# Admin matters

## 1. Selection of class reps

## 2. Activities for next 3 weeks

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Day</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21 Jul</td>
<td>Tues</td>
<td>L1: Course overview</td>
</tr>
<tr>
<td></td>
<td>22 Jul</td>
<td>Wed</td>
<td>L2: Object DBMS</td>
</tr>
<tr>
<td></td>
<td>24 Jul</td>
<td>Fri</td>
<td>No practical (prac 1 spec. available on website)</td>
</tr>
<tr>
<td>2</td>
<td>28 Jul</td>
<td>Tues</td>
<td>L3: Object DBMS</td>
</tr>
<tr>
<td></td>
<td>29 Jul</td>
<td>Wed</td>
<td>L4: Object DBMS</td>
</tr>
<tr>
<td></td>
<td>31 Jul</td>
<td>Fri</td>
<td>Practical 1: Object DBMS (db4objects)</td>
</tr>
<tr>
<td>3</td>
<td>4 Aug</td>
<td>Tues</td>
<td>L5: Object-Relational DBMS</td>
</tr>
<tr>
<td></td>
<td>5 Aug</td>
<td>Wed</td>
<td>L6: Object-Relational DBMS</td>
</tr>
<tr>
<td></td>
<td>7 Aug</td>
<td>Fri</td>
<td>Practical 2: Object-Relational DBMS (PostreSQL)</td>
</tr>
</tbody>
</table>
## Admin matters: Textbook chapters

<table>
<thead>
<tr>
<th>Part</th>
<th>Chapters</th>
<th>Topic</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>ch 1 to 3</td>
<td>Background</td>
<td>INF214 and COS326</td>
</tr>
<tr>
<td>Part 2</td>
<td>ch 4 to 9</td>
<td>The relational model and languages</td>
<td>INF214</td>
</tr>
<tr>
<td>Part 3</td>
<td>ch 10 to 15</td>
<td>Database analysis and design</td>
<td>INF214</td>
</tr>
<tr>
<td>Part 4</td>
<td>ch 16 to 19</td>
<td>Methodology</td>
<td>INF214</td>
</tr>
<tr>
<td>Part 5</td>
<td>ch 20 to 23</td>
<td>Selected database issues</td>
<td>INF261</td>
</tr>
<tr>
<td>Part 6</td>
<td>ch 24 to 26</td>
<td>Distributed DBMSs</td>
<td>INF261</td>
</tr>
<tr>
<td>Part 7</td>
<td>ch 27 to 29</td>
<td>Object DBMSs</td>
<td>COS326</td>
</tr>
<tr>
<td>Part 8</td>
<td>ch 30 to 31</td>
<td>The Web and DBMSs</td>
<td>COS326</td>
</tr>
<tr>
<td>Part 9</td>
<td>ch 32 to 35</td>
<td>Business intelligence</td>
<td>COS326</td>
</tr>
</tbody>
</table>
In this lecture

1. Examples of advanced DB applications
   i. CAD
   ii. GIS
   iii. Office Information Systems and Multimedia
   iv. Publishing
   (other examples: CASE, Telecoms networks)

2. Weaknesses of Relational DB systems

3. Storing Objects in Relational Databases

4. Origins of the object-oriented data model (reading for student)

5. OODB and OODBMS

6. Commercial OODBMSs

7. The OODBMS Manifesto (1989) (reading for student)
Example 1: Computer-aided design (CAD)

CAD database stores data relating to engineering designs e.g.
(1) Cars    (2) Aircraft
(3) Buildings (AutoCAD, Revit)
(4) Integrated Circuit chips

Some common characteristics for design data:

(1) design has many parts & interdependent subsystems
(2) large number of types & small number of instances
   cf. relational: few relations, many tuples
(3) design is not static, it evolves over time
(4) Change to one part may affect many other parts
(5) Often many design alternatives are considered at the same time
   => difficult to design a schema for a relational database

Refer to Section 27.1 (pg. 803) for full discussion
GIS database stores various types of spatial & temporal data used for e.g.

1. land management (urban & regional planning)
   - maps, spatial data for maps, etc.
2. underwater exploration
   - photographs, spatial data, etc.
Example 3: Office Information Systems (OIS)

OIS database stores data relating to computer control of information in a business

e.g.
- e-mail, documents, invoices etc.
- free-form text, photos, diagrams
- audio & video sequences

=> Data has richer structure than relational DB tuples
Example 4: Digital publishing

Database can store multimedia documents consisting of:

(1) text
(2) audio
(3) video
(4) image
(5) animation data

Relational structure not suitable for this data
Weaknesses of RDBMSs (1)

- **Poor representation of “real world” entities**
  - **Normalization** leads to relations that do not correspond to entities in the “real world”
  - Fragmentation of the “real world”

- **Information** is most commonly obtained from the DB through performing a join (intensive computation) of 2 or more tables

<table>
<thead>
<tr>
<th>Staff</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>staffNo</td>
<td>sName</td>
</tr>
<tr>
<td>SL21</td>
<td>John White</td>
</tr>
<tr>
<td>SG37</td>
<td>Ann Beech</td>
</tr>
<tr>
<td>SG14</td>
<td>David Ford</td>
</tr>
<tr>
<td>SA9</td>
<td>Mary Howe</td>
</tr>
<tr>
<td>SG5</td>
<td>Susan Brand</td>
</tr>
<tr>
<td>SL41</td>
<td>Julie Lee</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>branchNo</th>
<th>bAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>B005</td>
<td>22 Deer Rd, London</td>
</tr>
<tr>
<td>B007</td>
<td>16 Argyll St, Aberdeen</td>
</tr>
<tr>
<td>B003</td>
<td>163 Main St, Glasgow</td>
</tr>
</tbody>
</table>

*E.g. JOIN required to see branch address together with staff member’s name*
Weaknesses of RDBMSs (2)

Semantic Overloading

- Single construct for representing data and data relationships: *the relation*

- No mechanism to distinguish between relations & relationships or between different types of relationships

- Relational model is *semantically overloaded* e.g. ERD

```
Student

Registration

Course

relationship 'is written in' is not recorded in Relational DB

relationship 'is found in' is not recorded in Relational DB

This is a relation and a relationship
```
1. Write down a list of the relationships that are used in object-oriented programming

2. Explain why each of these relationships would be difficult to represent in a relational database.
Weaknesses of RDBMSs (3)

- **Limited Operations**
  - RDBMSs only have a fixed set of operations which cannot be extended i.e. **operations on sets of tuples (cf. Relational calculus & algebra)**
  - Not possible to specify new operations (cf. OOP methods)

- **e.g. GIS apps** use **data** (points, lines, etc) & **operations** (distance, **intersection**, **containment**, etc.)

Find all streets **intersecting** with Spuy Street?

Find all addresses **in** Minni Street?
Weaknesses of RDBMSs (4)

- **Homogeneous data structure**
  - Horizontal and vertical homogeneity is restrictive for ‘real world’ objects

- **Horizontal homogeneity**
  - each tuple composed of the same attributes

- **Vertical homogeneity**
  - values of an attribute come from same domain

- **Intersection of row and column must be atomic**
  - cannot store an object that can be decomposed into its components

  **Note:** can store a BLOB or CLOB (pointer to a file containing object) for viewing purposes
Weaknesses of RDBMSs (5)

◆ impedance
  – *Physics*
    » a measure of the opposition (resistance) to the flow of a current in an electrical circuit
  – *Software engineering*
    » a measure of the opposition *(difficulty)* caused by differences between two paradigms

◆ Embedded SQL
  – *for applications that access database data*
  – *SQL standard provides embedded SQL*
  – *creates problem of impedance mismatch*
Weaknesses of RDBMSs (6)

- Impedance mismatch
  - Mixing different programming paradigms
    - SQL: declarative
      (specify processing for multiple rows at a time)
    - C, C++, C#, VB, Java: procedural
      (specify processing of one row at a time)
  - Different formats to represent data
    - e.g SQL: Date, Time formats may be different from programming language formats
    - 30% of programming effort spent on conversions (1983)
  - Programming effort and runtime resources required for conversions
Weaknesses of RDBMSs (7)

- **RDBMS supports short-lived transactions**
  - Short-lived (business) vs long-duration (design)
  - e.g. `BEGIN TRANSACTION ..... END TRANSACTION`

- **Schema changes are difficult**
  - Database change requires program changes

- **Designed for content-based associative access**
  - Poor for navigational access
    - Movement between records,
    - **But:** can use stored procedures
Storing Objects in Relational Databases

◆ **Data persistence**
  – data outlives the execution of the program that created it e.g. data in a table in a DBMS

◆ **Storing objects in relational DB:**
  – An approach to achieving persistence with an OOP language:
    – e.g. use RDBMS as underlying storage engine

◆ **Mapping class instances** (i.e. objects) to one or more rows distributed over one or more relations

*Loss of semantic information e.g. inheritance*
E.g. of semantic information: inheritance (is-a)

Two tasks required for relational DB access:

1. design relations to represent class hierarchy
2. design how objects will be accessed
   - write code to convert between objects and rows
Mapping classes to relations (problem 1)

(1) Map each class or subclass to a relation

Schema (one option):

Staff (staffNo, fName, lName, position, gender, DOB, salary)
Manager (staffNo, bonus, mgrStartDate)
SalesPersonnel (staffNo, salesArea, carAllowance)
Secretary (staffNo, typingSpeed)

e.g. of query (who are the managers?):
SELECT * FROM Staff, Manager
WHERE (Staff.staffNo = Manager.staffNo)
Mapping classes to relations (problem 2)

(2) Map the hierarchy to a single relation

Shema:

Staff ( staffNo, fName, lName, position,
gender, DOB, salary, bonus, mgrStartDate,
salesArea, carAllowance, typingSpeed,
typeFlag )

e.g. of query (who are the managers?):

SELECT staffNo, fName, lName,
position, gender, DOB,
salary, bonus, mgrStartDate
FROM Staff WHERE typeFlag = "M";

where is the hierarchy info???
(3) Map the hierarchy to a single relation
* many NULL values*

<table>
<thead>
<tr>
<th>staffNo</th>
<th>name</th>
<th>position</th>
<th>salary</th>
<th>mgrStartDate</th>
<th>bonus</th>
<th>salesArea</th>
<th>carAllowance</th>
<th>typingSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL21</td>
<td>John White</td>
<td>Manager</td>
<td>30000</td>
<td>01/02/95</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SG37</td>
<td>Ann Beech</td>
<td>Assistant</td>
<td>12000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SG66</td>
<td>Mary Martinez</td>
<td>Sales Manager</td>
<td>27000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA9</td>
<td>Mary Howe</td>
<td>Assistant</td>
<td>9000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL89</td>
<td>Stuart Stern</td>
<td>Secretary</td>
<td>8500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL31</td>
<td>Robert Chin</td>
<td>Snr Sales Asst</td>
<td>17000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SG5</td>
<td>Susan Brand</td>
<td>Manager</td>
<td>24000</td>
<td>01/06/91</td>
<td>2350</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Origins of the object-oriented DBMS

- Started in Engineering and design domains
  - CAD

- Has become the preferred DBMS for:
  - Telecommunications applications
  - Financial applications

- OODMS market is small compared to RDBMS market
Origins of the object-oriented data model

- **Traditional database systems**
  - Persistence
  - Sharing
  - Transactions
  - Concurrency control
  - Recovery control
  - Security
  - Integrity
  - Querying

- **Semantic data models**
  - Generalization
  - Aggregation

- **Object-oriented programming**
  - Object identity
  - Encapsulation
  - Inheritance
  - Types and classes
  - Methods
  - Complex objects
  - Polymorphism
  - Extensibility

- **Special requirements**
  - Versioning
  - Schema evolution

**Reading for the student**
Terminology

- **OO data model**
  - (logical) data model that captures the semantics of objects supported in object-oriented programming

- **OO database**
  - persistent sharable collection of objects defined by an OO data model

- **OODBMS**
  - manager (software) of an OODB
**OODB and OODBMS**

- **Object-oriented database (OODB)**
- **Object-oriented DBMS (OODBMS)**

**PropertyForRent** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)

**PrivateOwner** (ownerNo, fName, mName, address, telNo)

**Client** (clientNo, fName, mName, address, telNo, prefType, maxRent)

**Lease** (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentFinish)
Commercial OODBMSs

Reading for student:

- GemStone from Gemstone Systems Inc.
- Objectivity/DB from Objectivity Inc.
- ObjectStore from Progress Software Corp.
- db4objects from Versant Corp
<table>
<thead>
<tr>
<th>Object-oriented characteristics</th>
<th>DBMS Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Complex objects</td>
<td>9. Data persistence</td>
</tr>
<tr>
<td>2. Object identity</td>
<td>10. Handle very large databases</td>
</tr>
<tr>
<td>3. Encapsulation</td>
<td>11. Concurrent users</td>
</tr>
<tr>
<td>4. Types or classes</td>
<td>12. Recovery from failure</td>
</tr>
<tr>
<td>5. Inheritance for types or classes</td>
<td>13. Simple way of querying data</td>
</tr>
<tr>
<td>6. Dynamic binding</td>
<td></td>
</tr>
<tr>
<td>7. DML must be computationally complete</td>
<td></td>
</tr>
<tr>
<td>8. Extendable data types</td>
<td></td>
</tr>
</tbody>
</table>