Welcome!

Course Outline

Donald Grasse, Executive Educational Instructor, (dgrasse@uchicago.edu)
Pavan Prathuru, Teaching Assistant, (pavanprathuru@uchicago.edu)

Course Outline

This course is designed to teach students the fundamentals of databases and database management. After completing the course, students will be able to:

- Design, implement, and manage relational databases effectively
- Write and optimize SQL queries for data retrieval and analysis
- Understand database administration, ensuring data integrity and optimal performance
- Apply principles in real-world scenarios

Why the Relational Database (November 5)

- What is a database and why do we structure them the way we do?
- An introduction to the relational model, the most widely used model used to organize data within a database

Mathematical and Logical Foundations of DBMS (November 6)

- Effectively interacting with, and querying, databases is ultimately about formal logic, organization, and indexing.
- To understand the method to DBMS, we will study set theory, a branch of mathematics that underpins nearly the rest of the entire course
- **Homework:** Set Theory Problem Set (Due End of Next Day)

Data Modeling (November 7)

- Conceptually, databases can be thought of in many ways, which has stakes for how they are organized
- We will study one of the most important tools for data modeling - the Entity Relationship model - and the rules for structuring a database to make sure it is efficient and easy to use
- **Homework:** Build Your Own E-R Diagram (Due End of Next Day)

Data Querying: Relational Algebra (November 8)

- The heart of database theory - and one of the tools that led to the development of relational database - is relational algebra, a defined algebraic structure based on set theory used for querying databases.
- Irrespective of how you will query databases in the future, relational algebra will be the foundation of your query
- **Homework:** Relational Algebra Problem Set (Due End of Next Day)

Synthesis and Assessment (November 9)

- How do set theory, databases, relational algebra, and data modeling fit together?
- This class will focus on synthesizing these ideas and placing the concepts in conversation with one another
- **Quiz 1: Due End of Day**

Data Querying in SQL (November 12)

- This class will bridge the theory of querying databases with practice
- We will learn the Structured Query Language (SQL) - the most important Database Language for DDL and DML
- **In Class Activity: Producing SQL Queries**

Data Security (November 13)

- Databases are incredibly valuable - sometimes so valuable that other actors who are not authorized access attempt to seize information from them
- This class will focus on key threats to data security, and how database administrators can help protect data

Data Governance (November 14)

- Data has to be managed appropriately during its whole life cycle - from the acquisition stage to the disposal stage
- This class will focus on data governance, an important and distinct concept from data management, including the best practices for administrators
- **Homework: Security and Governance Problem Set** (Due End of Next Day)

Beyond SQL (November 15)

- SQL is the most common - but not the only - tool for working with databases
- This class will focus on what is beyond SQL - including NoSQL and Postrelational databases

Synthesis and Assessment (November 16)

- Putting together the big picture
- **Quiz 2: Due End of Next Day**

Grading

The breakdown of grades will be as follows:

- Homework (60%)
- Quizzes (30%)
- Participation (10%)

Integrity

Each student is expected to turn in their own work. While collaboration is welcomed - and indeed expected - students should not directly copy their answers from either their peers or from online sources, including Artificial Intelligence (AI). Evidence of plagiarism is grounds for a failing grade on the assignment that was misrepresented or produced dishonestly.

Course Philosophy

- My goals for you:

Course Philosophy

- My goals for you:
    - Grasp the core concepts and underpinnings of those concepts to apply them **tomorrow**

Course Philosophy

- My goals for you:
  - Grasp the core concepts and underpinnings of those concepts to apply them **tomorrow**
  - Master the analytical frameworks so you can apply them in real life and as a spring board to branch into other techniques
- My method:

Course Philosophy

- My goals for you:
  - Grasp the core concepts and underpinnings of those concepts to apply them **tomorrow**
  - Master the analytical frameworks so you can apply them in real life and as a spring board to branch into other techniques
- My method:
  - Active learning though scaffolding

Lecture Overview

Today, we are going to:

- Understand where, when, and why database management comes into play

Lecture Overview

Today, we are going to:

- Understand where, when, and why database management comes into play
- Explain, at a high level, how DBMS operates, and its key functions

Lecture Overview

Today, we are going to:

- Understand where, when, and why database management comes into play
- Explain, at a high level, how DBMS operates, and its key functions
- Understand and explain the basis of the relational model

Introduction

- A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data

Introduction

- A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data
- A database is a collection of relevant information

Introduction

- A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data
- A database is a collection of relevant information
- The goal in DBMS is to provide a way to store and retrieve database information in an efficient and convenient way

Introduction

- A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data
- A database is a collection of relevant information
- The goal in DBMS is to provide a way to store and retrieve database information in an efficient and convenient way
- We have to achieve this while also protecting the data from threats and crashes to prevent anomalous results

History of Database Systems

- Before diving in, its important to understand how we got here

History of Database Systems

- Before diving in, its important to understand how we got here
- This helps clarify why things have evolved the way they did, and will give you some insight into how things could change in the future

History of Database Systems

- Information processing has driven computer growth, not the other way around

History of Database Systems

- Information processing has driven computer growth, not the other way around
- Punched carts were an early form of automation to collect census data in the United States

Herman Hollerith

History of Database Systems

- Early 1960s: Tapes and punch cards

History of Database Systems

- Early 1960s: Tapes and punch cards
- Sequencing was essential

History of Database Systems

- Early 1960s: Tapes and punch cards
- Sequencing was essential
- If someone got a raise that would be read into a new tape using the old tape, paying attention to the sequence of entry

History of Database Systems

- Early 1960s: Tapes and punch cards
- Sequencing was essential
- If someone got a raise that would be read into a new tape using the old tape, paying attention to the sequence of entry
- The new tape would become the master tape

History of Database Systems

- Late 1960s: Hard disks

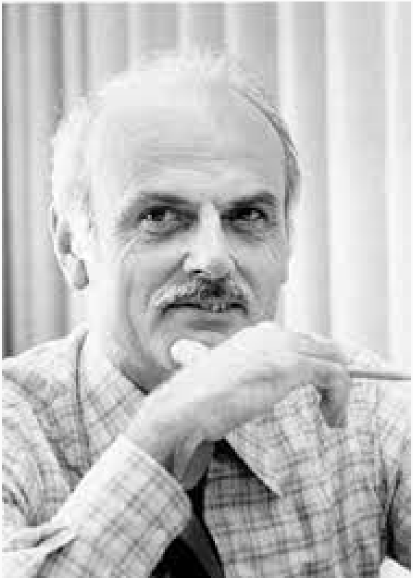
History of Database Systems

- Late 1960s: Hard disks
- Sequencing less important, data could be stored on the hard disk and accessed anywhere

History of Database Systems

- Late 1960s: Hard disks
- Sequencing less important, data could be stored on the hard disk and accessed anywhere
- Lists and trees could be saved, allowing for hierarchical databases

Edward F Codd defines the relational model and algebra



History of Database Systems

- 1980s: Relational databases become competitive

History of Database Systems

- 1980s: Relational databases become competitive
- IBM research developed a way to make relational databases efficient

History of Database Systems

- 1980s: Relational databases become competitive
- IBM research developed a way to make relational databases efficient
- The programmer was free to work at the logical level, with most efficiently related matters already being accounted for by relational model

History of Database Systems

- 1980s: Relational databases become competitive
- IBM research developed a way to make relational databases efficient
- The programmer was free to work at the logical level, with most efficiently related matters already being accounted for by relational model
- Since becoming dominant in the 1980s, the relational model has remained the most important and popular

History of Database Systems

- The World Wide Web led to explosive database growth

History of Database Systems

- The World Wide Web led to explosive database growth
- Database systems were deployed more extensively than ever before

History of Database Systems

- The World Wide Web led to explosive database growth
- Database systems were deployed more extensively than ever before
- Database systems had to support Web interfaces to data, and had to support high transaction-processing rates, as well as 24 $\times$ 7 availability

Database-Systems Applications

- Databases are an essential part of every enterprise today

Database-Systems Applications

- Databases are an essential part of every enterprise today
- Database access became more direct overtime

Database-Systems Applications

- Databases are an essential part of every enterprise today
- Database access became more direct overtime
- User interface hides this, but we interact with a database everyday

10 Minute Activity

- Talk in groups about a database that you interact with/interface with OR that you may/plan to in the future

10 Minute Activity

- Talk in groups about a database that you interact with/interface with OR that you may/plan to in the future
- Talk about how you think it may be organized to ensure security, efficiency, and convenience

Database-Systems Applications

- Enterprise information

Database-Systems Applications

- Enterprise information
  - Sales (customers, products, purchases)

Database-Systems Applications

- Enterprise information
  - Sales (customers, products, purchases)
  - Accounting (payment, receipts, account balances)

Database-Systems Applications

- Enterprise information
  - Sales (customers, products, purchases)
  - Accounting (payment, receipts, account balances)
  - Human resources (payroll, employee demographics, benefits, hours worked)

Database-Systems Applications

- Enterprise information
    - Sales (customers, products, purchases)
    - Accounting (payment, receipts, account balances)
    - Human resources (payroll, employee demographics, benefits, hours worked)
    - Manufacturing (supply chain, inventory in warehouses and stores, orders for items when you run out)

Database-Systems Applications

Financial information

- Banking (customer information, loans, accounts, transactions)

Database-Systems Applications

Financial information

- Banking (customer information, loans, accounts, transactions)
- Credit Card Transactions (individual purchases and monthly statements)

Database-Systems Applications

Financial information

- Banking (customer information, loans, accounts, transactions)
- Credit Card Transactions (individual purchases and monthly statements)
- Finance (stocks, bonds, market data)

Database-Systems Applications

Other organizations and institutions

- Universities (student information, course registration, grades)

Database-Systems Applications

Other organizations and institutions

- Universities (student information, course registration, grades)
- Airlines (reservations, schedules, plane locations)

Database-Systems Applications

Other organizations and institutions

- Universities (student information, course registration, grades)
- Airlines (reservations, schedules, plane locations)
- Telecommunications (calls, monthly bills, customers)

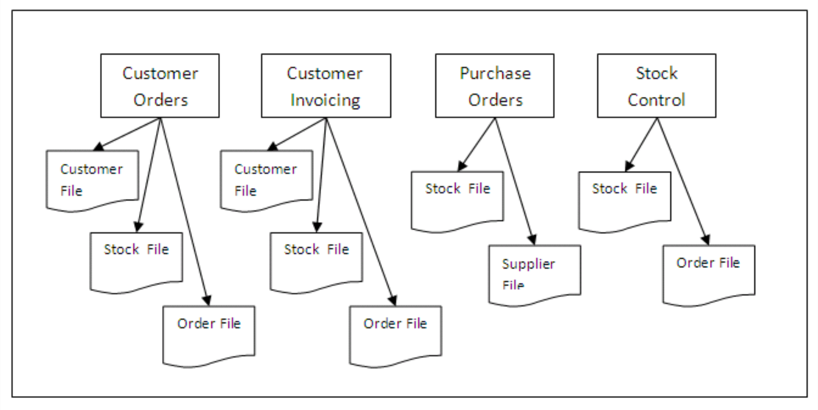
Purpose of Database Systems

- File-processing systems dominated storage of information

Purpose of Database Systems

- File-processing systems dominated storage of information
  - A collection of programs that store and manage files in computer hard-disk

File-processing systems



Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency

Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency
- Access

Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency
- Access
- Data Isolation

Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency
- Access
- Data Isolation
- Integrity problems

Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency
- Access
- Data Isolation
- Integrity problems
- Atomicity

Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency
- Access
- Data Isolation
- Integrity problems
- Atomicity
- Concurrent-access Anomalies

Purpose of Database Systems

The file processing system - despite its intuitive appeal - has some major downsides, which will inspire why we use database systems

- Data redundancy and inconsistency
- Access
- Data Isolation
- Integrity problems
- Atomicity
- Concurrent-access Anomalies
- Security

- To highlight the problems with a file-processing system, we will use the example of a university

- To highlight the problems with a file-processing system, we will use the example of a university
- The University wants to

- To highlight the problems with a file-processing system, we will use the example of a university
- The University wants to
  - Add new students, instructors, courses

- To highlight the problems with a file-processing system, we will use the example of a university
- The University wants to
  - Add new students, instructors, courses
  - Register students

- To highlight the problems with a file-processing system, we will use the example of a university
- The University wants to
    - Add new students, instructors, courses
    - Register students
    - Assign grades, GPAs, generate transcripts

Data redundancy and inconsistency

- Say a student is a double major in Music and Mathematics

Data redundancy and inconsistency

- Say a student is a double major in Music and Mathematics
- They became a double major their second year of college, and they moved after freshman year

Data redundancy and inconsistency

- Say a student is a double major in Music and Mathematics
- They became a double major their second year of college, and they moved after freshman year
- Now the file from the Music Department has a different address than the Mathematics Department

Data redundancy and inconsistency

- Say a student is a double major in Music and Mathematics
- They became a double major their second year of college, and they moved after freshman year
- Now the file from the Music Department has a different address than the Mathematics Department
- Redundancy and inconsistency are costly!

Difficulty Accessing Data

- The University wants to send out mailers to students who have parents that live far away to invite them to a visiting weekend.

Difficulty Accessing Data

- The University wants to send out mailers to students who have parents that live far away to invite them to a visiting weekend.
- The IT department is asked to generate a list of these students, but information is stored based on a file-processing system

Difficulty Accessing Data

- The University wants to send out mailers to students who have parents that live far away to invite them to a visiting weekend.
- The IT department is asked to generate a list of these students, but information is stored based on a file-processing system
- There is no easy way to accomplish this task!

Difficulty Accessing Data

- The IT department decides to write a script to generate a list of all students, their parents addresses, and select only those students who live far away

Difficulty Accessing Data

- The IT department decides to write a script to generate a list of all students, their parents addresses, and select only those students who live far away
- But now, the University realizes it wants to send different mailers based on whether students are upper or lower classman, so they need separate lists based on credit hours

Difficulty Accessing Data

- The IT department decides to write a script to generate a list of all students, their parents addresses, and select only those students who live far away
- But now, the University realizes it wants to send different mailers based on whether students are upper or lower classman, so they need separate lists based on credit hours
- You need a whole new script!

Data Isolation

- Sticking with the last example, you are compiling data from all departments to generate the list of students

Data Isolation

- Sticking with the last example, you are compiling data from all departments to generate the list of students
- The economics department has stored their data in .xls, the psychology department in .xlxs, the public policy department in .csv

Data Isolation

- Sticking with the last example, you are compiling data from all departments to generate the list of students
- The economics department has stored their data in .xls, the psychology department in .xlxs, the public policy department in .csv
- The Music department has stored students as columns but nearly every other department has stored them in rows

Data Isolation

- Sticking with the last example, you are compiling data from all departments to generate the list of students
- The economics department has stored their data in .xls, the psychology department in .xlxs, the public policy department in .csv
- The Music department has stored students as columns but nearly every other department has stored them in rows
- Because data is scattered and stored differently, writing a new application program to retrieve data is difficult

Integrity Problems

- Data values stored in a database must satisfy consistency constraints
    - A consistency constraint is a requirement that a value can only be in the database in a defined way

Integrity Problems

- Data values stored in a database must satisfy consistency constraints
    - A consistency constraint is a requirement that a value can only be in the database in a defined way
    - A student cannot have negative credit hours

Integrity Problems

- Say the university required that budgets for a department never go below zero (no deficit) and appropriate code was programmed into their budgeting sheets by each department to allow this

Integrity Problems

- Say the university required that budgets for a department never go below zero (no deficit) and appropriate code was programmed into their budgeting sheets by each department to allow this
- Now the University needs to save more, so the budget needs to be a surplus (everyone needs 10 dollars left over)

Integrity Problems

- Say the university required that budgets for a department never go below zero (no deficit) and appropriate code was programmed into their budgeting sheets by each department to allow this
- Now the University needs to save more, so the budget needs to be a surplus (everyone needs 10 dollars left over)
- Updating the system for the new consistency constraint is a nightmare

Atomicity

- Atomicity is the idea that an action should occur in its entirety or not at all

Atomicity

- Atomicity is the idea that an action should occur in its entirety or not at all
- Say the History department transfers 500 dollars to the Music departments budget

Atomicity

- Atomicity is the idea that an action should occur in its entirety or not at all
- Say the History department transfers 500 dollars to the Music departments budget
- The power goes out during the execution but everything boots back up fairly quickly

Atomicity

- Atomicity is the idea that an action should occur in its entirety or not at all
- Say the History department transfers 500 dollars to the Music departments budget
- The power goes out during the execution but everything boots back up fairly quickly
- The funds may have left History, but not be credited to Music

Concurrent-access Anomalies

- Two withdrawals at the same time

Concurrent-access Anomalies

- Two withdrawals at the same time
- Two registrations at the same time

Security Problems

- Enforcing constraints is difficult

What do we do?

- Because of all of these problems, there have been great advancements in designing databases that overcome these issues

View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.

View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data

View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data
- The system hides certain details of how the data are stored or maintained

Data Abstraction

- For the system to be usable, it must retrieve data efficiently

Data Abstraction

- For the system to be usable, it must retrieve data efficiently
- The need for efficiency has created complexity, which means things are hidden from the user to prevent them from being overwhelmed

Data Abstraction

- Physical level: how the data are actually stored

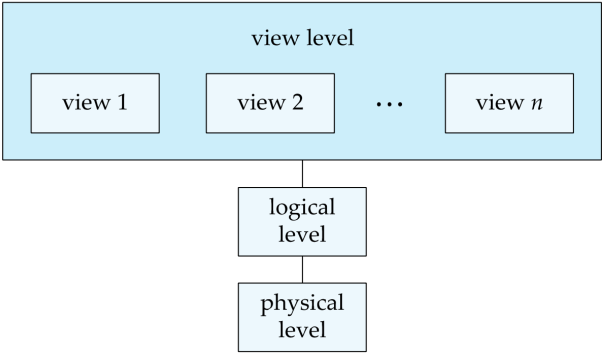
Data Abstraction

- Physical level: how the data are actually stored
- Logical level: what data are stored and what are their relationships

Data Abstraction

- Physical level: how the data are actually stored
- Logical level: what data are stored and what are their relationships
- View level

Data Abstraction



Data Abstraction

CREATE TABLE (ID varchar(5), name varchar(10), dept_name varchar(10) salary numeric(8, 2))

Data Abstraction

- Physical: storage locations

Data Abstraction

- Physical: storage locations
- Logical: type definition

Data Abstraction

- Physical: storage locations
- Logical: type definition
- View: who can see what

Instances and Schemas

- Schema: overall design of the database

Instances and Schemas

- Schema: overall design of the database
- Instance: information stored at a particular moment in time

Instances and Schemas

- Schema: overall design of the database
- Instance: information stored at a particular moment in time
- Schema is the variable declaration

Instances and Schemas

- Schema: overall design of the database
- Instance: information stored at a particular moment in time
- Schema is the variable declaration
- Instance is what we observe at a point in time

Data Models

- Data model: a conceptual tool for describing data, data relationships, data semantics, and consistency constraints

Data Models

- Data model: a conceptual tool for describing data, data relationships, data semantics, and consistency constraints
- Model provides a way of describing the design of a database at the physical, logical, and view levels

Data Models

- Relational model

Data Models

- Relational model
- Entity-Relationship Model

Data Models

- Relational model
- Entity-Relationship Model
- Object-Based Data Model

Data Models

- Relational model
- Entity-Relationship Model
- Object-Based Data Model
- Semistructured Data Model

Relational model

- Uses a collection of tables to represent both data and the relationships among those data

Relational model

- Uses a collection of tables to represent both data and the relationships among those data
- Each column has a unique name, and may correspond to an attribute

Relational model

- Uses a collection of tables to represent both data and the relationships among those data
- Each column has a unique name, and may correspond to an attribute
- Tables are also called relations

Relational model

- Uses a collection of tables to represent both data and the relationships among those data
- Each column has a unique name, and may correspond to an attribute
- Tables are also called relations
- Most widely used

Entity-Relationship Model

- Uses a collection of objects called entities (which are things in the real world distinguishable from other things) and relationships, which connect entities

Entity-Relationship Model

- Uses a collection of objects called entities (which are things in the real world distinguishable from other things) and relationships, which connect entities
- E-R is widely used in database design

Object-Based Data Model

- Object oriented programming has become the dominant software-development methodology

Object-Based Data Model

- Object oriented programming has become the dominant software-development methodology
- This has led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, functions, and object identity.

Semistructured Data Model

- Permits the specification of data where individual items of the same type may have different sets of attributes

Semistructured Data Model

- Permits the specification of data where individual items of the same type may have different sets of attributes
- Extensible Markup Language (XML) is widely used for semistructured data

Database Languages

- A database system provides a data-definition language to specify the database schema and a data-manipulation language to express database queries and update the database

Database Languages

- A database system provides a data-definition language to specify the database schema and a data-manipulation language to express database queries and update the database
- They are distinct, but go together as a way of defining a language (like SQL).

DML (Data Manipulation Language)

DML enables users to access or manipulate data as organized by the appropriate data model

- Retrieval of information stored in the database

DML (Data Manipulation Language)

DML enables users to access or manipulate data as organized by the appropriate data model

- Retrieval of information stored in the database
- Insertion of new information into the database

DML (Data Manipulation Language)

DML enables users to access or manipulate data as organized by the appropriate data model

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information

DML (Data Manipulation Language)

DML enables users to access or manipulate data as organized by the appropriate data model

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information
- Modification of information

DML (Data Manipulation Language)

Two different types based on what users need to specify

- Procedural DML: what is needed and how to get it

DML (Data Manipulation Language)

Two different types based on what users need to specify

- Procedural DML: what is needed and how to get it
- Declarative DML: what is needed without specifying how

DML (Data Manipulation Language)

- A query is a statement requesting the retrieval of information
- We call the part of a DML that that involves information retrieval a query language

DDL (Data Definition Language)

- DDL defines properties of the data and the database schema

DDL (Data Definition Language)

- DDL defines properties of the data and the database schema
- We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language

DDL (Data Definition Language)

- DDL defines properties of the data and the database schema
- We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language
- These statements define the implementation details of the databse schemes, which are usually hidden from the users

DDL (Data Definition Language)

4 big Integrity constraints

- Domain Constraints

DDL (Data Definition Language)

4 big Integrity constraints

- Domain Constraints
- Referential Integrity

DDL (Data Definition Language)

4 big Integrity constraints

- Domain Constraints
- Referential Integrity
- Assertions

DDL (Data Definition Language)

4 big Integrity constraints

- Domain Constraints
- Referential Integrity
- Assertions
- Authorization

Domain Constraints

- A domain of possible values must be associated with every attribute (integer types, date/time types)

Domain Constraints

- A domain of possible values must be associated with every attribute (integer types, date/time types)
- Declaring a domain limits the values that can be taken on

Domain Constraints

- A domain of possible values must be associated with every attribute (integer types, date/time types)
- Declaring a domain limits the values that can be taken on
- ‘Cheesecake’ is not a date

Domain Constraints

- A domain of possible values must be associated with every attribute (integer types, date/time types)
- Declaring a domain limits the values that can be taken on
- ‘Cheesecake’ is not a date
- ‘Catfish’ is not an integer

Domain Constraints

- A domain of possible values must be associated with every attribute (integer types, date/time types)
- Declaring a domain limits the values that can be taken on
- ‘Cheesecake’ is not a date
- ‘Catfish’ is not an integer
- These are tested easily by the system

Referential Integrity

- Ensures a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation

Referential Integrity

- Ensures a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation
- A course relation that includes dept_name should only list dept_name values that exist in a department relation

Referential Integrity

- Ensures a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation
- A course relation that includes dept_name should only list dept_name values that exist in a department relation
- Actions that violate are rejected

Assertions

- A condition that the database must always satisfy

Assertions

- A condition that the database must always satisfy
- (Domain constraints and referential integrity are assertions)

Assertions

- A condition that the database must always satisfy
- (Domain constraints and referential integrity are assertions)
- "Every department must have at least five courses offered per semester"

Assertions

- A condition that the database must always satisfy
- (Domain constraints and referential integrity are assertions)
- "Every department must have at least five courses offered per semester"
- If an assertion is created, the system tests it for validity, and if it is valid, any future modification is allowable only if it does not cause the assertion to be violated

Assertions

- A condition that the database must always satisfy
- (Domain constraints and referential integrity are assertions)
- "Every department must have at least five courses offered per semester"
- If an assertion is created, the system tests it for validity, and if it is valid, any future modification is allowable only if it does not cause the assertion to be violated
- A new department called "String Theory" with 4 courses would not be permitted

Authorization

- Access types permitted to users

Authorization

- Access types permitted to users
- Read authorization

Authorization

- Access types permitted to users
- Read authorization
- insert authorization

Authorization

- Access types permitted to users
- Read authorization
- insert authorization
- update authorization

Authorization

- Access types permitted to users
- Read authorization
- insert authorization
- update authorization
- delete authorization

DDL (Data Definition Language)

- Output of DDL is a data dictionary

DDL (Data Definition Language)

- Output of DDL is a data dictionary
- Includes metadata

DDL and DML Example: SQL

DML

- SQL is a nonprocedural language.

DDL and DML Example: SQL

DML

- SQL is a nonprocedural language.
- A query takes a table(s) as input(s) and always returns a single table

DDL and DML Example: SQL

DML

- SQL is a nonprocedural language.
- A query takes a table(s) as input(s) and always returns a single table
- Example **select** instructor.name
  from instructor
  where instructor.dept_name = 'History'

DDL and DML Example: SQL

DML

- SQL is a nonprocedural language.
- A query takes a table(s) as input(s) and always returns a single table
- Example **select** instructor.name
  from instructor
  where instructor.dept_name = 'History'
- The query returns one table of containing names of instructors from the instructor relation who are within the history department.

DDL and DML Example: SQL

DDL

- SQL allows for the definition of tables along with integrity constraints

DDL and DML Example: SQL

DDL

- SQL allows for the definition of tables along with integrity constraints
- Example **create table** department(
  name char(20),
  budget numeric(12,2));

DDL and DML Example: SQL

DDL

- SQL allows for the definition of tables along with integrity constraints
- Example **create table** department(
  name char(20),
  budget numeric(12,2));
- The DDL statement produces a table where names are 20 characters and budget is a numeric variable that can be 12 digits long (2 to the right of the decimal)

DDL and DML Example: SQL

DDL

- SQL allows for the definition of tables along with integrity constraints
- Example **create table** department(
  name char(20),
  budget numeric(12,2));
- The DDL statement produces a table where names are 20 characters and budget is a numeric variable that can be 12 digits long (2 to the right of the decimal)
- This updates the metadata (the schema of the database being an example)

Database Access from Application Programs

- SQL is not as powerful as a universal Turing machine

Database Access from Application Programs

- SQL is not as powerful as a universal Turing machine
- Turing completeness requires the computation of any computable function

Database Access from Application Programs

- SQL is not as powerful as a universal Turing machine
- Turing completeness requires the computation of any computable function
- SQL instead is designed to query and work with sets from a database (no looping constructs)

Database Access from Application Programs

- SQL is not as powerful as a universal Turing machine
- Turing completeness requires the computation of any computable function
- SQL instead is designed to query and work with sets from a database (no looping constructs)
- SQL cannot communicate over the network

Database Access from Application Programs

- Such computations and actions must be written in a host language, like C, C++, or Java

Database Access from Application Programs

- Such computations and actions must be written in a host language, like C, C++, or Java
- SQL queries are embedded to access data in the data base

Database Access from Application Programs

- Such computations and actions must be written in a host language, like C, C++, or Java
- SQL queries are embedded to access data in the data base
- Application programs are used to interact with the database like this

Database Access from Application Programs

Two ways to do this

- Providing an application program interface (API) that can be used to send DML and DDL statements to the database and retrieve the results

Database Access from Application Programs

Two ways to do this

- Providing an application program interface (API) that can be used to send DML and DDL statements to the database and retrieve the results
- Open Database Connectivity (ODBC) standard for use with the C language is a commonly used application program interface standard. Java Database Connectivity (JDBC) is another for Java

Database Access from Application Programs

Two ways to do this

- Providing an application program interface (API) that can be used to send DML and DDL statements to the database and retrieve the results
- Open Database Connectivity (ODBC) standard for use with the C language is a commonly used application program interface standard. Java Database Connectivity (JDBC) is another for Java
- By extending the host language syntax to embed DML calls within the host language program. Usually, a special character prefaces DML calls, and a preprocessor, called the DML precompiler, converts the DML statements to normal procedure calls in the host language

Database Design (Brief Overview)

We are using database systems in practice because we have...

- We have a large body of information

Database Design (Brief Overview)

We are using database systems in practice because we have...

- We have a large body of information
- This information doesn't exist in isolation (it relates)

Database Design (Brief Overview)

We are using database systems in practice because we have...

- We have a large body of information
- This information doesn't exist in isolation (it relates)
- The issues at hand are complex

Database Design (Brief Overview)

We are using database systems in practice because we have...

- We have a large body of information
- This information doesn't exist in isolation (it relates)
- The issues at hand are complex
- Database design is how we go about putting together the database schema

Database Design (Brief Overview)

- A high-level data model provides the database designer with a conceptual framework to specify requirements to the database user

Database Design (Brief Overview)

- A high-level data model provides the database designer with a conceptual framework to specify requirements to the database user
- From there, one moves into a conceptual-design phase (focus is not on physical storage yet)

Database Design (Brief Overview)

- A high-level data model provides the database designer with a conceptual framework to specify requirements to the database user
- From there, one moves into a conceptual-design phase (focus is not on physical storage yet)
- For the relational model - the focus of this course - this involves what attributes to capture and how to group attributes to form tables

Database Design (Brief Overview)

How would the design process work?

- Interface with database users (talk to university administrators)

Database Design (Brief Overview)

How would the design process work?

- Interface with database users (talk to university administrators)
- Learn data needs (generate transcripts, track who teaches what student)

Database Design (Brief Overview)

How would the design process work?

- Interface with database users (talk to university administrators)
- Learn data needs (generate transcripts, track who teaches what student)
- Here is what it may look like for a university, an example we will return to during the class

Database Design (Brief Overview)

- The University is organized into departments. Each department is identified by a unique name, is located in a building, and has a budget

Database Design (Brief Overview)

- The University is organized into departments. Each department is identified by a unique name, is located in a building, and has a budget
- Each department has a list of course offers. Each course has credits, and may or may not have prerequisites

Database Design (Brief Overview)

- The University is organized into departments. Each department is identified by a unique name, is located in a building, and has a budget
- Each department has a list of course offers. Each course has credits, and may or may not have prerequisites
- Instructors teach courses, are affiliated with a department, and have a salary

Database Design (Brief Overview)

- The University is organized into departments. Each department is identified by a unique name, is located in a building, and has a budget
- Each department has a list of course offers. Each course has credits, and may or may not have prerequisites
- Instructors teach courses, are affiliated with a department, and have a salary
- Students take courses, and have accumulated credit hours

Database Design (Brief Overview)

How would we do this?

- The Entity-Relationship Model

Database Design (Brief Overview)

How would we do this?

- The Entity-Relationship Model
- Normalization

Database Design (Brief Overview)

How would we do this?

- The Entity-Relationship Model
- Normalization
- A lot more in Lecture 3!

Discussion

- What are two disadvantages of database systems?

Discussion

- What are two disadvantages of database systems?
- Suppose you want to build a video site similar to YouTube. Consider disadvantages of keeping data in a file-processing system. Discuss the relevance of each of these points to the storage of actual video data, and to metadata about the video, such as title, the user who uploaded it, tags, and which users viewed it.

Discussion

- What are two disadvantages of database systems?
- Suppose you want to build a video site similar to YouTube. Consider disadvantages of keeping data in a file-processing system. Discuss the relevance of each of these points to the storage of actual video data, and to metadata about the video, such as title, the user who uploaded it, tags, and which users viewed it.
- Describe at least 3 tables that might be used to store information in a social-networking system such as Facebook

Introduction to the Relational Model

Structure of Relational Databases

- Relational database is a collection of tables, each of which is assigned a unique name

Structure of Relational Databases

- Relational database is a collection of tables, each of which is assigned a unique name
- A row in a table represents a relationship among a set of values

Structure of Relational Databases

- Relational database is a collection of tables, each of which is assigned a unique name
- A row in a table represents a relationship among a set of values
- In mathematics, a tuple is a sequence (or list) of values

Structure of Relational Databases

- Relational database is a collection of tables, each of which is assigned a unique name
- A row in a table represents a relationship among a set of values
- In mathematics, a tuple is a sequence (or list) of values
- A relationship between n values is a n-tuple

Structure of Relational Databases

- We call a table a relation

Structure of Relational Databases

- We call a table a relation
- We call a row a tuple

Structure of Relational Databases

- We call a table a relation
- We call a row a tuple
- We call columns attributes

Structure of Relational Databases

- Each attribute in a relation has a **domain**

Structure of Relational Databases

- Each attribute in a relation has a **domain**
- The domain is the set of permitted values

Structure of Relational Databases

- For all relations, the domains of attributes should be atomic

Structure of Relational Databases

- For all relations, the domains of attributes should be atomic
- **Atomic** means the elements of the domain are considered to be indivisible units

Structure of Relational Databases

- For all relations, the domains of attributes should be atomic
- **Atomic** means the elements of the domain are considered to be indivisible units
- Say we have an attribute in an Employee relation that contains phone numbers

Structure of Relational Databases

- For all relations, the domains of attributes should be atomic
- **Atomic** means the elements of the domain are considered to be indivisible units
- Say we have an attribute in an Employee relation that contains phone numbers
- Assume further someone can have more than one phone number

Structure of Relational Databases

- For example, the data may look like this

Structure of Relational Databases

- For example, the data may look like this
Name	Phone Number(s)
John Doe	(123) 456-7890, (987) 654-3210
Jane Smith	(555) 123-4567, (333) 888-9999

Table 1: Employee Information

Structure of Relational Databases

- For example, the data may look like this
-

Name	Phone Number(s)
John Doe	(123) 456-7890, (987) 654-3210
Jane Smith	(555) 123-4567, (333) 888-9999

Table 1: Employee Information

- The elements of the domain phone number have supbarts (number 1 and number 2). Not atomic

Structure of Relational Databases

- Even with one number, we can fail atomicity

Structure of Relational Databases

- Even with one number, we can fail atomicity
-
Name	Phone Number(s)
John Doe	c(123,456,7890)
Jane Smith	c(555,123,4567)

Table 2: Employee Information

Structure of Relational Databases

- Even with one number, we can fail atomicity
-
Name	Phone Number(s)
John Doe	c(123,456,7890)
Jane Smith	c(555,123,4567)

Table 2: Employee Information

- Atomicity matters because it simplifies data manipulation

Structure of Relational Databases

-

Structure of Relational Databases

- For some attributes we will have null values

Structure of Relational Databases

- For some attributes we will have null values
- Null values do not exist (NA), and can cause complications

Structure of Relational Databases

- For some attributes we will have null values
- Null values do not exist (NA), and can cause complications
- We will address these head on in lecture 4

Structure of Relational Databases

- We must have a way to specify how tuples within a given relation are distinguished

Structure of Relational Databases

- We must have a way to specify how tuples within a given relation are distinguished
- No two tuples are allowed to have the exactly all the same values for all of the attributes

Structure of Relational Databases

- We must have a way to specify how tuples within a given relation are distinguished
- No two tuples are allowed to have the exactly all the same values for all of the attributes
- A superkey is a set of one or more attributes that allows us to identify a tuple in the relation uniquely

Structure of Relational Databases

- We must have a way to specify how tuples within a given relation are distinguished
- No two tuples are allowed to have the exactly all the same values for all of the attributes
- A superkey is a set of one or more attributes that allows us to identify a tuple in the relation uniquely
- Otherwise, we have meaningless rows, which is redundancy

Structure of Relational Databases

- Formally, let R denote the set of attributes in the schema of a relation r.
- If we say that a subset K of R is a superkey for r, we are restricting consideration to instances of relations r in which no two distinct tuples have the same values on all attributes K.
- If $t_1 \neq t_2$ and both are in some relation r, then $t_1 K \neq t_2 K$

Structure of Relational Databases

- Superkeys may contain extraneous attributes
- If K is a superkey, so is K plus something else
- What we really want is a superkey where no subset of that superkey is also a superkey
- We call that a candidate key
- We use the term primary key to denote a candidate key that is chosen by the database designer as the means of identifying tuples within the relation

Structure of Relational Databases

- Primary keys should be **unique**

Structure of Relational Databases

- Primary keys should be **unique**
- The value should rarely - if ever **change**

Structure of Relational Databases

- Primary keys should be **unique**
- The value should rarely - if ever **change**
- One relation, r_1 may have an attribute which is a primary key in r_2 - this is a **foreign key** from r_1 to r_2

Structure of Relational Databases

- Primary keys should be **unique**
- The value should rarely - if ever **change**
- One relation, r_1 may have an attribute which is a primary key in r_2 - this is a **foreign key** from r_1 to r_2
- r_1 is the referencing relation, and r_2 is the referenced relation

Structure of Relational Databases

It is common to format schemas as follows

Instructor(<u>InstructorID</u>, InstructorName, Department_Name, Salary)
Course(<u>CourseID</u>, CourseName, Department_Name, Credits)

Structure of Relational Databases

Let's look at some examples

Structure of Relational Databases

Table 3: Employee Information

ID	Name	DepartmentName	Salary
1	John Smith	Sales	$60,000
2	Jane Doe	Marketing	$55,000
3	Bob Johnson	HR	$50,000
4	Alice Brown	Engineering	$70,000
5	Chris Lee	Accounting	$65,000
7	John Lee	Accounting	$33,000

Bad Keys

Example 1: Salary

Table 4: Employee Information

ID	Name	DepartmentName	Salary
1	John Smith	Sales	\$60,000
2	Jane Doe	Marketing	\$55,000

- Lets say Jane gets a 15,000 raise

Example 1: Salary

Table 4: Employee Information

ID	Name	DepartmentName	Salary
1	John Smith	Sales	$60,000
2	Jane Doe	Marketing	$55,000

- Lets say Jane gets a 15,000 raise
- Now she would have the same ID as John

Example 1: Salary

Table 4: Employee Information

ID	Name	DepartmentName	Salary
1	John Smith	Sales	\$60,000
2	Jane Doe	Marketing	\$55,000

- Lets say Jane gets a 15,000 raise
- Now she would have the same ID as John
- It'd no longer be unique!

Example 1: Salary

Table 4: Employee Information

ID	Name	DepartmentName	Salary
1	John Smith	Sales	$60,000
2	Jane Doe	Marketing	$55,000

- Lets say Jane gets a 15,000 raise
- Now she would have the same ID as John
- It'd no longer be unique!
- And what if we wanted to look at older data on Jane!

Example 2: DepartmentName

Table 5: Employee Information

ID	Name	DepartmentName	Salary
5	Chris Lee	Accounting	$65,000
7	John Lee	Accounting	$33,000

- Multiple people are in the same department, so it won't be unique

Example 2: DepartmentName

Table 5: Employee Information

ID	Name	DepartmentName	Salary
5	Chris Lee	Accounting	$65,000
7	John Lee	Accounting	$33,000

- Multiple people are in the same department, so it won't be unique
- Also...department assignments can change (what if John transfers?)

Example 3: Good Example - ID

Table 6: Employee Information

ID	Name	DepartmentName	Salary
1	John Smith	Sales	$60,000
2	Jane Doe	Marketing	$55,000
3	Bob Johnson	HR	$50,000
4	Alice Brown	Engineering	$70,000
5	Chris Lee	Accounting	$65,000
7	John Lee	Accounting	$33,000

- Its an artificial number assigned to employees so its fixed and unique

Example 4: Not Great - Concatenated Key with ID

Table 7: Employee Information

ID	Name	DepartmentName	Salary	ID2
1	John Smith	Sales	$60,000	John_1
2	Jane Doe	Marketing	$55,000	Jane_2

- ID2 contains ID so it will be unique

Example 4: Not Great - Concatenated Key with ID

Table 7: Employee Information

ID	Name	DepartmentName	Salary	ID2
1	John Smith	Sales	$60,000	John_1
2	Jane Doe	Marketing	$55,000	Jane_2

- ID2 contains ID so it will be unique
- But just using ID will also uniquely identify each tuple

Example 4: Not Great - Concatenated Key with ID

Table 7: Employee Information

ID	Name	DepartmentName	Salary	ID2
1	John Smith	Sales	$60,000	John_1
2	Jane Doe	Marketing	$55,000	Jane_2

- ID2 contains ID so it will be unique
- But just using ID will also uniquely identify each tuple
- So ID2 is just adding data to our table which we don't really need

Example: Tuples

In our example, a tuple may be (1, John Smith, Sales, 60,000)

Order Doesn't Matter

This is the same relation, tuples are preserved

Table 8: Employee Information

ID	Name	DepartmentName	Salary
4	Alice Brown	Engineering	\$70,000
2	Jane Doe	Marketing	\$55,000
3	Bob Johnson	HR	\$50,000
1	John Smith	Sales	\$60,000
7	John Lee	Accounting	\$33,000
5	Chris Lee	Accounting	\$65,000

Practice Exercises

Consider the following relational database

employee(person_name, street, city)
works(company_name,, person_name, salary)
company(company_name, city)

What are appropriate primary keys?

Practice Exercises

Consider the following relational database

employee(**person_name**, street, city)
works(company_name, **person_name**, salary)
company(**company_name**, city)

What are appropriate primary keys?

Practice Exercises

A peer argues that name is an appropriate primary key for this relation since no two people have the same first and last name. Another says it is a super key not a primary key. What is your position?

Table 9: Employee Information

ID	Name	DepartmentName	Salary
4	Alice Brown	Engineering	$70,000
2	Jane Doe	Marketing	$55,000
3	Bob Johnson	HR	$50,000
1	John Smith	Sales	$60,000
7	John Lee	Accounting	$33,000
5	Chris Lee	Accounting	$65,000

Practice Exercises

Neither. It is not a superkey or a primary key because someone with the same name may be hired in the future, even if no one has the same exact name right now.

Practice Exercises

Consider the following two relations:

CID	Name	Email
1	John Smith	john@example.com
2	Jane Doe	jane@example.com
3	Bob Johnson	bob@example.com

ProductID	Name	Price	CID
101	Product A	$20	2
102	Product F	$20	2
103	Product C	$25	1
104	Product E	$40	3

What is an insertion of a tuple that would violate the foreign key constraint.

Practice Exercises

Adding (10106, Product E, 40, 11) - because there is no customer 11