THERE'S a DATABASE in YOUR FUTURE

This section is a bit of a sales pitch for the course

Database programmers in high demand

- Oracle
- IBM
- MS SQL
- MySQL
- Postgres
- Visual Basic
THERE'S a DATABASE in YOUR FUTURE

- Programmers use database to support programs
- Database Administrators
- Database Designers
- Casual User
- Data Entry Clerk
  - if the economy takes a downturn ....
Business Tools use Database

Enterprise Resource Planning

<table>
<thead>
<tr>
<th>Commercial</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Oracle Financials</td>
<td>• OpenPro</td>
</tr>
<tr>
<td>• SAP Business/ERP</td>
<td>• Erp5</td>
</tr>
<tr>
<td>• Infor products</td>
<td>• Compiere</td>
</tr>
<tr>
<td>• Epicor Distribution</td>
<td>• OpenMFG</td>
</tr>
<tr>
<td>• Microsoft Dynamics</td>
<td>• Opentaps (Sequoia)</td>
</tr>
</tbody>
</table>
DB DATA is STRUCTURED

• Data packaged in Records
  – Like C Structs or C++ Classes

• Records have types
  – Employee info record

• Records have fields
  – Name, ID, Salary ....

• Fields have types
  – String, Integer ....

• DBs we do not study can handle semi-structured or unstructured data.
Example: Type vs Instance

• `EmpRecordType = (  
  name char(30),  
  birthdate datetype)`

• An employee RECORD might be:  
  ("Walter Della Mar",  
  "30-JAN-1971")
STRUCTURE VS CONTENT

Record Schema vs Instance
- Layout of employee fields
- vs Values for one employee

DB Schema vs Instance
- Layout of whole DB
- vs Contents at some instant
STRUCTURE in RELATIONAL DB

• Each table in the database
  – just ONE record TYPE.
  – plus some rules
  – That is the schema for the table.

• Schema for DB
  – collection of table schemas.
  – plus a set of rules

• Schema is relatively fixed over time

• Data may change rapidly.
  – At any time the whole set of data is called the database instance.
Some DATABASE GOALS

• Maintain Integrity
  – Avoid Inconsistency
  – Avoid Redundancy
• Security (we do not cover this)
• Ensure Recovery
• Concurrent Use
REDUNDANCY, CONSISTENCY, INTEGRITY

- Avoid Redundancy by not storing the same information in two places.
- Helps maintain Consistency
  - Takes two items to be inconsistent
- Integrity problem: employee record says she works for dept 11 but there is no dept 11
Representational Independence

- Use DB according to its Logical Definition
- Independent of Machine
- Independent of Stored Database
- Independent of Conceptual Database.
- Great Programming Interface!
Source of Independence

- User operates at logical level.
- Database translates Logical to implementation.
- Implementation is hidden and can be changed.
- Even moved to another vendor's DB
  - DB language (SQL) must have same syntax and meaning on both platforms