ISTE-608 Test Out Written Exam and Practical Exam Study Guide

Written Exam:

The written exam will be in the format of multiple choice, true/false, matching, short answer, and applied questions (ex. transposing muti-entity E-R diagrams into relations). In order to be fully prepared for the written exam you should be able to:

- identify three key differences between databases and file systems.
- identify three disadvantages of file systems. (Application program dependency, data duplication, data separation/isolation, long development times, higher maintenance requirements/costs)
- describe the meaning of the term "Database", including at least four of the following keywords: Shared, Self-describing, Organized, Logically Related, Persistent, Collection of Data.
- identify three types of data that can be stored in a database. (text, numbers, date/time, graphics/images, sound/video, programs, other objects)
- define and contrast the terms User Data, Metadata, Index, and Application Metadata.
- identify the correct definition of a database management system.
- describe the purpose of each DBMS subsystem (Design Tools, Run-time, and Engine).
- provide two disadvantages of a database as discussed in class. (DBAs/Specialized personnel, Installation/Management costs, Conversion costs, backup/recover, political issues/organizational conflict)
- recall the summary of steps to the database development lifecycle.
- identify DBMS independent steps from DBMS dependent steps.
- state what data modeling is any why it is important.
- define and contrast the terms Relation, Attribute, and Tuples, as they pertain to the relational model.
- describe and identify the difference between an Entity Class and an Entity Instance.
- recognize the following, when given an E-R diagram, and state what the meaning is: Entity, Attribute (simple vs composite; single-valued versus multi-valued; stored vs derived, and identifier)
- compare and contrast the Chen notation and the Crow's foot notation.

- describe what an identifier is and its usage.
- appropriately represent simple identifiers and composite identifiers, using either Chen or Crow's feet notation.
- describe a composite identifier.
- list the steps involved in creating a database.
- transpose an E-R diagram, with Chen or Crow's Feet notation, of a single entity into a relation.
- describe what a relation is and what rows and columns of a relation represent.
- describe at least five characteristics of a relation.
- determine whether or not a given relation conforms to all of the characteristics of a relation.
- describe a primary key in terms of its purpose and characteristics.
- list the steps involved in creating a database.
- list and describe the two types of SQL statements.
- classify SQL statements (CREATE, DROP, INSERT, and SELECT) as appropriate type of SQL statement (DDL/DML).
- know the required clauses of a SELECT statement.
- describe what a NULL value is.
- classify SQL statements (ALTER, UPDATE, and DELETE) as appropriate type of SQL statement (DDL/DML).
- state that normalization is part of the representational level of the development process.
- determine functional dependencies between attributes for a given relation.
- use proper notation for functional dependencies.
- describe a candidate key.
- describe a primary key.
- determine what candidate keys exist when given a relation and a list of functional dependencies.
- select an appropriate primary key based on any candidate keys present, when given a relation and a list of functional dependencies.
- describe what an anomaly is.
- identify the different types of anomalies: Insertion, Modification, and Deletion.

- identify the determinant when given a functional dependency.
- identify composite keys: primary and candidate.
- describe second normal form and its violation.
- identify any partial dependencies when given a relation and functional dependencies.
- correctly fix second normal form violations, including correctly denoting a foreign key and composing a reference statement, when given a relation and functional dependencies.
- describe a foreign key.
- describe what referential integrity is and why it is important.
- describe third normal form and its violation.
- identify any transitive dependencies, when given a relation and functional dependencies.
- correctly fix third normal form violations, including correctly denoting a foreign key and composing a reference statement, when given a relation and functional dependencies that cause transitive dependencies.
- describe Boyce-Codd normal form and its violation.
- correctly fix BCNF violations, including correctly denoting a foreign key and composing a reference statement, when given a relation and functional dependencies that include a determinant that is not a candidate key.
- modify the original and all resulting relations through BCNF, including correct foreign keys and reference statements, when given a relation and functional dependencies.
- describe the correlation between the quality of a data model and impact on the normalization process.
- determine the highest normal form that a relation is in when given a relation and a list of functional dependencies.
- describe the connection between the quality of an E-R diagram and the implications on the normalization process.
- describe the difference between a relationship class and a relationship instance.
- determine the degree of the relationship, when given an E-R diagram.
- distinguish binary relationships from recursive relationships, when given an E-R diagram.
- interpret the maximum cardinalities for a relationship and provide the cardinality ratio, when given an E-R diagram.

- interpret the minimum cardinalities for a relationship to determine if an entity is required to participate in a relationship or not, when given an E-R diagram.
- fully explain a relationship, using non-technical terms, that include the implications of the minimum and maximum cardinalities from both perspectives, when given an E-R diagram.
- compare and contrast “HAS-A” relationships to “IS-A” relationships.
- describe the difference between a supertype and a subtype.
- identify any supertypes and subtypes, when given an E-R diagram.
- distinguish a strong entity from a weak entity, when given an E-R diagram.
- determine the two main reasons supertype/subtype relationships are necessary.
- describe the implications of having a relationship at the supertype-level versus the subtype-level.
- determine whether the completeness constraint for the relationship is total or partial specialization and the implications of both on the supertype/subtype relationship, when given an E-R diagram.
- determine the disjointedness constraint and whether the overlap or disjoint rule is in effect and the implications on the supertype/subtype relationship, when given an E-R diagram.
- determine an appropriate discriminator based on if the disjoint or overlap rule is in effect for a supertype/subtype relationship, when given an E-R diagram.
- describe, in non-technical terms, the full supertype/subtype relationship including completeness and disjointedness constraints, when given an E-R diagram.
- correctly transpose the diagram into a set of relations in relational schema notation, when given an E-R diagram that includes strong and weak entities, with binary, recursive, and supertype/subtype relationships.
- describe foreign key placement for a 1:1, 1:N (N:1), M:N, or supertype/subtype relationship.
- explain the difference between the participation constraints (total participation and partial participation).
- determine whether the weak entity is ID-Dependent or Non-ID Dependent, when given an E-R diagram that includes a HAS-A relationship between a strong and a weak entity, along with functional dependencies.
- transpose an E-R diagram with proper placement of the foreign key based on whether the relationships are identifying or not, when given an E-R diagram that distinguishes between identifying and non-identifying relationships.

- correctly interpret and transpose a M:N relationship with attributes of the relationship, when given an E-R diagram.
- correctly interpret and transpose an associative entity, when given an E-R diagram.
- explain why a M:N relationship with attributes for the relationship is equivalent to two – 1:N relationships with an associative entity.
- transpose the E-R diagram and normalize all relations through BCNF, when given an E-R diagram and functional dependencies.
- state benefits of lookup tables.
- describe what referential integrity is, why it is important, and how it is implemented in a database.
- state that a transaction is a logical unit of work.
- state that a transaction can be composed of one or more statements.
- explain the purpose of COMMIT.
- classify COMMIT as a Transaction Control Language (TCL) statement.
- explain the purpose of ROLLBACK.
- classify ROLLBACK as a Transaction Control Language (TCL) statement.
- list the properties of a tranaction contained in the ACID acronym.
- describe the Atomicity property of a transaction.
- describe the Consistency property of a transaction.
- describe the Isolation property of a transaction.
- describe the Durability property of a transaction.
- compose a theoretical relational algebra statement that would satisfy a specified query, when given a set of relations. Operations include SELECTION, PROJECTION, UNION, DIFFERENCE, PRODUCT, INTERSECTION, and Joins (EQUIJOIN, NATURAL, LEFT OUTER, RIGHT OUTER, and FULL OUTER).
- derive the resulting relation, when given a set of relations and a theoretical relational algebra statement.
- state the conditions for union compatibility.
- determine if two relations are union compatible.
- specify the relational algebra operations that are commutative.

- describe the operations involved in performing a relational algebra join.
- describe how a relational algebra operation is accomplished in SQL.
- recognize the difference in structure between a SQL 89 formatted join and a SQL 92 formatted join.
- describe the structure of a subquery, including how results are passed from a nested query to the outer query.
- explain the difference between the WHERE clause and the HAVING clause of a SELECT statement.
- state which clauses of an SELECT statement are required and which are optional.
- state the proper order for clauses in a SELECT statement, including FROM, GROUP BY, HAVING, SELECT, ORDER BY, and WHERE.
- describe the structure of a highly abstract table.
- state the advantages and disadvantages of highly abstract tables.

Practical Exam:

If you receive an 80% or higher on the written exam, you will be allowed to take the practical exam. The practical will involve the use of MySQL in order to accomplish tasks provided for you to complete. You will only have access to the MySQL help system during the practical exam. In order to be fully prepared for the final practical, you should be able to accomplish the following through your knowledge of MySQL and SQL:

- create a database.
- open a database.
- write a CREATE TABLE statement with attributes of appropriate datatypes (CHAR, VARCHAR, INT, and DATE) and a constraint for the primary key.
- compose a DROP DATABASE statement.
- compose a DROP TABLE statement.
- execute commands to view metadata (SHOW DATABASES; SHOW TABLES; and DESCRIBE).
- create a script containing one or more DDL or DML statement.
- execute a script.
- write a SELECT statement that will view all attributes and all records from a single table.
- write a SELECT statement that contains only a subset of available attributes.

- apply a column alias in the SELECT clause of a SELECT statement.
- form an appropriate WHERE clause, of a SELECT statement, that includes logical operators (NOT, AND, and OR).
- find records where a specified attribute has one or more NULL values.
- insert a record that has one or more NULL values.
- recognize data that should be stored as SMALLINT and DECIMAL, when given an attribute description.
- use SIGNED and UNSIGNED appropriately for integer data types, when given an attribute description.
- apply NOT NULL constraints appropriately in a CREATE TABLE statement, when given attribute descriptions.
- set DEFAULT values appropriately in a CREATE TABLE statement, when given attribute descriptions.
- appropriately use the following relational operators in SELECT statements, as needed: >, >=, <, <=, =, <>, !=, IN, LIKE.
- form an appropriate WHERE clause, of a SELECT statement, that includes logical operator, BETWEEN.
- apply LIKE using wildcards ('_' and '%') to form a search pattern that meets specifications.
- perform basic calculations using an attribute in a SELECT statement.
- write a CREATE TABLE statement with attributes of appropriate datatypes (CHAR, VARCHAR, INT, and DATE), when given attribute specifications, and a constraint for the primary key.
- compose an UPDATE statement that will meet provided specifications.
- compose a DELETE statement that will meet provided specifications.
- form an ALTER TABLE statement that meets specifications provided.
- ALTER TABLE to DROP|ADD a PRIMARY KEY.
- ALTER TABLE to DROP|ADD a column meeting provided specifications.
- ALTER TABLE to SET|DROP a DEFAULT value specified.
- ALTER TABLE to MODIFY a COLUMN to meet provided specifications.
- write CREATE TABLE statements that will implement a provided schema (E-R diagram, transposed relations, and attribute specifications) with attributes of appropriate data types (CHAR, VARCHAR, INT (signed vs unsigned), SMALLINT (signed vs unsigned), TINYINT (signed vs

unsigned), DECIMAL, BOOLEAN, DATE, and ENUM), and correct use of NOT NULL constraints. Each statement will also include a correct primary key constraint and necessary foreign key constraint(s).

- determine if a NOT NULL constraint is needed when implementing a specific foreign key, when given an E-R diagram and transposed relations.
- organize a series of CREATE TABLE and INSERT statements so that the execution order doesn't cause any errors due to sequencing.
- define an attribute as AUTO_INCREMENT, when appropriate.
- compose an INSERT statement that will result in AUTO_INCREMENT being used to determine an attribute's value, when AUTO_INCREMENT was used to define the respective attribute.
- implement appropriate discriminating attribute(s) in a CREATE TABLE statement, when given an E-R diagram that includes a supertype/subtype relationship.
- determine and implement an appropriate ON UPDATE condition for a foreign key constraint, when given an E-R diagram and transposed relations.
- determine and implement an appropriate ON DELETE condition for a foreign key constraint, when given an E-R diagram and transposed relations.
- INSERT a record into a table that includes results from a SELECT statement, when given an E-R diagram and transposed relations.
- successfully implement a transaction that is composed of more than one statement, given a list of tasks and MySQL.
- compose a correct ALTER TABLE statement that will add an appropriate foreign key constraint.
- compose a correct ALTER TABLE statement that will drop an existing foreign key constraint.
- compose the SQL equivalent statement to produce the resulting relation, when given a relational schema implemented in SQL and a theoretical relational algebra statement. Statement operations can include SELECTION, PROJECTION, UNION, DIFFERENCE, PRODUCT, INTERSECTION, and Joins (EQUIJOIN, NATURAL, LEFT OUTER, RIGHT OUTER, and FULL OUTER).
- appropriately utilize DISTINCT when composing queries of relational algebra operations in SQL.
- write a CREATE TABLE statement that includes UNIQUE constraints for any candidate key(s) that weren't selected as the primary key.
- write a SELECT statement that includes table aliases.
- compose a SELECT statement that property joins multiple tables together, using either SQL 89 (inner join) or SQL 92 (inner and outer joins) format.

- compose a subquery (an outer query that includes one or more nested queries) that will accomplish a task that requires data to be passed from one or more tables to another.
- write a SELECT statement that appropriately utilizes any of the following single-row, character functions: LOWER(), UPPER(), LENGTH(), INSTR(), SUBSTR(), and CONCAT().
- write a SELECT statement that appropriately utilizes any of the following single-row, numeric functions: ROUND(), TRUNCATE(), and MOD().
- write a SELECT statement that appropriately utilizes any of the following multi-row functions: COUNT(), AVG(), SUM(), MIN(), MAX().
- correctly incorporate a GROUP BY clause, when using multi-row function(s) and also displaying other attributes as part of the SELECT clause of a SELECT statement.
- correctly incorporate a HAVING clause within a SELECT statement, when needing to eliminate records from a result set based on the results of an aggregate function.
- correctly incorporate an ORDER BY clause within a SELECT statement, when needing to sort a result set by specified criteria.