Object-Relational Mapping

- Consider a typical business scenario
  - Business data resides on relational DBMS's
  - Business applications are coded using OOPL's
  - These applications are data driven (i.e. data intensive)
  - A typical interaction between application and DBMS
    - Data is requested from the DBMS
    - The DBMS retrieves the data and sends it to the application
    - The application processes the data
    - Then, either an update or another data retrieval is requested, or a view is rendered using the retrieved data

Object-Relational Mapping

- Details of the interaction
  - Data request (application)
    - Uses an API to send SQL strings to the DBMS
    - API: DBMS specific (e.g., libpq.so) or agnostic (e.g., JDBC)
  - Data retrieval (DBMS and application)
    - The DBMS sends data back to the application
    - The application uses the API to process the returned data
  - Updates (DBMS and application)
    - The application uses the API to send SQL strings to the DBMS
    - The DBMS executes the updates and returns status and/or data
    - The application uses the API to check the result of the update

Object-Relational Mapping

- Object-Relational Impedance Mismatch
  - Design goals (data vs behavior)
  - Building blocks (tables/rows/fields vs classes/instances)
  - Type systems (e.g. BLOB vs PDFDocument)
  - Data retrieval (query based vs navigational access)
  - Data modification (DML vs setters)
  - Error handling (no recovery vs structured error handling)
  - Other
    - DBMS: referential integrity, transactions, concurrency control, etc
    - OOPL: inheritance, interfaces, relationships, reflection, etc

Object-Relational Model

- What is the optimal solution?
  - A single data model across PL and DBMS
- What is does a sub-optimal solution look like?
  - Bring the PL and DBMS data models as close as possible
  - Make this procedure as automatic as possible
  - Effectively isolate all this plumbing from the business layer
  - Allow freedom for choice (PL and DBMS)

Object-Relational Mapping

- ORM as one solution (not “the” solution)
  - Natural programming model
    - You program OOP, the mapping layer does the data plumbing
  - Classes can be used and tested independently of application
  - Minimize DBMS trips with optimized fetching strategies
    - A good tool is expected to do better than average programmers
  - Coding
    - Reduced coding time and total code size
    - Code is easier to read and maintain
  - Error frequency is significantly decreased

Object-Relational Mapping

- ORM Desirable Features (not exhaustive)
  - Transparency (POJOs/Beans)
  - Transitivity (relationships)
  - Persistent/transient instances (attached/detached)
  - Automatic dirty instance detection
  - Inheritance strategies (single table, class per table, etc)
  - Fetching strategies (lazy/eager)
  - Transaction control
  - Flexible, “sensible defaults” based configuration
  - Availability of development tools and learning resources

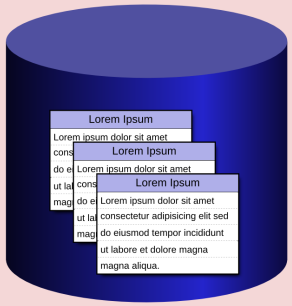
Object-Relational Mapping

- Other solutions
  - Native OODBMS's
    - db4o is a Java/.NET open source OODBMS (go check it out!)
    - Ozone is a Java open source OODBMS (older but advanced)
  - MS LINQ
    - LINQ stands for Language Integrated Query
  - Persistent Programming Language
    - No discrete boundary between program and database objects
  - Others...

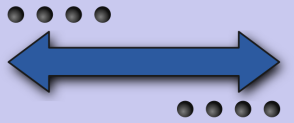
Object-Relational Mapping

- Further reading
  - There are good books on the subject and a number of (very) decent resources online
  - Ireland, C. et al. A Classification of Object-Relational Impedance Mismatch. DBKDA'09. (download if from IEEE Xplore, accessible via the UB Libraries subscription)
  - Minnaar, Douglas. Object-Relational Mapping as a Persistence Strategy. http://tinyurl.com/yeao2fq.

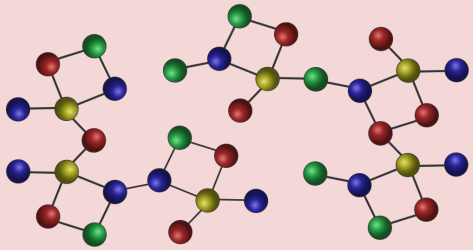
- application sends unchecked SQL strings over the Data API
- DBMS retrieves SQL results, marshalls, and sends it back
- application unmarshalls data



Database Back-End

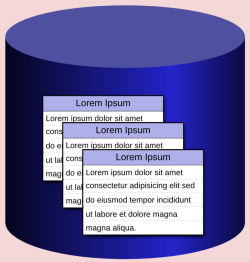


Data API



Application Objects

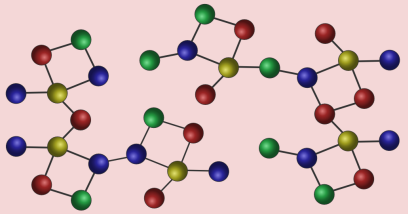
- application requests objects
- the ORM layer communicates with the DBMS, retrieves data and sends it back as objects
- plumbing is transparent



Database Back-End

Data API

ORM Layer



Application Objects