LECTURE 10 INTRODUCTION TO DATABASE

Acknowledgement

- The materials on databases are adapted from slides by Professor Peixiang Zhao for COP4710, as well as slides from earlier offering of COP4521 by Sharanya Jayaraman and Karen Works.

Database

- A database is a collection of data stored electronically, usually organized and structured in some form.
- Databases can be very simple – a text file or a CSV file, for example.
- When they are complex, we use formal methods of design and engineering principles, to build, update and maintain these systems.
- A DataBase Management System (DBMS) is the software used to interact between the database and an application that uses the database.
- Data in databases is usually stored as tables (Relational) or through some form of XML or JSON-based collections (NOSQL).

Database Management System (DBMS)

- System for providing EFFICIENT, CONVENIENT, and SAFE MULTI-USER storage of and access to MASSIVE amounts of PERSISTENT data
  - Lots of demand for such systems.

Banking System – A Motivating Example for Database

- Data
  - Information on accounts, customers, balances, current interest rates, transaction histories, etc.
- Massive
  - Many gigabytes at a minimum for large banks, more if keep history of all transactions, even more if keep images of checks -> Far too big for memory
- Persistant
  - Data outlives programs that operate on it

Banking System – A Database Example

- Safe
  - from system failures
  - from malicious users
- Convenient
  - simple commands to - debit account, get balance, write statement, transfer funds, etc.
  - also unpredicted queries should be easy
- Efficient
  - don't search all files to get the balance of one account, get all accounts with low balances, get large transactions, etc.
  - massive data! -> DBMS's carefully tuned for performance

Banking System – A Database Example

- Multi-user access: many people/programs accessing same database, or even same data, simultaneously. There are concurrency issues and needs careful control.
  - Alex @ ATM1 (New York): withdraw $100 from account #007

    ```
    get balance from database;
    if balance >= 100 then balance := balance - 100;
    dispense cash;
    put new balance into database;
    ```

  - Bob @ ATM2 (Tallahassee): withdraw $50 from account #007

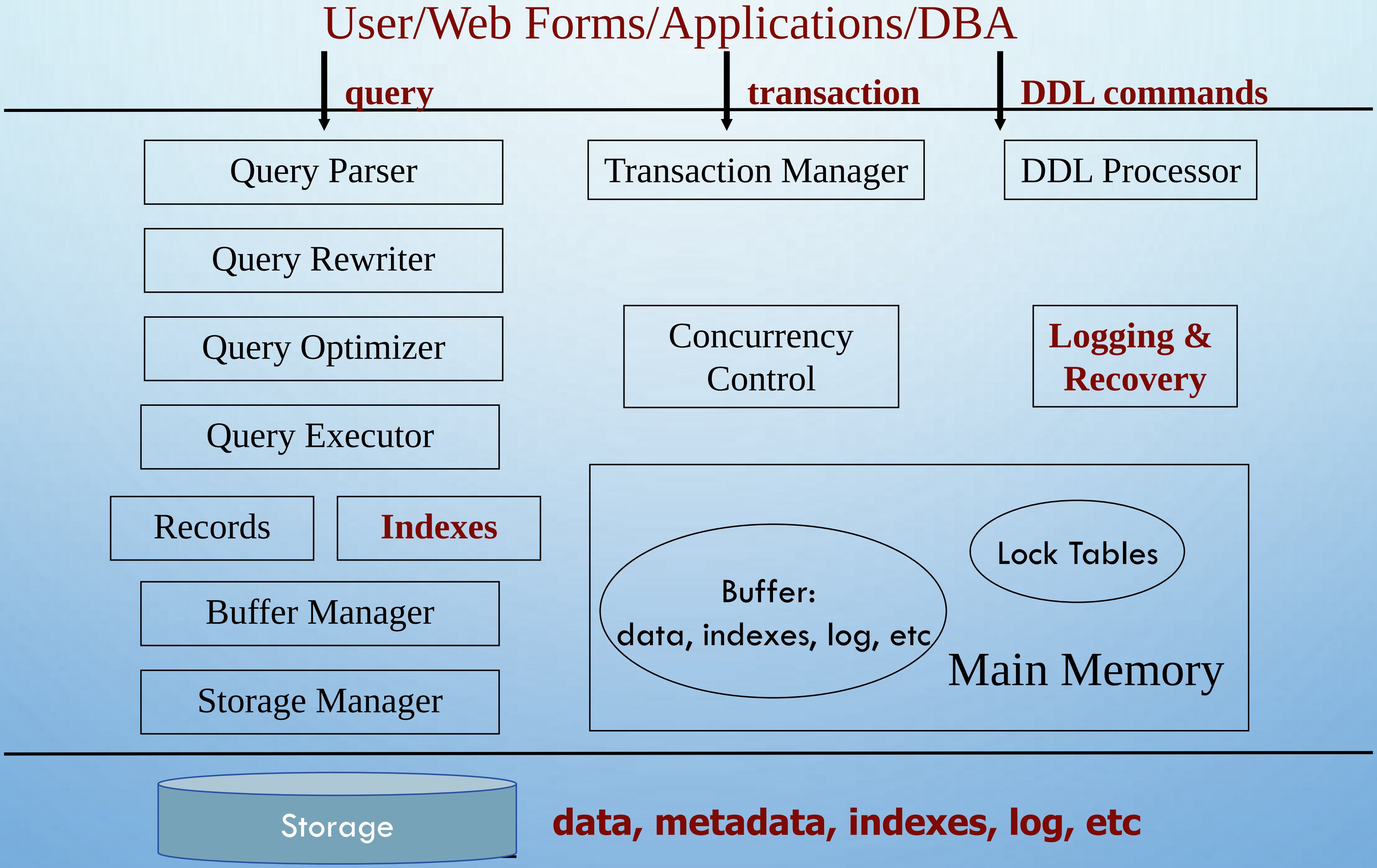
    ```
    get balance from database;
    if balance >= 50 then balance := balance - 50;
    dispense cash;
    put new balance into database;
    ```

  - **Initial balance = 200. Final balance = ??**

Banking System – A Database Example

- Can we do it on a regular file system?
  - Storing data: file system is limited
    - size limit by disk or address space
    - when the system crashes, we may lose data
    - password/file-based authorization insufficient
  - Query/update:
    - need to write a new C++/Java program for every new query
    - need to worry about performance
  - Concurrency: limited protection
    - need to worry about interfering with other users
    - need to offer different views to different users (e.g. registrar, students, professors)
  - Schema change:
    - entails changing file formats
    - need to rewrite virtually all applications
- This is what motivates DBMS!

DBMS ARCHITECTURE



Data Structuring: Model, Schema, Data

- Data model
  - Conceptual structuring of data stored in database
    - Higher level than schema and decides what can be in the schema
  - ex: data is set of records, each with student-ID, name, address, courses, photo
  - ex: data is graph where nodes represent cities, edges represent airline routes
- Schema versus data
  - **Schema**: is the blueprint of the database, specifying data fields and their types (how data is to be structured in the database).
    - defined at set-up time, rarely changes (also called "**metadata**")
  - **Data** is actual "**instance**" of database, changes rapidly
    - Data is stored in the databased at a particular time
  - vs. types and variables in programming languages

Schema vs. Data

- Schema: name, name of each field, the type of each field
  - Students (Sid:string, Name:string, Age: integer, GPA: real)
  - A template for describing a student
- Data: an example instance of the relation

Sid	Name	Age	GPA
0001	Alex	19	3.55
0002	Bob	22	3.10
0003	Chris	20	3.80
0004	David	20	3.95
0005	Eugene	21	3.30

DDL and DML

- Data definition language (DDL)
  - commands for setting up schema of database
- Data Manipulation Language (DML)
  - Commands to manipulate data in database:
    - RETRIEVE, INSERT, DELETE, MODIFY
  - Also called "query language"

Key Steps in building a (relational) DB application

- Step 1: Conceptual design
  - Use a modeling language to express what to model in the application
    - ER model or relational model are popular models
  - With the ER model, the output of this step is an ER diagram of the application domain
- Step 2: Select a type of DBMS
  - Relational DBMS is currently the most popular DBMS
- Step 3: Translate a model design to a relational schema (assume that a relational DBMS is selected)
  - Use a set of **rules** to translate from ER to relational schema
  - Use a set of schema refinement rules to transform the above relational schema into a **good** relational schema

Key Steps in building a DB application

Step 4: Implement the relational DBMS using a "database programming language" called SQL

Step 5: Write the application program (in C++, Python, Java, PHP) to handle user interactions and take care of things that the database does not do.

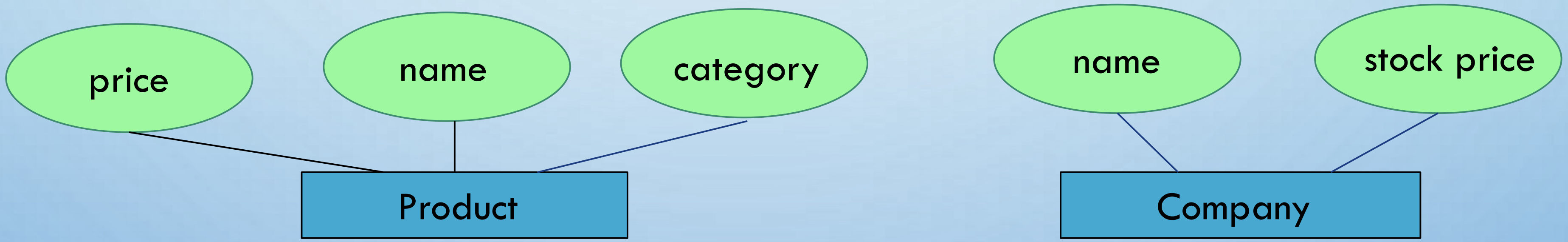
- Ordinary users do not know SQL, and cannot directly interact with the database.

ER Model

- A language to specify
  - What **information** a database must hold
  - What are the **relationships** among components of that information
- Proposed by Peter Chen in 1976
  - "*The Entity-Relationship Model --- Toward a Unified View of Data*". in **ACM transactions on database systems (TODS)**

Components of ER model

- ER model consists of entities, attributes, and relationships among entities
- Entities
  - Real-world objects distinguishable from other objects. Example: a company, a product, a job, a course, etc.
  - Described by a set of **attributes**



- Attributes
  - each has an **atomic** domain: string, integers, reals, etc.

(Binary) Relationship

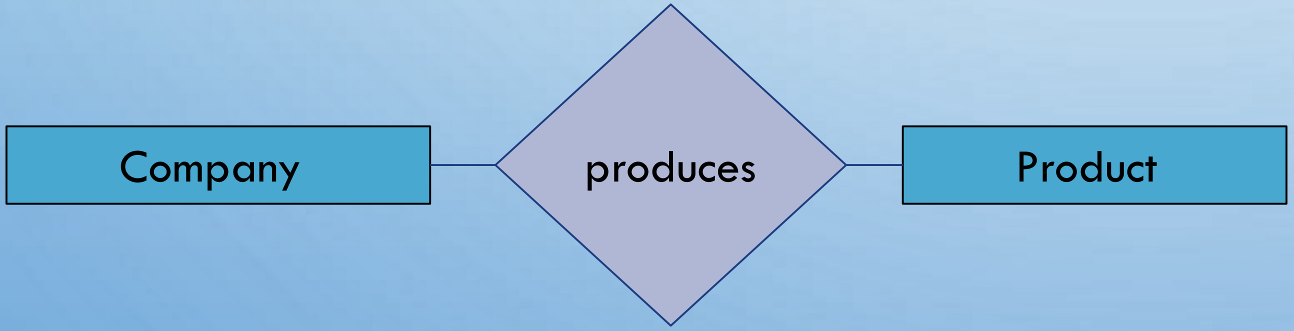
- Math definition:
  - The Cartesian product of two sets is the set of all ordered pair where the first element is from the first set and the second element is from the second set.
  - Let A = {1, 2, 3}, B = {a, b}. The Cartesian product of A and B is

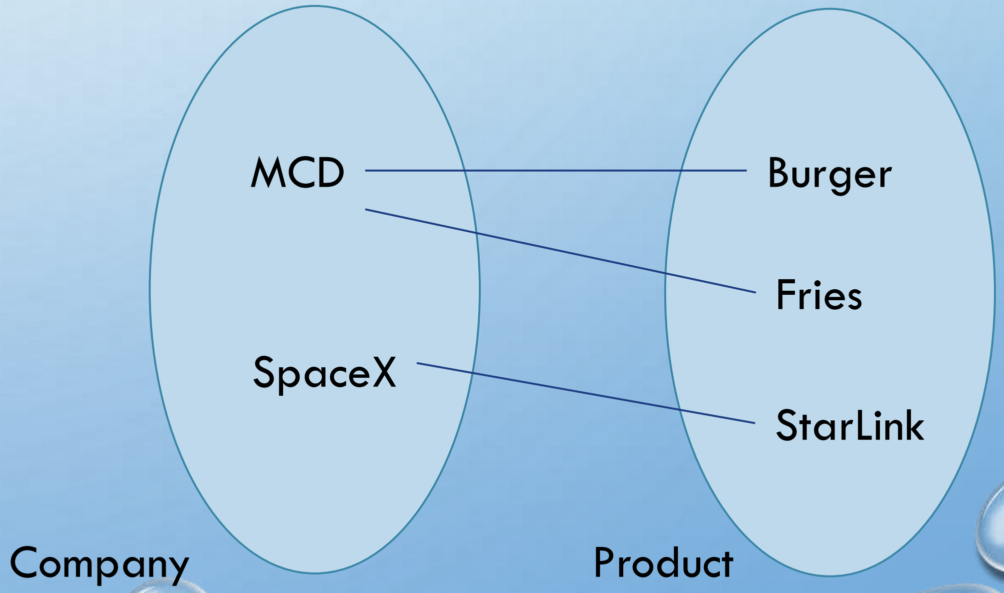
$$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$$

  - Let A, B be sets, a relation R (between A and B) is a subset of $A \times B$

(Binary) Relationship

- Math definition:
  - Let A, B be sets, a relation R (between A and B) is a subset of $A \times B$
  - Example: Company={MCD, SpaceX}, Product = {Burger, Fries, StarLink}
    - ❖ $Company \times Product = \{(MCD, Burger), (MCD, Fries), (MCD, StarLink), (SpaceX, Burger), (SpaceX, Fries), (SpaceX, StarLink)\}$
    - ❖ A produces relationship, $produces = \{(MCD, Burger), (MCD, Fries), (SpaceX, StarLink)\}$ is a subset of $Company \times Product$.



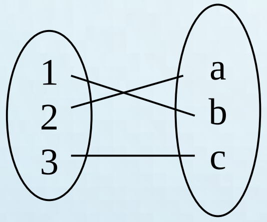


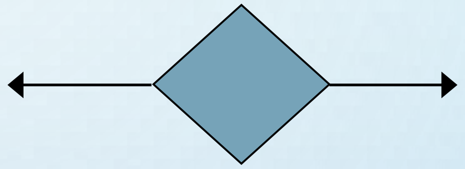
Exercise

- Let students be a set {‘John’, ‘Jane’, ‘June’}, classes be a set {COP4610, COP4530, COP4521}
  - What is the Cartesian production of students and classes
$$students \times classes = ???$$
  - Is {(John, COP4610), (John, COP4521), (Jane, COp4610), (June, COP4530)} a relation between students and classes?
  - Is {(John, COP4610), (John, COP4521)} a relation between students and classes?
  - Is {(John, COP4610), (John, COP4521), (John, COP3014)} a relation between students and classes?

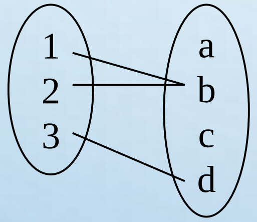
Multiplicity of E/R Relationship

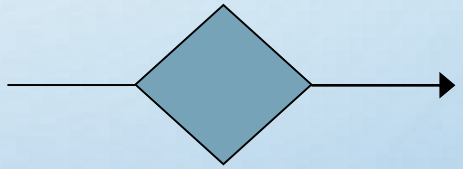
- **one-one:**



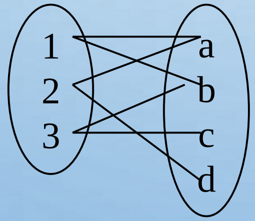


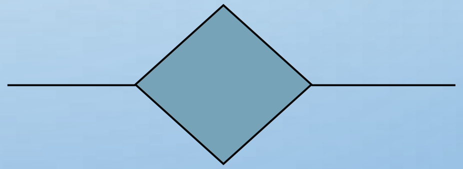
- **many-one**



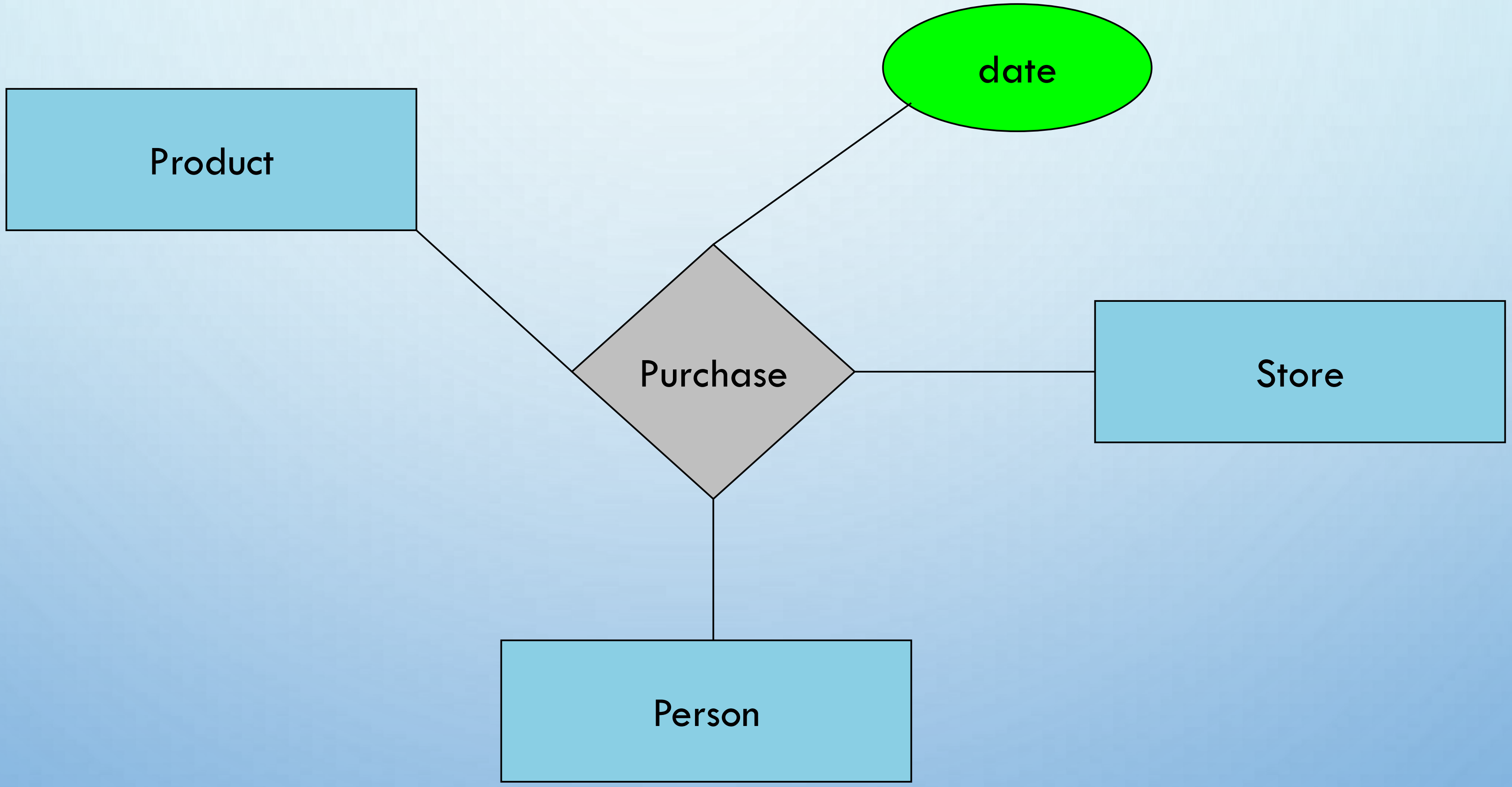


- **many-many**





Multiway relationship

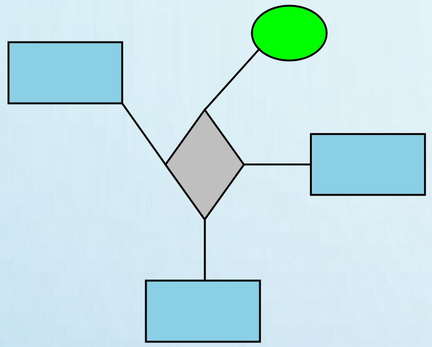


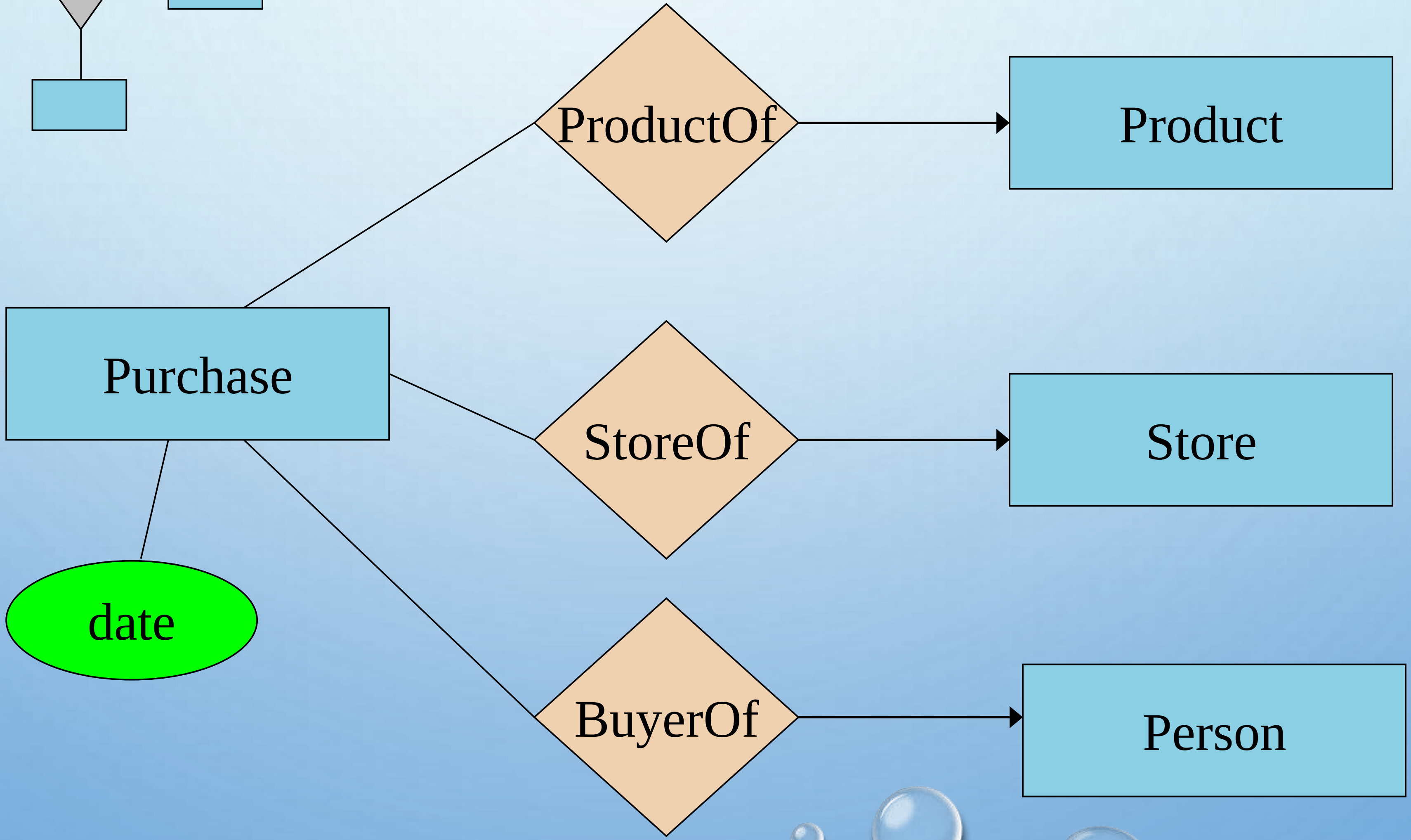
Converting n-ary relationship to binary relationship

- For an n-ary relationship R among entities $E_1, E_2, \ldots, E_n$:
    - Create a new entity R' representing the relationship
    - Connect R' to each E_i with a binary relationship
    - Move all attributes in R to R'

(The key of R' is the combination of keys of all E_i)

Convert multiway relationship to Binary



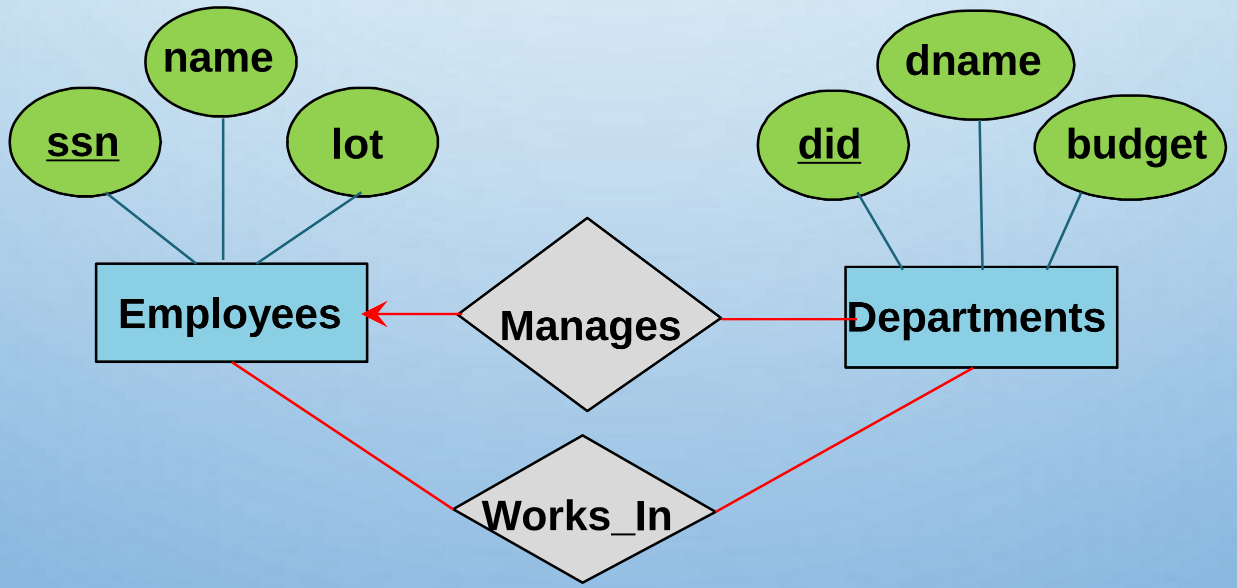


Relationships: summary

- Modeled as a mathematical set
- Binary and multiway relationships
- Converting a multiway one into many binary ones
- Constraints on the degree of the relationship
    - many-one, one-one, many-many
- Attributes of relationships

An ER-Model Example

- An employee can work in many departments, and a department can have many employees.
- Each department has at most one manager.



Relational Model

- E. F. Codd's Relational Database Model is a *mathematical structure* in which data resides
  - *Codd, Edgar Frank (June 1970). "A Relational Model of Data for Large Shared Data Banks" (PDF). Communications of the ACM. **13** (6): 377–87.*
- Conceptual model based on Codd's mathematical data model
  - All data in a database should be represented as relations (tables)

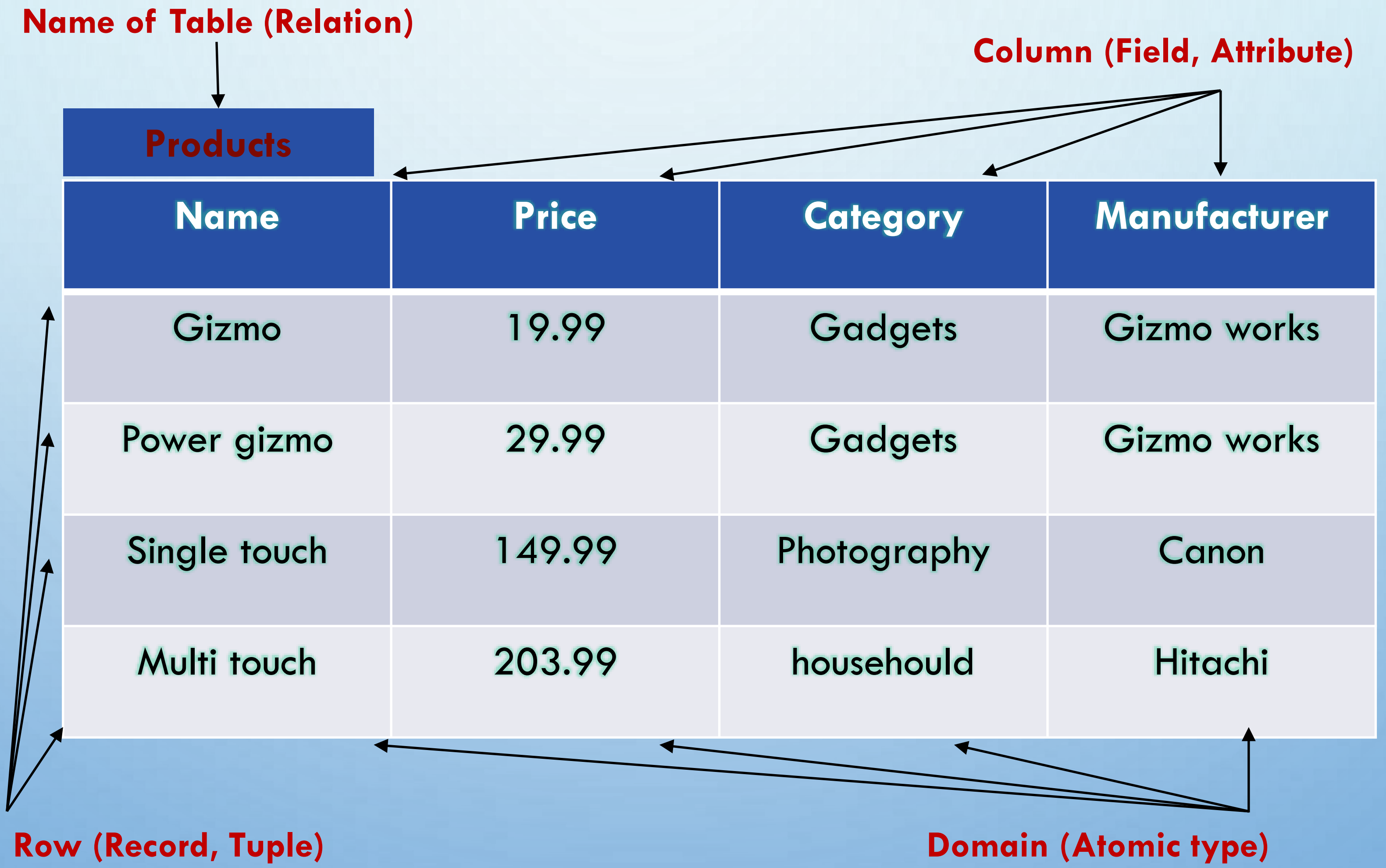
Relational Model

- Both ER model and relational model are used to model data
- ER model
    - Models the data using entities, relationships, and attributes
    - Well-suited for capturing the application requirements
    - Not well-suited for computer implementation
- Relational model
    - Model the data with one single concept: relation
    - The world is represented with a collection of relations (tables)
    - Well-suited for efficient implementation on computer

Relations (Tables)

- Each **relation** consists of
  - Rows (**tuple**s): representing individual record
  - Columns (**attribute**s): representing properties of the data
- A **relation** is a table with unique rows and unordered columns. Mathematically, it is a set of tuples (unique rows, no duplicates)
- An **attribute** is a named column of a relation, representing a data field
  - The set of possible value for an attribute is called a **domain**

An Example Relation



Products

Name	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	Gizmo works
Power gizmo	29.99	Gadgets	Gizmo works
Single touch	149.99	Photography	Canon
Multi touch	203.99	househould	Hitachi

Relation Schema

- Formal description of a table
- Schema of a relation: $R(A_1, A_2, \ldots, A_k)$, blueprint containing all the information needed to create the table
    1. Relation name
    2. Attribute names
    3. Attribute domains
- Schema of a database: $R_1(\ldots), R_2(\ldots),\ldots, R_n(\ldots)$
    - **A set of relation schemas**

Relational Schema

Employee(EmployeeNum:INTEGER,
LName:STRING,
FName:STRING,
DeptNum:INTEGER,
Age:INTEGER)

1) Primary key (EmployeeNum) Attribute Name is underlined. The primary key uniquely identify a record.
2) Attribute Name: Domain Name Pair (Age:INTEGER)
3) Domain Name is the data type of the attributes

Attribute Domain

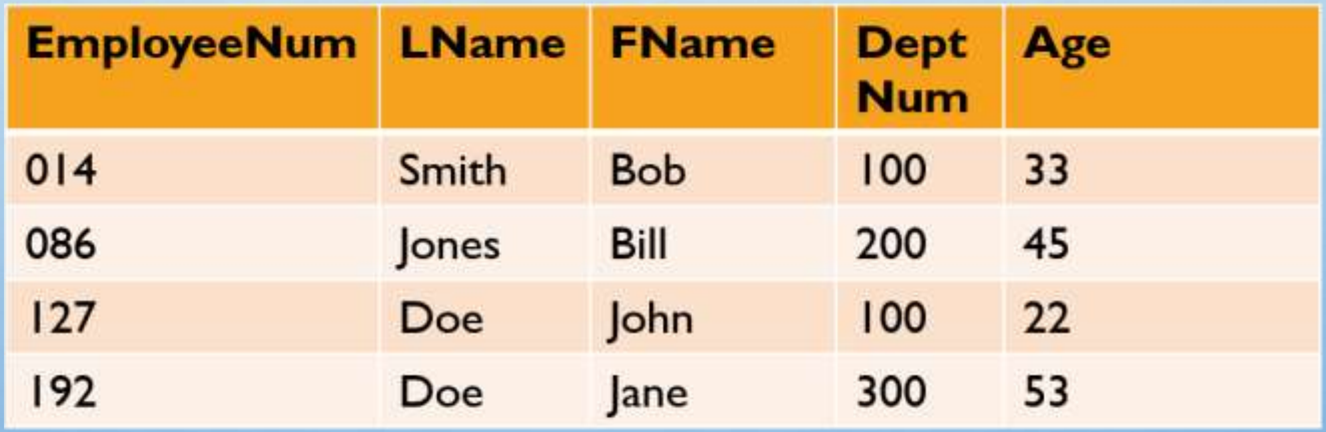
- A ***Domain*** is the set of values an attribute may take.
- An attribute’s domain:
  - data type of the attribute
  - optionally a user-defined set of values.

- Example: the domain of Age:
  - data type: integer
  - user-defined set of values: between 0 and 120 inclusive
    - ❖ The only values that may exist in the attribute Age of any tuple in the Employee relation can only be between 1 and 120.

Rules for Relations (Tables)

1. No two rows can have the exact same contents as each other: **rows are *unique*.**
2. All the rows of a particular column draw their values from the same attribute domain.
3. Attribute values are atomic.
   - An attribute cannot contain a list, array, object, or any compound data structure.

Employee relation



EmployeeNum	LName	FName	Dept Num	Age
014	Smith	Bob	100	33
086	Jones	Bill	200	45
127	Doe	John	100	22
192	Doe	Jane	300	53

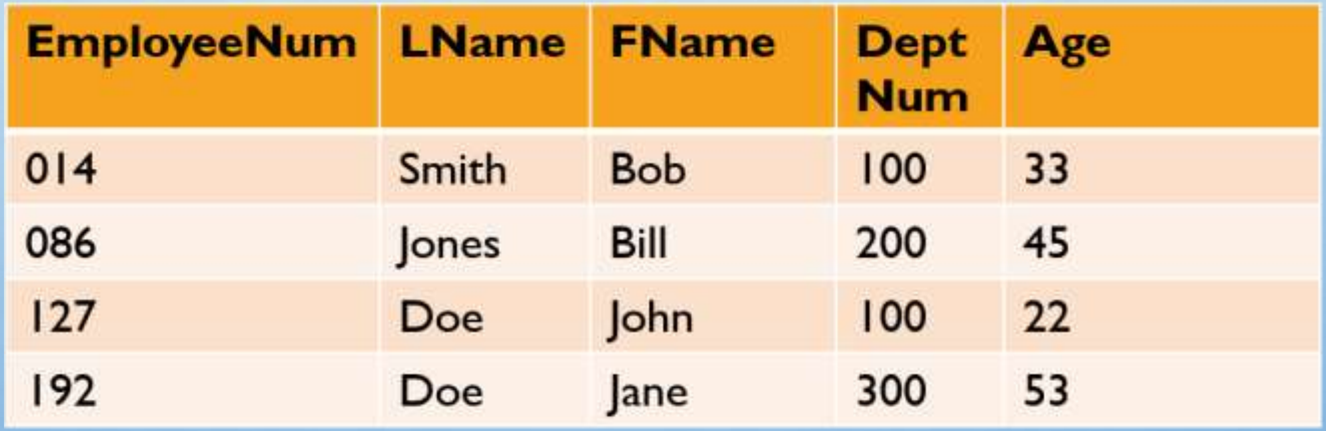
Rules for Relations (Tables)

4. There can be rules called integrity constraints declared on the tables which the tables automatically obey.

5. The order of the rows makes no difference.

6. The order of the columns makes no difference.

Employee relation

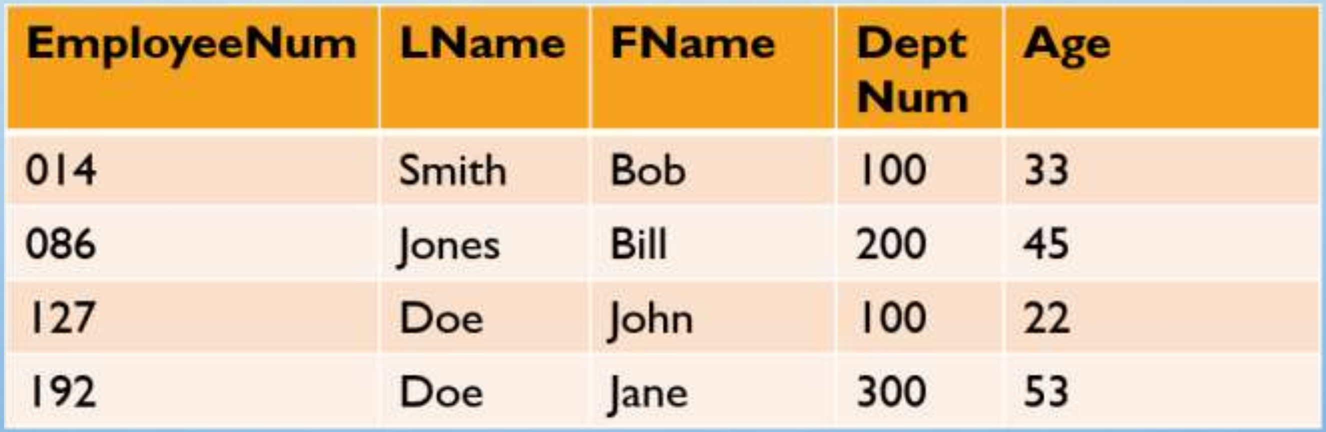


EmployeeNum	LName	FName	Dept Num	Age
014	Smith	Bob	100	33
086	Jones	Bill	200	45
127	Doe	John	100	22
192	Doe	Jane	300	53

Relations (Tables) and classes (in OOP)

- Tables are like OO classes.
- Each row is like an instance of the class.
- Each column represents a characteristic or instance variable.

Employee relation



EmployeeNum	LName	FName	Dept Num	Age
014	Smith	Bob	100	33
086	Jones	Bill	200	45
127	Doe	John	100	22
192	Doe	Jane	300	53

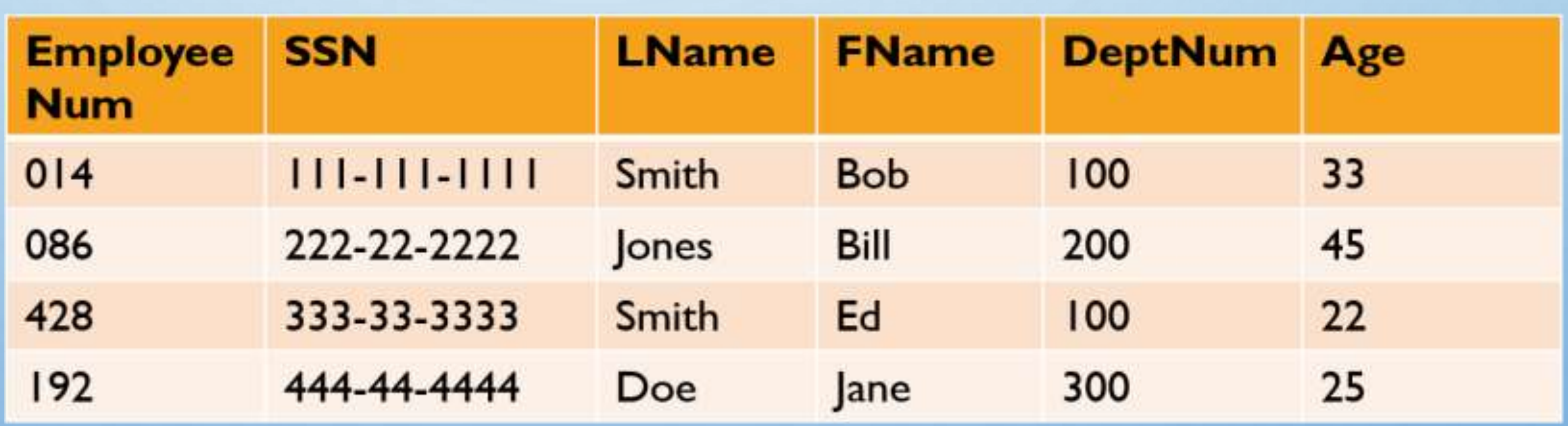
Key Attributes

- The key attribute is an attribute or a combination of attributes whose:
  - Values are unique for each row in the table and,
  - By definition, all rows in a relation are unique: this guarantees that there is a key for the relation. In the worst case, the key contains all attributes.
- *Primary Key* is the main identifier for rows in a table.
- Each table has one primary key.

Candidate Keys

- When there are multiple attributes that each could serve as a primary key, we call them *Candidate Keys*.
- Each one may be chosen as the *Primary Key*.

Employee relation



Employee Num	SSN	LName	FName	DeptNum	Age
014	111-111-1111	Smith	Bob	100	33
086	222-22-2222	Jones	Bill	200	45
428	333-33-3333	Smith	Ed	100	22
192	444-44-4444	Doe	Jane	300	25

Foreign Key

- A Foreign Key is a field whose values are keys in another relation
  - Must correspond to primary key of the second relation
  - Like a `logical pointer’
- Relations can be connected via keys: foreign key links the two tables via common values between the PK and the FK
  - This is the mechanism we use to link tables together in the Relational Model

Foreign Key

Employee

Employee Num PK	SSN	LName	FName	DeptNum	Age
014	111-111-1111	Smith	Bob	100	33
086	222-22-2222	Jones	Bill	200	45
428	333-33-3333	Smith	Ed	100	22
192	444-44-4444	Doe	Joe	300	25

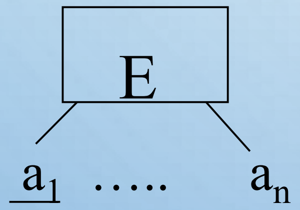
FamilyMember

Employee Num FK	LName	FName	Relationship
014	Smith	Susan	Wife
014	Smith	Billy	Son
014	Smith	John	Son
192	Doe	Edna	Wife
192	Doe	Mary	Daughter
428	Smith	Betty	Wife

Translate ER Diagram into Relations

- Translate entity

ER diagram



Relation Schema

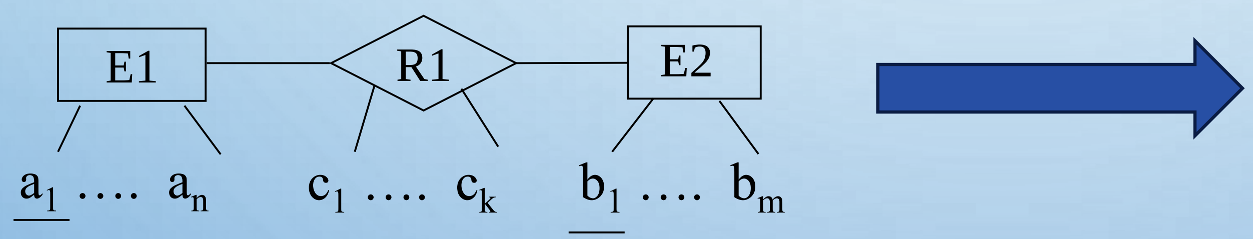
$E = (\underline{a_1,}\ \ldots,\ a_n)$

Translate ER Diagram into Relations

- Translate relation

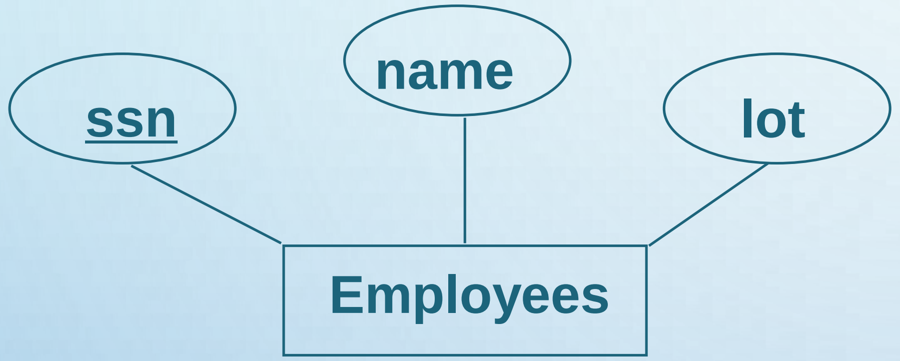
ER diagram

Relation Schema



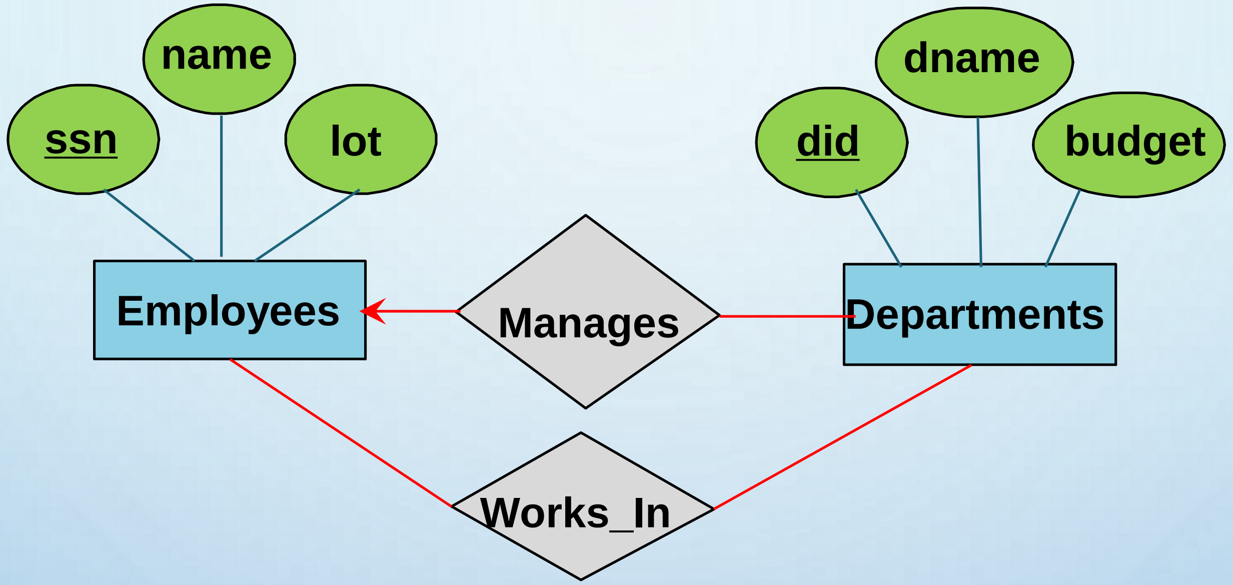
R1= ($\underline{a_1}$, $\underline{b_1}$, c_1, …, c_k)

Example



ssn	name	lot
123-22-3666	Attishoo	48
231-31-5368	Smiley	22
131-24-3650	Smethurst	35

Example



- Four tables:
  - Employees(<u>ssn</u>, name, lot)
  - Departments(<u>did</u>, dname, budget)
  - Manages(<u>ssn</u>, <u>did</u>)
  - Works_in(<u>ssn</u>, <u>did</u>)

Structured Query Language (SQL)

- SQL is the language used to create, query and change
  - The relation schema (the structure of the tables)
  - The data in the tables
- Once we have the relational model for an application, we can implement the model using SQL.