11/30:

Goals for course:

Create Entity/Relation models form application descriptions

Build relational models.

Identify redundancies in design and remove them using normalization techniques.

Create databases in an RDBMS and enforce data integrity constraints using SQL.

Write database queries using SQL.

Objectives:

* Knowledge of various database modeling techniques
* Understand the difference between logical and physical modeling
* Understand and implement both data definition and data management queries
* Understand the use of relational algebra and writing queries
* Understand entity relationship models, design, and functional dependencies
* Apply the Boyce-Codd Normalization and Multivalued dependencies
* Differentiate between database constraints and database triggers
* Understand indexes and its applications
* Understand views in relational databases, view modification.

Class policy

* Don’t seek perfection, just get an answer
* Show up early, which should not be too hard, he likes to start as early as possible
* Missed deadline = -100%, unless genuine excuse = -20%
* A, B, C are reasonable. D go to him privately. F don’t come back.

What will I learn?

Course Project

* Project will be defined by group by mid of Jan 2023
* At the end of quarter each group turns in report of project
* 2-3 people per group

Approached to storing data

* File processing
  + Older approach to storing data
  + Each user defines and implements the files needed for a specific software application is free to name data elements
  + Here each application is free to name data elements independently
  + Ancient ‘Boss’ is an example of this.
* Database Approach
  + Self-describing nature of a database system using meta data (data about data)
  + Insulation between programs and data and data abstraction using data modeling
  + Support of multiple views of the data using queries and views (allows you to abstract data about certain things, restricts certain things for certain users.)
  + Sharing data and multi-user transaction processing using access control and concurrency control (concurrency control updates the system so everyone is on the same page at the same time)

What is a database

* Database is commonly used to refer to any of the following
  + Personal address book
  + Colletion of word documents
  + Collection of excel spreadsheets
  + Very large flat file on which you run some statistical analysis functions
  + Data collected, maintained, and used in airline reservation

What is a system

* System- a group of interacting or interrelating elements that act according to a set of rules to form a unified whole
* Systems model:
  + System comprises multiple views
  + Man-made systems may have views such as concept, analysis, design, implementation, deployment, structure, behavior, input data, and output data views

DBMS

* A general-purpose software system that facilitates the process of defining, constructing, manipulating, and searching databases among various users and applications

12/2

History of Databases

File processing, is the older approach to storing data. Each user denies and implements the files needed for a specific software application as part of programming the application. Here each application is free to name data elements independently.

The main characteristics of a database approach versus a file-processing approach are the following:

* Self-describing nature of a database system – using Meta data
* Insulation between programs and data, and data abstraction – using data modeling
* Support of multiple views of the data = using queries and views.
* Sharing of data and multi-user transaction processing – using access control and concurrency control.

A database management system (DBMS) is a collection of programs that enable users to create and maintain a database

IDS – integrated data store. Developed internally at GE in the early 1960s

GE sold their computing division to Honeywell in 1969.

One of the first DBMSs: Network data model. Tuple-at-a-time queries. (Row or tuple)

COBASYL – COBOL people got to gather and proposed a standard for how programs will access a database. Lead by Charles Backman

Schema – network model

Information Management System

Early database system developed to keep track of purchase orders for Apollo moon mission. Hierarchical data model. Programmer-defined physical storage format.

Edgar Ted Codd was a mathematician working at IBM Research. He saw developers spending their time rewriting IMS and COVASYL programs every time the database’s schema or layout changed.

Database abstraction to avoid this maintenance: Store data base in simple data structures. Access data through high=level languages. Physical storage left up to implementation.

Early implementations of relational DBMS:

System R – IBM research

INGRES – Sonebarker Berkley

Oracle – larry Ellison

Object-oriented databases

Aboid relational-object impedance mismatch by tightly coupling objects and databases

Few of these original DBMSs from the 1980s still exist today but many of these technologies exist in other forms (JSON, XML)

1990s

No major advancements in database systems or application workloads

Microsoft creates SQL server

MySQL is written as a replacement of mSQL

Postgres gets SQL support

SQLite is started in early 2000s

Internet boom – everything became publicly available.

All the big players were heavyweight and expensive.

Rise of Data Warehouse

Rise of the special purpose OLAP DBMSs. Distributed / shared – nothing. Relational / SQL. Usually closed-source. Signficant performance benefits from using columnar data storage model.

12/5/2022

* Databases and how they are helpful
  + Academic
    - Active research, constantly changing
  + Programmer
    - Wide array of application involve using or accessing databases
  + Business
    - Many businesses need datbases
  + Student
    - Easier to get hired
* What is the goal
  + Fast and convenient access of information
  + RDMS
    - Relational database management system
  + Data and set of instruction to access / manipulate data
* When to use DBMS
  + Persistent storage of data
    - Data documentation
  + Consistency multiple user support
    - security
  + Centralized control of data
    - Control of redundancy, consistency and integrity
* Database approached
* Focusing on maps
* A model is a means of communication
* Emphasized selected aspects
* Described in some language/standardized notation
* Can be erroneous

12/9/2022

* Homogeneous DDBMS
  + You have the same software across all databases you have.
* Heterogeneous DDBMS
  + You have different software across databases.
* Federated
  + Each site may run different database systems
  + Data access if managed through a sing conceptual schema
    - The use of wrappers to ensure consistency across the environment
    - Each site must adhere to a centralized access policy
    - There may be a global schema.
* Data Modeling: the design process

12/12/2022

Lecture 5: Design: Entities and Constraints

* Only think of tables and tuples. Not data structures
* A key attribute can be composite in nature
  + Vehicle tag/plate – unique to car, includes car, state, number
* An entity type may have more than one key
  + The car entity type may have two keys:
    - VIN
    - Vehicle tag number, license plate number
* Each key is underlined
* In ER diagram, an entity type is displayed in a rectangular box
* Attributes are displayed in ovals
  + Each attribute is connected to its entity type
  + Components of a composite attribute are connected to the over representing the composite attribute
  + Each key attribute in underlined
  + Multivalued attributes displayed in double ovals
* See car example notes
* Constraints
  + Domain of attributes
    - Domain constraints
    - The set of allowed values for each attribute
    - Tech example
      * CWID: 8 digit number
      * First, last name: alpha string
      * DoB: Data
      * Ssn: 9-digit number
      * Passport: String (letter followed by 7 digits)
      * Program: Alpha string
    - Normally databases will require a value, not allowing NULL
    - NULL indicated that the value is “unknown”
  + Key
    - Foreign key constraint
    - K is a subset of R attributes
    - K is a superkey, such that K can sufficiently identify a unique tuple
    - Multiple superkeys and keys can exist.
    - Superkey K is a candidate key if K is minimal
  + Relation
    - It is a tuple. Is shown in table format
    - Programming example can be like a python dictionary
      * ID : (name, deptname, salary)
    - Unique ID gives a name, department name, and salary

12/14/22

Lecture 6: design: keys, foreign keys, relational algebra

We’re in the realm of queues

* Foreign key is when you break a larger table into two
  + Foreign key constrain: Value is on relation must appear in another
    - Referencing relation
    - Referenced relation
    - Ex.: dept\_name in INSTRUCTOR is a foreign key from INSTRUCTOR referencing DEPARTMENT
* Remember, it is a set of tuples, not a list of tuples
* Schema is the same irrespective of order of attributes
  + Attributes can be ordered however we want
* Degree of relation: number of attributes
* Degree of CARINALITY? : number of tuples or entries

QUEUES and RELATIONAL ALGEBRA

* Procedural vs non-procedural, or declarative
* “Pure” languages
  + Relational algebra
  + Tuple relational calculus
  + Bomain relational calculus
* Query input is a table or set of tables
* Query output is a table
* All data in the output table appears in one of the input tables
* Relational algebra supports operations like:
  + SUM
  + AVG
  + MAX
  + MIN
  + UNION
  + INTERSECTION
* Relational Algebra is a formal query languagebased on a set of operations on relations
* Fundamental operators
  + SELECT σ
  + PROJECTπ
  + RENAME ρ
  + CARTESIAN PRODUCT X
  + UNION U
  + SET DIFFERENCE –
  + NATURAL JOIN
  + INTERSECTION
  + ASSIGNMENT =
  + DIVISOIN/QUOTIENT %
  + THETA JOIN θ

12/16/22: looking at relationships

* Joining two relations
  + Let r and s be realtions on schemas Rand S respectively
  + The operation of a joins strives to merge the contents of the two relations into one relation based on attributes that are common to both relations
    - Implicit -> natural join
    - Eplicit -> equi-join
  + This is different from the union operator
* Relationships