Relational DBMS

* Data versus Information
  + Data is collected but not process -> unprocessed data
  + Information is data that has been processed -> processed data
  + Data
    - Consists of raw facts
    - Stored in the tables
  + Information
    - Interpretation of these facts
    - Manipulated and presented as information
* Database: a collection of information managed by DBMS software
* Typically, the information has a structure, and the database reflects this structure
* Database Management System (DBMS): a software system for manipulating databases
* DBMS supports
  + Logical views (Schema)
  + Physical views (Access methods)
  + Data Definition Language
  + Data Manipulation Language
  + Utilities
    - Transaction management
    - Concurrency control
    - Data integrity
    - Security
    - Disaster recovery
* Database Tables
  + A single database will usually have many tables
  + The ONLY place data is stored in a database is in it’s tables
  + Each table will hold information about only ONE “thing”
  + Each field in the table will further describe this one “thing”
  + Each record in the table will define on *instance* of the “thing”
* Storing Data in Databases
  + Field/Attributes
  + Record/Row
  + Data Item
* Early Databases
  + A close up of a map

    Description automatically generatedEvery program required code to be written to directly map file and record layouts, starting points, and chains
* The Relational Model
  + Adds a layer of separation between the application code and the files, records, fields, pointer, chains
  + Revolutionized database software
  + Data is stored in TABLES (“relations”)
  + Tables are related to each other by keys
  + Data is queried using a language (SQL) based on relational algebra
  + “self-describing”: contains within itself all information about itself, also known as metadata
* ACID
  + The DBMS software must maintain ACID compliance
  + A(tomicity): a transaction is all or nothing
  + C(onsistency): a transaction must maintain data consistency
    - All databases have consistency constraints
  + I(solation): multiple users/transactions cannot step on each other
  + D(urability): once a transaction is finished, the results cannot be lost
* Characteristics of Relational Databases
  + The purpose of a database is to help people track things of interest to them
  + Data is stroed in tables, which has rows and columns like a spreadsheet
  + A database stores BOTH data (in tables) AND relationships (between tables)
  + A database may have multiple tables, where each table stores data about a different thing
    - A STUDENT table and a CUSTOMER table
  + Each row in a table stores data about one occurrence of the thing of interest
    - One students data, one customers data
  + A database stores metadata: data about itself stored within itself
    - In the DBMS catalog, there is a table containing one row for every table in the database
* Summary
  + Databases are ubiquitous
  + DBMS software is extremely powerful and complex
  + Oracle and MS SQL Server are expensive, yet widely used
  + MySQL is free, and widely used
  + Relational DBMS software holds 90% of market share but
  + NoSQL and Hadoop are booming so folks can handle “Big Data”
  + Database design requires specialized skills
  + Database is a collection of interrelated stored data: think of an electronic filing system
  + DBMS is a software system/application for manipulating databases
  + DBMS supports logical and physical views, DDL, and DML
  + A single database normally contains many tables
    - Each table holds information about only one “thing”
    - Each filed in the table further describes this one “thing”
    - Each record in the table defines one instance of the “thing”

Relational Concepts

* A database model describes
  + Where/How does the DBMS physically store the data?
    - In Tables (relations)
  + How does the DBMS allow programs to do operation against the data?
    - Using SQL (an implementation of relational algebra)
  + How does the DBMS maintain constraints to protect the data?
    - Constraint definitions in SQL, OS level files permissions
* Constraints
  + When the DBMS is running, it will not allow any program to directly access any DBMS-managed files (through OS permissions)
* Non-Relational Models
  + Other Models: semi-structured, unstructured
    - Example: XML
    - Example: key, value pairs
    - Blocks of unstructured text
    - Object-Oriented Database
* Why the Relational Model?
  + The Relational Model has prevailed for over 30 years
  + Solid, Reliable Operating Principles
  + Can grow very large (multi-terabytes)
  + Efficient to manage, maintain
  + Available talent in the marketplace
  + Fabulous DBMS software alternatives
  + Design approach is simple, limited yet versatile
  + Data operations (via SQL) are simple, limited yet versatile
  + SQL is very powerful: few lines of code -> big results
* Relational Basics
  + We store data about a person, place, thing or event
  + Something we store data about is an entity
  + Data is stored in a two-dimensional table (relation), rows & columns
  + Each row represents one occurrence of an entity
  + Each row is unique, has a unique identifier
  + We store characteristics, or “attributes” about each occurrence of the entity
  + Each column of the table holds one attribute
  + A row or tuple
* The schema
  + A description of the relation in this format
    - RelationName(column-name1, column-name2, column-name3)
  + Or in this format with domain
    - RelationName(column-name1:datatype, column-name2:datatype, column-name3:datatype)
* Relational Basics
  + The relational model gives us a single way to represent data: as a two-dimensional table called a relation
    - Rows: each represent a movie
    - Columns: each represent a property of movies
  + Attributes: The columns of a relation are named by attributes (title, year, length, and genre)
  + Schemas: The name of a relation and the set of attributes for a relation is called the schema for that relation
    - Movies(title, year, length, genre)
  + Tuples: The rows of a relation, other than the header row containing the attribute names, are called tuples
    - (Gone With the Wind, 1939, 231, drama)
  + Domains: includes domain, or data type, for each attribute in a relation schema
    - Movies(Title: string, year: integer, length: integer, genre: string)
* Keys
  + A key is a combination of one or more columns that is used to identify rows in a relation
  + A composite key is a key that consists of two or more columns. Need NOT be contiguous. Also known as concatenated key
  + A candidate key is a key that determines all of the other columns in a relation
    - Identify a database record without referring to any other data
  + A primary key is a candidate key selected as the primary means of identifying rows in a relation
    - There is only one primary key per relation
    - The primary key may be a composite key
    - The ideal primary key is short, numeric, and never changes
  + The primary key is a unique key in the table that you choose that best uniquely identifies a record in the table
  + Atomic key: a primary key consisting of a single attribute
  + Natural key: a natural key is a column or set of columns that already exist in the table
  + A surrogate key is an artificial column added to a relation to serve as a primary key
    - E.g. personal table, since it’s possible for two people born on the same data to have the same name, need surrogate key
      * DBMS supplied
      * Short, numeric, and never changes: an ideal primary key
        + Integers in an automatically incrementing field, or as GUIDs (Globally Unique Identifier)
      * Has artificial values that are meaningless to users
        + Property\_id, GUID
      * Normally hidden from in forms and reports
      * Unlike primary keys, not all tables need surrogate keys, e.g. US states
  + A foreign key is the primary key of one relation that is placed into another relation to form a link between the relations
    - A foreign key can be a single column or a composite key
    - A FOREIGN KEY is a key used to link two tables together
    - Child table: containing foreign key
    - Referenced or parent table: containing the primary key
  + A foreign key constraint
    - Also called “Referential Integrity” or “RI”
    - In certain examples, DBMS can prevent from adding a PRODUCT whose SUPPLIED is not found in the database
    - Likewise, the constraint can prevent from deleting a supplier if that supplier has products in the database

Relational Algebra

* The Relational Algebra
  + Two groups of SQL
    - DDL – creating tables, columns
    - DML – querying and modifying data
  + Where did DML SQL come from?
* SQL Steps
  + Write a query in SQL, submit it
  + The DBMS engine “parses” the SQL, checks syntax
  + Builds an execution plan (based on Relational Algebra)
  + Executes the SQL
  + Returns an Answer Set (a table)
  + Stores the SQL in cache (Oracle Shared Pool)
* Preparing the query for execution
  + Its like algebra
  + Variables and Operations
    - The “variables” are relations
    - The operations are
      * Union
      * Intersection
      * Difference
      * Selectin
      * Projection
      * Cartesian Product
      * Join
* Set Operations
  + Some operations are “Set” operations
    - Think of each table as entire “set” of data
    - Consider two tables: R and S
      * Same attributes, same domains
  + Set Operations in Relational Algebra
    - R <union> S
      * All elements in R or S or both
    - R <intersection> S
      * The set of elements in R and S
    - R <difference> S
      * The set of elements in R but not in S
* Cartesian Product
  + Produce a new relation that is the combination of every tuple in on table combined with every tuple in the other
* Natural Join
  + Produce a new relation that combines matching tuples on a common attribute

Functional Dependency

* Database Design Steps
  + Collect all the information you can about the organization’s data (i.e. requirements)
    - What do they call their entities and attributes?
    - Data types and lengths
    - How and when do they use each piece of data?
    - Where does it come from? Where does it go to?
    - Who uses it for what purpose?
    - How many occurrences/instances of each entity do they deal with?
  + How?
    - Conduct interview
    - Review documentation
    - Review current systems and processes
      * User data entry screens
      * Paper forms
      * Paper reports
      * Computer reports
      * Process flows
  + Once data requirements are collected and documented, organize the data
    - Identify all Functional Dependencies
    - Put the data into 3rd Normal Form
    - Create a logical data model
* Functional Dependency (definition Ullman text, page 68)
  + If the values of one or more attributes (A1, A2, A3, etc.) of a relation functionally determine the value of another attribute (B) of that relation, then we can say that (B) is functionally dependent on (A1, A2, A3, etc.)
  + If I know the value of an attribute (or set of attribute), I can determine the value of another attribute
  + Example
    - FD = title + year -> length + genre + studioName
      * If 2 tuples have the same values in title + year, those 2 tuples will have the same values in length, genre, and studioName (TRUE)
    - FD = title + year -> starName
      * If 2 tuples have the same values in title + year, those 2 tuples will have the same values in starName (FALSE)
  + A formal definition of a Key
    - A set of attributes (A1, A2, A3, etc.) is a KEY for a relation if that set of attributes functionally determines all the other attributes of the relation
    - No subset of those attributes (A1, A2, A3, etc.) functionally determines all the other attributes of the relation
  + Combining and Splitting Rule
    - In a relation R(A, B, C, D)
      * If A+B -> C, and A+B -> D, then A+B -> C, D
  + Example
    - A relation of people in the U.S.
      * Name, SSN, address, city, state, ZIP, area code, 7-digit phone number
    - FD is a relationship of one attribute or field in a record to another
      * E.g. one field defines the other. SSN defines name
      * IF we know SSN, we can find name so name is functionally dependent on SSN. SSN -> Name
  + Transitive Rule
    - In a relation R(A, B, C)
      * If A -> B, and B -> C, then A -> C
  + Augmentation Rules
    - In a relation R(A, B, C)
      * If A -> C, then A + B -> C + B
  + Some FD Rules
    - If I know a true FD for a relation, I can deduce other true FD’s for that relation
  + Why do we care about all this?
    - If I am designing a database for an organization, I must consider all their data. I must organize that data into relations.
    - I need to understand Functional Dependencies so I can determine keys for every relation.
    - I need keys so that I can normalize the data
    - I must normalize the data to design the database properly

Normalization

* Database Design
  + Two approaches (techniques) to Database Design
  + Data Modeling
    - A conceptual picture of your database
    - Drawn with a computer based tool
  + Normalization
    - A method of cleaning up and organizing your data
    - Structures the data to fit the RELATIONAL MODEL
    - A set of rules you apply to the data to clean up the data
  + Best Approach
    - Combine the two
    - Construct a data model
    - Normalize the data as we Model
  + Iterative Process
    - Model some, normalize some, model some more, normalize some more
* Normalization
  + Process to eliminate data redundancy
  + Ensure data association/relationship
  + Easy in data maintenance
  + Increase design flexibility
  + There can be many normal forms but we will consider only first three
  + Others are considered too restrictive and generally not used in practice
  + Upon normalizing your design, you generally end up with more tables than you started with
  + In most practical applications, normalization achieves its best in 3rd normal form
* Rules of Normalization
  + A structure, defined, detailed process
  + Prepares data to make sure it complies with the rules of the RELATIONAL MODEL
  + Often seen by students as a CONFUSING process
  + Concepts are somewhat academic and theoretical
* Objective of Normalization
  + To arrange the data into a series of clearly defined, well-organized RELATIONS
  + Each with a primary key
  + All attributes are functionally dependent on the primary key
  + Each with required foreign keys referencing other relations
* Normalization: follow the Rules of “ONE”
  + One entity = One table
  + One occurrence of that entity = One row
  + One attribute = One column
  + All columns in the row describe one occurrence of that entity
  + All columns are functionally dependent on one primary key
* Rules of “ONE” example
  + Image a system used by a university’s Registrar’s office
  + We keep data about STUDENTS (entity)
  + For a student (one occurrence of the entity) we store FirstName, LastName, MiddleInitial, PhoneNumber, Address (attributes)
  + We identify a student by their StudentID (primary key)
  + Entity = STUDENT
  + Attributes = (StudentID, FirstName, LastName, MiddleInitial, PhoneNumber, Address)
  + One entity, one primary key, one student contained in one row, all columns are functionally dependent on the one primary key
* How do we achieve this?
  + Key analysis using functional dependencies
  + Identify and correct anomalies
  + Decomposition of a single table into multiple tables (end up with more tables)
* Anomalies
  + Redundancy: storing a piece of data more than once
  + Update/Delete errors
  + Decomposing this relation into two removes the anomalies
* Normalization Step-by-Step
  + First Normal Form
    - Remove any multi-valued cells and/or any rows requiring a specific sequence
  + Second Normal Form
    - For entities with concentrated keys, make sure that all attributes are dependent on the full key
  + Third Normal Form
    - Make sure that no attributes are dependent on any other non-key attributes
  + Fourth Normal Form
    - Eliminate multi-value dependencies
* First Normal Form
  + Remove Repeating Groups of attributes
  + Put repeating attributes into a new relation
    - Create a key for the new relation
      * Probably a concatenated key
      * The key of the original relation and an identifier for each occurrence of the Repeating Group of attributes
* Second Normal Form
  + Every non-key attribute is functionally dependent on the entire primary key
  + Only meaningful when there is a concatenated key
  + Create a new relation
    - Create a key for the new relation
  + Migrate only the dependent columns to the new relation
* Third Normal Form
  + No non-key attributes are dependent on any other non-key attributes
  + Create a new relation
    - Create a key for the new relation
  + Migrate only the dependent columns for the new relation
* Fourth Normal Form
  + A multi-valued dependency is this
    - Multiple independent attributes are dependent on the same determinant
  + Example: One customer may have multiple “contacts” of different “types”
  + 4NF is rarely used
* Summary
  + Normalization is a process to reduce data redundancy
  + Divides larger tables to smaller tables and links them using relationships
  + Easy to maintain data and increase design flexibility
  + Properly normalized design shouldn’t include derived fields and keep only the most meaningful data
* Transitive Dependency
  + When changing a non-key column, any other non-key columns may change
    - Need to eliminate transitive dependency in 3NF
* Examples
  + As you are designing a database, you must define your clients’ Data Requirements
  + During interviews, Clients often refer you to their existing reports, spreadsheets, screens, etc.

Data Modeling

* A technique used in Database Design
* Like an architect’s drawings of a house being built
  + The architect draws the model of the database
  + The technicians (DBAs for example) then build the database according to the architect’s design
* Different Techniques
  + iDEF1X
  + Crow’s Feet
  + Chen
* Data Modeling
  + IEM Symbols
    - An optional end of a relationship is depicted by a small circle (o)
    - The one of end is depicted by a small vertical line (---)
    - The many end of a relationship is depicted by a “crows” foot (--<-)
  + Types of DB Design
    - Logical database design
      * Implementation Independent
      * Establishes the entities, fields and relationships between fields
    - Physical database design
      * Refine the logical design in terms of the constraints and characteristics of DBMS
  + Purpose of Design Types
    - The main reason for having two separate stages in the design process is so that the designer can focus on two separate issues
    - In the logical step, the focus is on the business process and requirements
    - In the physical step, the focus is on the technical requirements
  + Correct Database design is concerned with
    - Avoidance of data redundancy
    - Application performance
    - Data independence
    - Data security
    - Application development
  + A screenshot of a social media post

    Description automatically generatedProcess
    - Gather all data requirements
    - Normalize the data to 3rd normal form
    - From 3NF schemas, draw the data model (ERD)
    - Review the ERD with your customers verifying it against the requirements
    - Upon signoff, generate DDL
  + Cardinality
    - How many of THESE are related to how many of THESE
    - Typically: zero, one, or many
    - On both ends of the relationship
  + Optionality
    - Is the relationship mandatory (one or more) or optional (zero)
  + Relationships
    - Mandatory relationship is where there must be at least one matching record in each entity
    - Optional relationship is where there may or may not be a matching record in each entity
  + A picture containing screenshot

    Description automatically generatedEx. Customer and Order Table
    - Customer is the parent; customers can exist whether they buy a product or not
    - Order, is the child record, Fk migrates from the Customer. It is mandatory that Order has a CustomerID (dependent entity)
    - An order cannot exist without a customer
  + Prepare for Data Modeling
    - Once data requirements are clear, then
    - Decide the Business Area you are modeling
      * Failure to restrict your model to a single business area will make the data modeling process much more complex
    - Organize all the data items in ENTITIES and ATTRIBUTES
    - Determine an attribute that can serve as a PRIMARY KEY for each entity
    - If no appropriate candidate key exist, then plan to create a SURROGATE key
  + Begin the Process of Data Modeling
    - Draw a rectangle to represent each ENTITY
    - Write the NAME of the entity above the rectangle (entity names should be singular)
    - Draw a RELATIONSHIP line between each related entity
    - Draw CARDINALITY and OPTIONALITY symbols on both ends of each relationship line
    - Resolve any many-to-many relationships by creating an ASSOCIATION (“child”) entity between the two “parent” entities
  + 1. Drawing a horizontal line across each entity rectangle and enter the name of the primary key attribute above the line
    - As you define the primary keys and group the attributes within entities, you will NORMALIZE the data
  + 2. The list all the remaining attributes within the rectangle below the line
  + 3. Identify any foreign key attributes with an “FK”
  + 4. Walk through the model with your customers
  + Reading the Relationships
    - An order belongs to one customer
    - A customer can place zero, one or many orders
    - An order may contain one or many OrderDetails
    - A product may be purchased on zero, one or many OrderDetails
    - Read relationship clockwise
  + More Model Constructs
    - Square Edge = “Independent Entities”
    - Round Edge = “Dependent Entities” (weak)
    - Dashed Line = a dependent entity where the parent’s key does not migrate to primary
    - Solid Line = a dependent entity where the parent’s key migrates to primary
  + ID Dependent Entities: An ID-dependent entity is an entity whose identifier (key) includes the identifier of another entity
  + Subtype/Supertype
    - The supertype contains all common attributes, while the subtypes contain specific attributes
    - The supertype may have a discriminator attribute which indicates the subtype
  + Relationships
    - Relationship are associations between entities
      * Generally a verb connecting two or more entities
        + An employee has an Address
        + A manager is an Employee
        + A manager manages a Department
    - One-to-One
      * Each record in entity A is related the only ONE record in entity B. Each record in entity B is related the only ONE record in entity A.
    - One-to-Many
      * Each record in entity A is related to zero, one or more record in entity B. Each record in entity B is related to only ONE record in entity A.
    - Many-to-Many
      * Each record in entity A is related to zero, one or more records in entity B. Each record in entity B is related to zero, one or more records in entity A.
  + Resolving a Many-to-Many relationship
    - Can exist in the logical design but not in the physical design
    - To resolve the relationship
      * Create an intersection entity
      * Create a Unique ID for the new entity which consists of a combination of the unique IDs of the two original entities
  + Association Entity
    - Created to resolved a Many-to-Many relationship
    - A dependent Child between two independent parents
    - Relationship to parents is mandatory
    - A composite primary key, a portion inherited from each parent
    - Rounded corners indicate a dependency
  + Data Modeling Software
    - Drawing a model
      * Entities, attributes, relationships, cardinality, optionality
      * Physical characteristics like data type & length
      * Constraints like PK, FK, Nullable
    - Different tools support different capabilities
      * Some can only DRAW
      * Some capture the intelligence behind the drawing
      * Some can generate SQL DDL
    - Different tools have different licensing
      * Free = less capable (mostly)
    - Other concerns
      * Windows, Mac, Linux
      * Local install or cloud
      * Drawing technique: Crows Foot (IE) vs. iDEF1X vs. UML
      * Data Modeling vs full blow Database Administration Console vs SQL Editor
  + Step-By-Step for Data Modeling
    - Draw each TABLE
    - Double-click to bring up COLUMN dialog
    - Add each column, data type, constraints
    - Create relationships
      * Click on Child Key, then on Parent Key
      * This duplicates keys in the model
      * Modify cardinality/optionality, if needed
    - Export DDL
      * Database: forward engineer (creates DDL) and reverse engineer (created DataModel)
      * Or, File: Import/Export