Study Guide

# Chapter 1

* Data in context (INFORMATION): Data that has been processed in such a way that the knowledge of the person who uses the data is increased
* Metadata: Data that describe the properties or characteristics of end-user data and the context of that data.
* Disadvantages of file processing systems: Program-Data dependence, duplication of data, limited data sharing, lengthy development times, excessive program maintenance
* Database Approach
  + Relational Databases: Establish relationships between entities by means of common fields included in a file
  + Database Management System (DBMS): Software system that enables the use of a database approach
  + Advantages of database approach: Program-data independence, planned data redundancy, improved data consistency, improved data sharing, increased productivity of application development, enforcement of standards, improved data quality, improved data accessibility and responsiveness, reduced program maintenance, improved decision support
* Components of the database environment: Data modeling and design tools, repository, DBMS, database, application programs, user interface, data and database administrators, system developers, end users
* Database Development Process
  + Planning-Enterprise Modeling, Planning-Conceptual Data Modeling, Analysis-Conceptual Data Modeling, Design-Local Database Design, Design-Physical Database Design and Definition, Implementation-Database implementation, Maintenance-Database maintenance
* ANSI/SPARC Three Schema Architecture
  + External Schema: View of managers and other employees who are the database users
  + Conceptual Schema: Combines different external views into a single, coherent, and comprehensive definition of the enterprise’s data.
  + Internal Schema: Consists of two schemas: Logical schema and physical schema. Logical schema is representation of data for a type of data management technology. Physical schema describes how data are to be represented and stored in secondary storage using a DBMS
  + Data Independence: The data is not dependent of the format it’s being stored in
* Roles in the Database Development Process: Business analysts, systems analysts, database analysts and data modelers, users, programmers, database architects, data administrators, project managers, technical experts
* Range of Database Systems
  + Personal Database: Designed to support one user. Cannot easily be shared with other users
  + Multitier Client/Server Databases: Intended to support departments or divisions (25-100 people)
  + Enterprise Applications: Entire organization or enterprise

# Chapter 1.5

* Analysis
  + Information Flow Diagram
    - Information Flow
      * Everything in the database must come from somewhere
      * Everything in the input documents must go somewhere
      * Everything in the database must be used for something
      * Everything on the output documents must come from somewhere
    - Never connect two documents
    - Never connect two tasks
    - Get document and task names from the project description
* Specification
  + EER
    - Model constraints that can be modeled
  + Task Decomposition
    - Model tasks AND constraints that cannot be modeled using the EER
    - Psuedo code defines expected input DATA and steps to generate output DATA
* Design
  + Database Design (Relational Model) see chapter 4
  + Tasks (Psuedo-code w/ SQL)
    - Psuedo code that interacts with the database is replaced by actual SQL
* Implementation
  + Database Implementation on an actual DBMS
  + API using a programming language (w/ SQL)

# Chapter 2

* Entity-Relationship Model (E-R Model)
  + Notation
    - Entity Types
      * Strong: Exists independently of other entity types. Always have a unique characteristic called an identifier that is an attribute or a combination of attributes that uniquely distinguish each occurrence of that entity.
      * Weak: Existence depends on some other entity type. Does not typically have its own identifier
      * Associative: Entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.
    - Attributes
      * Identifier: attribute whose value distinguishes individual instances of an entity type.
      * Partial Identifier: an attribute on a weak entity that is later combined with identifier of the owner
      * Optional: an attribute that may not have a value
      * Derived: attribute whose values can be calculated from related attribute values plus possibly data not in the database
      * Multivalued: attribute that may take on more than one value for a given entity instance
      * Composite: attribute that has meaningful component parts which are more detailed attributes
    - Relationship Degrees
      * Unary: relationship between instances of a single entity type
      * Binary: relationship between the instances of two entity types and is the most common type of relationship encountered in data modeling
      * Ternary (n-ary): simultaneous relationship among the instances of three entity types
    - Relationship Cardinality
      * Mandatory One: Entity instance must have and can only have one relationship
      * Mandatory Many: Entity instance must have one or more relationships
      * Optional One: Entity instance can have up to one relationship or none
      * Optional May: Entity instance can have many relationships or none
* Business Rules
  + Characteristics of a GOOD Business rule: declarative, precise, atomic, consistent, expressible, distinct, business-oriented
  + Data names – what makes a good data name? Relate to business & not technical, be meaningful, be unique, be readable, be composed of words taken from an approved list, be repeatable, follow a standard syntax
  + Data definition – what makes a good data definition? Gathered from same source as all other requirements, usually accompanied by diagrams, stated in the singular and defined by what it is and not what it isn’t
* Entities
  + Entity TYPE: collection of entities that share common properties or characteristics
  + Entity INSTANCE: single occurrence of an entity type
  + Entity type versus system input, output, or user: a true data entity will have many possible instances, each with a distinguishing characteristic, as well as one or more other descriptive pieces of data
  + Strong Entity Type: one that exists independently of other entity types. Always have a unique characteristic, called an identifier
  + Weak Entity Type: Existence depends on some other entity type.
  + What makes a good entity type NAME: singular noun, specific to organization, concise, abbreviation
* Attributes
  + Required Attribute: Attribute that must be present for each entity instance
  + Optional Attribute: an attribute that may not have a value
  + Simple (atomic) attribute: an attribute that cannot be broken down into smaller components that are meaningful for the organization
  + Composite attribute: an attribute that has meaningful component parts, which are more detailed attributes
  + Single-valued attribute: can only have one value
  + Multi-valued attribute: may take on multiple values in one entity instance
  + Stored attribute: Stored in system
  + Derived attribute: values can be calculated from related attribute values plus possibly data not in the database
  + Identifier: attribute whose value distinguishes individual instances of an entity type
  + Partial Identifier: an attribute on a weak entity that is later combined with identifier of the owner
  + What makes a good attribute NAME? singular noun or phrase, unique, follow a standard format
* Relationships
  + Relationship TYPE: meaningful association between entity types
  + Relationship INSTANCE: an association between entity instances
  + Attributes on a relationship: attributes can be associated with a many-to-many (or one-to-one) relationship
  + When should a relationship become an associative entity? All the relationships for the participating entity types are “many” relationships the resulting associative entity type has independent meaning to end users, has one or more attributes in addition to the identifier, participates in one or more relationships independent of the entities related in the relationship
  + Time dependent data – using a time-stamp: a time stamp is simply a time value such as date and time that is associated with a data value

# Chapter 3

* Supertypes and Subtypes
  + Notation: Supertype is connected with a line to a circle, which in turn is connected with a line to subtype that has been defined
  + Attribute inheritance: the property by which subtype entities inherit values of all attributes and instance of all relationships of the supertype
  + When to use supertype/subtype relationships: there are attributes that apply to some but not all instances of an entity type, the instances of a subtype participate in a relationship unique to that subtype
  + Generalization: process of defining a more general entity type from a set of more specialized entity types
  + Specialization: top down process, the direct revers of generalization.
  + Combining Specialization/Generalization: depends on several factors, such as the nature of the problem domain, previous modeling efforts, and personal preference
  + Completeness Constraints
    - Total Specialization: each entity instance of the supertype must be a member of some subtype in the relationship
    - Partial Specialization: an entity instance of the supertype is allowed not to belong to any subtype
    - Disjoint rule: Supertype can be only one subtype
    - Overlap rule: Supertype can belong to multiple subtypes
  + Subtype Discriminator: a composite attribute when there is an overlap rule
* How attributes are inherited in a HIERARCHY: attributes are assigned at the highest logical level that is possible in the hierarchy; subtypes that are lower in the hierarchy inherit attributes not only from their immediate supertype but also from all supertypes higher in the hierarchy, up to the root.

# Chapter 4

* Relation: named, two-dimensional table of data
  + Definition
    - Data structure: data organized in form of tables, with rows and columns
    - Data manipulation: powerful operations used to manipulate data stored in relations
    - Data integrity: model includes mechanisms to specify business rules that maintain the integrity of data when they are manipulated
  + Keys
    - Primary key: attribute or a combination of attributes that uniquely identifies each row in a relation
    - Composite key: primary key that consists of more than one attribute
    - Foreign key: attribute in a relation that serves as the primary key of another relation
  + Properties of a Relation [IMPORTANT]
    - 1. Each relation has a unique name
    - 2. An entry at the intersection of each row and column is atomic
    - 3. Each row is unique
    - 4. Each column has a unique name
    - 5. The sequence of columns is insignificant
    - 6. The sequence of rows is insignificant
    - Removing multi-valued attributes: the second property of relations states that no multivaluted attributes are allowed in a relation
  + Integrity Constraints: rules limiting acceptable values and actions, whose purpose is to facilitate maintaining the accuracy and integrity of data
    - Domain Constraints: All values in a column of a relation must be from the same domain. A domain is a set of values that may be assigned to an attribute
    - Entity Integrity: Ensures that every relation has a primary key and guarantees that every primary key is non-null
      * Entity Integrity RULE: “No primary key attribute (or component of a primary key attribute) may be null.”
    - Referential Integrity: if there is a foreign key in one relation, either each foreign key value must match a primary key value in another relation or the foreign key value must be null.
  + Insertion Anomaly: To insert a new row, the user must supply values for multiple primary keys because of redundancy
  + Deletion Anomaly: Deleting a row removes other data that you didn’t want deleted
  + Modification Anomaly: The same record must be updated in multiple places due to redundancy
* NULL values: a value that may be assigned to an attribute when no other value applies or when the applicable value is unknown.
* Mapping EER Diagrams to Relations
  + Mapping regular entities: Regular entities are transformed into a relation in an E-R diagram. Name given to relation is generally same as entity type. Each simple attribute becomes attribute of relation. Identifier becomes primary key of corresponding relation.
  + Mapping weak entities: Create new relation and include all simple attributes as attributes of this relation. Include primary key of the *identifying* relation as a foreign key attribute. Primary key of new relation is combination of primary key of *identifying* relation and partial identifier of weak entity type
  + Mapping binary relationships:
    - One-to-many: Create relation for each entity type participating. Include primary key attribute of entity on one side of the relationship as a foreign key on the many side
    - Many-to-many: Create new relation that is included as a foreign key in the primary keys of each participating entity type. These attributes become the primary key of the new relation. Nonkey attributes associated with the many-to-many relationship are included in this new relation
    - One-to-one: Two relations are created, one for each participating entity type. Primary key of one of these relations is included as foreign key in other.
  + Mapping associative entities:
    - Identifier not assigned: Default primary key for associative relation is composite key that consists of two primary key attributes from other two relations. These become foreign keys that reference the other two relations.
    - Identifier assigned: Create new associative relation. Primary key is the identifier assigned. Primary keys for the two participating entity types are included as foreign keys in the associative relation.
  + Mapping unary relationships:
    - One-to-many relationships: Foreign key attribute is added to relation, which references primary key values of the same relation
    - Many-to-many relationships: Two relations created, one to represent entity type in relationship and an associative relation to represent many-to-many relationship itself. Primary key of associative relation consists of two attributes, both taken from primary keys of other relation. Nonkey attributes of the relationship are included in the associative relation.
  + Mapping ternary (n-ary) relationships: Create new associative relation. Default primary key consists of three primary key attributes for participating entity types. These attributes act in role of foreign keys that reference individual primary keys of participating entity types. Attributes of associative entity type become attributes of new relation.
  + Mapping supertype/subtype relationships: Create a separate relation for supertype and for each subtype; assign to relation created for supertype the attributes common to all members of supertype, including primary key; assign to relation for each subtype the primary key of the supertype and only those attributes that are unique to the supertype; assign one or more attributes of supertype to function as subtype discriminator
* Normalization
  + Not First Normal Form (NF^2): any data model suffering redundancy
  + First Normal Form (1NF): any multivalued attributes have been removed, so there is a single value at the intersection of each row and column of the table
  + Second Normal Form (2NF): any partial functional dependencies have been removed
  + Third Normal Form (3NF): any transitive dependencies have been removed
  + Steps to moving from NF^2 to 3NF
    - NF^2 to 1NF: there must be no repeating groups in the relation; a primary key has been defined which uniquely identifies each row
    - 1NF to 2NF: create new relation for each primary key attribute that is a determinant in a partial dependency, that attribute is the primary key of the new relation; move nonkey attributes that are only dependent on this primary key attribute from old relation to new relation
    - 2NF to 3NF: for each nonkey attribute that is a determinant in a relation, create a new relation, that attribute becomes the primary key of the new relation; move all attributes that are functionally dependent only on primary key of new relation from old to new relation; leave attribute that serves as primary key in new relation in the old relation to serve as foreign key allowing you to associate the two relations
  + Keys
    - Determinants: attribute on left side of arrow in functional dependency
    - Candidate Key: attribute, or combination of, that uniquely identifies row in relation
      * Unique Identification: for every row, value of key must uniquely identify that row. Implies that each nonkey attribute is functionally dependent on that key
      * Nonredundancy: no attribute in key can be deleted without destroying property of unique identification
      * “A candidate key is always a determinant, whereas a determinant may or may not be a candidate key”
    - Functional Dependency: constraint between two attributes or two sets of attributes
      * Partial: nonkey attribute is functionally dependent on part (but not all) of the primary key
      * Transitive: functional dependency between the primary key and one or more nonkey attributes that are dependent on the primary key via another nonkey attribute