**CSCI 4333.2**

8/28/2024

**Database Basics**

by K. Yue

**1. Introduction**

* It is important to be familiar with the basic terms and concepts of databases in this course.
* A database system is built by using a Database Management System (DBMS).
* One popular DB engine ranking: <http://db-engines.com/en/ranking>.
* We focus on Relational DBMS (RDBMS).
* Examples of Relational DBMS:
	1. Access: most popular 'departmental' DBMS
	2. Oracle: Most popular commercial DBMS
	3. MS SQL server: Likely second most popular commercial DBMS
	4. MySQL: most popular open source DBMS
	5. MariaDB: highly compatible to MySQL (more open source than MySQL)
	6. Postgres: popular open source DBMS known for innovation and functionality.
	7. SQLite: most popular portable DB engine.
* Relational DBMS basically use the relational model (with extensions).
* There are many other models. Examples:
	1. Object-Oriented Database (OODB): e.g., db4o, Gemstone, etc.
	2. Big Data:
		+ Document DB: e.g., MongoDB, CouchDB
		+ Key-Value DB: e.g., Redis, LevelDB
		+ Wide Column DB: optimized over large dataset; store columns together, not rows. E.g. Cassandra and HBase.
		+ Graphical DB: e.g., Neo4J

**2. Users**

* Users drive requirements. It is always important to find out the types of users.
* There are many types of users in a RDB. Three major kinds:
	1. End users: usually do not use SQL to access the database directly. Examples:
		+ front-end users
		+ managers and staff
		+ domain experts
	2. DB application developers and data analysts
		+ Develop DB solutions using SQL.
		+ With various levels of access privileges.
	3. DB administrators
		+ Manage the entire DB, such as:
			- conceptual and physical database design and implementation
			- security
			- user account management
			- backup and recovery
			- performance tuning
* The likely role of most of you: DB and application developers and data analysts.
* You are likely not the end users of the DB system you built.
* Thus, do *not* build the database for yourself.

**3. DB Development Phases**

* The classical waterfall software development life cycle can be useful as a basis to understand the various phases of database development.
	1. Requirement: conceptual modeling.
		1. Planning
		2. Analysis
	2. Design:
		1. Logical modeling and design
		2. Physical Design
	3. Implementation
	4. Testing
	5. Maintenance
* There are many other software lifecycle models.
* DB development is a kind of software development.



**3.1 Data Modeling**

General conceptual model. See, for example: [https://en.wikipedia.org/wiki/Conceptual\_model\_(computer\_science)](https://en.wikipedia.org/wiki/Conceptual_model_%28computer_science%29).

1. Capture domain knowledge and requirements from the business and application perspectives.
2. Driven by requirements.
3. Construct a conceptual model iteratively.

See, for example: <https://en.wikipedia.org/wiki/Conceptual_schema>.



**Identify and capture user requirements:**

1. Likely the most tedious and difficult parts for many traditional applications.
2. Collect documents of existing systems.
3. Study documents of existing systems.
4. Talk with domain experts and end users.
5. Model the problem using a modeling language, such as UML, ER, etc.
6. Document the captured requirements: e.g., modeling, requirement specifications, data dictionary, etc.
7. Iteratively refine and correct the model until enough details are captured.

**3.2 Design database solutions**

1. Select the appropriate data model of the database.
2. Select the appropriate DBMS.
3. Design the *logical* model.
4. Design the architecture of the DB system.
5. Design the physical database.
6. Design external views.
7. Design individual components.

**3.3. Implementation and testing**

1. Implement and test design.
2. Optimize performance.

**4. The Three-Layered DB Architecture**

* The three-layered database architecture is well known and you can get a lot of information about it from the Web. Examples:
	+ A simple one: <https://www.tutorialspoint.com/Three-Level-Architecture-of-Database>
	+ A more nuanced one: <http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm>
	+ Consult Figure 2.4 of Ricardo.
* Use the *layer* pattern to manage complexity. The *layer pattern* is an important concept in Computer Science and software architecture.
* Three levels:
	+ External or view level: Describes a part of the database for a particular user group, Provide the right level of abstraction and security control.
	+ Logical level: Describe logical structure of the entire database.
		1. Some practitioners call the 'logical level' the 'conceptual level'. This can cause confusion as other distinguish between 'conceptual model' and 'logical model'.
	+ Internal/Physical level: Describe physical storage structure of the database.
* Provide data independence:
	+ Logical data independence:
		1. between logical database and external views.
		2. Changes in the logical database may not affect the external views.
	+ Physical data independence:
		1. between logical database and physical database.
		2. Changes in the physical database do not affect the correctness of the logical database.
* The logical level is the focus.

