

Programmazione di Applicazioni Data Intensive *Introduzione*

*Laurea in Ingegneria e Scienze Informatiche
DISI – Università di Bologna, Cesena*

G. Domeniconi, G. Moro, R. Pasolini

DISI Università di Bologna, Cesena

name.surname@unibo.it



Definizione

- Si definiscono *data intensive* le applicazioni in cui la gestione ed elaborazione dei dati costituiscono l'aspetto di maggiore importanza e complessità
 - usato anche in contrapposizione a *compute intensive*, riferito ad applicazioni maggiormente basate su elevate capacità computazionali
- “We call an application data-intensive if *data is its primary challenge: the quantity of data, the complexity of data, or the speed at which it is changing.*”

M. Kleppmann, “Designing Data-Intensive Applications” (2014)



Motivazioni: Crescente Quantità di Dati

- I dati sono prodotti **costantemente e in grandi quantità**
 - La crescita del **World Wide Web** ha portato ad avere una enorme quantità di informazione disponibile pubblicamente
- Questi dati sono prodotti in **varie forme, strutturate** (dati numerici, serie temporali, ...) **e non** (testi, immagini, ...)
- *“In 2006, the amount of digital information created, captured, and replicated was $1,288 \times 10^{18}$ bits. In computer parlance, that's 161 exabytes or 161 billion gigabytes. This is about 3 million times the information in all the books ever written.”*

D. Reinsel et al., “The Expanding Digital Universe”, IDC white paper (2007)
- *“We must harness the Internet’s energy before the information it has unleashed buries us.”*

V. G. Cerf, “An Information Avalanche” (2007)



Problematiche

Raccolta

- Gestire nuovi dati *generati in continuazione*

Memorizzazione

- *Archiviazione efficiente* in termini di spazio e di tempo

Ricerca

- *Reperimento efficiente* dei dati d'interesse
- Sintesi dei dati (medie, totali, ...) per migliore fruibilità

Analisi

- Estrazione efficiente di *conoscenza utile economicamente*
- Possibilità di aggiornare la conoscenza incrementalmente
- Elaborazioni *distribuite* di dati sparsi su molteplici nodi

Progettazione

- Progettazione *Model-driven* di *applicazioni UI platform-independent*
 - Es: web application, mobile, desktop, ecc...



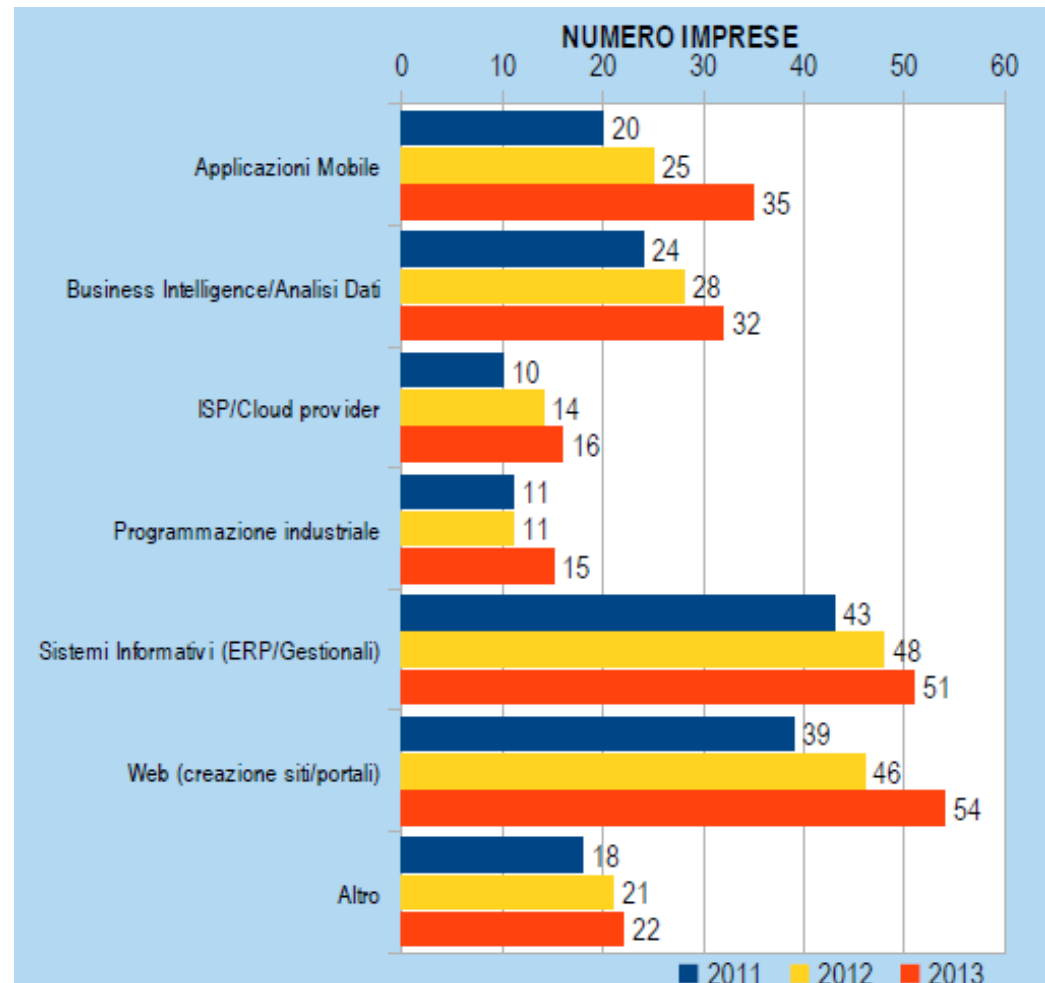
Tecnologie

- I **mezzi tecnologici** disponibili per la trattazione dei dati sono in costante sviluppo e sempre più accessibili
- Infrastrutture **scalabili** per **l'elaborazione parallela e distribuita** dei dati sono ormai ampiamente diffuse
 - anche in forma di servizi offerti da terze parti (*Infrastructure as a Service*, vedi ad es. Amazon Web Services)
- Diversi progetti di **software libero e open source** hanno raggiunto livelli di maturità tali da essere preferibili a costose soluzioni commerciali
- È possibile dalle proprie applicazioni reperire informazioni e dati da **social network** attraverso API
 - Es: twitter, facebook, LinkedIn....



Tipologia prodotti software forniti dalle imprese ICT

- Numero di imprese ICT con relativi prodotti software nella provincia di Forlì-Cesena
- Le aziende di software che fanno siti web sono le più numerose
 - Più di quelle che fanno sistemi informativi o applicazioni mobile



Caso di studio

- Come caso di studio concreto per il corso è la realizzazione di un semplice **sito di e-commerce** e l'analisi dei dati raccolti dalle attività dei suoi utenti

Benvenuto, utente ([Accedi](#) | [Registrati](#))

Il tuo carrello è vuoto

Negozio on-line



Prodotto
Questa è la descrizione
Prezzo: 9,99 €

Recensioni utenti

★★★★★

Ottimo prodotto

★★★☆☆

Buono ma ha dei difetti...

Caso di studio: motivazioni

Un sito di e-commerce rappresenta un caso di studio esaustivo di applicazione data intensive

- I più grandi portali di e-commerce (es. Amazon) con cataloghi di **milioni di prodotti** necessitano di sistemi estremamente ottimizzati per la **memorizzazione** e la **ricerca** dei dati, in grado di gestire richieste di **migliaia di utenti allo stesso tempo**
- Gli ordini eseguiti dai clienti e i voti che danno ai prodotti costituiscono una grande mole di **informazione strutturata**, analizzabile in modo efficiente per fornire **suggerimenti di acquisto**, sia generali che personalizzati sui singoli clienti
- Dalle **recensioni testuali** dei clienti può essere stimato il loro **grado di soddisfazione** verso i singoli prodotti e si può valutare la reputazione dei rispettivi marchi



Caso di studio: requisiti

- Il sito presenta un catalogo di **prodotti**, organizzati in una gerarchia di **categorie**
- Gli **utenti** del sito devono essere in grado di autenticarsi con i loro rispettivi account
- Ogni utente può raccogliere prodotti in un carrello per poi compiere l'ordine di questi prodotti
- L'utente può consultare gli **ordini** fatti in passato
- Ogni prodotto è corredato da **recensioni**, compilate dagli utenti che lo hanno acquistato in passato
 - Ogni recensione corrisponde ad uno specifico **acquisto** del prodotto
 - Ogni recensione è costituita da un punteggio (1-5 stelle) e del testo



Caso di studio: dati di esempio

- Per ottenere un esempio realistico di applicazione, forniamo un database prepopolato con un **set di dati di esempio estratti da Amazon.com** (versione USA di Amazon)
- La quantità di dati è relativamente contenuta rispetto ai casi reali più *data intensive*, in modo da rendere più rapide le attività di laboratorio che prevedono l'analisi dell'intero set
 - 11.000 prodotti organizzati in decine di categorie
 - 11.000 utenti, esecutori di 100.000 ordini e autori di 500.000 recensioni
- Le tecnologie che presentiamo sono ad ogni modo **scalabili**, usate nella realtà anche per moli di dati molto più grandi



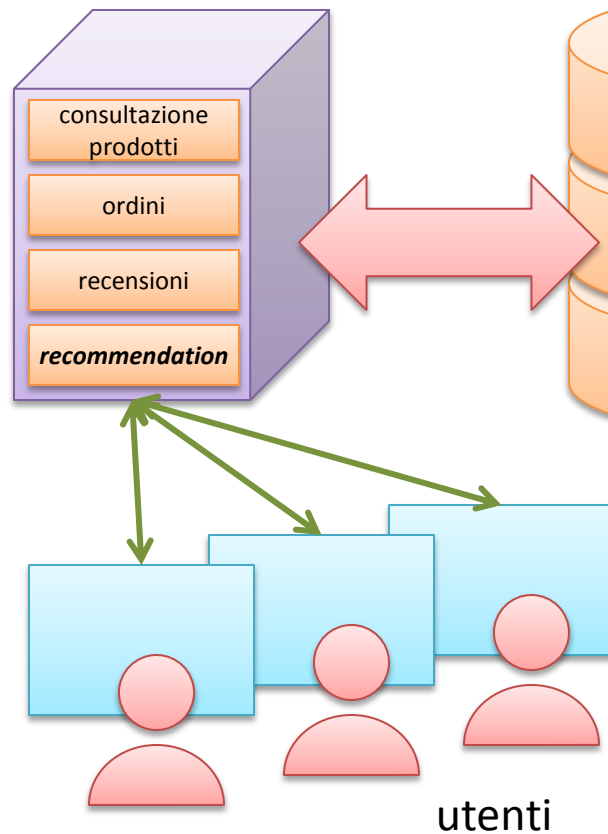
Caso di studio: dati analizzabili

- Lo storico degli acquisti degli utenti e le loro recensioni costituiscono una mole potenzialmente enorme di dati
- Questi dati possono essere analizzati per estrarre informazioni utili e fornire un migliore servizio ai clienti e ai venditori
- Agli utenti, possono essere messi in evidenza i prodotti a cui potrebbero essere maggiormente interessati (***recommendation***)
- Dalle recensioni di ciascun prodotto (potenzialmente migliaia), è possibile estrarre informazioni sommarie sull'opinione generale degli acquirenti nei suoi confronti (***customer satisfaction***)

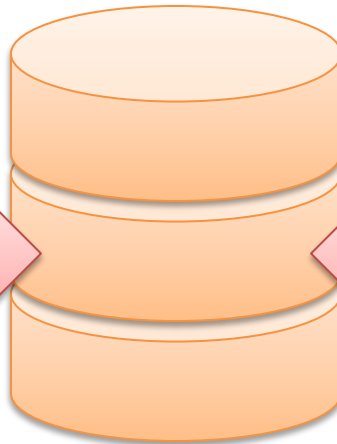


Caso di studio: componenti

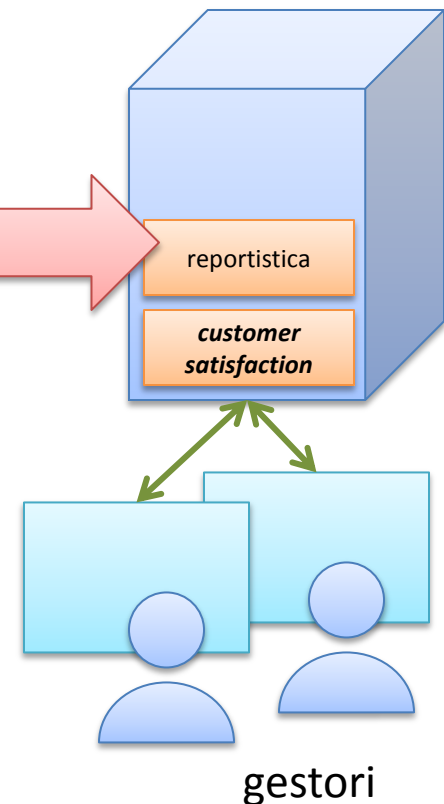
Sito web: fornisce
il servizio agli
utenti



Base di dati



**Strumenti di
analisi:** usati dai
gestori del sito



Caso di studio: realizzazione

- Durante il corso saranno presentati due approcci differenti per la realizzazione dell'applicazione presentata
- Dapprima vediamo lo *sviluppo model-driven*, in cui dal modello costruito in fase di progettazione viene generato automaticamente il codice dell'applicazione
- Dopo *implementeremo manualmente* l'applicazione, vedendo nel dettaglio le tecnologie di base usate per la realizzazione, le problematiche ricorrenti e le soluzioni ad esse
- In seguito presenteremo ulteriori tecnologie usate per *l'analisi dei dati e l'estrazione di informazione*, che useremo per aggiungere funzionalità avanzate all'applicazione



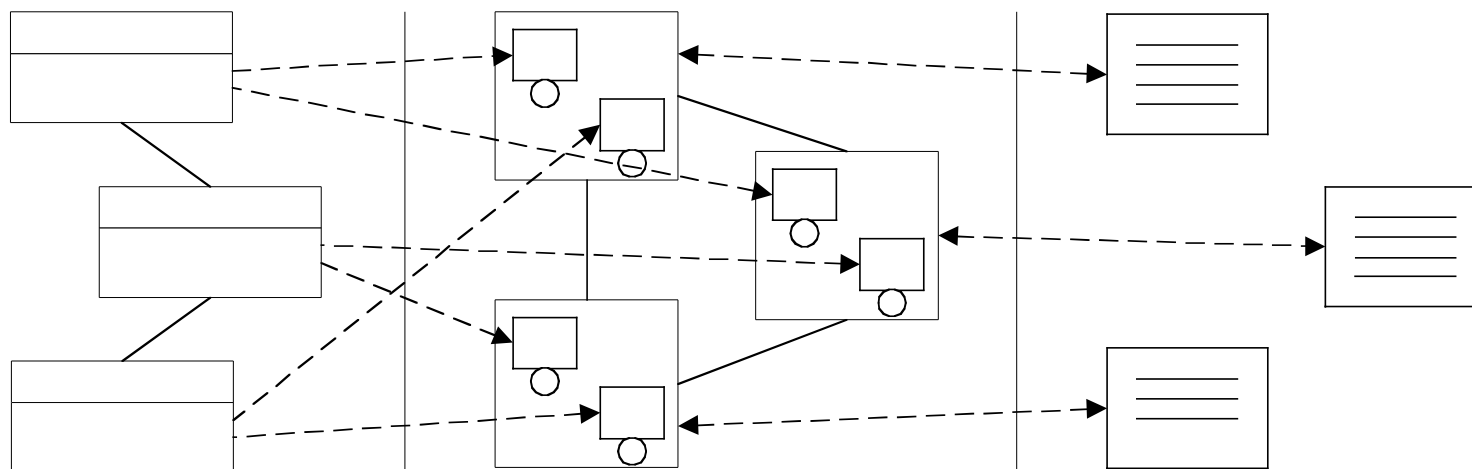
IFML

- *Interaction Flow Modeling Language* (IFML) è un linguaggio visuale standard per la modellazione di applicazioni
 - È un'evoluzione di WebML, linguaggio di modellazione sviluppato appositamente per applicazioni Web data intensive
- Il modello IFML descrive le interazioni dell'utente col front-end dell'applicazione (*User Interactions*)
 - Sono definite le viste che compongono l'interfaccia (*Container*), gli elementi contenuti (*Component*), i collegamenti tra loro (*Navigation* e *Data Flow*), le possibili interazioni (*Event*) e i loro effetti (*Action*)
- Il modello è **indipendente** dalla piattaforma di esecuzione e non impone requisiti sul layout e sullo stile del front-end
 - IFML può descrivere applicazioni che girano su molteplici piattaforme differenti: Web, mobile, desktop ecc.



Creazione di una applicazione UI

Modello Dati + Modello Interazioni + Modello Presentazione



Struttura del contenuto
(entità, relazioni)

Struttura dell'applicazione
(unità, pagine, link, site view)

Presentazione
(stili)

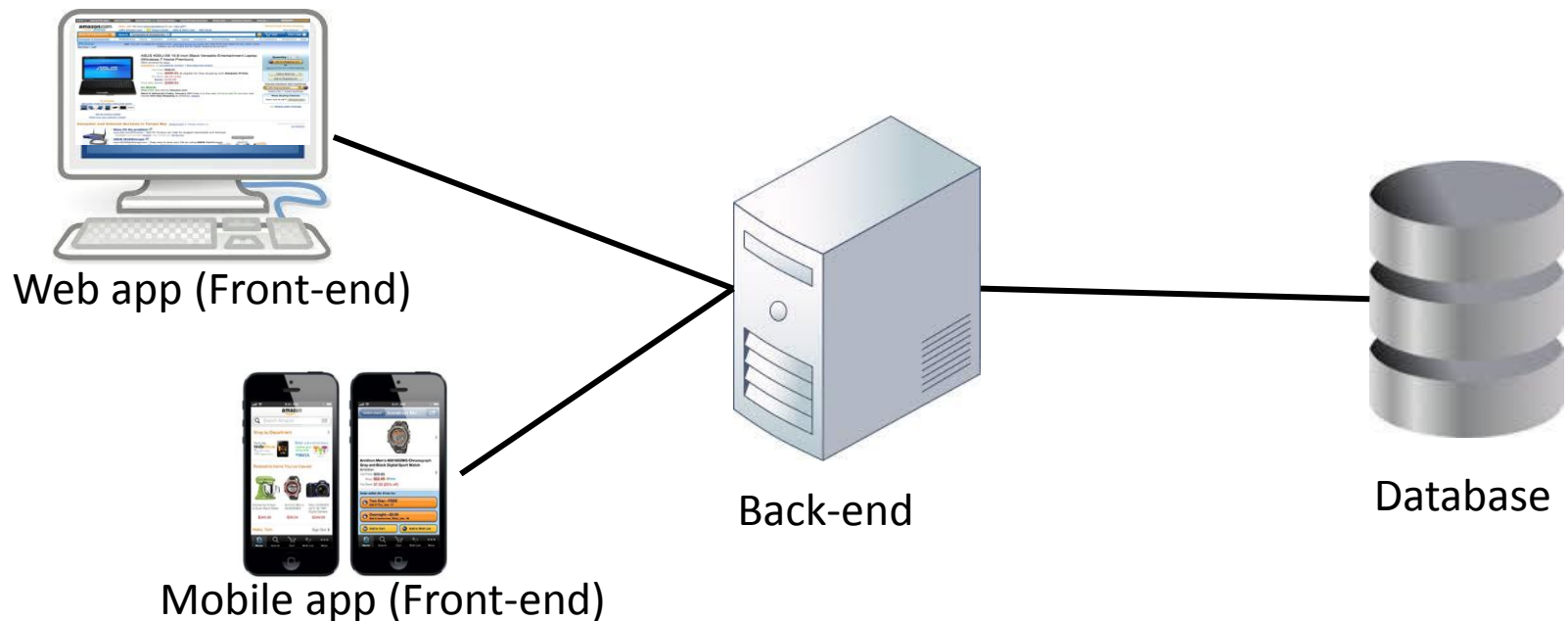
WebRatio

- **WebRatio** è un ambiente basato su Eclipse per lo sviluppo *model-driven* di applicazioni basato su diversi linguaggi
 - ER (*Entity/Relationship*) per il modello dei dati del dominio
 - IFML per la definizione delle User Interaction
 - BPMN (*Business Process Model and Notation*) per la definizione dei processi di business tramite diagrammi di flusso
- WebRatio consente lo **sviluppo agile e rapido** di un'applicazione Web, in modo completamente visuale
- Dal progetto sono **generati in automatico** lo schema del DB e l'implementazione basata su Java EE dell'intera applicazione
 - È possibile definire stili di presentazione e componenti personalizzati
- WebRatio sarà usato per costruire interamente l'applicazione



IFML: Progettazione Model-driven platform-independent

- Al giorno d'oggi tutti i più grandi e-store permettono la navigazione e l'utilizzo sia da web app che da mobile
- Con IFML e WebRatio è possibile modellare la parte UI indipendentemente dalla piattaforma di destinazione e generare automaticamente il codice relativo all'app desiderata



Database Relazionali

- I database basati sul **modello relazionale** costituiscono l'approccio più comune alla memorizzazione di grandi masse di dati con frequenti operazioni di lettura e scrittura
- Uno dei RDBMS più usati è **PostgreSQL**: software libero aderente allo standard SQL e con funzionalità avanzate
- L'uso di un database relazionale richiede la definizione di uno **schema dei dati**
 - Dei **vincoli** devono essere definiti per garantire la correttezza dei dati
 - L'uso di opportuni **indici** è fondamentale per l'efficienza
- **JDBC** è l'interfaccia standard di Java per l'accesso ai RDBMS
- Vedremo come si costruirebbe da zero lo schema del database usato nell'applicazione, completo di vincoli e indici



Applicazioni Web

- I servizi fruibili via **Web** (social network, e-commerce ecc.) sono tra gli esempi più comuni di applicazioni data intensive
 - Usate da molteplici utenti contemporaneamente, che consumano e producono dati in continuazione
- Diverse applicazioni Web sono basate sulla piattaforma **Java**
 - in particolare applicazioni con forti requisiti di affidabilità ed efficienza
- Vedremo le tecnologie di base per le applicazioni Web in Java
 - **servlet** per la logica applicativa (*controller*)
 - **JSP** per la creazione dinamica di pagine Web (*view*)
- Queste saranno utilizzate per la costruzione dell'applicazione di e-commerce presentata come caso di studio



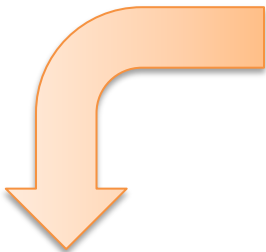
Object-Relational Impedance Mismatch

- L'uso di un database relazionale per la persistenza di dati di un'applicazione object-oriented comporta problemi dati dalla discrepanza tra i due paradigmi, detta *impedance mismatch*
- Nel paradigma object-oriented abbiamo oggetti identificati dalla loro posizione in memoria, spesso composti da altri oggetti (collegati tramite puntatori) accessibili tramite metodi
- Nel database relazionale, i dati corrispondenti ad un oggetto composto sono distribuiti su diverse tabelle, identificati e riferiti tramite chiavi: sono necessari join tra tabelle
- La conversione dei dati tra i due modelli richiede soluzioni non banali da progettare



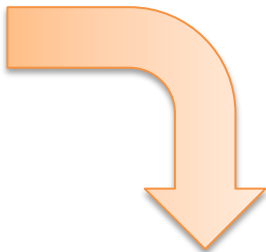
Esempio: Ordine ad Oggetti e su Database Relazionale

ad oggetti



Ordine #6758		
Utente: Bill		Data: 4/3/2016
prodotto	quantità	prezzo
E.T. [DVD]	1	€ 2,99
Matrix [DVD]	1	€ 4,99
...

relazionale



(Order)

number: 6758

date: 4/3/2016

user: (User) Bill

items: 1x (Product) E.T. [DVD]

1x (Product) Matrix [DVD]

...

```
class Mapper {  
  order read(...) {...}  
  void write(...) {...}  
}
```

id	name	...
123	Bill	...
...
order	item	...
3854	235	...
3854	568	...
...

id	date	user	...
...
6758	4-3-16	123	...
...			
...			
Id	name		...
235	E.T. [DVD]		...
568	Matrix [DVD]		...
...



Object-Relational Mapping

- Con *object-relational mapping* ci si riferisce alle soluzioni a livello di progetto e implementazione adottate per risolvere l'impedance mismatch, per cui esistono diversi approcci
 - Devono permettere di leggere e scrivere oggetti persistenti sul DB, garantendo consistenza, accessi concorrenti tramite transazioni ecc.
- Le soluzioni basate su *Data Access Object* prevedono la creazione di uno strato dell'applicazione che incapsuli la logica di accesso al database e la separi dal resto dell'applicazione
- Vedremo in primo luogo come realizzare uno strato DAO per la nostra applicazione basato su JDBC
- Si vedrà che l'implementazione dei DAO richiede di scrivere grandi quantità di codice: non è l'approccio più conveniente



Framework per la Persistenza

- I **framework di persistenza** forniscono un'implementazione dei meccanismi tipici di object-relational mapping, riducendo notevolmente il carico di lavoro degli sviluppatori
- Una volta configurato correttamente in base alle esigenze e al modello dei dati della propria applicazione, un framework offre un API di alto livello per gestire oggetti persistenti su DB
- Come software di riferimento useremo **Hibernate**, un framework di persistenza open source ampiamente diffuso
- Vedremo come sia possibile sostituire il DAO basato su JDBC con uno basato su Hibernate, più rapido da implementare
 - Vedremo nel dettaglio la dichiarazione del mapping tra classi Java e tabelle del DB e l'uso delle API per la gestione ai dati persistenti



Suggerimenti di Acquisto Impersonali: Regole Associative

- I dati raccolti da un'applicazione data intensive possono essere analizzati per estrarre informazioni di alto livello
- Nel caso del sito di e-commerce, possiamo analizzare i dati generati dall'attività degli utenti per **suggerire prodotti** che possono potenzialmente essere interessati ad acquistare
- Dall'analisi dei prodotti venduti nei singoli ordini, è possibile estrarre **regole associative** che indichino quali prodotti siano **frequentemente venduti insieme** ad altri
- Vedremo come realizzare un semplice algoritmo per individuare, per ciascun prodotto, quelli maggiormente correlati ad esso



Suggerimenti di Acquisto **Personal**i:

Recommendation di Prodotti

- Analizzando le recensioni date dagli utenti, i sistemi di **recommendation** possono fornire **suggerimenti personalizzati per ciascun utente** in base ai propri gusti
 - Diversi delle regole associative, che non considerano i singoli utenti
- **RecDB** è un'estensione integrata in PostgreSQL che consente di ottenere recommendation direttamente dal DB, tramite semplici query SQL con una clausola aggiuntiva
- **Apache Mahout** è una libreria Java open source che fornisce diversi algoritmi di recommendation configurabili, con varie possibili fonti di dati e funzionalità per valutarne l'accuratezza
- Vedremo come integrare le recommendation nel nostro sito, mostrando suggerimenti precalcolati periodicamente



Full Text Search

- I dati testuali di un'applicazione (descrizioni, recensioni, ...) sono *destrutturati*, non sono analizzabili direttamente come i dati strutturati (numeri, date, ...) e devono essere invece convertiti in rappresentazioni appropriate
- I sistemi di *Full Text Search* consentono di indicizzare grandi moli di dati testuali ed effettuare in essi ricerche di singole parole, frasi, parole simili ecc.
- *Apache Lucene* è una libreria Java open source per costruire indici di documenti testuali ed effettuare ricerche su essi
- *Hibernate Search* è un'estensione di Hibernate che sfrutta Lucene per consentire la Full Text Search sui dati persistenti
- Integreremo Search nella nostra applicazione per consentire la ricerca full text di prodotti e recensioni



Analisi della Customer Satisfaction

- Una volta strutturati, i dati testuali possono essere analizzati per estrarre conoscenza potenzialmente utile
- Dall'analisi delle recensioni scritte dai clienti, è possibile **dedurre il loro grado di soddisfazione** nei confronti dei prodotti acquistati, etichettandolo come positivo o negativo
- Un possibile approccio consiste nell'individuare **parole note a priori** che esprimono sentimenti positivi e negativi e valutare ogni recensione in base alla frequenza con cui appaiono
- Un approccio più avanzato è basato sul **machine learning**: un algoritmo di apprendimento analizza recensioni pre-etichettate ed estrae in automatico un modello di conoscenza utilizzabile per dedurre la polarità di altre recensioni



Organizzazione del Corso

Lezioni (secondo semestre)

- 50 ore tra lezioni in aula e esercitazioni in laboratorio

Modalità di esame (6 CFU)

- Prova orale con:
 - Discussione di progetto di laboratorio di gruppo (1-3 studenti) concordato con il docente, su uno o più argomenti del corso, con un argomento per ogni partecipante (gruppo con 3 membri -> 3 argomenti).
 - Domande sul programma del corso

Materiale didattico

- Slide fornite dal docente
- Riferimenti bibliografici (vedi prossime slide)
- Software e set di dati disponibili in laboratorio e scaricabili gratuitamente



Riferimenti (1)

- Introduzione
 - I. Gorton, D. K. Gracio; “Data-Intensive Computing – Architectures, Algorithms and Applications”; Cambridge University Press; 2013
 - M. Kleppmann; “Designing Data-Intensive Applications”; O’Reilly; 2014
- IFML e WebRatio
 - Specifica OMG di IFML: <http://www.omg.org/spec/IFML/>
 - M. Brambilla, P. Fraternali; “Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML”; Morgan Kaufman; 2014. Link: <http://www.sciencedirect.com/science/book/9780128001080>
 - IFML: <http://www.ifml.org/>
 - WebRatio: <http://www.webratio.com/>



Riferimenti (2)

- Database relazionali
 - PostgreSQL: <http://www.postgresql.org/>
- Sviluppo Java Web
 - Documentazione API di Java EE: <https://docs.oracle.com/javaee/7/api/>
- Object-Relational Mapping
 - Hibernate: <http://hibernate.org/orm/>
 - C. Bauer, G. King, G. Gregory; “Java Persistence with Hibernate”; Manning; 2013



Riferimenti (3)

- Recommendation e machine learning
 - F. Ricci, L. Rokach, B. Shapira, P. B. Kantor; “Recommender Systems Handbook”; Springer; 2011
 - RecDB: <https://github.com/Sarwat/recdb-postgresql>
 - Apache Mahout: <http://mahout.apache.org/>
 - S. Owen, R. Anil, T. Dunning, E. Friedman; “Mahout in Action”; Manning; 2012
- Full Text Search
 - Apache Lucene: <http://lucene.apache.org/core/>
 - Hibernate Search: <http://hibernate.org/search/>

