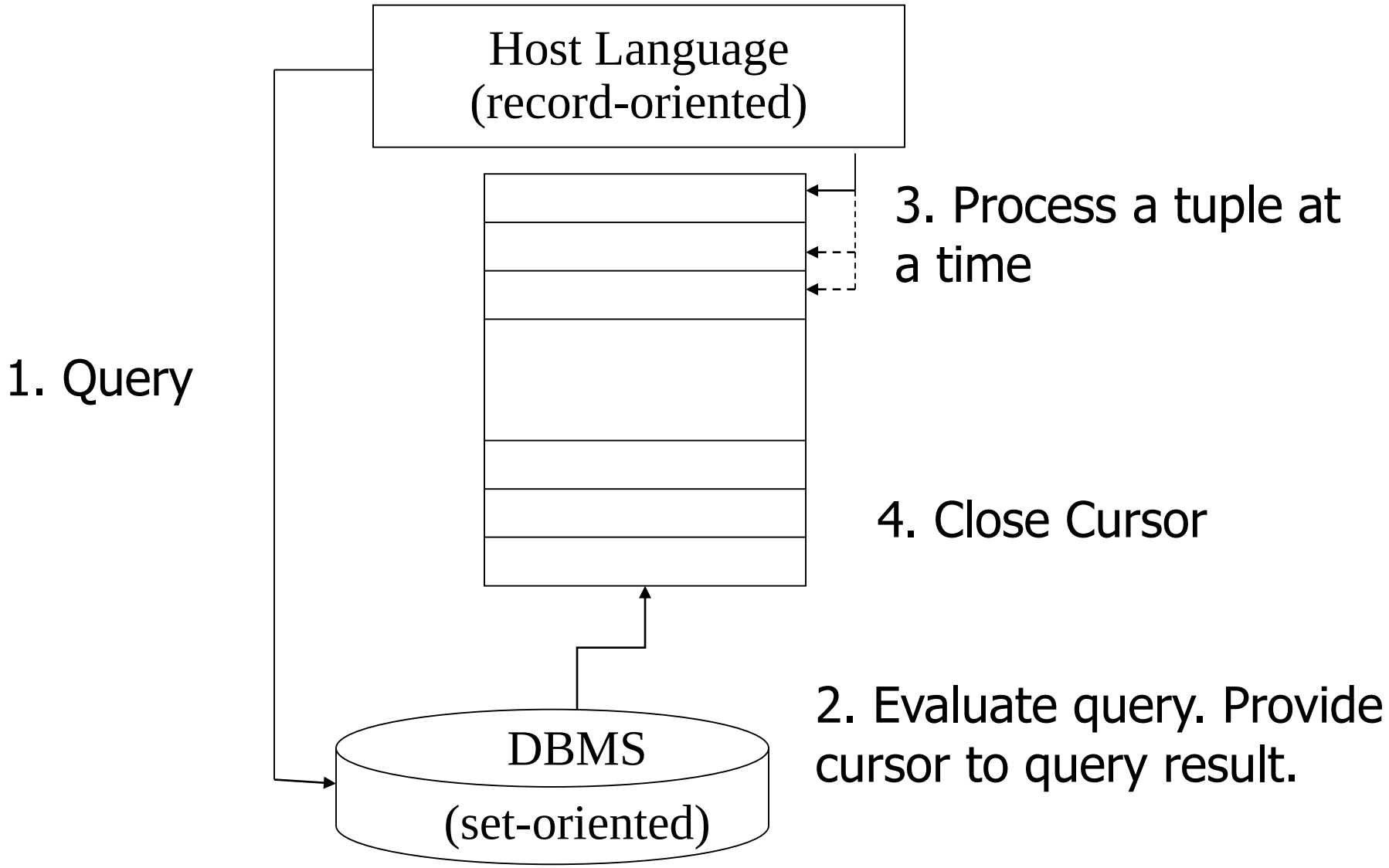
Embedded SQL



JDBC

- Java DataBase Connectivity
- Standard to access databases using Java
- Approach:
  - Create a *connection* to the database
  - Create a *statement* to be executed by the database
  - Set parameters of the *statement* (optional)
  - Execute the *statement;* return *ResultSet* (aka cursor)
  - Read tuples from *ResultSet*
- JDBC is not restricted to SQL!

Create Connection to the DB

Connection conn = DriverManager.getConnection(
urlDB, username, password)

- Need JDBC driver (provided by DBMS)
- urlDB: Identifies the database uniquely
  - N.B. one server could provide multiple DBs.
- Username, Password: as usual
- Other settings provided by configuration
  - e.g., buffer pool, app heap, TA level, ...

Output the names of all Profs

```
Statement s = conn.createStatement();
ResultSet r;

s.execute(„SELECT name FROM professor“);
r  = s.getResultSet();
while (r.next()) {
   output(r.getString(1));
}
r.close();
```

Parameterized Queries

```
PreparedStatement s = conn.prepareStatement(
   „SELECT name FROM prof WHERE level = ?“);
ResultSet r;
...

s.setString(1, „AP“);
r = s.executeQuery();
while (r.next()) ...
```

Tipps and Tricks
Connection Pooling

- Create several connections to the database
- Grab an unused connection before accessing DB
- Execute statement using that connection

- Why? Do not block the database with heavy queries

- Rule of thumb: 5 – 10 connections
  - (too many connections will hurt performance and avail.)

Tipps and Tricks
Cursor Caching

- Use PreparedStatements!
- Example:
  insert into professor(name, level) values(?,?)
- Why? Avoid overhead (optimizer) for every call
- Disadvantage? Optimizer has no statistics

JDBC Summary

- Simple protocol to send messages to the database
  - Database is typically deployed as a server!
- SQL Syntax not checked at compile time!!!
  - For Java, those are just strings
- (Type) Safety of parameters checked at running time
- All JDBC Statements raise SQLExceptions
  - Should be caught!
- New Standard: SQLJ

SQL J

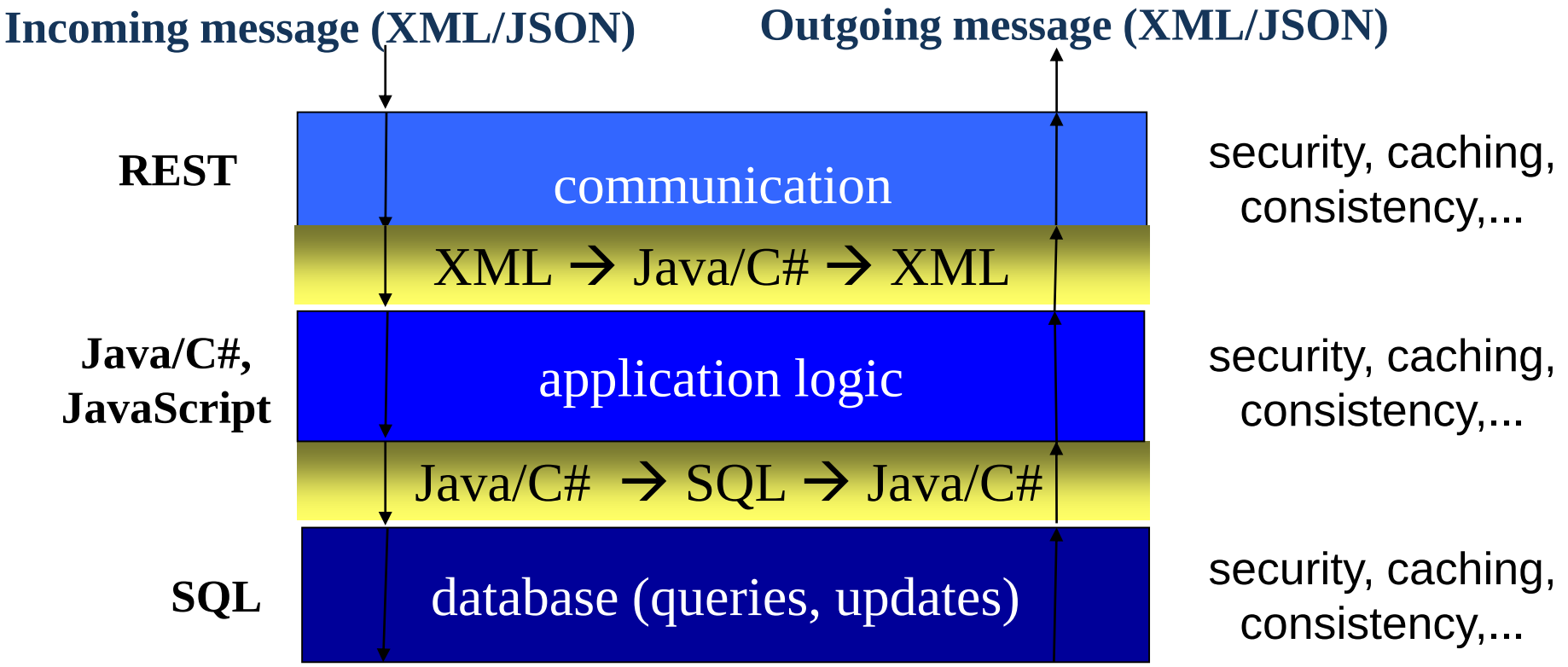
- SQL embedded in Java
- Use preprocessor at compile time for type safety, SQL syntax
- Annotate SQL statements with #sql
- Iterator (Cursor) similar to JDBC

```
#sql iterator ProfIterator(String name, String level);
ProfIterator myProfs;
#sql myProfs = { SELECT name, level FROM Professor };
while (myProfs.next()) {
   System.out.println(myProfs.name() + myProfs.level());
}
```

Object-Relational Mapping (e.g., Hibernate)

- With JDBC and SQL-J, programmers wear two hats
  - Object-oriented programming with Java
  - Database programming with SQL
  - Two languages, two data models, two type systems, ...
  - Duplicate work for logging, caching, error handling, security

Traditional Multi-tier Architecture



Problem: Every layer reinvents the wheel!!!

- security, caching, consistency, error handling, data model, ...
- huge overheads during development (technology jungle)
- huge overheads during deployment (configuration)
- huge overheads during operation (RPCs, duplicate work)

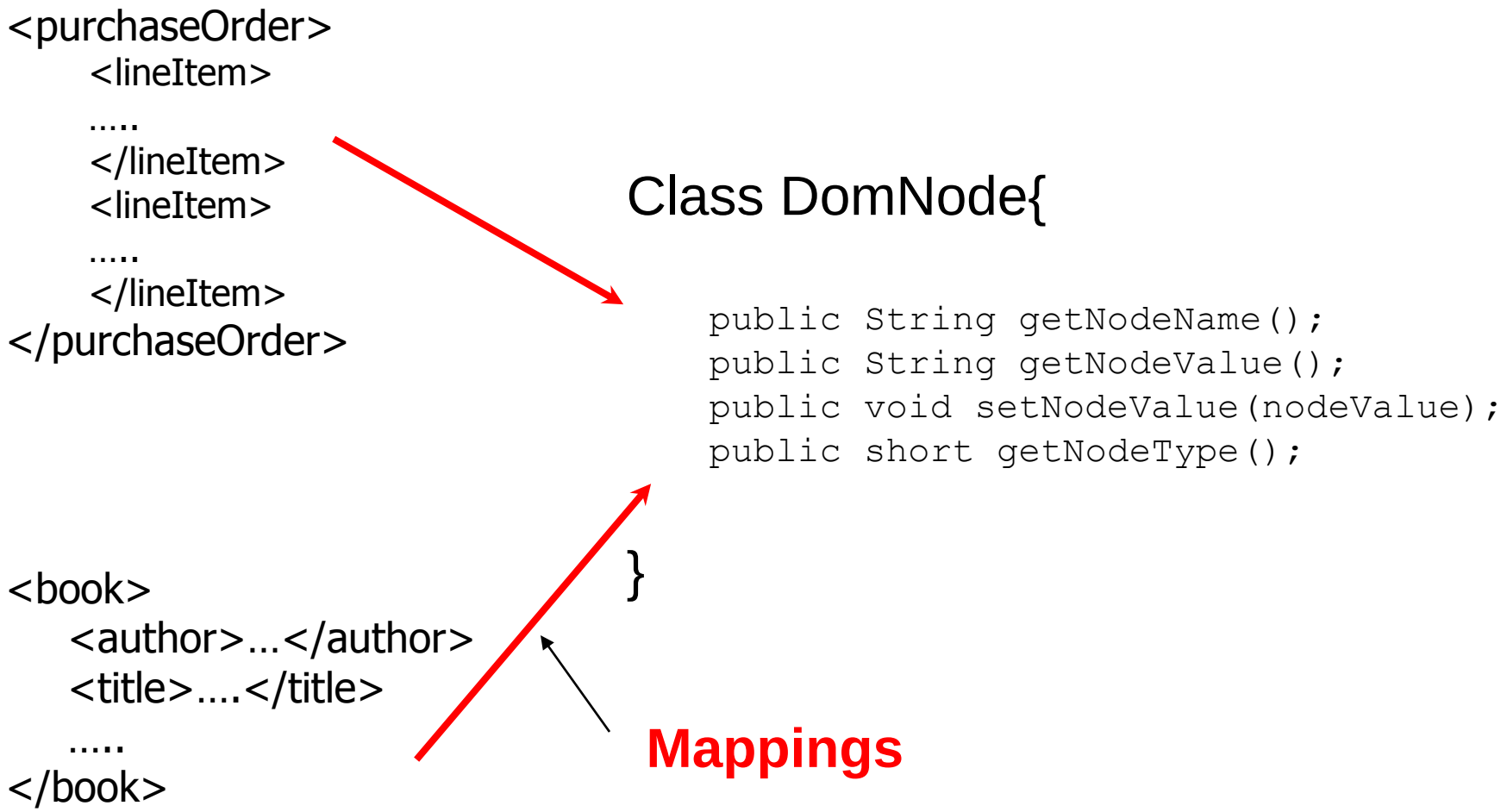
Object-Relational Mapping (e.g., Hibernate)

- With JDBC and SQL-J, programmers wear two hats
  - Object-oriented programming with Java
  - Database programming with SQL
  - Two languages, two data models, two type systems, ...
  - Duplicate work for logging, caching, error handling, security

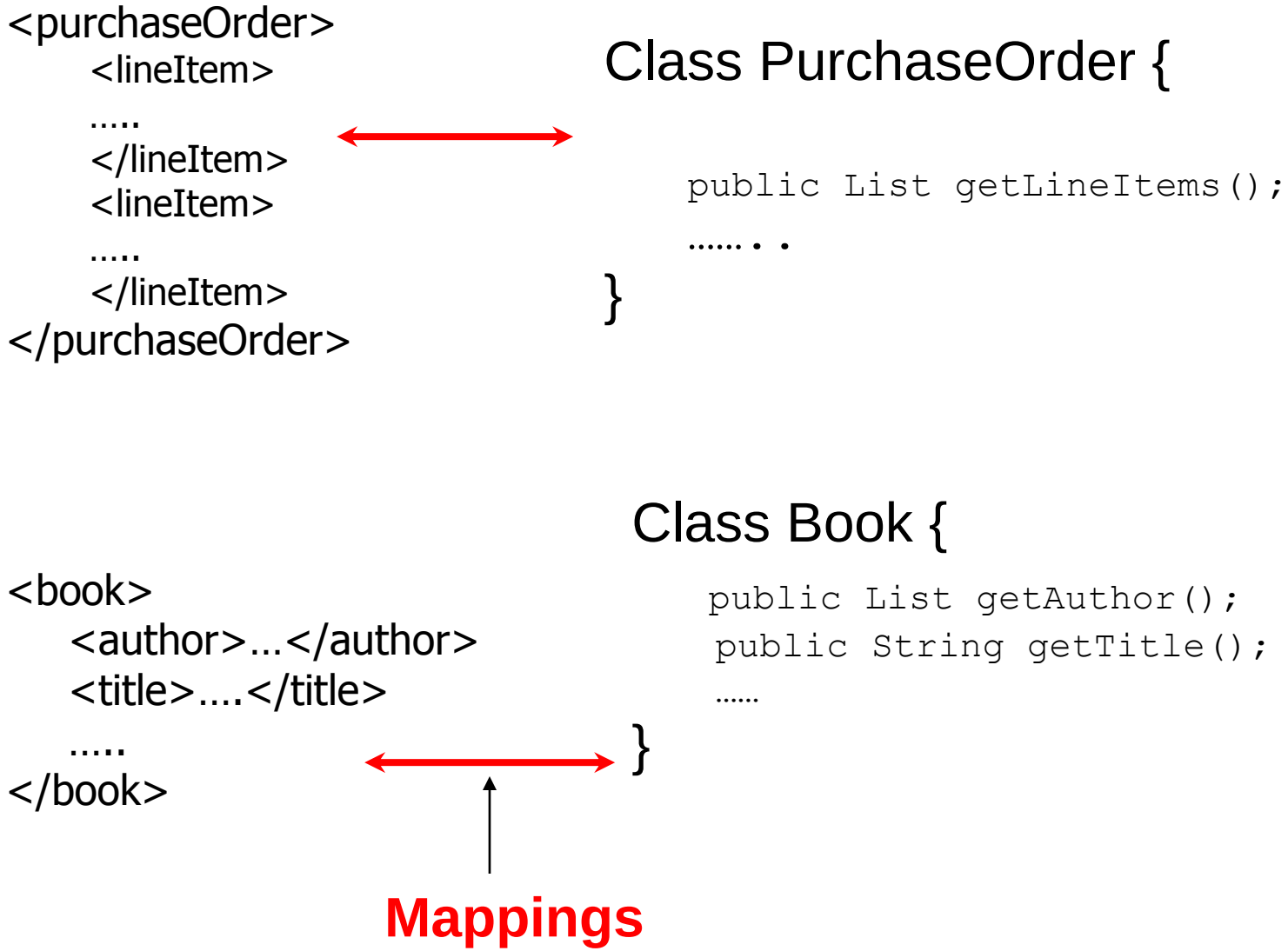
- Idea: Automate the database programming
  - DDL: generate „create table" from XML, annotations
  - Queries: generate „select" from getters and setters
  - **Make everything look like Java**

- Idea applicable to relational and XML!
- *Please, do not use in project! We learn the bare bones here!!!*

XML Mapping to generic structures



Mapping to non-generic structures



Other Approaches

- New programming languages
  - e.g., Ruby, XQuery, etc.
  - integrate app scripting and database programming
  - address additional impedance mismatch with Web

- PL/SQL (stored procedures)
  - bring application logic to database: „;“, „while“, blocks, ...
  - rather than database logic to application
  - huge performance advantages

- LINQ (language integrated queries)
  - provide a super-data model at application layer
  - (mother of all Hibernates)