People & Contact Information

- **Instructor:** Prof. Betty O’Neil  
  Email: eoneil AT cs DOT umb DOT edu (preferred contact)  
  Web: http://www.cs.umb.edu/~eoneil  
  Office: Science Building, 3rd Floor, Room 169 (S-3-169)

- **TA:** Mohammad.Hadianpour  
  Email: Mohammad.Hadian001 AT umb DOT edu  
  Office: S-3-124A

- **Course-related emails (for instructor and for TA)**  
  Subject line should start with [CS430] or [CS630]

Course Info

- **Lecture Hours**  
  Mon and Wed, 7:00-8:15pm

- **Office Hours**  
  Mon and Wed 3:30-3:45pm, 5:30-6:45 in S/3/169  
  By appointment (send email)

- **Class URL**  
  http://www.cs.umb.edu/cs630/

Textbooks

- **Textbooks**  
  Database Management Systems, 3rd Edition  
  by Ramakrishnan and Gehrke

  Please purchase a print copy (under $50).  
  Note: No electronic devices are allowed in open-book exams. No book sit in front to share mine.

Prerequisites

- **Data Structures and Algorithms, Java**  
  CS310 officially, but CS210 or equivalent is acceptable  
  If an undergraduate, take cs310 before or with cs430/cs630, not after (rule against out-of-order courses)

- **Familiarity with UNIX/Linux OS**  
  Exercises will be executed on Oracle 12c server running on a Linux machine in the CS dept (DBS3 Machine)  
  Access is via sqlplus on topcat.cs.umb.edu, another Linux machine, and later; JDBC

Grading

- **Final exam (150 points) — open book**
- **Midterm (100 points) — open book**  
  Open book does NOT include electronic devices!

- **Approximately 6 homework assignments**  
  10 points each, drop lowest score.  
  Assignments for CS630 will have additional questions  
  Assignments are individual — submit your own work only!  
  No plagiarism! See student code of conduct
Course Materials

- Class URL
  http://www.cs.umb.edu/cs630/

- Make sure you obtain a cs630 Unix course accounts
  - Go to S/3/158 and get assistance
  - Or login to users.cs.umb.edu as user apply from anywhere (password given in class)
  - New apply process (alternative way): See documentation here.

University Policies

ACADEMIC CONDUCT:
It is the expressed policy of the University that every aspect of academic life—not only formal coursework situations, but all relationships and interactions connected to the educational process—shall be conducted in an absolutely and uncompromisingly honest manner. The University presupposes that any submission of work for academic credit indicates that the work is the student's own and is in compliance with University policies. In cases where academic dishonesty is discovered after completion of a course or degree program, sanctions may be imposed retroactively, up to and including revocation of the degree. Students are required to adhere to the Code of Student Conduct, including requirements for academic honesty, delineated in the University of Massachusetts Boston Bulletin, found at: http://www.umb.edu/life_on_campus/policies/community/code.

University Policies

Accommodations:
The University of Massachusetts Boston is committed to providing reasonable academic accommodations for all students with disabilities. This syllabus is available in alternate format upon request. Students with disabilities who need accommodations in this course must contact the instructor to discuss needed accommodations. Accommodations will be provided after the student has met with the instructor to request accommodations. Students must be registered with the Ross Center for Disability Services, UL 211, www.ross.center@umb.edu 617.287.7430 before requesting accommodations from the instructor.

Course Overview

- Relational Data Model
- Relational Algebra
- Structured Query Language
  - The most important part of the course
- Conceptual design — the ER model
- Database application development
  - Java/JDBC, PL/SQL
- Design Theory
- Database Security
  - If time, intro to GraphQL, a new query methodology

What is a DBMS?

- Specialized software that provides:
  - Uniform and transparent access to data
  - Application-independence
  - Application/user is oblivious to internal data organization
  - Data organization may change, but applications need not change
  - Efficient access to data
  - Fast search capabilities, indexing
  - Data consistency
    - E.g., cannot delete student record if grade records still in DBMS
  - Concurrent access to data
  - Persistent storage and recovery from failure
  - Security

Why study databases?

- Databases are ubiquitous
  - Behind all web service providers there is a DBMS
    - In some cases a very large-scale one (EBay, Facebook, etc.)
    - Amazon uses DBs for money handling
  - Corporations use DBs for business processes, HR, etc
  - Scientific computing relies on very large amounts of data
    - Human genome data
    - Biochemistry data (protein sequences)
    - Astronomy data
    - High-energy physics
- DBAs are very well-paid!
  - And even in other IT areas, DBMS skills are a must
A bit of history …

- First data stores were file systems
  - Does not conform to transparency and uniformity desiderata
  - Search (within file) most often linear
  - Not portable
  - Doesn’t handle concurrency properly
  - Sequential access only
- Early DBMS appeared in the 60s
  - Driven by banking and airline industry
  - Relatively small record size, and many concurrent accesses
  - Two prominent models: hierarchical model (tree) and network model (graph)
  - Lack of support for high-level query languages

A bit of history (contd.)

- Relational Databases
  - Major breakthrough, paper written by Codd (1970)
  - Relations (tables) with rows (records) and columns (fields)
  - Relationships and constraints among tables
  - Structured Query Language (SQL): high-level, declarative
  - Data definition/manipulation language
  - Fast search – use of index structures
  - Data access language independent from internal organization
- Newer paradigms
  - Object-oriented and multimedia DB
  - Data Stream Management Systems (DSMS)
  - Big Data, MapReduce

Levels of Abstraction

Data

Physical Schema

Conceptual Schema

Views define how users see data

Describes files and indexes used

Defines logical data structure

Views 1 2 3

Database Management Systems 3rd ed, Ramesh R. and Gehrke

System architecture

Clients access the database server over the Internet

Oracle 12c on dbs3.cs.umb.edu

Your login on topcat.cs.umb.edu or pe07.cs.umb.edu

Your development system at home

Client software, server software, and the SQL interface

Client

Database server

Database management system

Application software

Visual Basic

Java driver

Oracle

ADO

JDBC

.NET driver

Obsolete!

Common options for accessing Oracle data

Java application

.NET application

Visual Basic 6 application

JDBC

ADD/.NET

ADO

Java driver

.NET driver

OLE DB

Oracle
An application that uses an application server

We won’t be using an application server in this class, but we do in CS636, a follow-on course, with Java running in the application server (tomcat).

The Relational Model

Data Model

- **Structure of data**
  - Relational model uses tables: just rows, columns, no ‘pointers’
  - Vs. Programming languages deal with arrays, collections, etc. with explicit references from one item to another
- **Operations on the data**
  - Queries: operations that retrieve information
  - Modifications: operations that change data
- **Constraints**
  - Domain constraints (the simplest): e.g., age must be numeric
  - Other constraints: each student has unique matriculation #
- **Prominent Data Models**
  - Relational model
  - Object-relational model, semi-structured model (XML), E-R

Relational Model

- **Relational database**: a set of relations
- **Relation**:
  - two-dimensional table, with rows and columns
  - #Rows = cardinality
  - #Columns = degree (or arity)
  - Each row represents an entity
    - A student, a course, a movie
  - Each column represents a property of the entity
    - Student age, student matriculation #, student gpa
  - Column values are atomic (e.g., integer or string) within given domain
  - Rows are also called tuples or records; columns are also called fields or attributes

“Students” Relation or Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Cardinality = 3, Degree = 5
Use of “string” as a domain in R&G

- If you already know some SQL, you know there is no data type “string” in SQL, but we see “string” in many schemas in the R&G book
- The first use is on pg. 59, for students, as on the last slide.
- In SQL, we have char(20) for example, for 20-character strings.
- So think of “string” as standing for strings with some length limit.

More about Relations

- Relations are sets of tuples
  - Sets are NOT ordered
  - Do NOT retrieve by row number, but by content!
- Relation Instance
  - Contents of a relation may change over time
  - E.g., Students join or leave the university
  - Instance represents set of tuples at a certain point in time
- Schemas may change too
  - Although this is not frequent in practice
  - Changing schema can be very expensive

Instance of “Students” Relation

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Cardinality = 3, Degree = 5

Another Instance of “Students”

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
<tr>
<td>53660</td>
<td>Korth</td>
<td>korth@math</td>
<td>22</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Cardinality = 4, Degree = 5

Typical Application life cycle

- Install application: create tables for application, once and for all
- Use application: add/delete rows, update rows, query rows
Of course, some applications create tables dynamically.

Keys

- A key of a relation is a set of fields K such that:
  1. No two distinct tuples in ANY relation instance have same values in all key fields, and
  2. No subset of K is a key (otherwise K is a superkey)
- Key may not be unique
  - Multiple candidate keys may exist
  - One of the keys is chosen (by DBA) to be the primary key
  - Keys are shown underlined in schema
- In the relational model, duplicate tuples do not exist!
  - But most DBMS implementations do allow duplicates
  - Key constraints must be set by DBA to avoid duplicates
  - Rule of thumb: every table should have a primary key
Example of Keys

Students(sid: string, name: string, login: string, age: integer, gpa: real)

- sid is a key; (sid, name) is a superkey

```
<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>
```

- In practice, it is not easy to know when there exists a unique attribute combination in the data (e.g., names)
- artificial keys are created: student ID, customer ID, etc.
- SSN is also often used for keys (but what if multinational...)

```
Murach's Oracle SQL and PL/SQL, C1
© 2014, Mike Murach & Associates, Inc.
Slide 32
Murach Example: The Vendors table viewed here in Oracle SQL Developer
```

```
The Murach Invoices table

<table>
<thead>
<tr>
<th>Row</th>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>2</td>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>
```

```
The relationship between two tables
```

```
The columns of the Murach Invoices table
```

```
The columns of the Vendors table
```

```
The primary key
```

```
The foreign key
```

```
Murach's Oracle SQL and PL/SQL, C1
© 2014, Mike Murach & Associates, Inc.
Slide 34
The columns of the Murach Invoices table
```

```
The relationship between two tables
```

```
The columns of the Vendors table
```

```
The primary key
```

```
The foreign key
```

```
Murach's Oracle SQL and PL/SQL, C1
© 2014, Mike Murach & Associates, Inc.
Slide 35
The relationship between two tables
```

```
The columns of the Vendors table
```

```
The primary key
```

```
The foreign key
```

```
Murach's Oracle SQL and PL/SQL, C1
© 2014, Mike Murach & Associates, Inc.
Slide 36
The relationship between two tables
```

```
The columns of the Vendors table
```

```
The primary key
```

```
The foreign key
```

```
Murach's Oracle SQL and PL/SQL, C1
© 2014, Mike Murach & Associates, Inc.
Slide 37
The relationship between two tables