# Notes on Database and SQL Fundamentals

## Vocabulary / Fundamental Concepts

* Data – raw facts represented symbolically
* Information – data organized or prepared so as to stand to inform someone
* Knowledge – information used for actual decision-making
* Database – collection of related, structured data
* Database Management System (DBMS) – collections of programs that define/maintain and allow the manipulation of databases
* DBMS alternative? File-based storage
* DBMS is preferable because of additional functionality as detailed below and because of reduced redundancy
* Applications that are very data-intensive, highly specialized, and demand high performance may not use a DBMS at all!
  + Why? DBMS functionality can come at the cost of a performance overhead

## Advantages & Functions of Databases

* Efficiency: Query Processing & Optimization
* Safe, Multi-User Access: Transaction Management
* Concurrency
* Recovery
* Controlled Redundancy
* Reliability: Integrity Constraints
* Safety: Security & Privacy
* Efficiency: Physical Databases & Indexing
* Convenient: Database Administration
* Convenience: Database application development and Interface
* Can be programmed via “frameworks”
* Can run in conjunction with “middleware”
* Convenience: Standards & Interoperability (SQL in its various standards – SQL86 through to SQL 2008 ANSI/ISO standards
* The 2011 standard is 68 pages long in English
* Convenience: SQL is a high level *declarative* language
  + You only need to say *what* you want; you do not need to specify *how* to get it

DBMSs routinely handle terabytes/day-scale data with thousands of changes/second from hundreds of thousands of users.

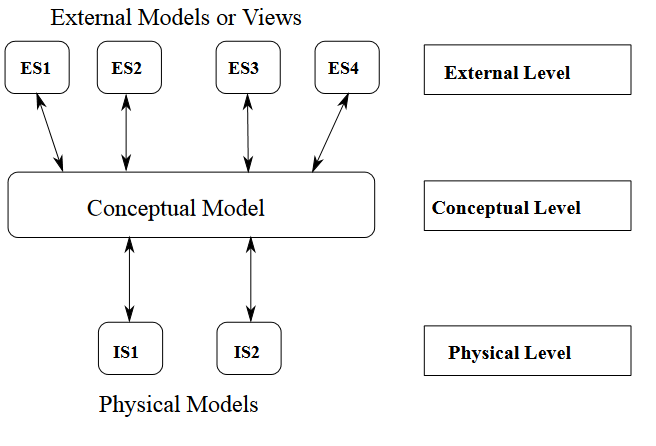
* Most people access databases indirectly, by way of applications – either web or desktop
* Database applications can be programmed via high level “frameworks”

## Key Database People

* DBMS implementor – builds the DBMS system
* Database designer – designs database solution to meet organization’s needs
* Database application developer – writes the codes that operates the database and makes it useful – typically involves interfaces like the web or desktop apps
* Database administrator - maintains the DBMS, deals with backup, recovery, user accounts, HW and network issues, generally keeps things running smoothly
* Users - everyone, everywhere nearly all the time

## Three-schema architecture and Data Independence

* Each of the following are independent of one another:
  + What individual classes of users sees (“external view”)
  + The overall global logical design of the DB (“conceptual level”)
  + The physical storage details (HW, indexes, disks, etc.)
* Each of the lower levels can be hidden from the one above it
* Changes in one level needn’t impact the others
  + Can change indexes, file structure, disks without altering conceptual level → “Logical Data Independence”
  + Can change parts of the conceptual schema without changing individual views → “Physical Data Independence”



## Data Models

What is a data model?

* A data model is a tool for describing data, their relationships, semantics, and integrity constraints (rules for assuring the data are accurate, meaningful).
* A framework for organizing and manipulating data

There are three essential components in any data model:

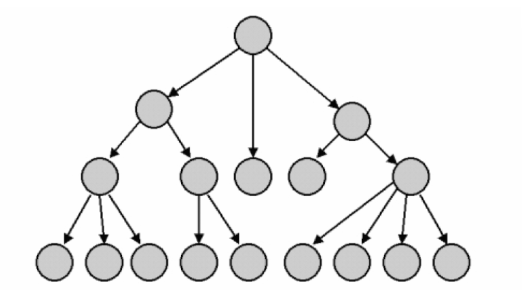
1. Representation (Data and their inter-relationships)
2. Operations (CRUD – Create, Read (i.e. query), Update, Delete)
3. Integrity Constraints (Statements which specify the correctness of databases.)

### Semantic Data Models

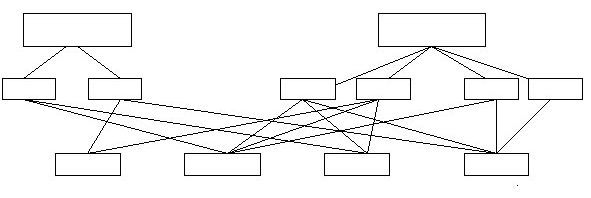
* Used to describe the data at a conceptual level
* Usually converted to a lower-level record-based logical models that can be implemented in DBMSs
* Examples:
  + **Entity-Relationship (ER) model – this will be the exclusive model for this class**
  + NIAM / ORM – “Object-Role Modeling”
  + Binary Data Model
  + Many others

### Record-Based Logical Models

* Used to specify both the overall structure of the database and a high level description of the implementation of that structure in a DBMS.
* Most commercial database systems are based on these models. Examples:
  + **Relational Model – based on inter-related tables: Will be our records-based model for the course**
  + Hierarchical Model – use in, for example LDAP/Active Directory:



* + Network Model – like hierarchical but allows multiple parents/poly-hierarchy:



**Sneak Preview**

In a subsequent competency you will “get your hands dirty” with the most prevalent modern semantic data model, the Entity-Relationship (ER) model. Using the ER model, you will be able to turn a detailed description of a data problem into a high-level plan for the database that will solve it. Later on in the competency, you’ll learn how to turn that high level plan into a lower-level, records-based plan that can be directly implemented in the RDBMS via SQL. For now though, your focus is on understanding the basics of relational databases and getting some hands-on experience with SQL.