Database System Concepts: Ch.1 - Introduction

database-management system (DBMS) - a collection of interrelated data and a set of programs used to access those data

* the term *database* is usually used to refer to the collection of data itself

online transaction processing - mode of use for a database where there exists a large number of users; however, each user retrieves relatively small amounts of data and perform small updates to the system

data analytics - mode of use for a database that is used to process data to draw conclusions and infer rules/ decision procedures which are then used to drive business decisions

Data Models

Relational Model - uses a collection of tables to represent both data and the relationships among those data

Entity-Relationship Model - E-R data model uses collection of objects called entities and relationships among these objects.

* this is more of a way of modeling the data logically while the relational model is more of the implementation based on the model

NOTE: Focus on the above models. These are the most commonly used.

Semi-structured Data Model - permits the specification of data where individual data items of the same type may have different sets of attributes

* *JSON* and *XML* are widely used semi-structured data representations
* this data model is covered more in chapter 8 of the text

Object-based data model - store objects in relational tables; this type of model basically extends the relational model with notions of encapsulation, methods, and object identity

Data Abstraction

Database system developers use complex data structures to represent data in the database for efficiency reasons. The DB developers hide the complexity from users using several levels of *data abstraction*

Physical - lowest level of abstraction that describes how the data are actually stored

Logical - the next highest level, describes what data are stored in the database and what relationships exist among them
* this level should be *physical data independent*
* the level used by database administrators

View - highest level of abstraction, describes only part of the entire database
* exists to simplify database-user interaction with the system
* DB system can provide different views for the same database

index - data structure used to support efficient retrieval of records; form part of the physical level
* **NOTE:** I want to learn about these data structures, find another resource after building up some more 'lower-level' programming chops in spare time, not the most important topic

Instances and Schemas

instance - the collection of information stored in a database at a particular moment

schema - the overall design of the database

physical schema - describes the database design at the physical level

logical schema - describes the database design at the logical level

subschemas - schemas that describe the different views of a database

Database Languages

Data-definition Language (DDL)

data-definition language (DDL) - specifies the database schema, also specifies additional properties of the data

data storage and definition language - special type of DDL that specifies storage structure and access methods used by the database system

the DDL is used to provide facilities to specific constraints in the database

the constraints can be a simple predicate; however, these can be costly to test each time the database is updated so DBs tend to implement only integrity constraints that can be tested with minimal overhead:

* *domain constraints* - domains (set of possible values) are the most elementary form of an integrity constraint; tested by the system each time new data is entered into the database

* *referential integrity* - ensure that a certain value appears in one relation but also appears in another relation
  - ex: For a database containing information on a department and course, we want to be sure the department actually exists in the university. So, the department name value in a course table must also appear in some record of the department relation.

  - if violated, simply reject the action that caused the violation

* *authorization* - constraints applied to limit what users have access to
  - most common: read authorization, insert authorization, update authorization, and delete authorization

data dictionary - special type of table that can be accessed and updated only by the database system itself
* metadata is added to the data dictionary whenever a DDL statement is processed

Data-Manipulation Language

data-manipulation language (DML) - enables users to access or manipulate data as organized by the appropriate data model
- types of access: retrieval, insertion, deletion, modification

procedural DMLs - require a user to specify what data are needed and how to get those data

declarative DMLs - require a user to specify what data are needed without specifying how to get those data

the *query processor* component of a database system translates DML queries into sequences of actions at the physical level of the database system
* this component is covered in chapters 15 and 16 and chapter 22 covers query processing in i common parallel and distributed settings

Database Access from Application Programs

Generally, non-procedural languages such as SQL aren't as powerful as a universal turning machine (general-purpose programming language)

- SQL doesn't support input from user, output to displays, network communication, etc.

these types of computations must be written in a host language (C, C++, Python, Java, etc.) with embedded SQL statements

- to access the database, DML statements need to be sent from the host to the database for execution, most commonly done via an API used to send DML and DDL statements to the DB and get results

Database Design

NOTE: This section only briefly covers the database design process, the details are fleshed out through the test. Be sure to use the information in these chapters to supplement material from COP4710. These two resources will make up the bulk of the learning material for database systems design.

conceptual-design - phase where a high-level data model provides the database designer with a conceptual framework in which to specify the requirements of the user
- focus is to describe the data and their relationships rather than on specifying physical storage details
- deliverable is a specification of user requirements
- how?
    * use ER model or employ normalization (ch.6 and 7 cover these along with COP4710

logical-design - phase where the designer maps the conceptual scheme onto the implementation data model of the system to be used

physical-design - phase in which i the designer uses system-specific schema to define the physical features of the database (file org, internal storage structures, etc. [ch.13])

Database Engine

Functional components of a database system include: storage manager, query processor, transaction management component

Storage manager

storage manger - component of a database system that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system
* the storage manager is responsible for the interaction with the file manger by translating DML statements into file-system commands

components of the storage manger:
* authorization and integrity manager - tests integrity constraints and checks authority of users access
* *Transaction manager* - ensures DB remains in consistent state, concurrent transactions execute w/o conflict
* *File manager* - manage the allocation of space on disk storage and data structures used to represent the data on disk
* *Buffer Manager* - fetches data from disk storage into main memory; decided what to cache in main memory; mission critical part since it enables the database to handle data sizes much larger than the size of main memory

data structures used by the storage manager:
    * *data files* - stored the DB
    * *data dictionary* - stores metadata about the structure of the database, schema of the database
    * *indices* - an index provides pointers to data items that hold a particular value

NOTE: The details of storage media, file structures, and buffer management are covered in halter 12 and 13, then methods for accessing data are discussed in chapter 14

<u>query processor:</u>

query processor - allows the database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of the system
    * translates updates and queries written in a nonprocedural language (at the logic level) into an efficient sequence of operations at the physical level

Transaction Manager

transaction manager - allows app developers to treat a sequence of database accesses as though they were a single unit that either happens in its entirety or not at all

requirements:
* *atomicity* - all-or-nothing requirement, either all actions within a transaction occur or the transaction doesn't
* *consistency* - correctness requirement, the system must preserve consistency of the database
* *durability* - persistence requirement, successful transaction results must persist

transaction - collection of operations that performs a single logical function in a database application
* each transaction unit is of both atomicity and consistency

components:
* *recovery manager* - ensures atomicity and durability properties
  - a failed transaction must have no effect on the state of the database
* *concurrency-control manager* - controls the interaction among the concurrent transactions to ensure the consistency of the database

Database and Application Architecture

NOTE: On page 23 of the text, there is a figure that gives the general architecture of a database system

two-tier architecture - database applications reside at the client make, and invokes database system functionality at the server machine through query language statements

three-tier architecture - used in most modern database applications; the client machine acts as a front end and does no actually contain any direct database calls; web browsers and mobile applications are the most commonly used application clients

* instead of invoking DBS functionally directly with the DB over the network, the front end communicates with an application server which in turn communicates with the database system
* the business logic of the application is stored on the application server instead of on the client machine