Università degli Studi di Modena e Reggio Emilia
Dipartimento di Ingegneria “Enzo Ferrari”

Corso di Laurea in Ingegneria Informatica

Sviluppo di un’applicazione per l’analisi della produzione aziendale

Relatrice
Chiar.ma Prof.ssa Sonia Bergamaschi

Laureando
Marco Tedeschini

ANNO ACCADEMICO 2021/2022

MultiData S.r.l.

MultiData S.r.l. è un'Azienda leader nel settore dell'automazione e vede la sua specializzazione in impianti di lavorazione della gomma e delle materie plastiche.

- Azienda nata nel 1994 in provincia di Modena
- Si occupava della trasformazione dei quadri elettromeccanici in quadri con PC e PLC
- Ad oggi produce software indipendenti di controllo utilizzati sulle linee di produzione

MultiData
BUILDING SOFTWARE FOR YOUR COMPANY

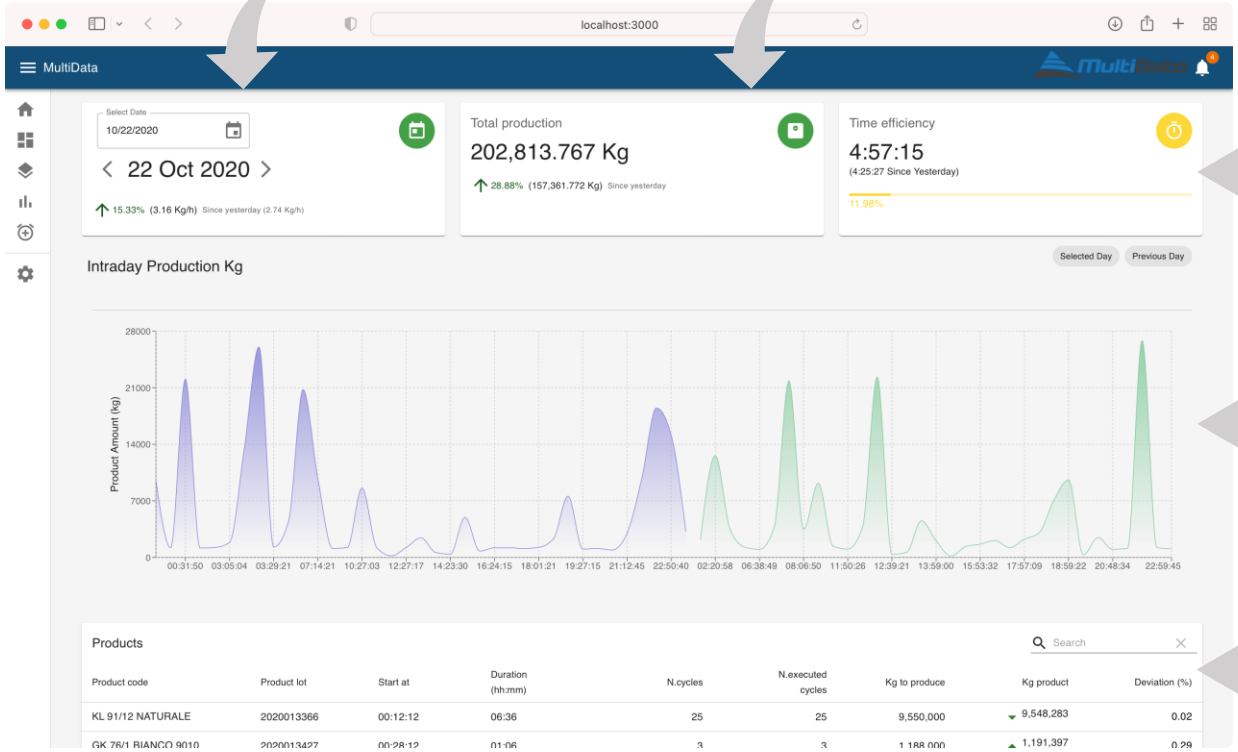
Durante il tirocinio mi è stato assegnato il compito riguardante la realizzazione di un progetto software per l'Azienda.
In particolare, mi è stato chiesto di creare *ex novo* un'applicazione per gestire e analizzare i dati raccolti dai processi produttivi degli impianti dei suoi clienti.

Risultati ottenuti

3

Selezione della data

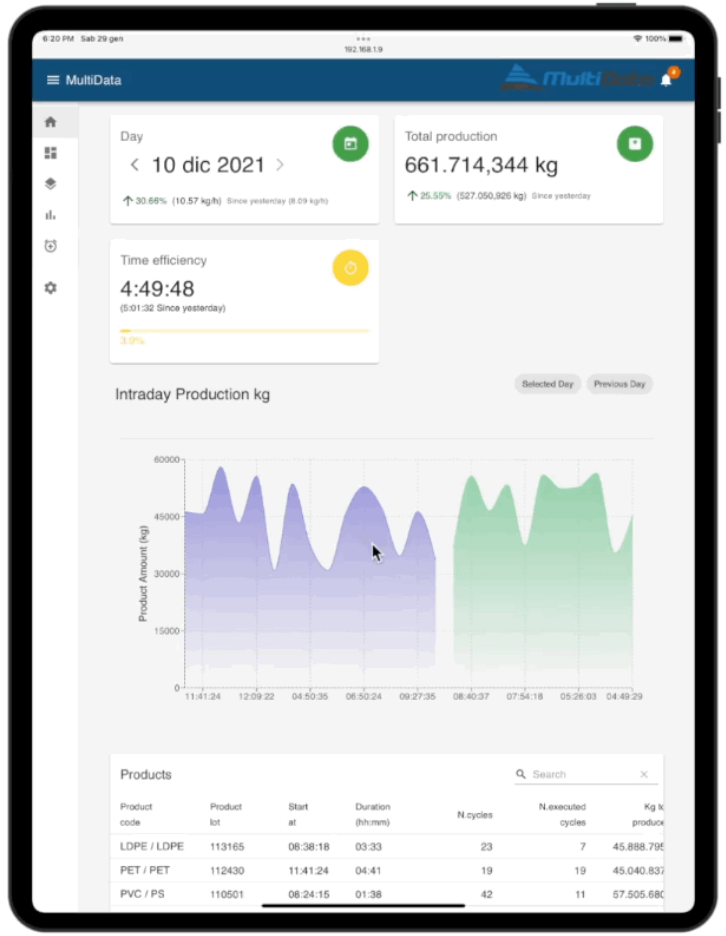
Produzione giornaliera totale



Efficienza giornaliera

Produzione delle ultime 48 ore

Dati delle produzioni del giorno

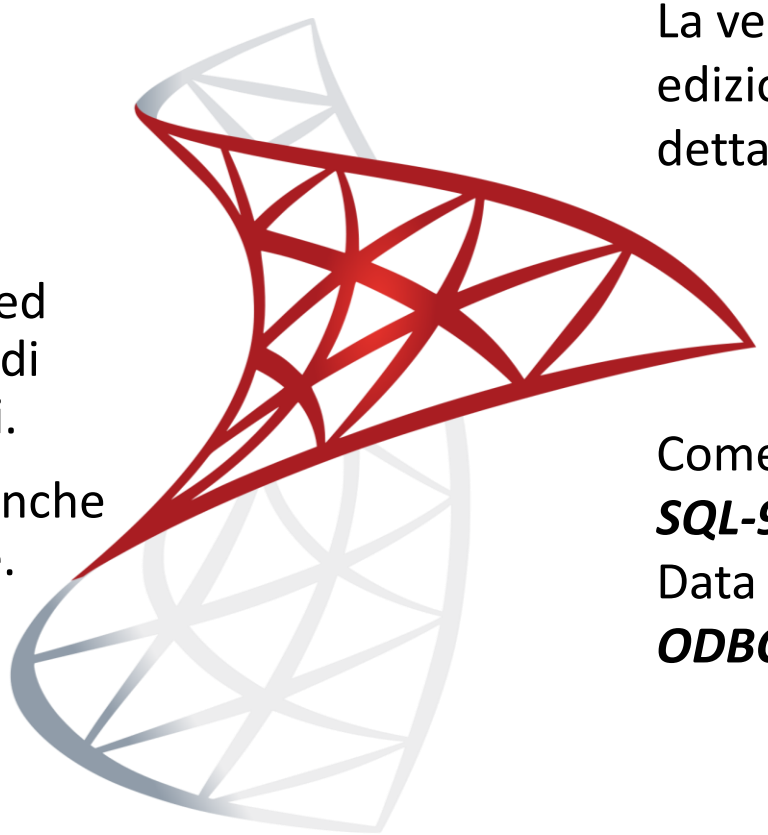


Microsoft SQL Server

MultiData utilizza ***MS SQL***: il più famoso *RDBMS* prodotto da ***Microsoft***.

La prima versione risale al 1989 ed era in grado di lavorare con basi di dati di medio-piccole dimensioni.

Dal 2000 in poi si iniziò a usare anche per operare con grandi database.



La versione attuale (2019) comprende 5 edizioni che differiscono solo per limiti dettati da licenza.

Come dialetto usa ***Transact-SQL***, variante di ***SQL-92*** e sfrutta come protocollo, Tabular Data Stream (***TDS***), pur supportando anche ***ODBC*** (Open Database Connectivity).

Django

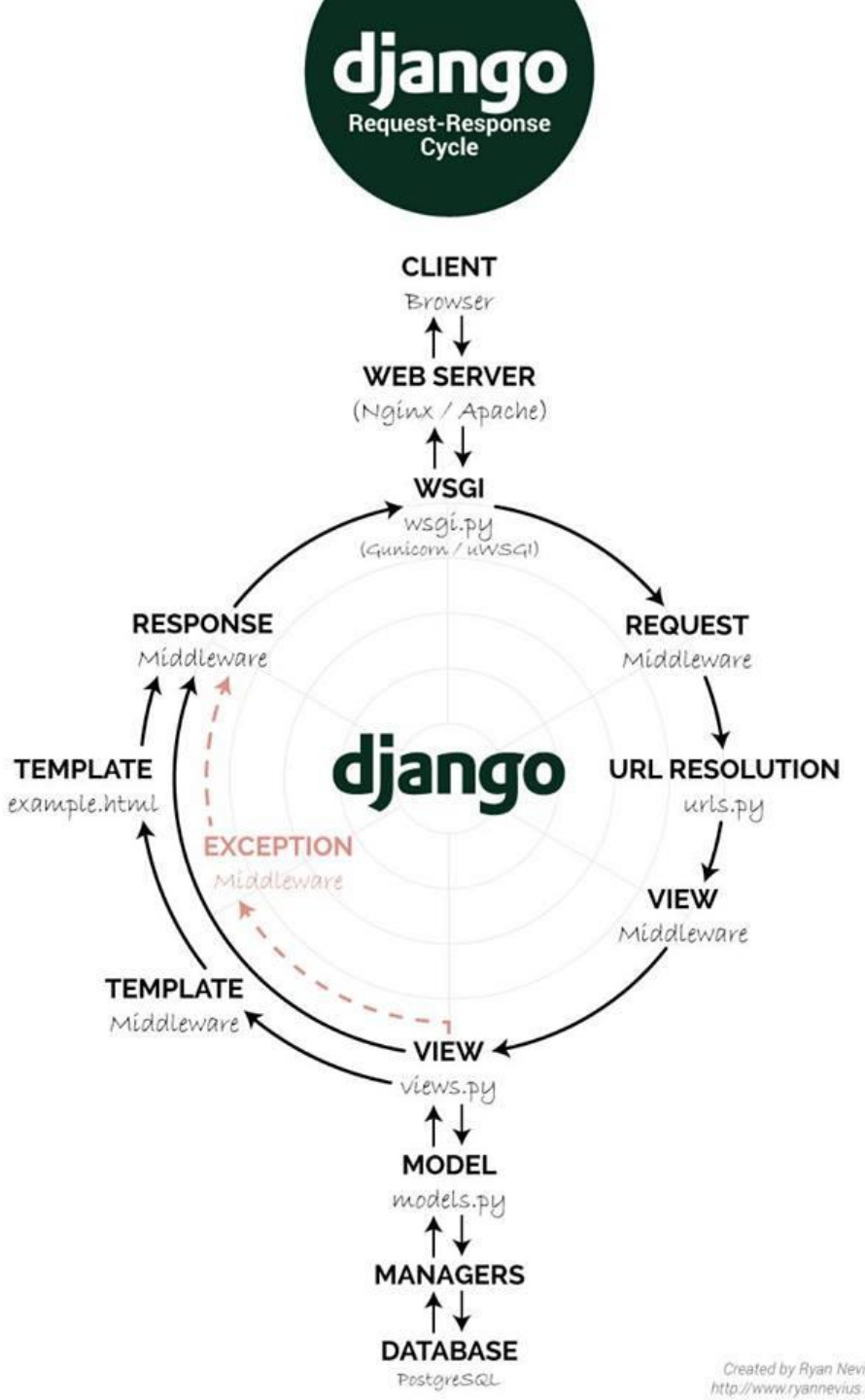
Python è un linguaggio di programmazione semplice e intuitivo, e offre una ricchissima libreria standard.

Django è uno dei web framework di Python più conosciuti. È open source e permette uno sviluppo rapido e fornisce un collegamento per diversi DBMS.

Non fornisce di default il supporto a MS SQL, ma un modulo di terze parti creato da Microsoft (mssql-django) ne permette l'integrazione.

Django REST Framework (DRF) è uno strumento per creare web API.

Nel ciclo richiesta/risposta i dati letti dal database vengono **deserializzati**, modellati ed infine **serializzati** per essere inviati al frontend.

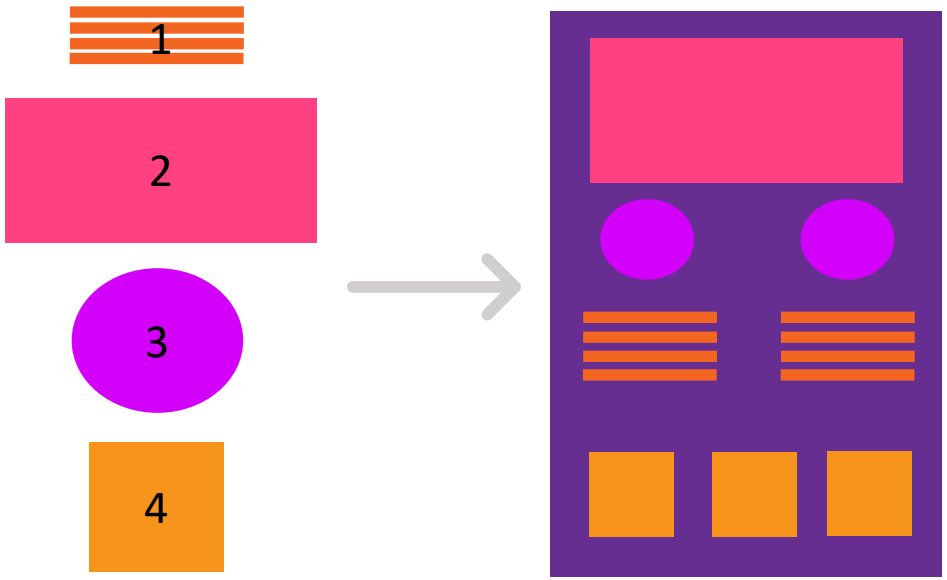


ReactJS e la logica dei *componenti*

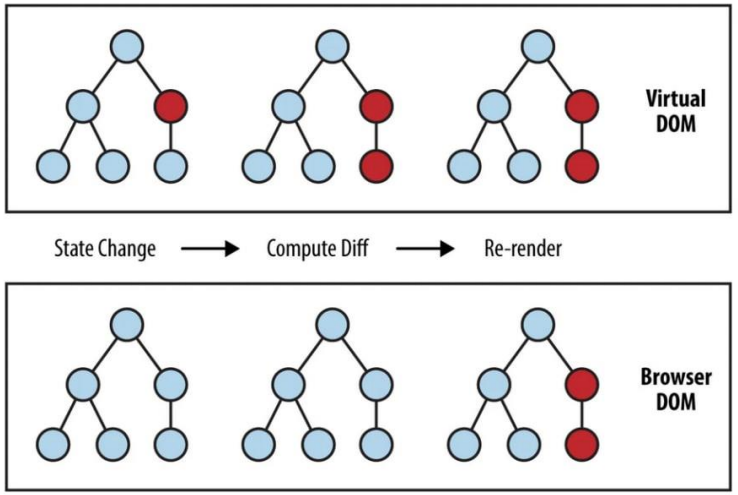
ReactJS è una libreria ***JavaScript*** adatta allo sviluppo web ed è utilizzata per la creazione di interfacce utente basate su componenti.

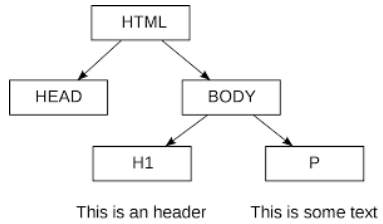
I componenti:

- Sono come "blocchi" di costruzione
- Si possono comporre
- Possono essere "ereditati"
- Sono funzioni JavaScript



Prestazioni migliori grazie al ***VirtualDOM***, per aggiornare il "vero" ***DOM*** (Document Object Model) solo quando necessario.





Bundle ("impacchettamento"):

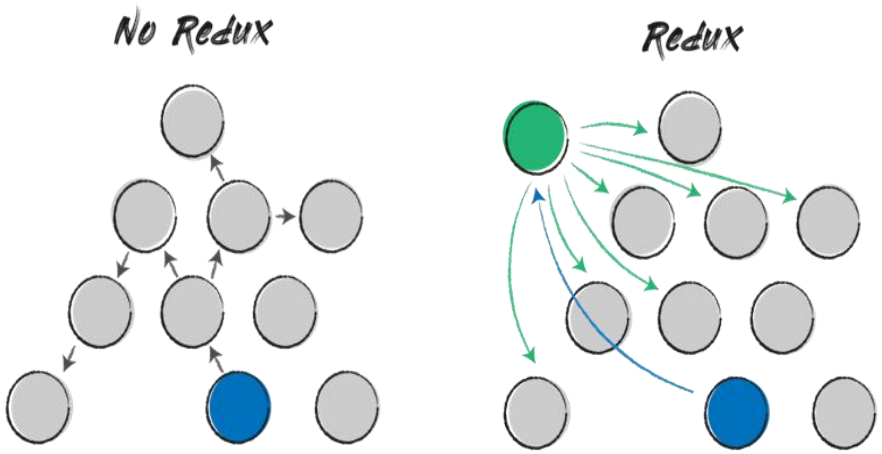
- ***Webpack*** per ottimizzare le prestazioni
- ***Babel*** per il transpile

ReactJS & Redux

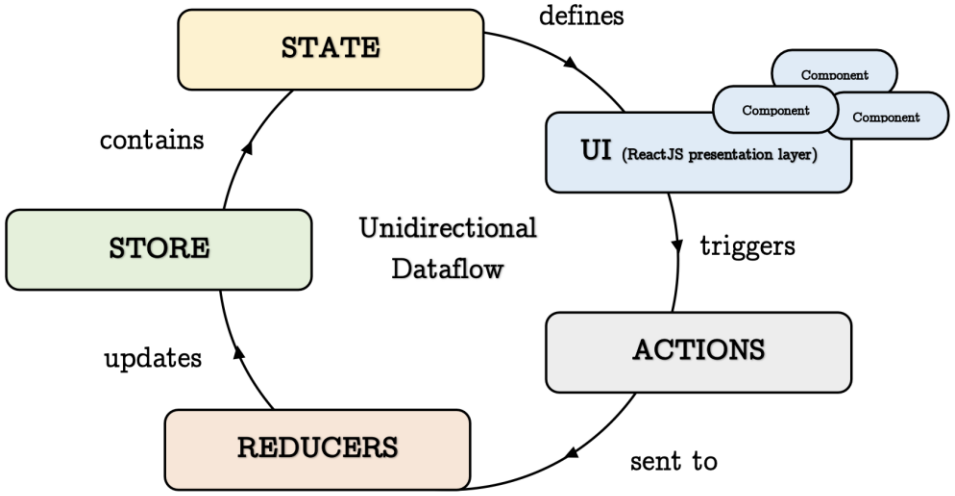
Redux è una libreria utile per immagazzinare lo stato di un'applicazione.

Utilizzandola insieme a React è possibile creare un software in cui si ha una netta divisione delle responsabilità tra la presentazione e la logica insieme allo stato dell'applicazione (**decoupling**).

ReactJS si occuperà, con l'utilizzo di componenti ***stateless*** di gestire solamente la presentazione grafica.



Queste due tecnologie combinate implementano il pattern a flussi di dati unidirezionale (**one-way dataflow** o **unidirectional dataflow**).



Questo schema permette di ottenere un'applicazione a stati predicibili.

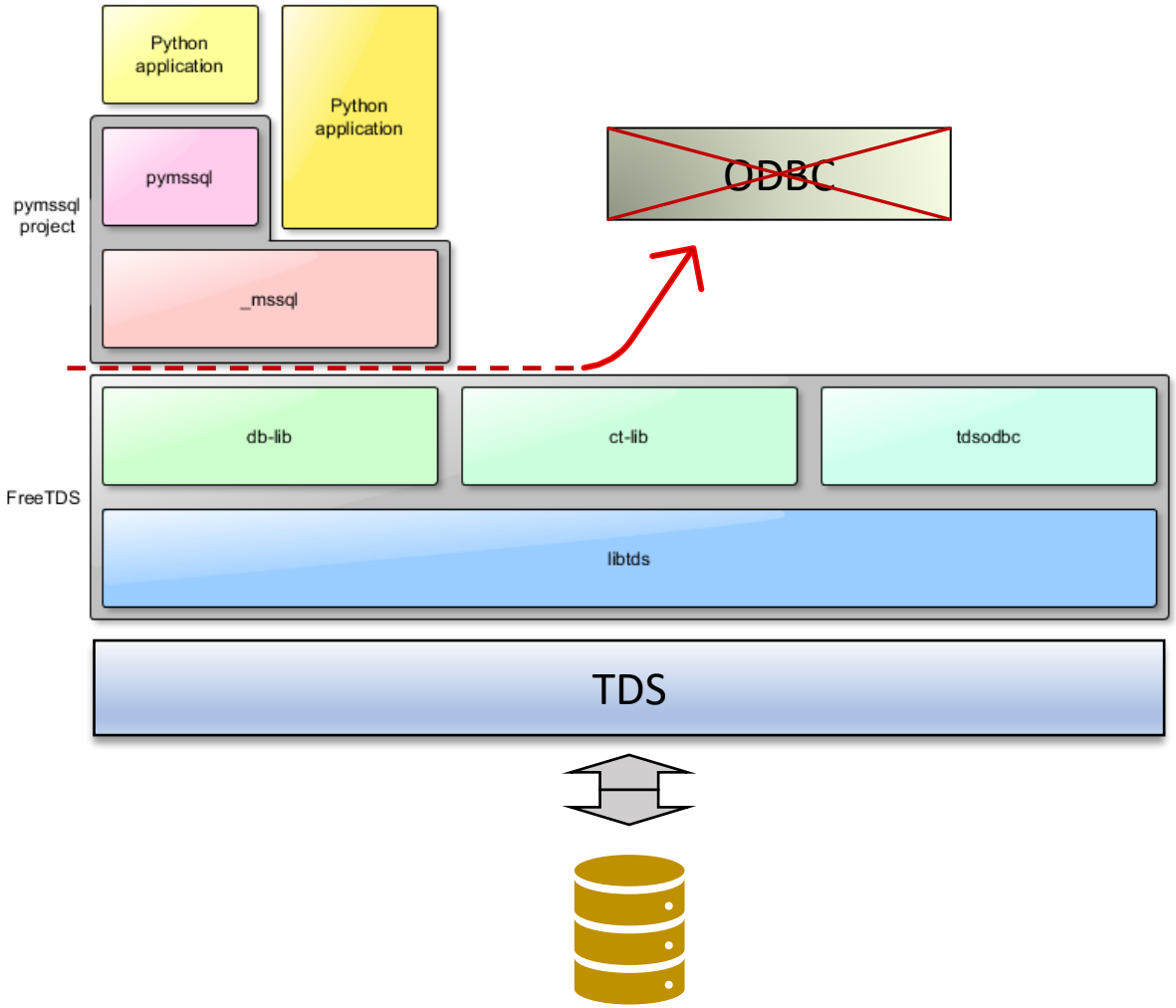
Come garantire la compatibilità con ARM?

Django-mssql si appoggia alla libreria ***pyodbc*** che per l'appunto utilizza ODBC un'API standard per la connessione del client al DBMS, che però non è in alcun modo supportata sulle piattaforme ***ARM***.

ARM indica una famiglia di microprocessori che hanno come caratteristica un basso consumo energetico in rapporto alle prestazioni.

Ad oggi alcune aziende come ad esempio *Apple* e *Microsoft* hanno già linee di computer che la utilizzano.

Al fine di poter supportare maggiori dispositivi, ho voluto modificare il progetto *Microsoft* integrando, un'interfaccia scritta in *Cython* sulle basi del driver **FreeTDS**: ***pymssql***.



Pymssql vs Pyodbc

Legenda

scarto percentuale > 50%

25% < scarto percentuale ≤ 50%

0% ≤ scarto percentuale ≤ 25%

PYMSSQL

	100	500	1000	1500	5000	10000	20000
INSERT	0,3992	1,5147	3,0597	4,2575	15,2018	29,6158	58,4723
INSERT BULK	0,0143	0,0570	0,1137	0,0956	0,1855	0,4328	0,8112
DELETE	0,6213	2,9383	6,0667	9,0643	29,6221	58,7147	117,8011
DELETE BULK	0,0077	0,0084	0,0124	0,0140	0,0242	0,0349	0,0564
UPDATE	0,3499	1,7149	3,4022	5,1923	17,6674	35,1007	67,9372
UPDATE BULK	0,0059	0,0061	0,0077	0,0111	0,0259	0,0412	0,0706
SELECT	0,3425	1,7089	3,3362	5,2002	20,3558	39,0009	102,7149
SELECT INDEXED	0,3644	1,8320	3,6269	5,4623	18,5850	35,5954	72,3342

PYODBC

	100	500	1000	1500	5000	10000	20000
INSERT	0,5799	2,8832	6,1336	7,1567	24,5000	45,0105	82,0485
INSERT BULK	0,0224	0,0593	0,1438	0,0941	0,1787	0,3696	0,7400
DELETE	1,1073	5,0211	10,1602	15,2595	51,6549	102,1771	203,7903
DELETE BULK	0,0101	0,0129	0,0142	0,0163	0,0289	0,0409	0,0691
UPDATE	0,4232	2,0703	4,1775	6,4361	21,5054	43,4564	84,8792
UPDATE BULK	0,0060	0,0059	0,0079	0,0113	0,0238	0,0479	0,0818
SELECT	0,3136	1,6872	3,5043	5,8821	20,3421	46,7149	113,0616
SELECT INDEXED	0,3047	1,6795	3,0656	4,7322	15,0539	30,2514	61,8203

pymssql riporta risultati migliori, e nello specifico nelle operazioni semplici.

La mancanza di un **layer** (ODBC) rende il codice più rapido. Quanto più la stessa operazione viene ripetuta, tanto più questo tempo inciderà sulla durata dell'esecuzione.

Ho riscontrato ottimi risultati nelle ripetizioni di singole "***delete***" dove *pymssql* è più veloce del 70%.

Risultati analoghi si riscontrano anche nelle "**insert**", dove il grafico è molto meno lineare.

La differenza in percentuale aumenta fino alle 1000 query di inserimento, dove *pymssql* dimezza i tempi di esecuzione di ***pyodbc***, per poi vedere la "forbice" assottigliarsi nei test successivi.

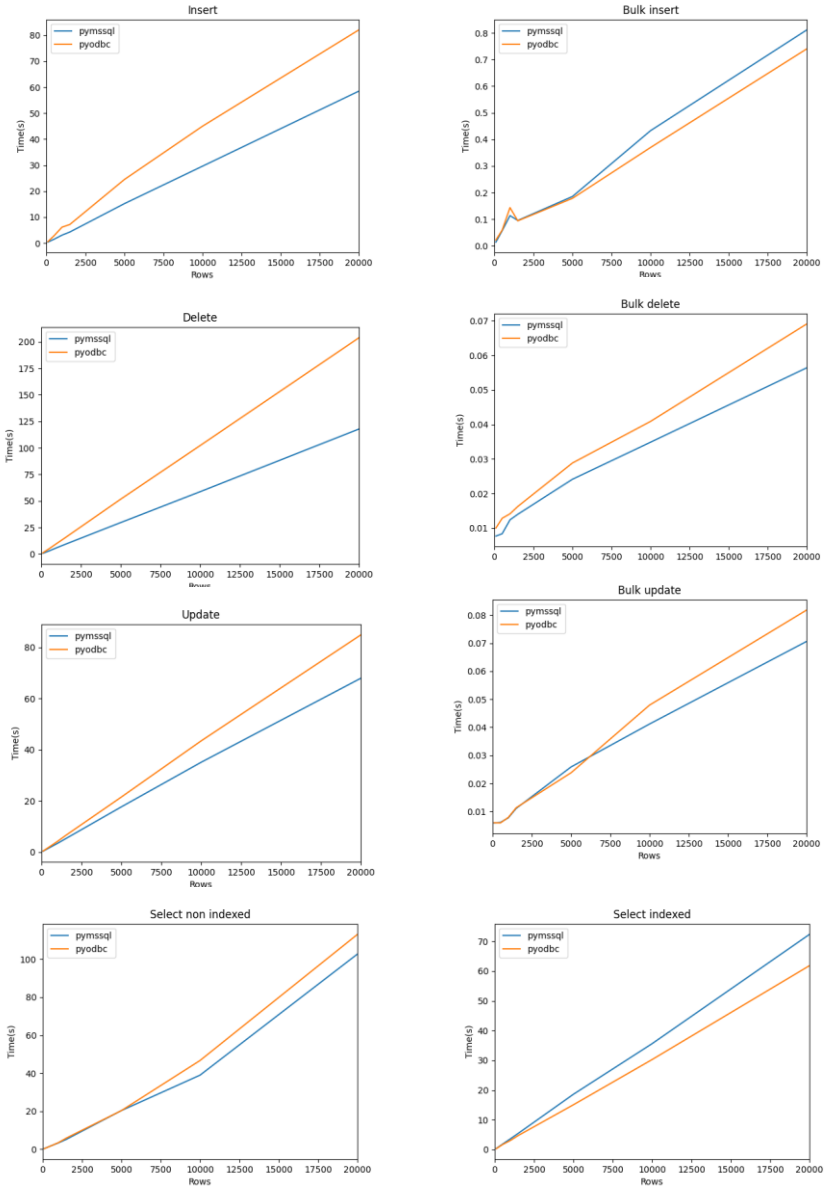
Pymssql vs Pyodbc

pyodbc offre un'ottima integrazione con i thread del DBMS. Nelle operazioni *bulk* riporta tempi migliori su maggiori volumi di dati, quando l'onerosa creazione dei thread è compensata dalla loro efficienza.

Nelle "*selezioni indicizzate*" ***pyodbc*** riporta tempi più rapidi, probabilmente dovuti a una migliore implementazione del supporto a tale tipologia di query.

Penso che la libreria possa ambire a un futuro utilizzo in altri lavori, anche se alcune fasi dello sviluppo non possono dirsi concluse.

Per gestire database con accessi limitati e con volumi di dati piccoli, le prestazioni non solo sono sufficienti, ma possono essere considerate un punto di forza.



Grazie per l’attenzione