominici

xxx

sessionize 2023 MOST ACTIVE SPEAKER