**AIT 524: Database Management Essentials**

**Test 1 Study Guide**

Test 1 consists of two parts that are equally weighted:

* **Part 1** includes 50 multiple-choice/true-false questions and covers the topics listed below. Questions are based on the material covered in the textbook, presented in the Power-Point presentations and open-source materials provided on Bb for Classes 1-5.
* **Part 2** will assess how well students can apply concepts learned in class to solve real-world problems. Students will be asked to:
  + Write detailed business rules using an Entity-Relationship Model (ERD) for the given database
  + Draw an ERD for the given business rules
  + Write and/or explain queries using operations of relational algebra

Test 1 is closed book, closed notes. Students will be taking Test 1, Part 1, on Blackboard. Test 1, Part 2, will be taken on Blackboard and paper.

Topics include but are not limited to:

**Database Systems:**

* **ad hoc query**—A “spur-of-the-moment” question.
* **centralized database**—A database located at a single site.
* **data**—Raw facts, that is, facts that have not yet been processed to reveal their meaning to the end user.
* **data anomaly**—A data abnormality that exists when inconsistent changes to a database have been made. For example, an employee moves, but the address change is corrected in only one file and not across all files in the database.
* **data dependence**—A data condition in which the data representation and manipulation are dependent on the physical data storage characteristics.
* **data dictionary**—A DBMS component that stores metadata—data about data. Thus, the data dictionary contains the data definition as well as its characteristics and relationships. A data dictionary may also include data that are external to the DBMS. Also known as an *information resource dictionary*. See also *active data dictionary*, *metadata*, and *passive data dictionary*.
* **data inconsistency**—A condition in which different versions of the same data yield different (inconsistent) results.
* **data independence**—A condition that exists when data access is unaffected by changes in the physical data storage characteristics.
* **data integrity**—In a relational database, refers to a condition in which the data in the database is in compliance with all entity and referential integrity constraints.
* **data management**—A process that focuses on data collection, storage, and retrieval. Common data management functions include addition, deletion, modification, and listing.
* **data redundancy**—A condition that exists when a data environment contains redundant (unnecessarily duplicated) data.
* **data warehouse**—Bill Inmon, the acknowledged “father of the data warehouse,” defines the term as “an integrated, subject-oriented, time-variant, nonvolatile collection of data that provides support for decision making.”
* **database**—A shared, integrated computer structure that houses a collection of related data. A database contains two types of data: end-user data (raw facts) and metadata. The metadata consist of data about data, that is, the data characteristics and relationships.
* **database design**—The process that yields the description of the database structure. The database design process determines the database components. Database design is the second phase of the database life cycle.
* **database management system (DBMS)** —Refers to the collection of programs that manages the database structure and controls access to the data stored in the database.
* **database system**—An organization of components that defines and regulates the collection, storage, management, and use of data in a database environment.
* **desktop database**—A single-user database that runs on a personal computer.
* **distributed database**—A logically related database that is stored over two or more physically independent sites.
* **enterprise database**—The overall company data representation, which provides support for present and expected future needs.
* **field**—A character or group of characters (alphabetic or numeric) that defines a characteristic of a person, place, or thing. For example, a person’s Social Security number, address, phone number, and bank balance all constitute fields.
* **file**—A named collection of related records.
* **information**—The result of processing raw data to reveal its meaning. Information consists of transformed data and facilitates decision making.
* **islands of information**—A term used in the old-style file system environment to refer to independent, often duplicated, and inconsistent data pools created and managed by different organizational departments.
* **knowledge**—The body of information and facts about a specific subject. Knowledge implies familiarity, awareness, and understanding of information as it applies to an environment. A key characteristic of knowledge is that “new” knowledge can be derived from “old” knowledge.
* **logical data format**—The way in which a human being views data.
* **metadata**—Data about data, that is, data concerning data characteristics and relationships. See also *data dictionary*.
* **multiuser database**—A database that supports multiple concurrent users.
* **operational database**—A database that is designed primarily to support a company’s day-to-day operations. Also known as a *transactional* database or *production* database.
* **performance tuning**—Activities that make a database perform more efficiently in terms of storage and access speed.
* **physical data format**—The way in which a computer “sees” (stores) data.
* **production database**—The main database designed to keep track of the day-to-day operations of a company. See also *transactional database*.
* **query**—A question or task asked by an end user of a database in the form of SQL code. A specific request for data manipulation issued by the end user or the application to the DBMS.
* **query language**—A nonprocedural language that is used by a DBMS to manipulate its data. An example of a query language is SQL.
* **query result set**—The collection of data rows that are returned by a query.
* **record**—A collection of related (logically connected) fields.
* **single-user database**—A database that supports only one user at a time.
* **structural dependence**—A data characteristic that exists when a change in the database schema affects data access, thus requiring changes in all access programs.
* **structural independence**—A data characteristic that exists when changes in the database schema do not affect data access.
* **Structured Query Language**—A powerful and flexible relational database language composed of commands that enable users to create database and table structures, perform various types of data manipulation and data administration, and query the database to extract useful information.
* **transactional database**—A database designed to keep track of the day-to-day transactions of an organization. See also *production database*.
* **workgroup database**—A multiuser database that supports a relatively small number of users (usually fewer than 50) or that is used for a specific department in an organization.

**Data Models**

* **attribute**—A characteristic of an entity or object. An attribute has a name and a data type.
* **business rule**—Narrative descriptions of a policy, procedure, or principle within an organization. Examples: A pilot cannot be on duty for more than 10 hours during a 24-hour period. A professor may teach up to four classes during any one semester.
* **class**—A collection of like objects with shared structure (attributes) and behavior (methods). A class encapsulates an object’s data representation and a method’s implementation. Classes are organized in a class hierarchy.
* **class hierarchy**—The organization of classes in a hierarchical tree where each “parent” class is a superclass and each “child” class is a subclass. See also *inheritance*.
* **conceptual model**—The output of the conceptual design process. The conceptual model provides a global view of an entire database. Describes the main data objects, avoiding details.
* **conceptual schema**—A representation of the conceptual model, usually expressed graphically. See also *conceptual model*.
* **connectivity**—Describes the classification of the relationship between entities. Classifications include 1:1, 1:M, and M:N.
* **constraint**—A restriction placed on data. Constraints are normally expressed in the form of rules. Example: “A student’s GPA must be between 0.00 and 4.00.” Constraints are important because they help to ensure data integrity.
* **Crow’s Foot notation**—A representation of the entity relationship diagram using a three-pronged symbol to represent the “many” sides of the relationship.
* **data definition language (DDL)**—The language that allows a database administrator to define the database structure, schema, and subschema.
* **data management language (DML)**—The language (set of commands) that allows an end user to manipulate the data in the database (SELECT, INSERT, UPDATE, DELETE, COMMIT, and ROLLBACK).
* **data model**—A representation, usually graphic, of a complex “real-world” data structure. Data models are used in the database design phase of the database life cycle.
* **entity**—Something about which someone wants to store data; typically a person, a place, a thing, a concept, or an event. See also *attribute*.
* **entity instance**—A term used in ER modeling to refer to a specific table row. Also known as an *entity occurrence*.
* **entity occurrence**—See *entity instance*.
* **entity relationship diagram (ERD)**—A diagram that depicts an entity relationship model’s entities, attributes, and relations.
* **entity relationship (ER) model (ERM)**—A data model developed by P. Chen in 1975. It describes relationships (1:1, 1:M, and M:N) among entities at the conceptual level with the help of ER diagrams.
* **entity set**—In a relational model, refers to a grouping of related entities.
* **extended relational data model (ERDM)**—Sometimes referred to as the enhanced entity relationship model; the result of adding more semantic constructs (entity supertypes, entity subtypes, and entity clustering) to the original entity relationship (ER) model.
* **external model**—The application programmer’s view of the data environment. Given its business-unit focus, an external model works with a data subset of the global database schema.
* **external schema**—The specific representation of an external view, that is, the end user’s view of the data environment.
* **hardware independence**—Means that a model does not depend on the hardware used in the implementation of the model. Therefore, changes in the hardware will have no effect on the database design at the conceptual level.
* **hierarchical model**—No longer a major player in the current database market; important to know, however, because the basic concepts and characteristics form the basis for subsequent database development. This model is based on an “upside-down” tree structure in which each record is called a segment. The top record is the root segment. Each segment has a 1:M relationship to the segment directly below it.
* **inheritance**—In the object-oriented data model, the ability of an object to inherit the data structure and methods of the classes above it in the class hierarchy. See also *class hierarchy*.
* **internal model**—In database modeling, refers to a level of data abstraction that adapts the conceptual model to a specific DBMS model for implementation.
* **internal schema**—Depicts a specific representation of an internal model, using the database constructs supported by the chosen database. (The internal model is the representation of a database as “seen” by the DBMS. In other words, the internal model requires a designer to match the conceptual model’s characteristics and constraints to those of the selected implementation model.)
* **logical design**—A stage in the design phase that matches the conceptual design to the requirements of the selected DBMS and is, therefore, software-dependent. It is used to translate the conceptual design into the internal model for a selected database management system, such as DB2, SQL Server, Oracle, IMS, Informix, Access, and Ingress.
* **logical independence**—A condition that exists when the internal model can be changed without affecting the conceptual model. (The internal model is hardware independent because it is unaffected by the choice of computer on which the software is installed. Therefore, a change in storage devices or even a change in operating systems will not affect the internal model.)
* **many-to-many (M:N or \*..\*) relationship**—One of three types of relationships (associations among two or more entities) in which one occurrence of an entity is associated with many occurrences of a related entity and one occurrence of the related entity is associated with many occurrences of the first entity.
* **method**—In the object-oriented data model, a named set of instructions to perform an action. Methods represent real-world actions. Methods are invoked through messages.
* **network model**—A data model standard created by the CODASYL Data Base Task Group in the late 1960s. It represented data as a collection of record types and relationships as predefined sets with an owner record type and a member record type in a 1:M relationship.
* **object**—An abstract representation of a real-world entity that has a unique identity, embedded properties, and the ability to interact with other objects and with itself.
* **object-oriented data model (OODM)**—A data model whose basic modeling structure is an object.
* **object-oriented database management system (OODBMS)**—Data management software used to manage data found within an object-oriented database model.
* **object/relational database management system (O/RDBMS)**—A DBMS based on the extended relational model (ERDM). The ERDM, championed by many relational database researchers, constitutes the relational model’s response to the OODM. This model includes many of the object-oriented model’s best features within an inherently simpler relational database structural environment.
* **one-to-many (1:M or 1..\*) relationship**—One of three types of relationships (associations among two or more entities) that are used by data models. In a 1:M relationship, one entity instance is associated with many instances of the related entity.
* **one-to-one (1:1 or 1..1) relationship**—One of three types of relationships (associations among two or more entities) that are used by data models. In a 1:1 relationship, one entity instance is associated with only one instance of the related entity.
* **physical independence**—A condition that exists when the physical model can be changed without affecting the internal model.
* **physical model**—A model in which the physical characteristics (location, path, and format) are described for the data. Both hardware- and software-dependent. See also *physical design*.
* **relation**—In a relational database model, an entity set. Relations are implemented as tables. Relations (tables) are related to each other through the sharing of a common entity characteristic (value in a column).
* **relational database management system (RDBMS)**—A collection of programs that manages a relational database. The RDBMS software translates a user’s logical requests (queries) into commands that physically locate and retrieve the requested data. A good RDBMS also creates and maintains a data dictionary (system catalog) to help provide data security, data integrity, concurrent access, easy access, and system administration to the data in the database through a query language (SQL) and application programs.
* **relational diagram**—A graphical representation of a relational database’s entities, the attributes within those entities, and the relationships among those entities.
* **relational model**—Developed by E. F. Codd (of IBM) in 1970, it represents a major breakthrough for users and designers because of its conceptual simplicity. The relational model, based on mathematical set theory, represents data as independent relations. Each relation (table) is conceptually represented as a matrix of intersecting rows and columns. The relations are related to each other through the sharing of common entity characteristics (values in columns).
* **relationship**—An association between entities.
* **schema**— A logical grouping of database objects (tables, indexes, views, queries, etc.) that are related to each other. Usually, a schema belongs to a single user or application.
* **semantic data model**—The first of a series of data models that more closely represented the real world, modeling both data and their relationships in a single structure known as an object. The SDM, published in 1981, was developed by M. Hammer and D. McLeod.
* **software independence**—A property of any model or application that does not depend on the software used to implement it.
* **subschema**—In the network model, the portion of the database “seen” by the application programs that produce the desired information from the data contained within the database.
* **table**—A (conceptual) matrix composed of intersecting rows (entities) and columns (attributes) that represents an entity set in the relational model. Also called a *relation*.

**The Relational Database Model**

* **associative entity**—See *composite entity*.
* **attribute domain**—See *domain*.
* **bridge entity**—See *composite entity*.
* **candidate key**—See *key*.
* **composite entity**—An entity designed to transform an M:N relationship into two 1:M relationships. The composite entity’s primary key comprises at least the primary keys of the entities that it connects. Also known as a *bridge entity*. See also *linking table*.
* **composite key**—A multiple-attribute key.
* **data dictionary**—A DBMS component that stores metadata—data about data. Thus, the data dictionary contains the data definition as well as its characteristics

and relationships. A data dictionary may also include data that are external to the DBMS. Also known as an *information resource dictionary*. See also *active data dictionary*, *metadata*, and *passive data dictionary*.

* **determination**—The role of a key. In the context of a database table, the statement “A determines B” indicates that knowing the value of attribute A means that the value of attribute B can be looked up (determined).
* **domain**—In data modeling, refers to the construct used to organize and describe an attribute’s set of possible values.
* **entity integrity**—The property of a relational table that guarantees that each entity has a unique value in a primary key and that there are no null values in the primary key.
* **flags**—Special codes implemented by designers to trigger a required response, to alert end users to specified conditions, or to encode values. Flags may be used to prevent nulls by bringing attention to the absence of a value in a table.
* **foreign key (FK)**—See *key*.
* **full functional dependence**—A condition in which an attribute is functionally dependent on a composite key but not on any subset of that composite key.
* **functional dependence**—Within a relation R, an attribute B is functionally dependent on an attribute A if and only if a given value of the attribute A determines exactly one value of the attribute B. The relationship “B is dependent on A” is equivalent to “A determines B” and is written as A B.
* **homonyms**—Indicates the use of the same name to label different attributes; generally should be avoided. Some relational software automatically checks for homonyms and either alerts the user to their existence or automatically makes the appropriate adjustments. See also *synonym*.
* **index**—An ordered array composed of index key values and row ID values (pointers). Indexes are generally used to speed up and facilitate data retrieval. Also known as an index key.
* **index key**—See *index*.
* **key**—An entity identifier based on the concept of functional dependence; may be classified as follows: *Superkey*: An attribute (or combination of attributes) that uniquely identifies each entity in a table. *Candidate key*: A minimal attribute that is itself a superkey. *Primary key (PK)*: A candidate key selected as a unique entity identifier. *Secondary key*: A key that is used strictly for data retrieval purposes. For example, a customer is not likely to know his or her customer number (primary key), but the combination of last name, first name, middle initial, and telephone number is likely to make a match to the appropriate table row. *Foreign key:* An attribute (or combination of attributes) in one table whose values must match the primary key in another table or whose values must be null.
* **key attribute**—The attribute(s) that form(s) a primary key. See also *prime attribute*.
* **null**—In SQL, refers to the absence of an attribute value. Note: A null is not a blank.
* **primary key (PK)**—In the relational model, an identifier composed of one or more attributes that uniquely identifies a row. See also *key*.
* **referential integrity**—A condition by which a dependent table’s foreign key must have either a null entry or a matching entry in the related table. Even though an attribute may not have a corresponding attribute, it is impossible to have an invalid entry.
* **relational schema**—The description of the organization of a relational database as seen by the database administrator.
* **secondary key**—A key that is used strictly for data retrieval purposes. For example, a customer is not likely to know his or her customer number (primary key), but the combination of last name, first name, middle initial, and telephone number is likely to make a match to the appropriate table row. See also *key*.
* **superkey**—See *key*.
* **synonym**—The use of different names to identify the same object, such as an entity, an attribute, or a relationship; should generally be avoided. See also *homonym*.
* **system catalog**—A detailed system data dictionary that describes all objects in a database.
* **unique index**—An index in which the index key can have only one pointer value (row) associated with it.

**Entity Relationship (ER) Modeling**

* **binary relationship**—An ER term used to describe an association (relationship) between two entities. Example: PROFESSOR teaches COURSE.
* **cardinality**—Assigns a specific value to connectivity. Expresses the range (minimum to maximum) of allowed entity occurrences associated with a single occurrence of the related entity.
* **composite attribute**—An attribute that can be further subdivided to yield additional attributes. For example, a phone number (615-898-2368) may be divided into an area code (615), an exchange number (898), and a four-digit code (2368). Compare to *simple attribute*.
* **composite identifier**—In ER modeling, a key composed of more than one attribute.
* **connectivity**—Describes the classification of the relationship between entities. Classifications include 1:1, 1:M, and M:N.
* **derived attribute**—An attribute that does not physically exist within the entity and is derived via an algorithm. Example: Age = current date – birth date.
* **existence-dependent**—A property of an entity whose existence depends on one or more other entities. In an existence-dependent environment, the existence-independent table must be created and loaded first because the existence-dependent key cannot reference a table that does not yet exist.
* **existence-independent**—An entity that can exist apart from one or more related entities. It must be created first when referencing an existence-dependent table to it.
* **identifiers**—The ERM uses identifiers to uniquely identify each entity instance. In the relational model, such identifiers are mapped to primary keys in tables.
* **identifying relationship**—A relationship that exists when the related entities are existence-dependent. Also called a *strong relationship* or *strong identifying relationship* because the dependent entity’s primary key contains the primary key of the parent entity.
* **iterative process**—A process based on repetition of steps and procedures.
* **mandatory participation**—A term used to describe a relationship in which one entity occurrence must have a corresponding occurrence in another entity. Example: EMPLOYEE works in DIVISION. (A person cannot be an employee if he or she is not assigned to a company’s division.)
* **multivalued attribute**—An attribute that can have many values for a single entity occurrence. For example, an EMP\_DEGREE attribute might store the string “BBA, MBA, PHD” to indicate three different degrees held.
* **non-identifying relationship**—A relationship that occurs when the primary key of the dependent (many side) entity does not contain the primary key of the related parent entity. Also known as a *weak* relationship.
* **optional attribute**—In ER modeling, refers to an attribute that does not require a value, therefore it can be left empty.
* **optional participation**—In ER modeling, refers to a condition where one entity occurrence does not require a corresponding entity occurrence in a particular relationship.
* **participants**—An ER term used to label the entities that participate in a relationship. Example: PROFESSOR teaches CLASS. (The teaches relationship is based on the participants PROFESSOR and CLASS.)
* **recursive relationship**—A relationship that is found within a single entity type. For example, an EMPLOYEE is married to an EMPLOYEE or a PART is a component of another PART.
* **relationship degree**—Indicates the number of entities or participants associated with a relationship. A relationship degree can be unary, binary, ternary, or higher level.
* **required attribute**—In ER modeling, refers to an attribute that must have a value. In other words, it cannot be left empty.
* **simple attribute**—An attribute that cannot be subdivided into meaningful components. Compare to *composite attribute*.
* **single-valued attribute**—An attribute that can have only one value.
* **strong relationship**—When two entities are existence-dependent; from a database design perspective, this exists whenever the primary key of the related entity contains the primary key of the parent entity.
* **ternary relationship**—An ER term used to describe an association (relationship) between three entities. Example: A CONTRIBUTOR contributes money to a FUND from which a RECIPIENT receives money.
* **unary relationship**—An ER term used to describe an association within an entity. Example: A COURSE is a prerequisite to another COURSE.
* **weak entity**—An entity that displays existence dependence and inherits the primary key of its parent entity. Example: A DEPENDENT requires the existence of an EMPLOYEE.
* **weak relationship**—A relationship that exists when the PK of the related entity does not contain a PK component of the parent entity. Also known as a *non-identifying relationship*.

**Normalization of Database Tables**

* **atomic attribute**—An attribute that cannot be further subdivided to produce meaningful components. For example, a person’s last name attribute cannot be meaningfully subdivided into other name components; therefore, the last name attribute is atomic.
* **atomicity**—A property of transactions that states that all parts of a transaction must be treated as a single logical unit of work in which all operations must be completed (committed) to produce a consistent database.
* **Boyce-Codd normal form (BCNF)**— A special form of third normal form (3NF) in which every determinant is a candidate key. A table that is in BCNF must be in 3NF. See also *determinant*.
* **denormalization**—A process by which a table is changed from a higher level normal form to a lower level normal form. Usually done to increase processing speed. Potentially yields data anomalies.
* **dependency diagram**—A representation of all data dependencies (primary key, partial, or transitive) within a table.
* **determinant**—Any attribute in a specific row whose value directly determines other values in that row. See also *Boyce-Codd normal form (BCNF)*.
* **first normal form (1NF)**—The first stage in the normalization process. It describes a relation depicted in tabular format, with no repeating groups and with a primary key identified. All nonkey attributes in the relation are dependent on the primary key.
* **granularity**—Refers to the level of detail represented by the values stored in a table’s row. Data stored at their lowest level of granularity are said to be atomic data.
* **key attribute**—The attribute(s) that form(s) a primary key. See also *prime attribute*.
* **nonkey attribute**—See *nonprime attribute*.
* **nonprime attribute**—An attribute that is not part of a key.
* **normalization**—A process that assigns attributes to entities in such a way that data redundancies are reduced or eliminated.
* **partial dependency**—In normalization, a condition in which an attribute is dependent on only a portion (subset) of the primary key.
* **prime attribute**—A key attribute, that is, an attribute that is part of a key or is the whole key. See also *key attribute*.
* **repeating group**—In a relation, a characteristic describing a group of multiple entries of the same type that exist for a single key attribute occurrence. For example, a car can have multiple colors (top, interior, bottom, trim, and so on).
* **second normal form (2NF)**—The second stage in the normalization process in which a relation is in 1NF and there are no partial dependencies (dependencies in only part of the primary key).
* **surrogate key**—A system-assigned primary key, generally numeric and auto-incremented.
* **third normal form (3NF)**—A table is in 3NF when it is in 2NF and no nonkey attribute is functionally dependent on another nonkey attribute; that is, it cannot include transitive dependencies.
* **transitive dependency**—A condition in which an attribute is dependent on another attribute that is not part of the primary key.