An Overview

Database system plays a central role in our work and life. For example, there is a significant database component behind any non-trivial web application. Big data and machine learning have also been picking up, now that we have collected so much data.

Database is also one of very few computer science areas that is relatively self-contained, and supported by a solid theoretical (mathematical) body of knowledge, i.e., the relational database (RDB) theory.

In this course, we will focus on the back end, i.e., RDB based database design and SQL query design.

In particular, we will address such issues as how to design a “solid” database, put in data, revise them if we have to, and get out information correctly and fast.
The other stuff...

Although we will also touch on the *front end*, e.g., HTML based data presentation, the details of this latter part will be left to subsequent courses, such as *CS3020 Web Programming*, *CS3030 Adv. Web Programming*, and *CS3820 Human Computer Interface*.

You will explore critical project management issues in, e.g., *CS3720 System Analysis and Design* and *CS4140 Software engineering*.

Security has become a (h)uge part of database system. We will push database security issues to other courses, such as *CS4420 Computer Security*.

If you want to learn Big data, sign up *CS 4500 Special Topics on Big Data Analytics*.
A bunch of names

A database is based on data, which carries information, the lifeblood of every enterprise, e.g., Amazon, Facebook, etc.

There are essentially two aspects of data. The logical aspect provides a meaning of data, and a physical one addresses its storage.

For example, the number 108 could mean the temperature in Phoenix on some day during last summer, the product of $1^1, 2^2$ and $3^3$, the number of a local road between Plaistow, MA and Rochester, NH, a room number, etc.. Physically, it takes a byte to store this number.

We once put all the data on paper, now they are kept in various kinds of storage media, in terms of bits, bytes, fields, records, and files.

A database is a collection of related files. A database management system (DBMS) manages databases.
Why do we need data?

Every enterprise, such as a company, a college, etc., keeps lots of data for various purposes. A college keeps students’s transcripts to maintain academic records, enrollment data, i.e., how many students applied, get admitted, paid deposits, enrolled, at what time, etc., to maintain a business record; and other stuff, such as parking violation record and fee.

Because of the huge amount of data we have to work with, the boring nature of the related processing, and, most importantly, such processing is often mechanical thus can be automated, today, almost everybody uses computers to process data, often via database applications, because those data are related.

We are going to learn some of the basic concepts and practical skills involved in database applications in this course.
DBMS

DBMS is a highly sophisticated piece of software that is used to manage all database related applications, including all the involved files, mainly data files and index files. Those index files put data into order so that we can get a quick access.

The relationship among, or the structure of, data, often called data modeling, was originally thought to be hierarchical. Examples include the organization chart, the menu structure, etc.
Alternatives

Sometimes, things can be a bit more complicated. For example, a manager manages employees, while the manager herself is also an employee. Also, friendship is a relationship among persons. Thus, we can use a more general network, or graph, model to characterize the relationship among data. Below shows a simple collection of relationships among entities.

DBMS based on these two models were once quite popular, such as IMS by IBM to support the Apollo program in the 1960’s.
Relational model

Edgar Codd (1923–2003) had a different idea. He wrote a few papers in the 1960’s and 1970’s, and suggested to look at the relation between data in terms of mathematical set theory. Codd won the Turing Award in 1981 for this work.

Oracle is among the companies that picked up Codd’s idea and built much more satisfactory DBMS, based on this RDB model.

Today, RDB is the dominant data model in this business, which is also what we are going to talk about most of the time in this course.

There are tons of DBMS in the market. They all try to address such issues as how to set up a database, through a data declaration language (DDL); and how to design queries to get the information we need, e.g., Who has the highest, and the lowest, GPA among her peers? with a data manipulation language (DML).
What will we use?

All these RDB based systems provide various features according to the *Structured Query Language* (SQL) standard, which started in 1989 with the most recent one being SQL 2011, mostly completed, and SQL 2015/2016 is already(?) on its way.

http://www.jcc.com/resources/sql-standards

From a user’s perspective, these systems are more or less the same. The difference lies in the interface, internal power, features, and reliability. All come down to the price.

We will use *MySQL* ([http://www.mysql.com](http://www.mysql.com)) in this course since it does its job reasonably well, it is free and (thus) widely used, although it only partially implements the SQL standard in an incremental manner with its most recent version being 5.7.

It has since been bought by Oracle. 😞
The essentials

Database is supported by a very solid mathematical foundation. 😊

Since we often put in huge amount of data into a database, we want to have a database that contains all the information we need, but with as little redundancy as possible. The normalization theory can guide us through the whole process of taking out as much fat as we wish, with more and more effort.

On the DML side, the relational algebra will guide us through the process of writing correct, and fast, SQL queries.

The related topics, and their applications, are essentially covered in Chapters 3-8, or Units 4 through 9 in the course page. In particular, Chapters 3, 5 and 6 provide the very foundation of the RDB theory, Chapter 4 a visual tool for setting up the database, and Chapters 7 and 8 provide some examples of application.
MA2200, **MA2250**, or MA3200

Both the normalization process and the relation algebra depend heavily on math stuff, mainly set related operations as union, intersection, Cartesian product; subsets, relations, and functions.

That is why these courses, especially *MA 2250 MA for CS*, are listed as one of the prerequisites of this one.

If you are not that familiar with them, it is a great idea to check out a book on these subjects from the library.

You can also do a Google search on “Relational Algebra tutorial” with answers (819,000 results as of September 11, 2017). Khan Academy provides a very light treatment, too light for this course.

Let's do a (p)review of mathematics in its minimum, and lots of them will come in later.
Little (p)review

A set is a collection of objects. For example, we have a set of apples, a collection of chairs, a bunch of students, and a couple of prerequisites for this course: Intro to Programming (database programming) and Finite Math, Discrete Math, or Math for CS.

Let $A, B$ be two sets, we can apply the following operations to them to get back another set:

- **Union**: $A \cup B = \{x | x \in A \lor x \in B\}$
- **Intersection**: $A \cap B = \{x | x \in A \land x \in B\}$
- **Complement**: $A - B = \{x | x \in A \land x \notin B\}$

and check if there is a relation between two sets, which returns a answer of “yes” or “no”:

- **Subset**: $A \subseteq B \equiv \forall x, x \in A \Rightarrow x \in B$. 

11
Visually speaking

*Venn diagrams* are often used to specify the relationship among sets.

For example, the following tells the union and intersection operations, respectively.

Let $A$ be $\{1, 2\}$, $B$ be $\{2, 3\}$. We have that $A \cup B = \{1, 2, 3\} = B \cup A$, $A \cap B = \{2\} = B \cap A$, $A - B = \{1\}$, but $B - A = \{3\}$ and $A \not\subseteq B$.

Clearly, both Union and Intersection are commutative, but Complement is not.
Sequences and tuples

A sequence of objects is a list of these objects in some order. For example, $(7, 21, 57)$ represents the sequence $7, 21, 57$, which is not the same as \{7, 21, 57\}.

**Question:** What is the difference between a set and a sequence?

**Answer:** Order.

A finite sequence with $k$ elements is usually called a $k$–tuple. Particularly, a 2-tuple is called a pair.

For example, $(7, 21, 57)$ is a 3-tuple, or a triplet, while $(7, 21)$ is a pair.
Cartesian product

This is something that we use quite a bit in RDB to get data from multiple places.

Let $A, B$ be two sets, we can define the *Cartesian product* on $A$ and $B$ as follows:

$$A \times B = \{(a, b) | a \in A \land b \in B\}$$

For example, if $A = \{1, 2\}$ and $B = \{x, y, z\}$, then

$$A \times B = \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\}.$$  

**Question:** What is $A \times B \times \{a, b, c\}$?

**Answer:** A bunch of eighteen triplets.

**Question:** Why eighteen? What is the size of $A \times B \times C$, in general?

**Answer:** $|A \times B \times C| = |A| \times |B| \times |C|$. 

14
Function

A function, just like a Java method, or a Python function, takes in one or more input(s) and sends out an output, e.g., \( f(a) = b \). A function is a special mapping in the sense that the following always holds: for all \( x, y, x = y \Rightarrow f(x) = f(y) \).

Let \( f \) be a function, the collection of its possible inputs and outputs are called its domain, and range, respectively.

\[
f : D \rightarrow R.
\]

For example, the absolute value function, \( \text{abs} \), is obviously a function in this sense. We have that:

\[
\text{abs} : \mathbb{Z} \rightarrow \mathbb{Z}
\]

\[
\text{abs}(x) = \begin{cases} 
  x & \text{if } x \geq 0, \\
  -x & \text{otherwise}.
\end{cases}
\]

The above also provides a procedure to compute \( \text{abs}(x) \) for any \( x \).
Relation

Now, we come to the central idea of RDB, *relation*.

A *predicate* is a function with a binary range: \( \{T, F\} \), where ‘\( T \)’ (‘\( F \)’) stands for “true” (“false”).

Since \( x \leq y \) is either true or false, depending on if the current value of \( x \) is no more than that of \( y \), ‘\( \leq \)’ is a predicate.

A *predicate* whose domain is a set of \( k \)–tuples, \( k \geq 2 \), is called a *relation*. Thus, ‘\( \leq \)’ is also a relation, since its domain, the collection of all the pairs \( (x, y) \) such that \( x \leq y \), is a set of 2-tuples.

Khan academy does have a pretty good one on functions and relations.
The gist of RDB

Let $R$ be a $n$–ary relation, $(a_1, \ldots, a_n)$ be a $n$–tuple, $R(a_1, \ldots, a_n)$ means that the latter is true.

If “A freshman John Doe, with id 1111, does live in 123 Main St.” is true, we can represent this fact as $(1111, \text{JohnDoe}, \text{123MainSt.}, \text{Freshman})$.

In general, we can have a predicate “A person with Name $X$, Id $Y$, status $Z$, lives in $A$. “ and then collect all the true statements based on this predicate at the moment as a relation and represent it as the following table:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Address</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>John Doe</td>
<td>123 Main St.</td>
<td>Freshman</td>
</tr>
<tr>
<td>2222</td>
<td>Mary Smith</td>
<td>1 Lake St.</td>
<td>Freshman</td>
</tr>
<tr>
<td>1234</td>
<td>Joe Blow</td>
<td>6 Yard Ct.</td>
<td>Junior</td>
</tr>
</tbody>
</table>

This is the mathematical basis of RDB, a collection of such tables.
The other stuff

Besides the essential ones as mentioned on page 9, we also discuss some other stuff.

Unit 10 is on data recovery, since we bet our last penny on the security of database. We have to figure out how to get it back if something happens.

DBMS is quite expensive, thus it is used by many people during the same time, i.e., concurrently. This may cause some side effects. Unit 11 studies the related issues, touching on the important subject of concurrent processing, now that even our smart phone has multiple cores.

Finally, RDB, albeit popular, does have some restrictions. Object database, the subject of the last unit sheds some light from a OOP perspective, which you should know really well from the earlier programming courses.
Practically speaking

Once the database is set up, we will try to get out information by *querying a database*, i.e., *applying operations on those relations as sets.*

We can pose SQL queries, one at a time, in the SQL prompt, but this is not the norm.

To have a sophisticated application, such as an interactive session (WebReg or Moodle), we also need to have other features, such as the conditional, and the loop structures. We thus often embed SQL queries in a *host language*, which provides the aforementioned general features.

People often use such languages as *C++* and *Java* as the host language, but we will use the WEB friendly *PHP* ([www.php.net](http://www.php.net)) because...
... another very important issue is the presentation part of the database application. This part gets in the data from an end user, and throws back the result after certain, hopefully correct, manipulation.

In today’s world, there is no better and/or more popular platform than WEB. Thus, many of the database applications directly use the HTML format to write up their presentation platform. We will do the same.

To summarize, practically, we use HTML to write up a WEB friendly interface, embedded with database programming code written in PHP, which sends queries to, and receives the result from, (a) MySQL based database(s).
On the lab work

This course is intended to provide a solid background in DB related concepts, and a strong DB programming component. Thus, besides the textbook, there are two separate, but related, lab notes, for the latter effort.

Considering the amount of the work involved, although there will be a roughly once-a-week lab, students are expected to spend a significant amount of time on their own 😊 to go through the material as contained in the lab notes and complete the assigned lab works. 😊

Moreover, because of the complexity of the DB applications, it is typically a product of team work. Thus, besides individual programming assignments as found in the lab notes, there will also be a team based project. 😊
On turing

We will do all the work on turing with details being provided in §3 of the PhP labnotes.

To remotely work with turing, you can use, e.g., SSH (Secure Shell Exchange), which can be obtained through https://shareware.unc.edu/.

SSH comes with two pieces: file transfer and remote log-in. You can use any text editor such as Notepad++ to come up with a PhP script, use the file transfer piece of SSH to upload the script to turing, then get it executed with its URL.

When we get there, you can use the log-in piece to get into turing to set up databases and play out queries.

We will go through the whole nine yards of this process in you-know-where.