**Chapter 1**

Characteristics of Databases

* Help people track things of interest in databases
* Data stored in tables with reach row and column like a spread sheet
* Each row in a table is an occurrence or “instance”
* Database stores data and relationships

Naming Conventions

* Table Names are all written in capital letters (ex. STUDENT)
* Column names are written normally without space (ex. StudentName)

Databases Create Information

* Data records facts and figures
* Information is knowledge from data

Components of a Database System

* User 🡪 Database Application 🡨SQL 🡪 DBMS 🡨 🡪 Database
  + Applications are the computer programs you use to work with
  + DBMS creates, processes, and administers databases
    - Create database
    - Create tables
    - Create supporting structures (indexes)
    - Modify (insert, update, delete) database data
    - Enforce rules
  + SQL is the standard that is used in DBMS

Microsoft Access

* Low-end product that attempts to hide much of the technology from the user
* BDMS plus application generator

Three Types of Database Design.

* From existing data
  + Analyze spreadsheets and extra data and design it using normalization
* New systems development
  + Create data model from application requirements transform data into database design
* Database redesign
  + Migrate databases to newer databases. Reverse engineer and design new using normalization

The Relational Database Model

* All current major DBMS products are based on it
* E.F. Codd (in case this is a bonus question like last year)
* Based on mathematics called relational algebra

**Chapter 2**

SQL DDL, DML, and SQL/PSM

* Data Definition Language (DDR)
  + Used for creating tables, relationships, and other structures. Defines things
* Data manipulation language (DML)
  + Used for queries and data modification. It manipulates data
* SQL/Persistent Stored Modules (SQL/PSM)
  + Adds procedural programming capabilities
    - Variables
    - Control-of-flow statements

SQL TCL/DCL

* Transaction control language (TCL)
  + Used to mark transaction boundaries and control transaction behaviour
* Data control language (DCL)
  + Used to grant or revoke database permissions from users and groups

DISTINCT

* Selects only unique values
* Let’s you hand in your assignments in lab without being sent back down again for the third or fourth time <\_<

Wildcards

* SQL Server, MySQL, just about every database
  + \_ = exactly one character
  + % = any set of characters
* Except access, fuck you access
  + ? = exactly one character
  + \* = any set of characters

SQL Built-in Functions

* COUNT
* SUM
* AVG
* MIN
* MAX
* RTRIM
  + Returns a character string after truncating all trailing blanks

Subqueries

* A SELECT statement within brackets is a subquery.
* Can be nested the same way if statements can be nested in JAVA

Joins

* Outer Join
  + Only rows specified are shown
  + All rows are shown even when there is no matching

**Chapter 3**

Entity

* Identifiable thing that users want to track
  + Customers, computers, sales, etc.

Relation

* Two-dimensional table that has rows relating to columns creating some sort of relationship between the two values
* Has to have only one entry per cell

Alternative Terminology

* Table = Relation = File
* Column = Attribute = Field
* Row = Tuple (Isn’t this a DBZ character?) = Record

Functional Dependency

* Occurs when the value of one (or more) attribute(s) determines the value of the second
  + Usually the ID, or primary key proper
* Attribute on the left side of the functional dependency is called the determinant

Composite Determinants

* Determinant of a functional dependency that consists of the more than one attribute
  + If A 🡪 (B, C) then A🡪 B and A 🡪 C

What makes determinant values unique?

* Determinants are unique if a relation determines every column in the relation

Candidate and Primary and Surrogate and Foreign keys

* Candidate key
  + Determines all the other columns in a relation (determinant)
* Primary key
  + Candidate key selected as primary means of identifying rows
* Surrogate keys
  + Artificial column added to the relation that is only there to be a primary key
    - DBMS supplied, often auto-incremented, artificial value, often ID
* Foreign key
  + Primary key of another relation to form a link between two tables
* Should be noted that keys are often underlined in databases

Referential Integrity Constraint

* Statement that limits the values of the foreign key to those already existing as primary key values

Modification Anomalies

* Deletion anomaly
* Insertion anomaly
* Update anomaly

Normal Forms

* First Normal Form
  + Table that qualifies as a relation
* Second Normal Form
  + All of its non-key attributes are dependent on all of the primary keys
* Third Normal Form
  + No determinants except for the primary key
* Boyce-Codd Normal Form(BCNF)
  + Every determinant is a candidate key
    - Make as many tables as needed
    - Make determinant primary key
    - Leave copy of determinant as foreign key in the original relation
* Fourth Normal Form
  + Multivalued dependencies are put into their own relation(table)
* Project-Join Fifth Normal Form
  + Table is split apart but not put back together
  + Every non-trivial join dependency is implied by the candidate keys

**Chapter Four**

Assessing Table Structure

* Count rows and examine columns
* Example data values and determine
  + Multivalued dependencies
  + Functional dependencies
  + Candidate keys
  + Primary keys
  + Foreign keys

Type of Database

* Updatable normally has BCNF as its ideal

Normalization Advantages and Disadvantages

* Advantages
  + Eliminate modification anomalies
  + Reduce duplication of data
* Disadvantages
  + More complicated SQL queries
  + More work on the DBMS slowing the system down

Common Design Problems

* Multi-value, multi-column problem
  + Multiple values are stored in one column
  + To fix it use separate table to store multiple values
* Inconsistent values
  + Different users, or different data sources use slightly different forms of the same data
    - Different descriptions, spelling inconsistencies, etc
    - To fix use referential integrity check to check keys
    - Use SQL GROUP BY clause
* Missing values
  + No value has been provided
  + NULL values are ambiguous
  + Possible that the database is asking questions that only works on certain demographics
    - Such as asking childbirth questions to a male user
* General-purpose remarks column
  + Often contains important data however, people really suck at describing what it is that they are trying to say so it is a maze of garbage answers to figure out what it is that they mean.

**Chapter Five**

The Data Model

* Data model is a plan or blueprint for a database design
* It is generalized and abstract

E-R Model

* Entity-Relationship Model is a set of concepts and graphical symbols to create a conceptual schema
  + Entity Class
    - Collection of entities of a given type
    - Ex. CUSTOMER
  + Entity Instance
    - Occurrence of a particular entity
    - Ex. CustomerID 1234
  + Attributes
    - All instances of a given entity class (column values)
      * CustomerName, CustomerNumber, Phone, Email, etc
  + Identifiers
    - Primary Key
  + Composite identifier
    - Composite Key

Degree of the Relationship

* Binary relationship
  + Two entities
  + Ex. EMPLOYEE SKILL
* Ternary relationship
  + Three entities
  + CLIENT ARCHITECT PROJECT has a CLIENT and an ARCHITECT

Cardinality

* Cardinality means “count”
* Maximum cardinality
  + Maximum numbers of entity instances that can participate
  + Minimum number of entity instances that must participate
    - If zero it is considered optional and gets a little o when cardinality is drawn
* Parent and Child Entities
  + One-to-Many relationship
    - The parent is on the one side and it can have many children but children can only have one parent
    - Ex. EMPLOYEE can have a number of computers but a COMPUTER can be assigned to only one EMPLOYEE
  + HAS-A Relationship
    - EMPLOYEE has one or more COMPUTERs or COMPUTER has one assigned EMPLOYEE

ID-Dependent Entities

* It is a child entity whose identifier includes the identifier of another entity (parent)
* Ex. An APARTMENT is the child of a BUILDING

Weak Entities

* Weak entity whose existence depends on another entity
  + The identifier of the parent does not appear in the identifier of the child

Subtype Entities

* Special case of supertype entity
  + STUDENT supertype has the subtype UNDERGRADUATE or GRADUATE
* Supertype contains all common attributes
* Exclusive
  + One supertype relates to, at most, one subtype
  + This is marked with an X in a circle ( X )
  + Ex. STUDENT is exclusively an UNDERGRADUATE or a GRADUATE, cannot be both

Subtypes: IS-A Relationships

* Relationships connecting supertypes and subtypes is called IS-A relationships

Strong Entity Patterns

* The primary key association is apparent
  + Ie. COMPANY has CompanyName that optionally can have many DEPARTMENTS with the primary key DepartmentName

ID-Dependent Relationships: The Association Pattern

* A many to many relationship that has the two primary keys of the parents be the primary keys of the child
  + Ex. PART’s primary key PartNumber and COMPANY primary key COMPANY name has child QUOTATION with PartNumber and CompanyName as its foreign keys

Archetype/Instance Pattern

* PAINTING: PRINT
* CLASS:SECTION
* HOUSE\_MODEL:HOUSE

Recursive Relationships

* Occurs when an entity has a relationship with itself
* For example in a race cars would have a relation with other instances of the car entity

**Chapter 6**

Steps for doing so

* Create a table for each entity
  + Specify primary and candidate keys
  + Specify properties for each column
    - Null or notNull
    - Data types
    - Constraints
    - Default
* Create relationships by placing foreign keys
  + Keep track of their relationships and cardinality
  + Strong or weak relationships
  + Keep in mind any ID-dependent (super and subentities)
* Specify logic for enforcing minimum cardinality
  + O-O
    - Parent optional child optional
  + M-O
    - Parent mandatory and child optional
    - Every child must have a parent
    - Referential integrity constraints are properly defined
    - Foreign key is NOT NULL
  + O –M
    - Parent optional child mandatory
    - Triggers and application codes will need to be written
  + M-M
    - Mandatory parent and child
    - Worst case scenario as it is a very tricky process to update or delete any row on either side
  + Cascading updates and deletes
    - What happens to the parent will carry over to the child for any to all values
    - Mostly for weak entities

Application Programming: Triggers

* Trigger is a stored program that is executed by the BDMS whenever a specific event occurs
* Used to enforce specific minimum cardinality enforcement

**Chapter 7**

Alter Table Statement

* Alter Table changes table structure, properties, or constraints after it has been created
  + Adding constraints
    - Or dropping them
  + Updating
  + Cascade
  + Delete
    - DROP
  + Truncate
    - TRUNCATE TABLE
      * Removes data only

Views

* Virtual table that is constructed from other tables or views
* No data of its own, obtains data from tables or other views

Embedding SQL in Program Code

* SQL cursors are used to select one row at a time from pseudo-files
* Can have cursors via triggers and stored procedures
  + DECLARE SQL Cursor CURSRO FOR (SELECT \* FROM CUSTOMER);
    - OPEN SQLCursor:
      * MOVE SQLCursorer to first row
        + WHILE (SQLCursor not past last row)

LOOP

{SQL action statements go here)

REPEAT LOOP UNTIL DONE;

* + - CLOSE SQLCursor;

Stored Procedures

* Program that is stored within the database and compiled when used
* Can receive input parameters and they can return results
* Provide greater security as stored procedures are always stored on the database server
* Decreased network traffic
* SQL can be optimized by DBMS compiler