In other words, a somewhat random list of words and concepts that are necessary to move on…

*Read Chapter 1, including the historical notes on pages 29 - 31.*
Concept #1: Databases & Database Management Systems
What is a Database?

According to the book:
- Collection of interrelated data
- Set of programs to access the data
- A DBMS contains information about a particular enterprise
- DBMS provides an environment that is both convenient and efficient to use.

Another definition (know these):
- A database is a collection of organized, interrelated data, typically relating to a particular enterprise
- A Database Management System (DBMS) is a set of programs for managing and accessing databases
Some Popular Database Management Systems

- Commercial “off-the-shelf” (COTS):
  - Oracle
  - IBM DB2 (IBM)
  - SQL Server (Microsoft)
  - Sybase
  - Informix (IBM)
  - Access (Microsoft)
  - Cache (Intersystems – nonrelational)

- Open Source:
  - MySQL
  - PostgreSQL

Note: This is not a course on any particular DBMS!
Some Database Applications

Anywhere there is data, there could be a database:
- Banking: accounts, loans, customers
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Course context is an “enterprise” that has requirements for:
- Storage and management of 100’s of gigabytes or terabytes of data
- Support for 100’s or more of concurrent users and transactions
- Traditional supporting platform, e.g., Dell PowerEdge R720xd, 68 processors, 16GB RAM each, 50TB of disk space
Prior to the availability of COTS DBMSs, database applications were built on top of file systems – coded from the ground up.

**Drawbacks of this approach:**
- Difficult to reprogram sophisticated processing, i.e., concurrency control, backup and recovery, security
- Re-inventing the wheel can be expensive and error-prone.
- “We need a truck, let’s design and build our own truck.”

According to the book, this leads to:
- Data redundancy and inconsistency
- Multiple files and formats
- A new program to carry out each new task
- Integrity constraints (e.g. account balance > 0) become embedded throughout program code, etc.

Database systems offer proven solutions for the above problems.
Even to this day, engineers will occasionally propose custom-developed file systems.

So when should we code from scratch, and when do we buy a DBMS??

- How much data?
- How sophisticated is the processing of that data?
- How many concurrent users?
- What level of security?
- Is data integrity an issue?
- Does the data change at all?
Concept #2: Levels of Abstraction
Levels of Abstraction

- **Physical level** - defines low-level details about how data item is stored on disk.

- **Logical level** - describes data stored in a database, and the relationships among the data (usually conveyed as a data model, e.g., an ER diagram).

- **View level** - defines how information is presented to users. Views can also hide details of data types, and information (e.g., salary) for security purposes.
Levels of Abstraction

- **Physical data independence** is the ability to modify the physical schema without having an impact on the logical or view levels.

- Physical data independence is important in any database or DBMS.

- Similarly one could define *logical data independence*, but that would not be as meaningful.
Concept #3: Instances vs. Schemas
The difference between a database schema and a database instance is similar to the difference between a data type and a variable in a program.

A database schema defines the structure or design of a database.

More precisely:

- A logical schema defines a database design at the logical level; typically an entity-relationship (ER) or UML diagram.
- A physical schema defines a database design at the physical level; typically a DDL file.

An instance of a database is the combination of the database and its contents at one point in time.
Concept #4: Data Models
What is a Data Model?

- The phrase “data model” is used in a couple of different ways.

- Frequently used (use #1) to refer to an overall approach or philosophy for database design and development.

- For those individuals, groups and corporations that subscribe to a specific data model, that model permeates all aspects of database design, development, implementation, etc.
What is a Data Model?

- **Common data models:**
  - Relational model
  - Object-oriented model
  - Object-relational model
  - Semi, and non-structured data models (XML)
  - Various other NoSQL models (graph, document, key/value)

- **Legacy data models:**
  - Network
  - Hierarchical
During the early phases of database design and development, a “data model” is frequently developed (use #2).

The purpose of developing the data model is to define:
- Data
- Relationships between data items
- Semantics of data items
- Constraints on data items

In other words, a data model defines the logical schema, i.e., the logical level of design of a database.

A data model is typically conveyed as one or more diagrams (e.g., ER or UML diagrams).

This early phase in database development is referred to as data modeling.
Examples of entity-relationship diagrams:

- Authors current (UML-ish) notation:
- Older (Chen) notation:

- Widely used for database modeling.
Regardless of the model, the end result is the same – a relational database consisting of a collection of tables:

<table>
<thead>
<tr>
<th>customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>12 Alma St.</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>4 North St.</td>
<td>Rye</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>3 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>Turner</td>
<td>123 Putnam Ave.</td>
<td>Stamford</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>100 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>Lindsay</td>
<td>175 Park Ave.</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>72 North St.</td>
<td>Rye</td>
</tr>
</tbody>
</table>

(a) The customer table

<table>
<thead>
<tr>
<th>customer-id</th>
<th>account-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>A-101</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>A-201</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-215</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>A-102</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>A-305</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>A-217</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>A-222</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-201</td>
</tr>
</tbody>
</table>

(c) The depositor table

<table>
<thead>
<tr>
<th>account-number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>A-305</td>
<td>350</td>
</tr>
<tr>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>A-217</td>
<td>750</td>
</tr>
<tr>
<td>A-222</td>
<td>700</td>
</tr>
</tbody>
</table>

(b) The account table
Concept #5: Query Languages
A query language is used to create, manage, access, and modify data in a database.

The list of query languages is quite long:
- [Link](http://en.wikipedia.org/wiki/Query_languages)

The most widely used query language is Structure Query Language (SQL).

At a high-level, SQL consists of two parts:
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
**Data Definition Language (DDL)**

- DDL is used for defining a (physical) database schema (see the book for a more complete example):

  ```
  create table account ( 
    account-number char(10), 
    branch-name   varchar(16), 
    balance       integer, 
    primary key (account-number)
  )
  ```

- Given a DDL file, the DDL compiler generates a set of tables.

- The authors also define a subset of DDL called *Data storage and definition language* for specifying things such as:
  - Location on disk
  - Physical-level formatting
  - Access privileges
Data Manipulation Language (DML)

- DML is used for accessing and manipulating a database.

- Two classes of DMLs:
  - Procedural – user specifies how to get the required data.
  - Non-procedural – user specifies what data is required, but not how to get that data.

- SQL is usually referred to as a non-procedural query language.
SQL Examples

- Find the name of the customer with customer-id 192-83-7465:
  ```sql
  select customer.customer-name
  from customer
  where customer.customer-id = '192-83-7465'
  ```

- Find the balances of all accounts held by the customer with customer-id 192-83-7465:
  ```sql
  select account.balance
  from depositor, account
  where depositor.customer-id = '192-83-7465' and
  depositor.account-number = account.account-number
  ```

- Databases are typically accessed by:
  - Users through a command line interface
  - Users through a query or software editing tool, e.g., MySQL Workbench
  - Application programs that (generally) access them through embedded SQL or an application program interface (e.g. ODBC/JDBC)
Concept #6: Database Users
Database Users

Users are differentiated by the way they interact with the system:

- Naïve users
- Application programmers
- Specialized users
- Sophisticated users
The DBA coordinates all the activities of the database system; has a good understanding of the enterprise’s information resources and needs.

DBA duties:
- Granting user authority to access the database
- Acting as liaison with users
- Installing and maintaining DBMS software
- Monitoring performance and performance tuning
- Backup and recovery

According to the book, the DBA is also responsible for:
- Logical and Physical schema definition and modification
- Access method definition
- Specifying integrity constraints
- Responding to changes in requirements

These latter tasks are frequently performed by a software or systems engineer specialized in database design.
Concept #7: DBMS Structure
Overall DBMS Structure
Overall DBMS Structure

Users, Programs

Queries
Commands

Database
Server

Query Processor

DDL Interpreter
HLL Compiler
& Linker
Query Evaluation
Engine

DML Compiler
Parser, etc.
Optimizer

Storage Manager

Authorization
& Integrity
Manager

Buffer Manager
File Manager

Transaction Manager

Backup
& Recovery
Concurrency
Control

Storage

Data
Data Dictionary

Indices
Statistical Data
The following components of a DBMS are of interest to us:

- transaction manager
- buffer manager
- file manager
- authorization and integrity manager
- query optimizer
A transaction is a collection of operations that performs a single logical function in a database application.

The transaction manager performs two primary functions:
- backup and recovery
- concurrency control

Backup and recovery ensures that the database remains in a consistent (correct) state despite failures:
- system, power, network failures
- operating system crashes
- transaction failures.

Concurrency-control involves managing the interactions among concurrent transactions.
The **buffer manager** loads data into main memory from disk as it is needed by the DBMS, and writes it back out when necessary.

The buffer manager is responsible for:

- loading pages of data from disk into a segment of main memory called “the buffer”; a.k.a. “the cache”
- determining which pages in the buffer get replaced
- writing pages back out to disk
- managing overall configuration of the buffer, decomposition into memory pools, page time-stamps, etc.

Sound familiar?
The **file manager** is responsible for managing the files that store data.

- formatting the data files
- managing free and used space in the data files
- defragmenting the data files
- inserting and deleting specific data from the files
The **authorization & integrity manager** performs two primary functions:

- **data security**
  - ensure that unauthorized users can’t access the database
  - ensure that authorized users can only access appropriate data

- **data integrity**
  - in general, maintains & enforces integrity constraints
  - maintains data relationships in the presence of data modifications
  - prevents modifications that would corrupt established data relationships
A given query can be implemented by a DBMS in many different ways.

The *query optimizer* attempts to determine the most efficient strategy for executing a given query.

The strategy for implementing a given query is referred to as a *query plan*. 