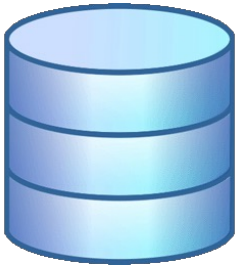
PERSISTENZA DEGLI OGGETTI

Persistenza

- **persistènza** s. f. [der. di *persistere*]. – Il fatto di persistere, di protrarsi nel tempo per una durata notevole e senza variazioni sensibili

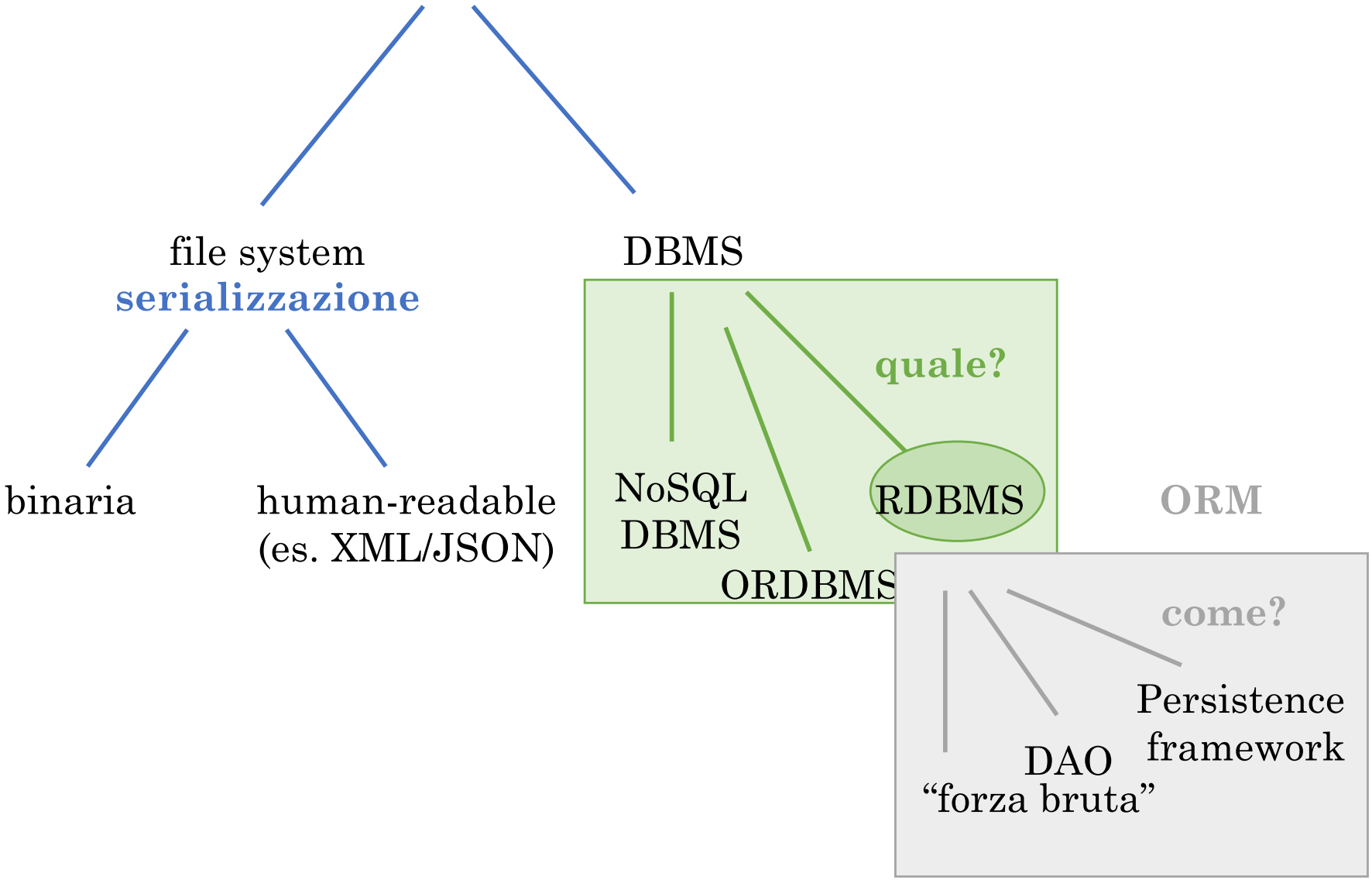
- **Persistenza:** salvataggio dei dati dalla memoria di lavoro dell'applicazione che li ha creati in modo che possano essere utilizzati in successive esecuzioni dell'applicazione
- far sopravvivere dei dati all'esecuzione dell'applicazione
  - **save:** salvataggio dei dati in una memoria non volatile
    - su un file system o su un database
  - **restore:** loro recupero successivo in una nuova esecuzione dell'applicazione



Persistenza

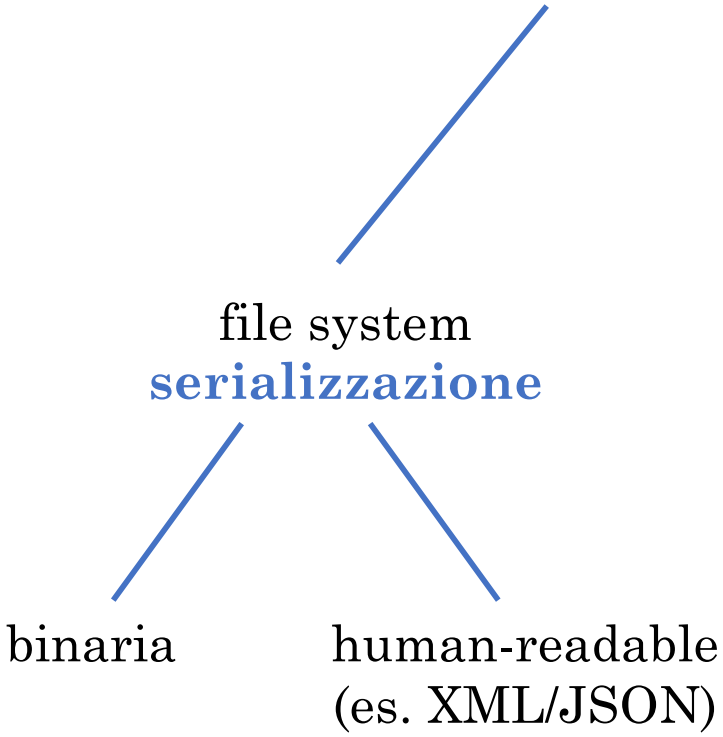
- La maggior parte dei sistemi sw ha requisiti di persistenza
  - Task critici delle applicazioni sono spesso legati alla persistenza dei dati
  - Nei sistemi object-oriented, ci sono diversi modi in cui gli oggetti possono essere resi persistenti
  - La scelta del persistence method (modo di rendere persitenti gli oggetti) è una parte importante del design di un’applicazione

Persistenza



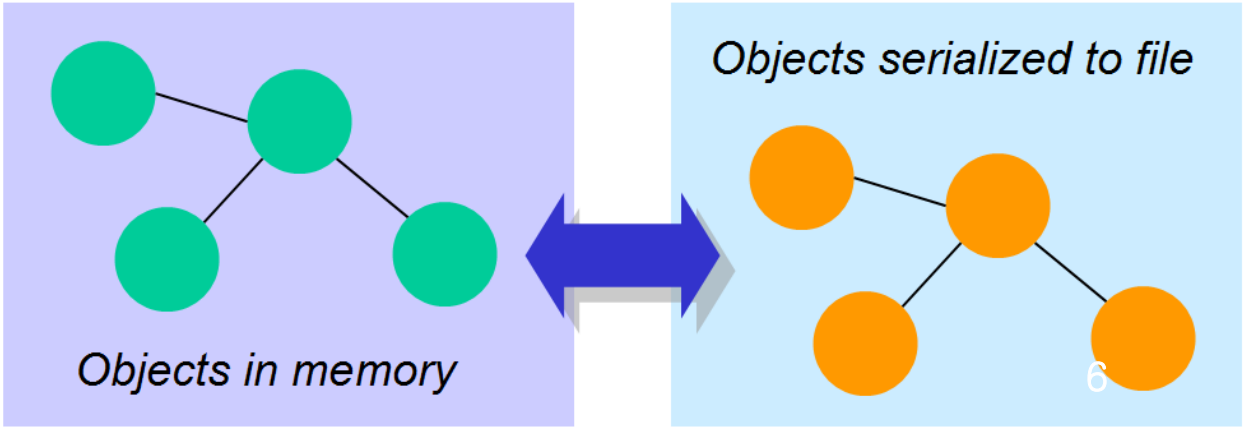
Nota: come? ha senso anche negli altri casi

Persistenza



Serializzazione

- **Serializzazione:** processo per salvare un oggetto in un supporto di memorizzazione lineare (ad esempio, un file o un'area di memoria), o per trasmetterlo su una connessione di rete
- può essere in forma **binaria** o può utilizzare **codifiche testuali** (ad esempio i formati XML o JSON) direttamente leggibili dagli esseri umani
- l'oggetto potrà poi essere successivamente ricreato nello stesso identico stato dal processo inverso (**deserializzazione**)



Serializzazione Binaria

- **`Serializable`**
- **`java.io.Serializable`**

- Cosa accade per un grafo di oggetti?
  - Chiusura transitiva

- Cosa accade se si vuole serializzare solo parte del grafo?
  - Possibilità di specificare attributo come transiente

Serializzazione Human Readable (XML/JSON)

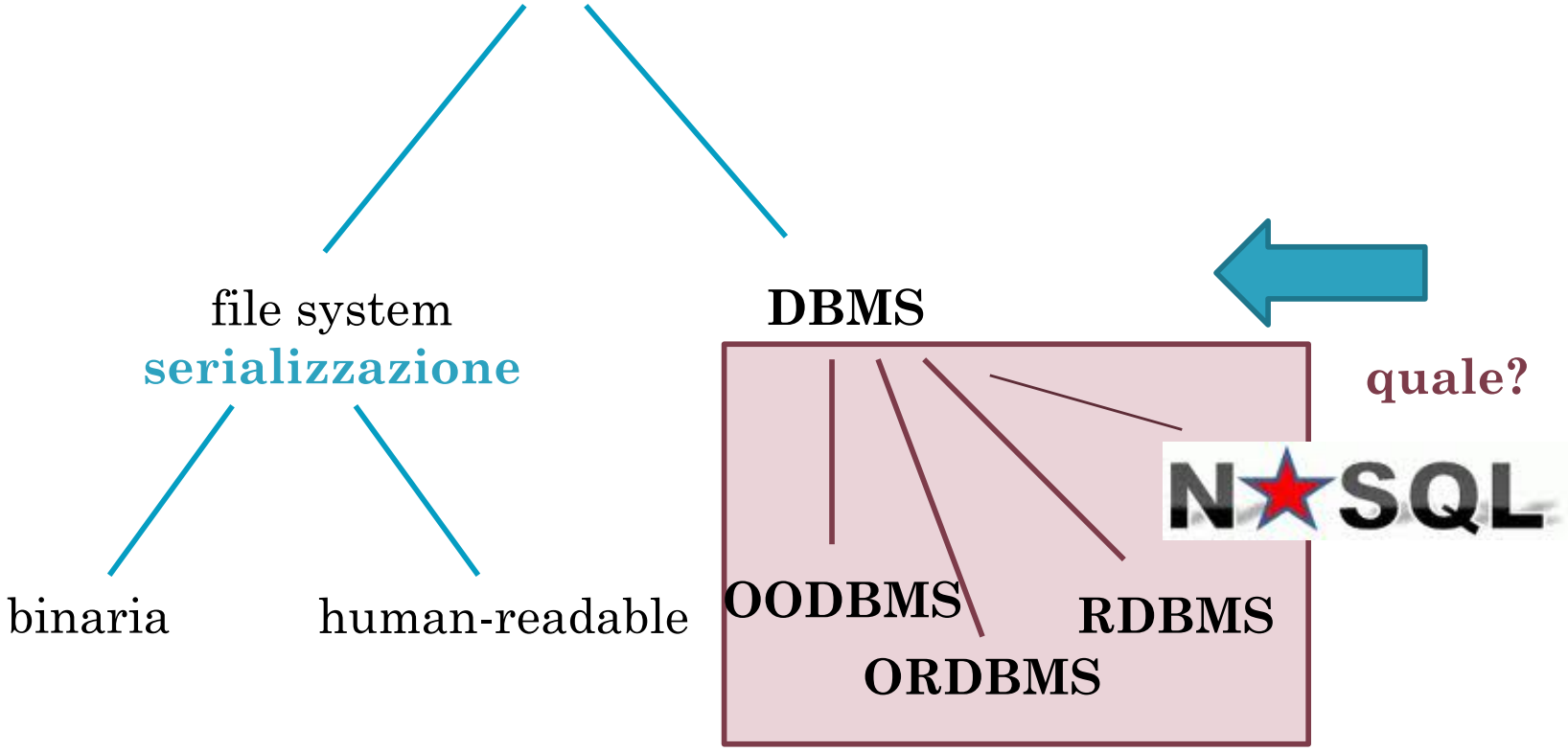
- Librerie (es Xstream, GSON)

- **`toXML, toJSON`**
- Producono stringa da oggetto

- **`fromXML, fromJson`**
- Ricostruiscono oggetto da stringa

- Diversi modi per gestire riferimenti tra oggetti

Persistenza



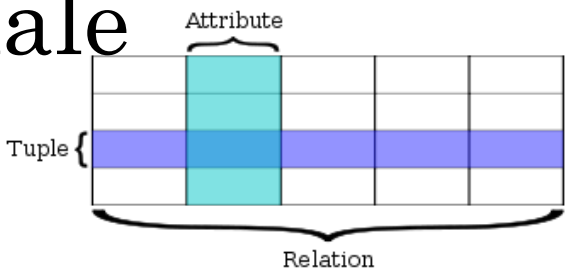
Uso di Basi di Dati

- La maggioranza delle applicazioni Client-Server usano un DBMS come supporto alla memorizzazione persistente dei dati

- Esigenze diverse nella gestione della persistenza
  - Possibilità di interrogazione
  - Supporto di transazioni
  - Accesso concorrente, sicuro, ...

  Vedi introduzione corso BD ☺

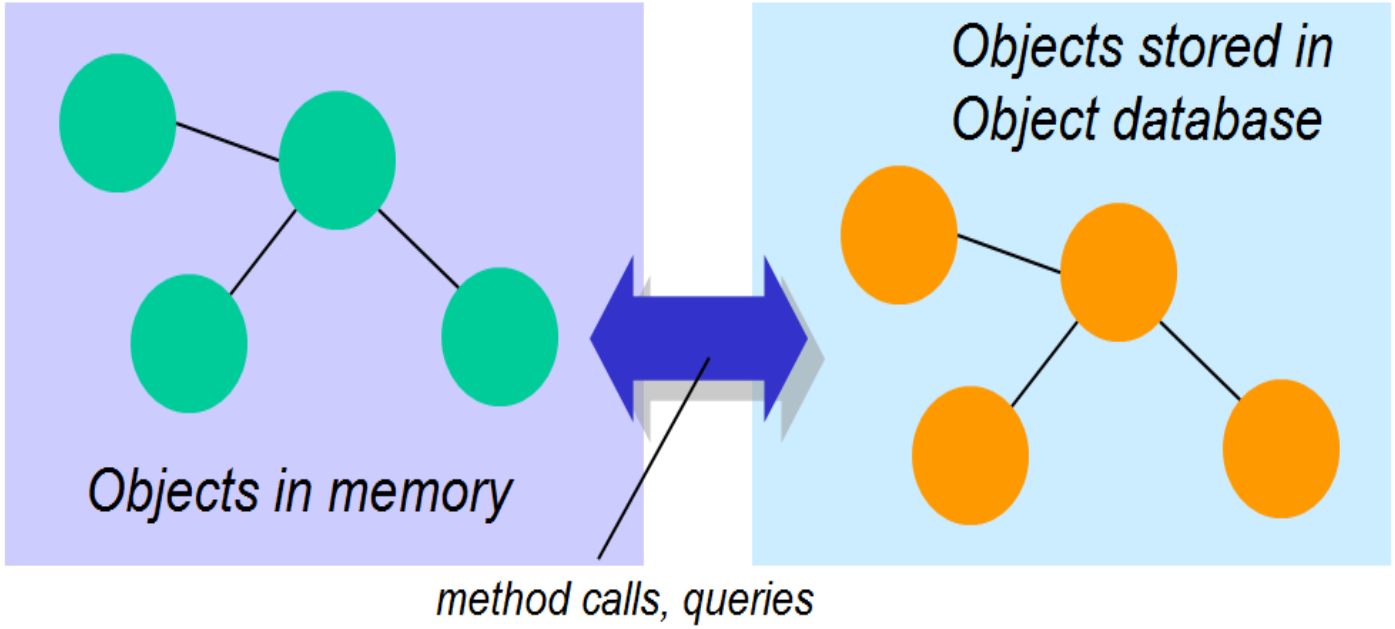
RDBMS e Modello Relazionale



- Dati rappresentati da semplici strutture tabellari: le relazioni [E. Codd, 1970]
- Associazioni rappresentate tramite il meccanismo chiavi-chiavi esterne
- Accesso ai dati tramite linguaggi dichiarativi di alto livello (SQL)
- Separa la rappresentazione logica e fisica dei dati
- Grandissimo successo a livello commerciale (Oracle, DB2, SQL Server, etc 90% mercato DBMS)
- Le applicazioni utilizzano SQL all'interno di/lo combinano con un (altro) linguaggio di programmazione

Alternative al Modello Relazionale

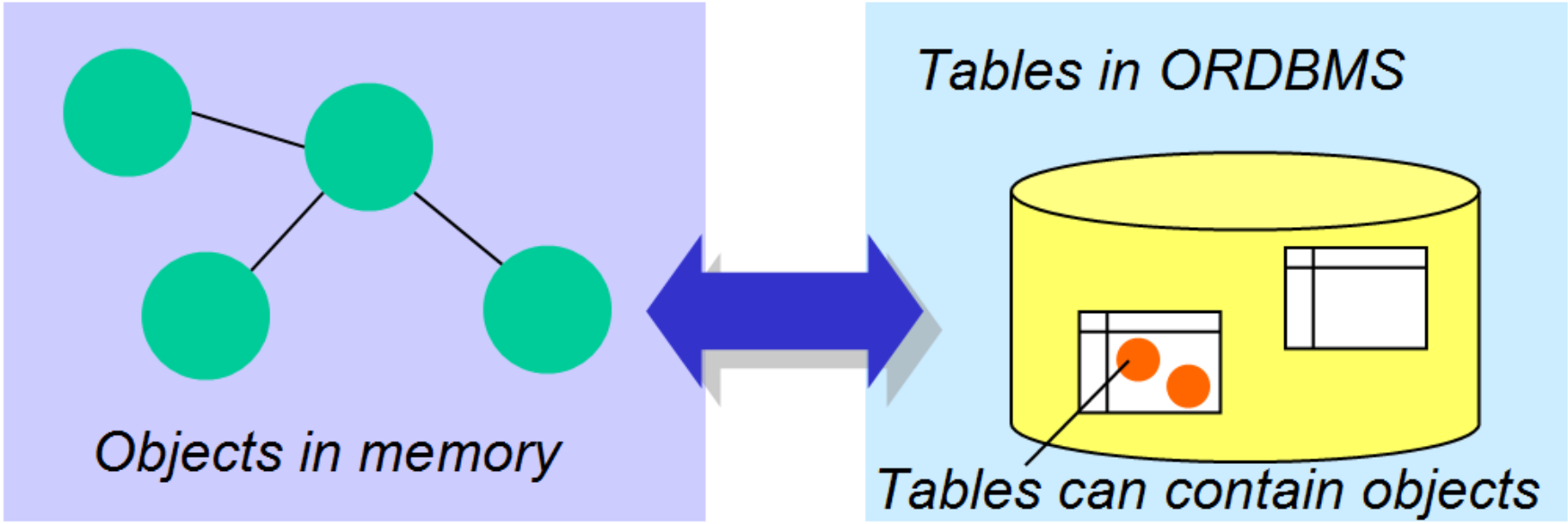
- **Object Data Model – OODBMSs**
  - offre **persistenza ortogonale** agli oggetti
  - Atkinson et al. Manifesto del 1989
  - standard proposto da Object Data Management Group (ODMG)
  - interrogazione principalmente navigazionale



Alternative al Modello Relazionale

- **Object-Relational Model**
  - modello ibrido (o “post-relazionale”) – gli oggetti possono essere memorizzati in tabelle di basi di dati relazionali
  - Un passo oltre i modelli Nested/Extended Relational (no 1NF)
  - Stonebraker (1990) e altri
  - Lo standard SQL ha caratteristiche OR dalla versione del 1999, raffinate e ampliate in quella del 2003
  - Tutti i principali RDBMSs commerciali quali Oracle, DB2, SQL Server, … supportano caratteristiche OR

ORDBMS


Tables in ORDBMS
Objects in memory
Tables can contain objects

Modello Object Relational

- Il modello object relational è un'estensione del modello relazionale con le seguenti caratteristiche:
  - superamento della prima forma normale
  - **nuovi tipi di dato user-defined**:
    - tipi strutturati, tipi riga, tipi semplici
    - tipi riferimento, tipi collezione
  - **metodi** per modellare le operazioni sui tipi definiti dall'utente
  - nuovi modi per modellare le **associazioni**
  - **ereditarietà** di tipi e tabelle
- Ha avuto il vantaggio commerciale di essere supportato dai maggiori RDBMS vendor

ORDBMS – Esempio SQL:2003

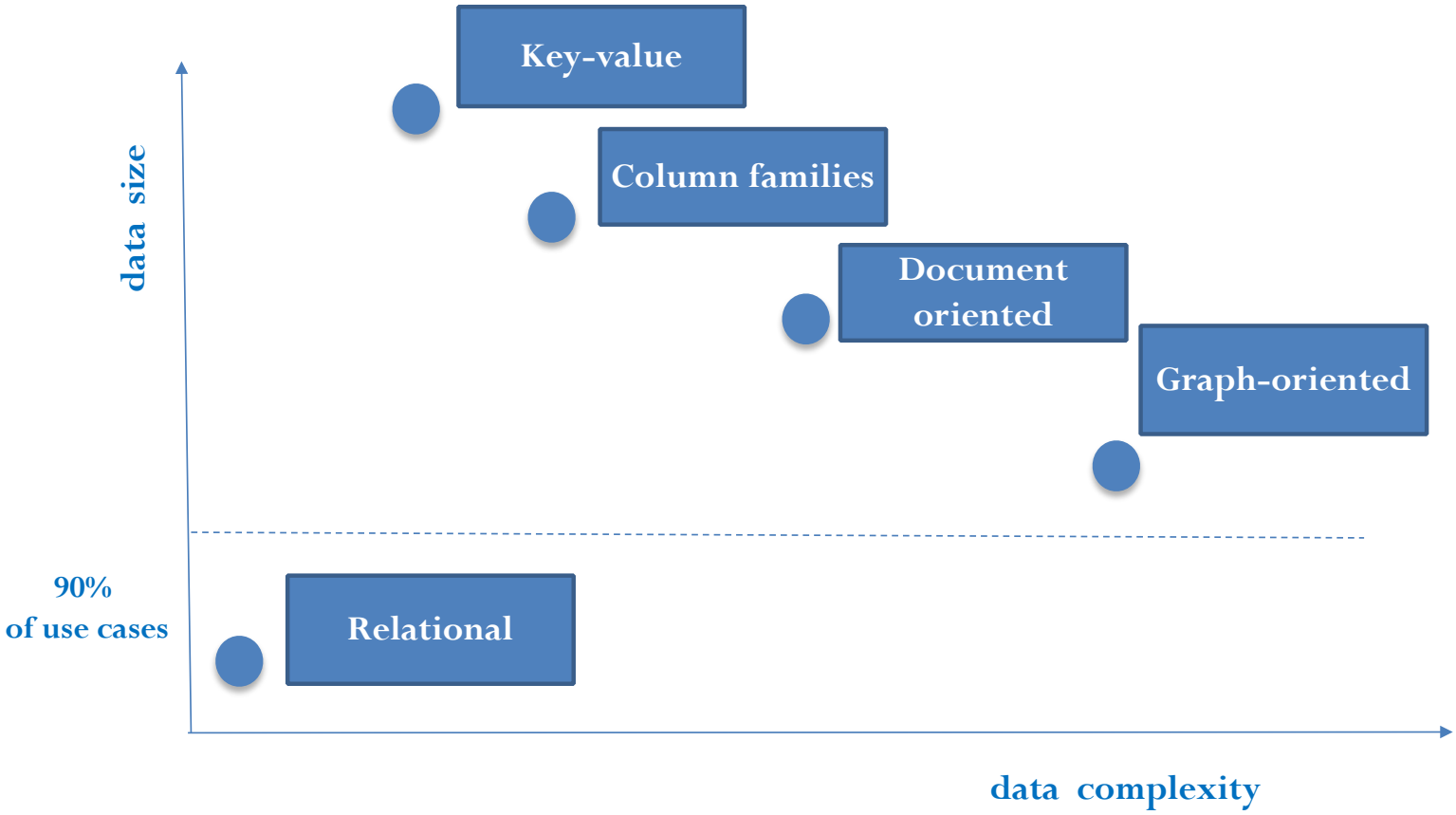
```
CREATE TYPE t_cliente AS
  (codCli   DECIMAL(4),
   nome           VARCHAR(20),
   telefono       CHAR(15),
   dataN    DATE,
   telefono CHAR(15) MULTISET,
  residenza ROW (citta VARCHAR(20),
                 via VARCHAR(20),
                 no VARCHAR(10),
                 cap INTEGER))
NOT INSTANTIABLE
NOT FINAL
INSTANCE METHOD punteggio() RETURNS
  INTEGER;

CREATE  TABLE Clienti OF t_cliente;
```

```
CREATE TYPE t_VIP
UNDER t_cliente AS
  (bonus INTEGER)
NOT FINAL;

CREATE TABLE VIP OF t_VIP
UNDER Clienti;

CREATE INSTANCE
METHOD punteggio()
RETURNS INTEGER
FOR t_cliente
BEGIN
  RETURN (
    SELECT COUNT(*)
          FROM Noleggio
          WHERE
  codCli=SELF.codCli);
END;
```

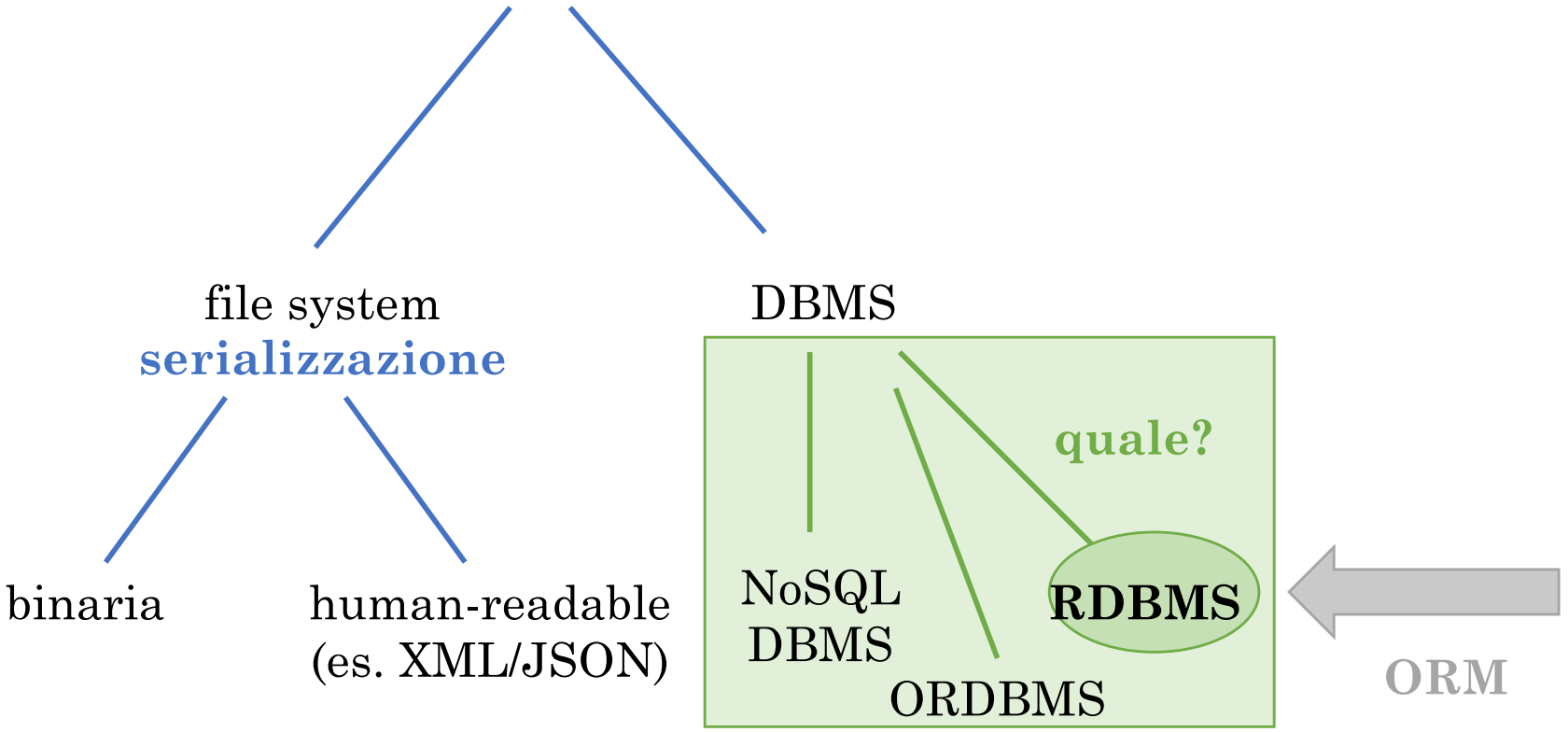
DBMS NoSQL

- Insieme di sistemi anche molto diversi tra di loro
- Termine nasce nel 2009 per denotare un «movimento»
- Motivazioni comuni
- Impedance mismatch
- Da «integration databases» a «application databases»
  - Attack of the **cluster**
  - Attack of **big data**

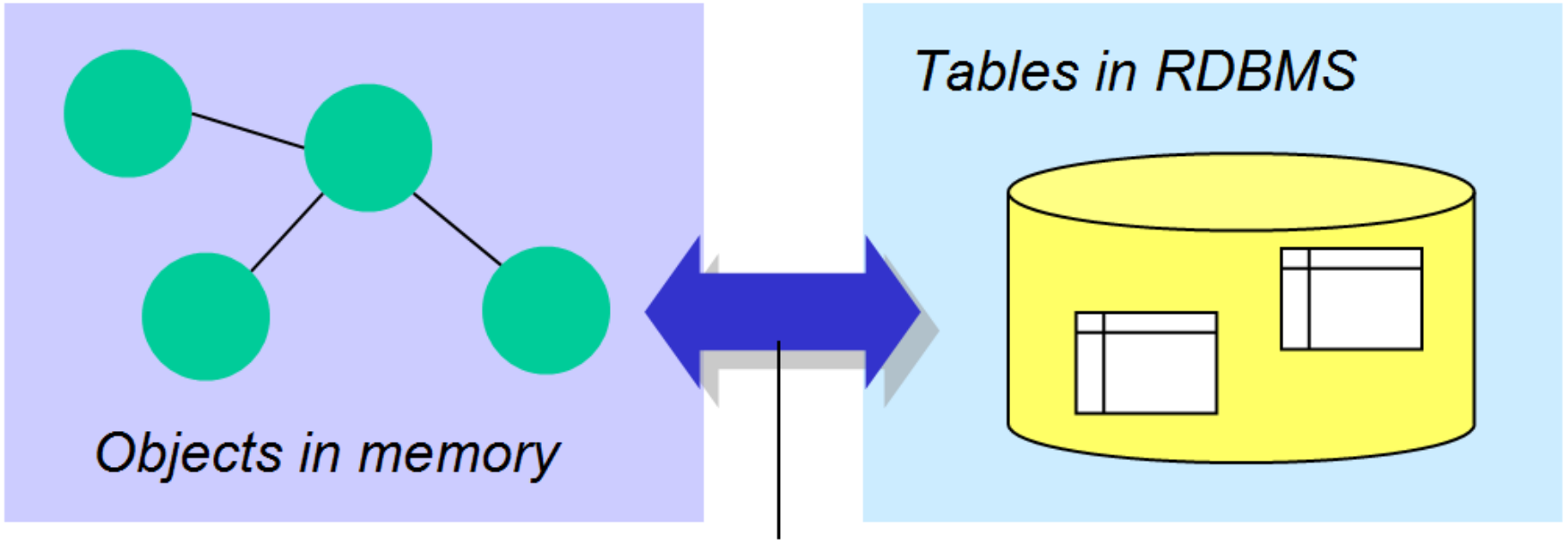
MODELLI DEI DATI NoSQL



Persistenza



OBJECT RELATIONAL MAPPING



OBJECT RELATIONAL MAPPING

- **Impedance mismatch**

  mancata corrispondenza tra modello OO e relazionale
  - Modello OO: descrive il dominio in termini di oggetti e loro proprietà, che possono essere attributi (i.e. valori) o associazioni ad altri oggetti
  - Modello relazionale: descrive il dominio in termini di relazioni tra valori

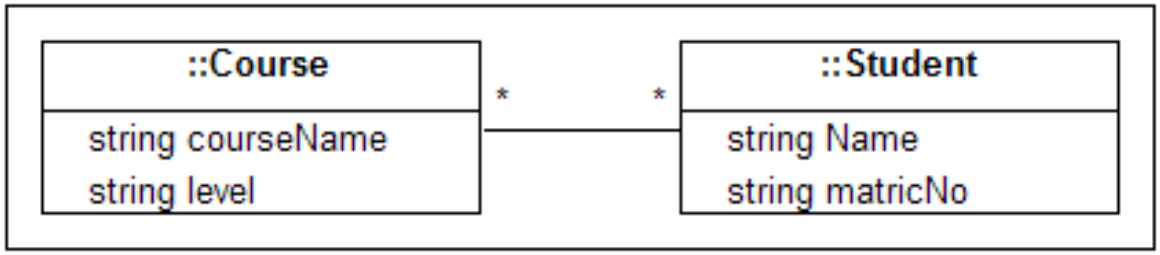
- Differenze fondamentali tra i modelli
  - Coesione
  - Incapsulamento
  - Granularità dei tipi di dato (es. classe indirizzo)
  - Ereditarietà e polimorfismo
  - Identità
  - Navigabilità

OBJECT RELATIONAL MAPPING

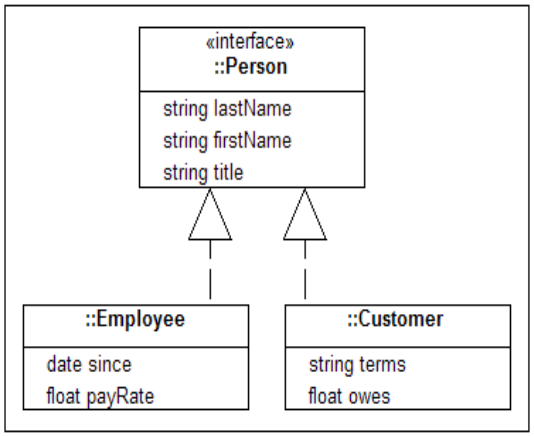
“Usare un database relazionale per supportare la persistenza dei dati contenuti negli oggetti è come rientrare a casa alla sera e, anzichè parcheggiare, smontare pezzo per pezzo la propria auto per poi rimontarla al mattino prima di ripartire per andare al lavoro”

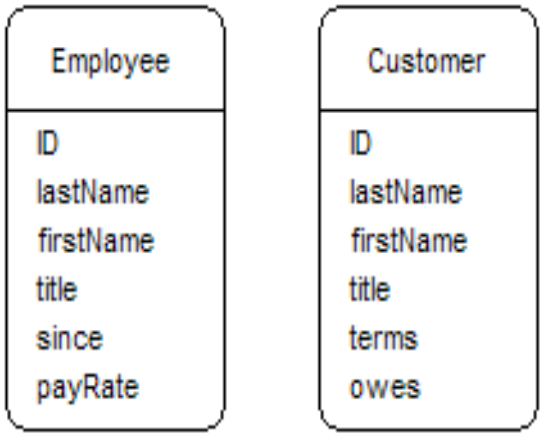
OR MAPPING: BASICS

- Classe → Tabella
  - ogni attributo → Colonna
  - oid (identificatore) → colonna numerica autoincrementata (sequence)
- Attributi di tipo collezione o strutturato → Tabella
- Associazioni one-to-one e one-to-many → Chiave esterna
- Associazioni many-to-many e classi associative → Tabella

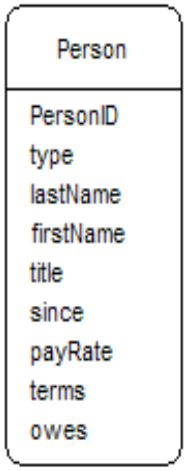


OR MAPPING: INHERITANCE

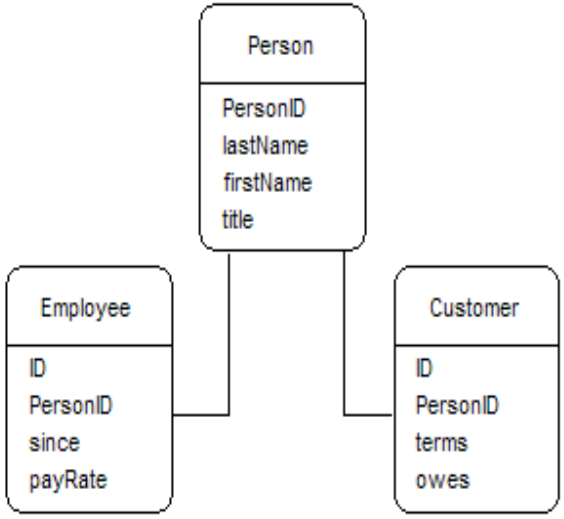




Horizontal mapping



Filtered mapping

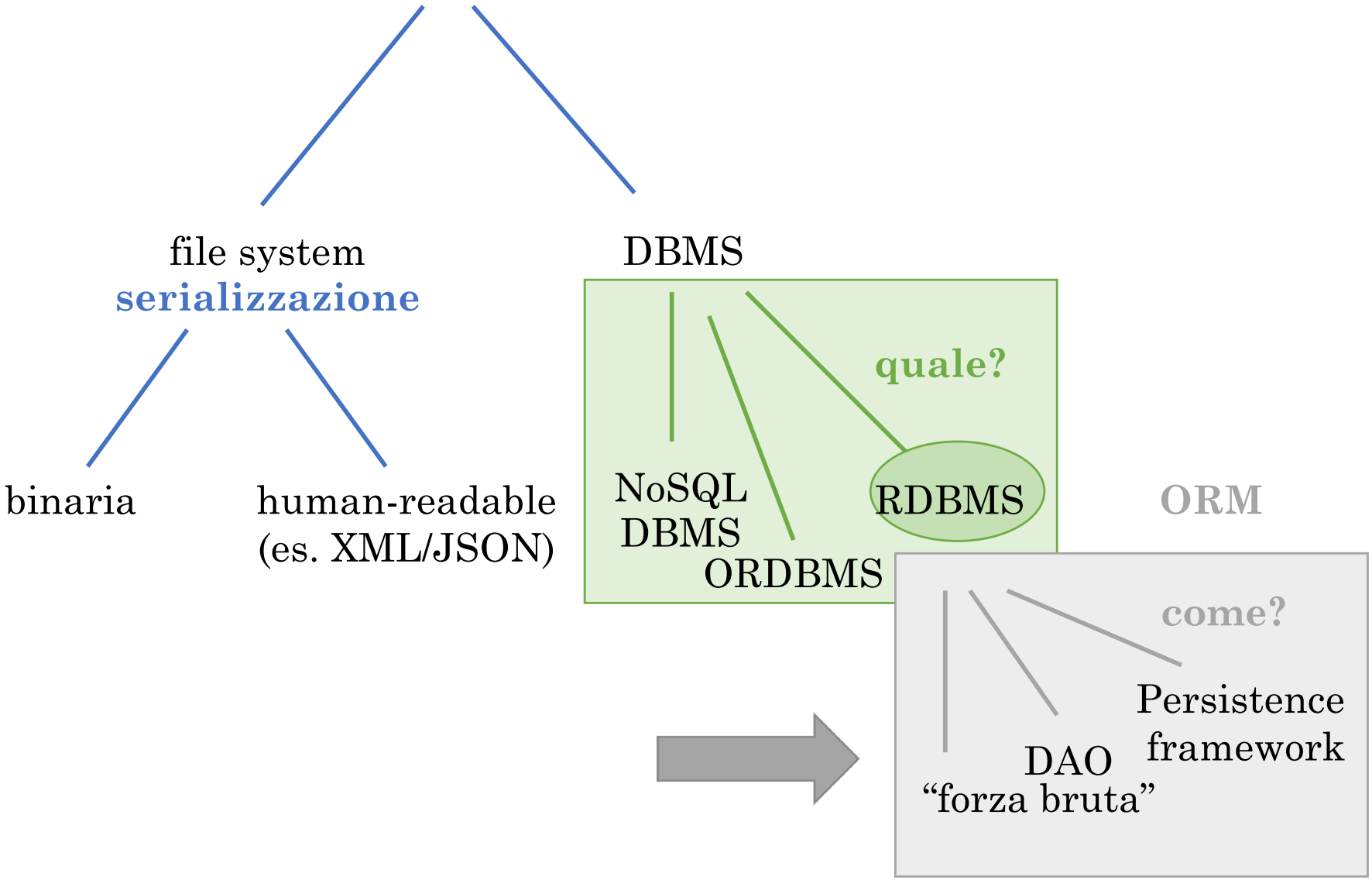


Vertical mapping

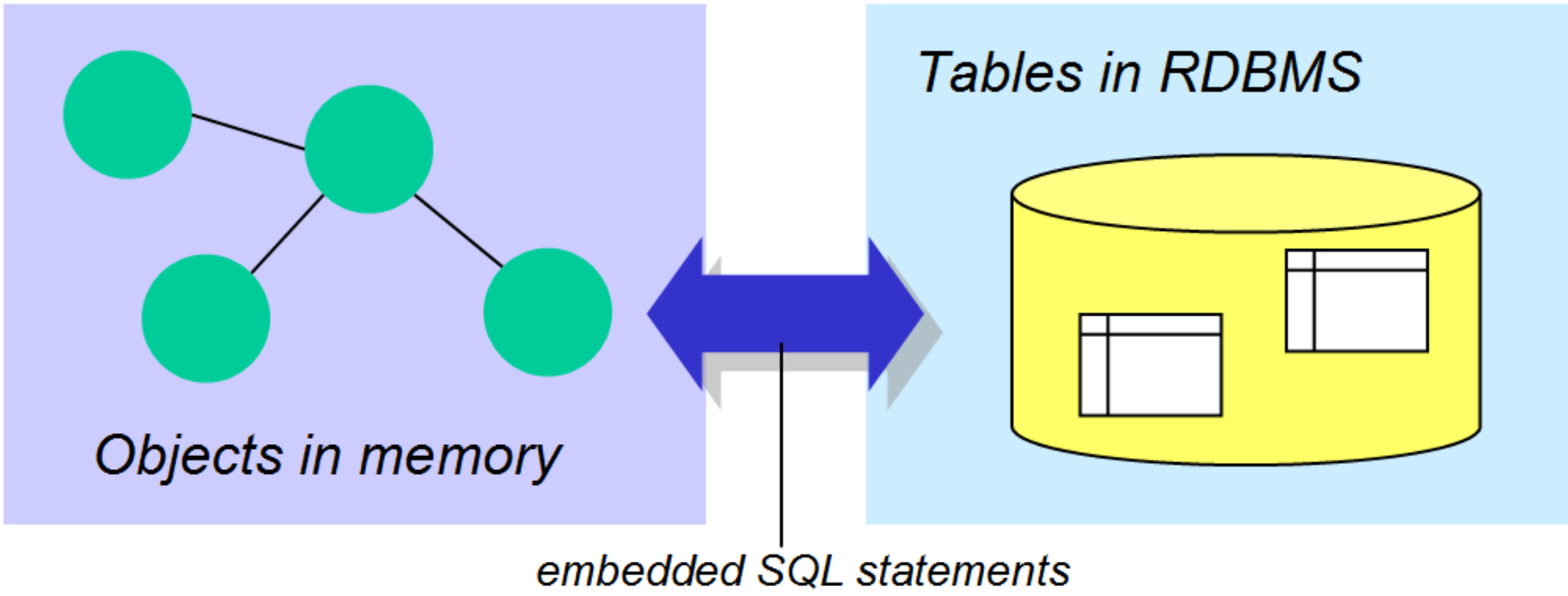
OBJECT RELATIONAL MAPPING

- Progettazione logica di basi di dati vista a BD
- Modello ER (esteso) vs class diagram UML
  - … a parte i vincoli di cardinalità “rovesciati”
  - identificazione (interna, mista, esterna), entità deboli
  - associazioni n-arie

Persistenza



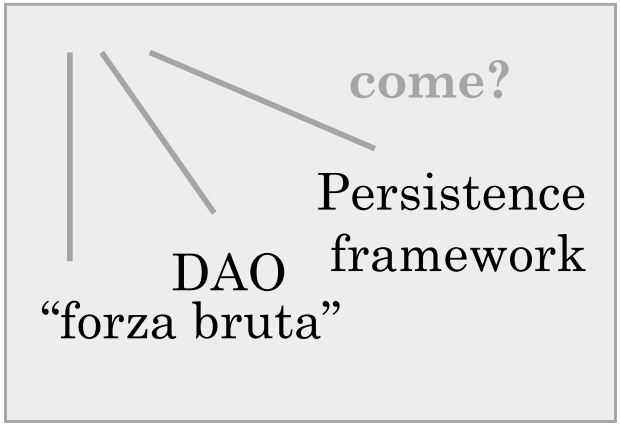
OBJECT RELATIONAL MAPPING



Object Relational Mapping – Come?

- **Gap semantico**, cioè differenza di significato, tra il linguaggio di programmazione che usiamo per scrivere il codice e il modo in cui accediamo ai dati nella base di dati
- Mapping run-time tra il sistema software e la base di dati
- Chi-come si genera questo mapping code?

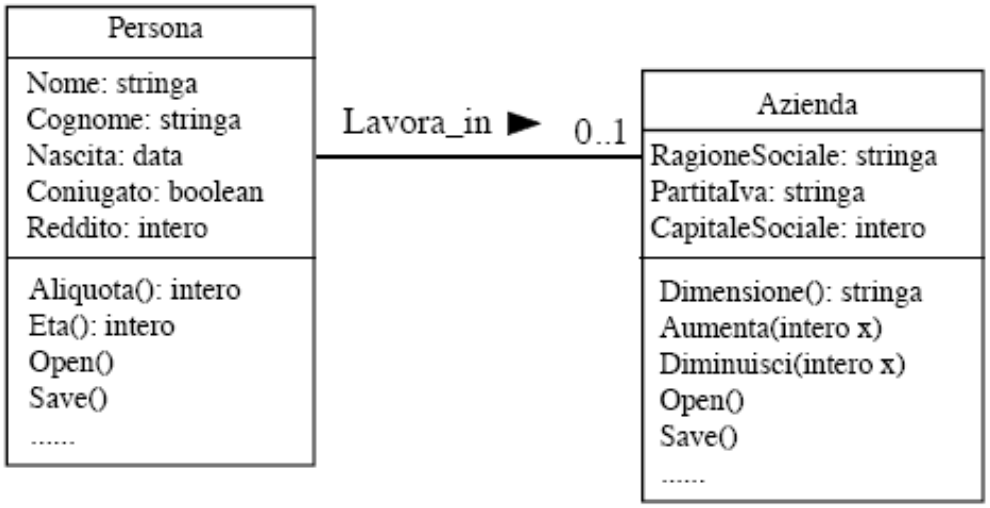
- Due tipologie di problemi
  - Qualità dell'architettura software risultante
  - Qualità del software risultante e processo di sviluppo



Object Relational Mapping – Forza Bruta

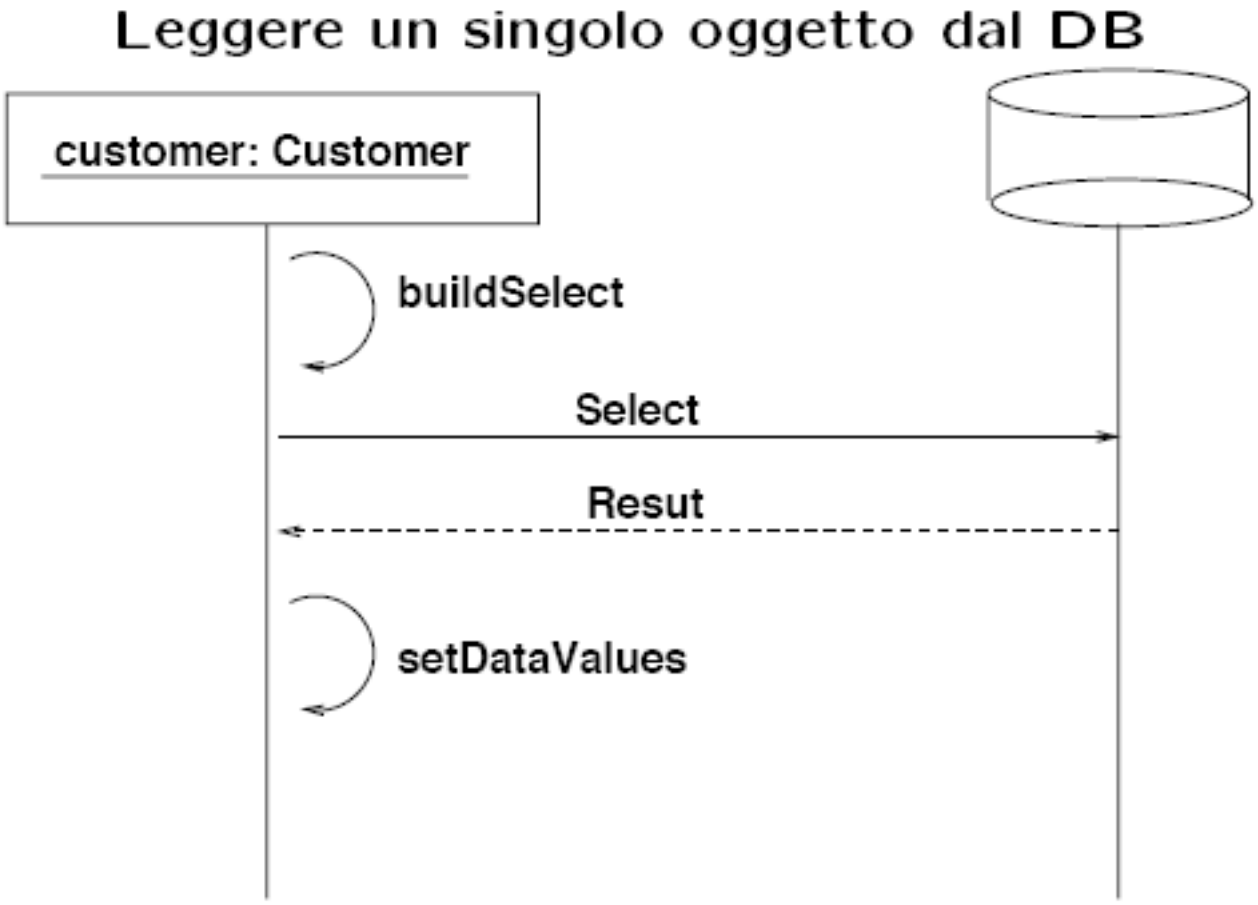
- Le classi dell'applicazione (in particolare le classi di dominio) vengono “equipaggiate” con metodi che interagiscono direttamente con la base di dati, ovunque necessario
- la logica di accesso ai dati sul DB viene incorporata nelle classi di dominio
- è ragionevole quando l'applicazione e sufficientemente semplice e si può fare a meno di uno strato di incapsulamento (persistence classes)

Object Relational Mapping – Forza Bruta



- Nelle classi che modellano oggetti del dominio, si ha un metodo per popolare un oggetto con i dati del DB (Open()), o per rendere persistenti i cambiamenti effettuati sull'oggetto (Save()), etc.

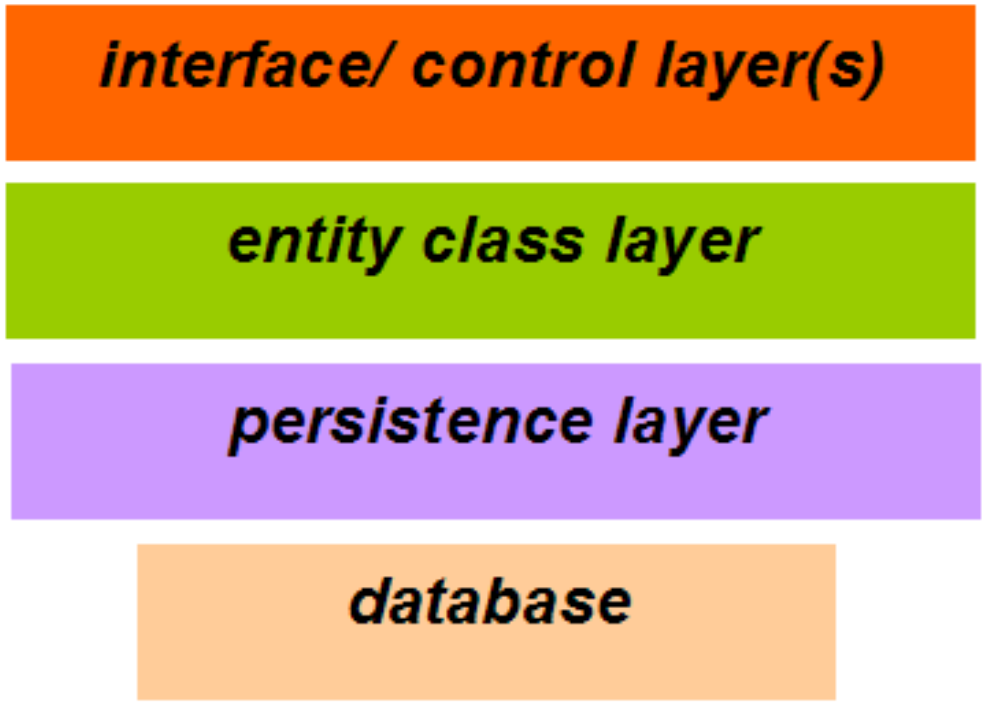
Object Relational Mapping – Forza Bruta



Object Relational Mapping – Forza Bruta

- Non è una strategia di incapsulamento
- Accoppia fortemente lo strato delle classi di dominio al database
- Richiede che chi progetta l'applicazione abbia conoscenza dettagliata del database
- Si offre un canale diretto dalla logica dell'applicazione al DB,
- Violando
  - interfacciamento esplicito: non è definito in maniera chiara il passaggio di informazione fra l'applicazione ed il DB
  - information hiding: non vengono nascosti all'applicazione i dettagli della base dati

Object Relational Mapping – DAO: Persistence Layer

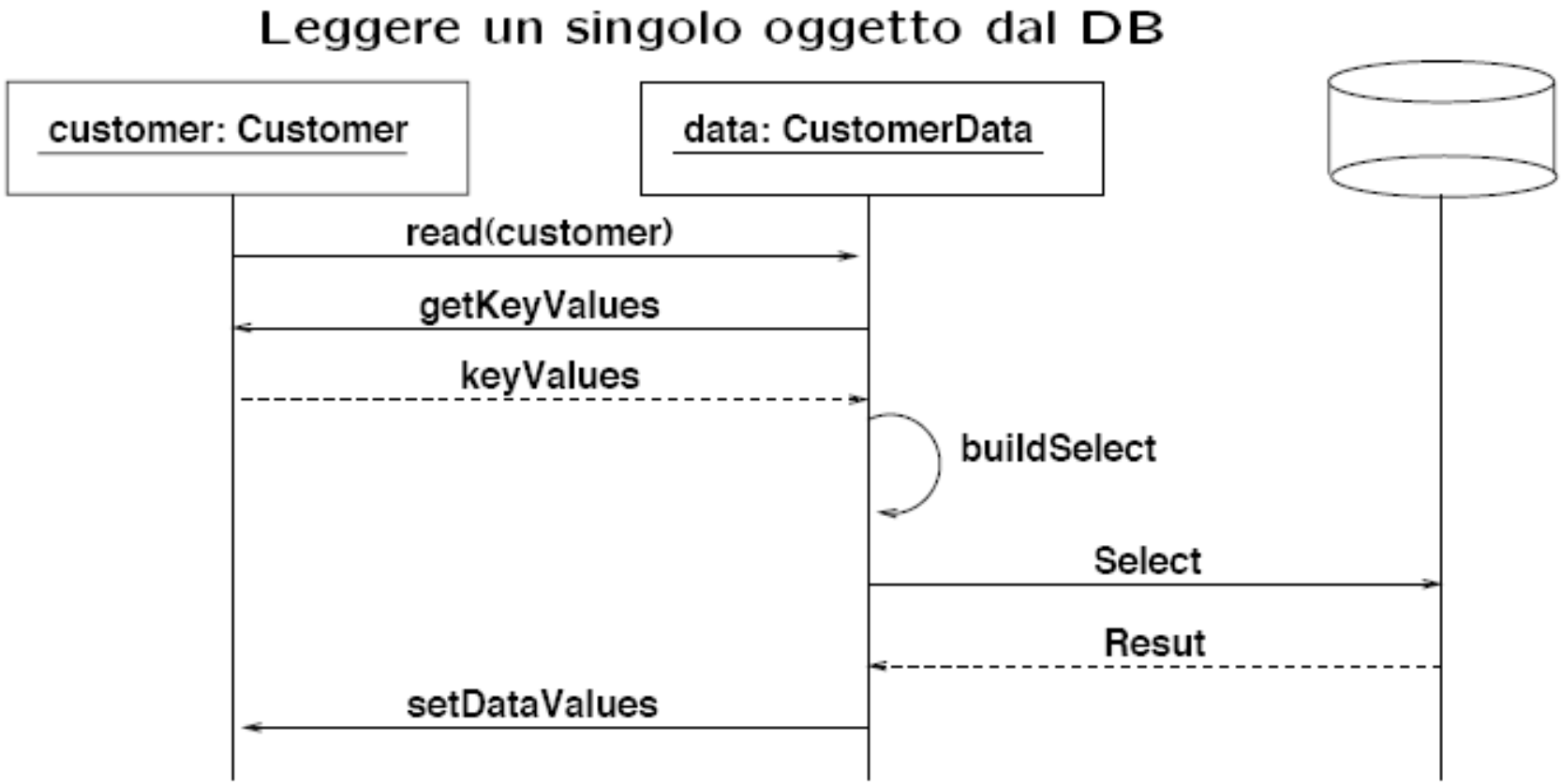


a typical application architecture

Object Relational Mapping – DAO

- Uno strato dell'applicazione (chiamato appunto DAO o persistence layer) demandato completamente a gestire la comunicazione fra l'applicazione ed il DBMS
- Il mapping è realizzato attraverso l'uso di un opportuno linguaggio (e.g., JDBC) per interfacciarsi ad SQL
- l'accesso al DB viene però opportunamente incapsulato nelle classi DAO:
  - nasconde alla logica di business il codice di accesso ai dati
  - fornisce un interfacciamento esplicito del codice
  - migliora la modularità, risolve problemi di
  - accoppiamento tipici dell'approccio forza bruta

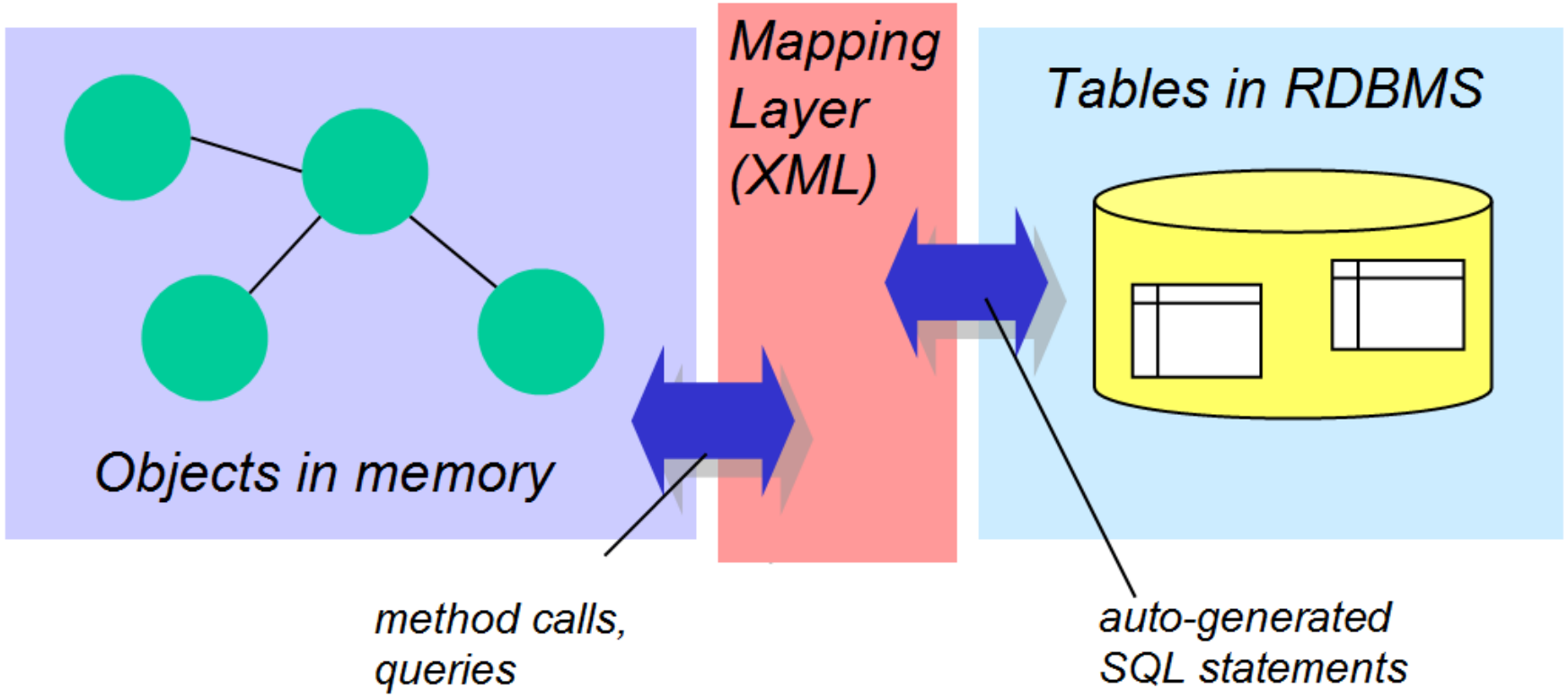
Object Relational Mapping – DAO



Object Relational Mapping – DAO

- Tutta la logica di accesso al DB è completamente incapsulata nelle classi DAO
- Cambiamenti del DB influenzano solo le DAO
- La classe CustomerData si fa carico di gestire il codice SQL, mentre tutto ciò è trasparente rispetto alla classe Customer (che è una domain class), la quale invoca metodi indipendenti dal DB
- L'approccio tipico è quello di avere un DAO per ciascuna domain class

OR Mapping Frameworks



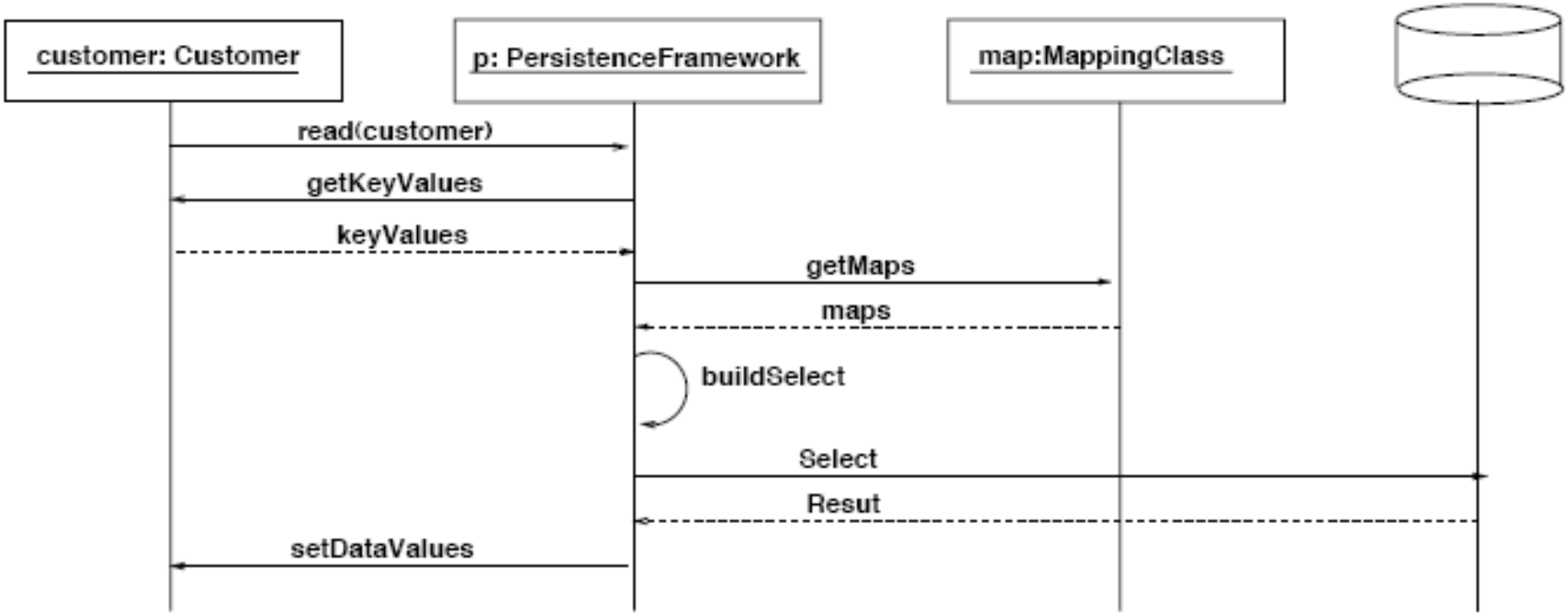
Object Relational Mapping – Persistence Framework

- Il 30-40% di una applicazione JDBC è relativo alla trasformazione di tuple in oggetti e viceversa
- Molti diversi persistence framework/ORM toolkits per i diversi linguaggi oo
- scopo:
  - eliminare codice noioso e ripetitivo che instanzia gli oggetti a partire dalle tuple
  - Isolare lo sviluppatore di applicazioni da estensioni/variazioni SQL vendor-specific
  - permettere allo sviluppatore di applicazioni di sfruttare a pieno l'object-orientation, modellando e gestendo i dati senza essere vincolato alla visione relazionale
- La manipolazione dei dati viene effettuata a livello di oggetti, non (solo) a livello di comando SQL

Object Relational Mapping – Persistence Framework

- Framework predefinito per la gestione della persistenza
  - l'obiettivo è liberare il programmatore quanto più possibile dalla necessità di scrivere codice SQL nell'applicazione
  - il codice SQL viene generato automaticamente sulla base di informazioni di meta-livello fornite dal programmatore (ad es. all'interno di file di configurazione)
  - incapsulamento completo: il programmatore vede il DB solo quando configura il framework
- incapsula pienamente la logica di accesso al DB
- meta-dati rappresentano la corrispondenza tra le classi di dominio e tabelle, nonché le associazioni tra le classi di dominio

Object Relational Mapping – Persistence Framework



Object Relational Mapping – Persistence Framework

- Funzionalità chiave:
  - alcuni oggetti sono inizialmente definiti come persistenti – da quel momento, modifiche a questi oggetti sono ripercosse in maniera trasparente nella base di dati e non è necessario scrivere codice specifico per aggiornare il database
  - è il framework a gestire il mapping tra gli oggetti e le tabelle del database relazionale dove sono effettivamente memorizzati
  - tale mapping è generalmente definito in file XML

- funzionalità di base (CRUD) e altre funzionalità:
  - transazioni
  - gestione della concorrenza
  - caching

Object Relational Mapping – Persistence Framework … Non è tutto così semplice

- Scelte diverse in fase di mapping possono portare a sistemi con prestazioni molto diverse (strutture di memorizzazione e velocità di accesso ai dati)
- Necessità di sperimentazione e tuning, possibilmente con l'aiuto di un mapping tool o di un persistence layer
- Inoltre
  - Lo schema può essere condiviso da altre applicazioni e sotto il controllo di un DBA
  - Il database può essere pre-esistente come sistema legacy che non possiamo modificare
  - Reverse-engineering del mapping

Object Relational Mapping – Persistence Framework … Svantaggi

```
SELECT
[Project9].[ContactID] AS [ContactID],[Project9].[C1] AS [C1],[Project9].[C2] AS [C2],[Project9].[ContactID1] AS [ContactID1],[Project9].[SalesOrderID] AS [SalesOrderID],
[Project9].[TotalDue] AS [TotalDue]
FROM ( SELECT        [Distinct1].[ContactID] AS [ContactID],        1 AS [C1],        [Project8].[ContactID] AS [ContactID1],        [Project8].[SalesOrderID] AS [SalesOrderID],
      [Project8].[TotalDue] AS [TotalDue],        [Project8].[C1] AS [C2]
      FROM
(SELECT DISTINCT   [Extent1].[ContactID] AS [ContactID]
         FROM  [DBA].[Contact] AS [Extent1]
                  INNER JOIN [DBA].[SalesOrderHeader] AS [Extent2]
ON  EXISTS (SELECT   cast(1 as bit) AS [C1]
                FROM    ( SELECT cast(1 as bit) AS X ) AS [SingleRowTable1]
                LEFT OUTER JOIN  (SELECT [Extent3].[ContactID] AS [ContactID]
                      FROM [DBA].[Contact] AS [Extent3]  WHERE [Extent2].[ContactID] = [Extent3].[ContactID] )AS [Project1] ON cast(1 as bit) = cast(1 as bit)
                LEFT OUTER JOIN  (SELECT  [Extent4].[ContactID] AS [ContactID]
                      FROM [DBA].[Contact] AS [Extent4]  WHERE [Extent2].[ContactID] = [Extent4].[ContactID] ) AS [Project2] ON cast(1 as bit) = cast(1 as bit)
                WHERE ([Extent1].[ContactID] = [Project1].[ContactID]) OR (([Extent1].[ContactID] IS NULL) AND ([Project2].[ContactID] IS NULL))  )
) AS [Distinct1]
 LEFT OUTER JOIN
 (SELECT  [Extent5].[ContactID] AS [ContactID],  [Extent6].[SalesOrderID] AS [SalesOrderID],  [Extent6].[TotalDue] AS [TotalDue],   1 AS [C1]
           FROM  [DBA].[Contact] AS [Extent5]
            INNER JOIN [DBA].[SalesOrderHeader] AS [Extent6]
  ON  EXISTS (SELECT    cast(1 as bit) AS [C1]
                FROM    ( SELECT cast(1 as bit) AS X ) AS [SingleRowTable2]
                LEFT OUTER JOIN  (SELECT  [Extent7].[ContactID] AS [ContactID]
                      FROM [DBA].[Contact] AS [Extent7]  WHERE [Extent6].[ContactID] = [Extent7].[ContactID] )AS [Project5] ON cast(1 as bit) = cast(1 as bit)
                LEFT OUTER JOIN  (SELECT  [Extent8].[ContactID] AS [ContactID]
                      FROM [DBA].[Contact] AS [Extent8]  WHERE [Extent6].[ContactID] = [Extent8].[ContactID] )AS [Project6] ON cast(1 as bit) = cast(1 as bit)
               WHERE ([Extent5].[ContactID] = [Project5].[ContactID]) OR (([Extent5].[ContactID] IS NULL) AND ([Project6].[ContactID] IS NULL))
```

Object Relational Mapping – Persistence Framework … Svantaggi

La paginata di SQL generato della slide precedente corrisponde alla seguente interrogazione SQL

```
select  Extent6.ContactID,  1 as C1,  1 as C2,
Extent6.ContactID as  ContactID1,
Extent6.SalesOrderID  as SalesOrderID,
Extent6.TotalDue  as TotalDue
from  DBA.SalesOrderHeader as Extent6
order by Extent6.ContactID  asc
```

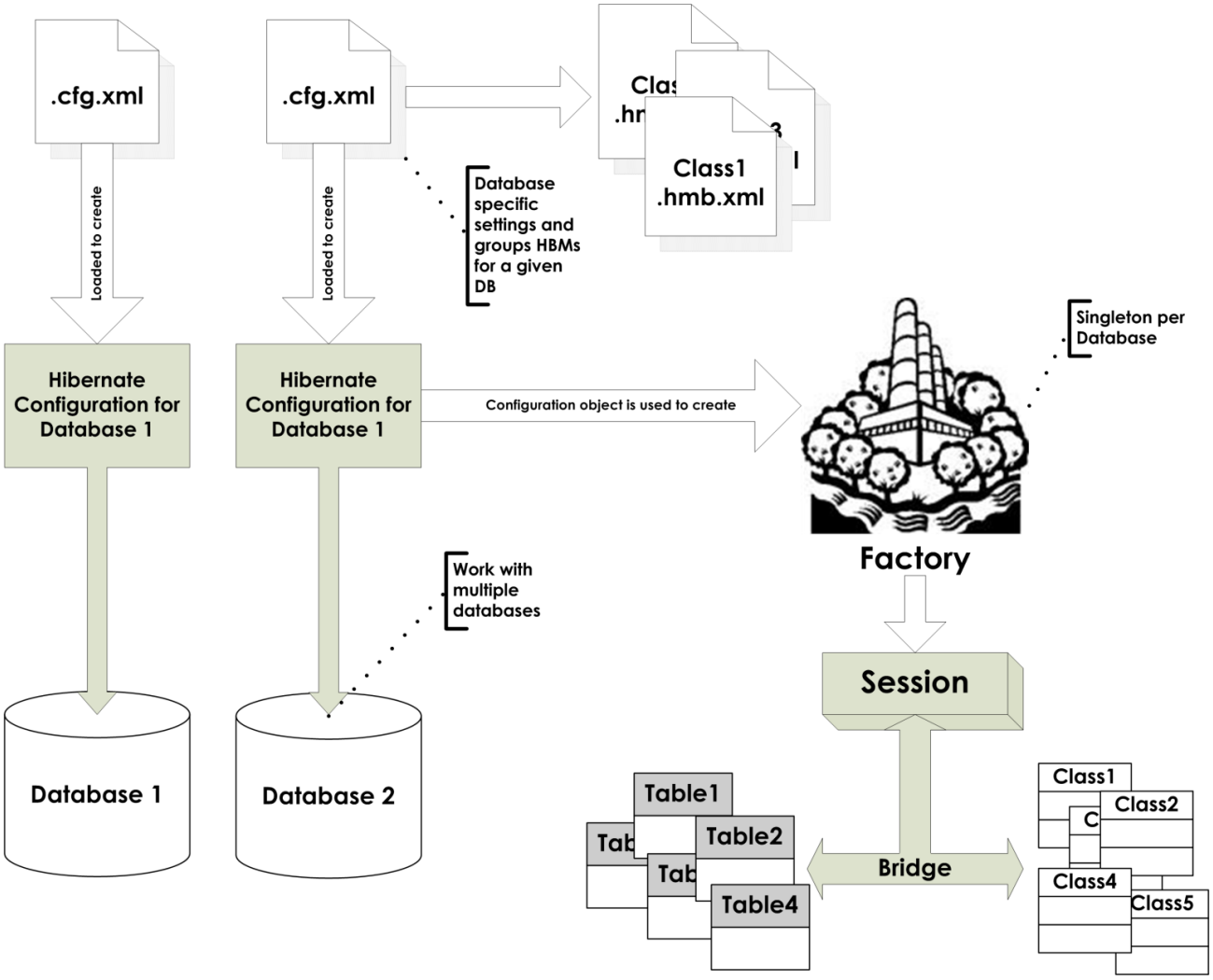
Per l'ottimizzatore non è proprio lo stesso…

Un Persistence Framework - Hibernate

5 principali ingredienti di un'applicazione che fa uso di Hibernate per la gestione della persistenza

1. le **classi di dominio** realizzate in Java
2. una **base di dati**, e.g. realizzata in H2-PostgreSQL-MySQL-...
3. un file che definisce il **mapping** di ogni classe persistente
4. uno o più file di **configurazione** di Hibernate
5. le **interfacce Hibernate** per l'accesso alla base di dati: Session, Transaction e Query – package org.hibernate

Hibernate – Quadro generale



www.hibernate.org

"Java Persistence with Hibernate"
Christian Bauer and Gavin King – Manning

Integration vs Application Database

- integration databases
  - SQL fornisce un meccanismo di integrazione tra applicazioni
  - Tante applicazioni una base di dati
  - Necessità di un livello esterno (viste)

- application databases
  - In contesto Web le strutture che vengono scambiate tra applicazioni sono più ricche
  - Per ridurre il numero di comunicazioni
    - nested record, liste, etc.
  - Rappresentate in XML or JSON
  - Disaccoppiamento tra il DB interno di un'applicazione e il modo con cui comunica con il mondo esterno

Web (anni 2000)

Polyglot Persistence

- DBMS diversi rispondono bene a diverse esigenze

- Ad es., nel contesto di un sito e-commerce
  - Memorizzare dati transazionali
  - Memorizzare dati di sessione
  - Navigare il grafo dei clienti
  - Effettuare analisi (OLAP)
  - …

Polyglot Persistence