# CS4.301 Data & Applications

Ponnurangam Kumaraguru (“PK”)
#ProfGiri @ IIIT Hyderabad

 pk.profgiri  /in/ponguru  @ponguru  Ponnurangam.kumaraguru

# Weak Entity Types

An entity that does not have a key attribute and that is identification-dependent on another entity type.

A weak entity must participate in an identifying relationship type with an owner or identifying entity type

Entities are identified by the combination of:

- A partial key of the weak entity type
- The particular entity they are related to in the identifying relationship type

**Example:**

- A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
- Name of DEPENDENT is the *partial key*
- DEPENDENT is a *weak entity type*
- EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

# Weak Entity Types

| Concept | Symbol | Meaning |
| --- | --- | --- |
| Strong Entity | Single rectangle | Independent existence |
| Weak Entity | Double rectangle | Depends on strong entity |
| Identifying Relationship | Double diamond | Defines weak entity's identity |

# Attributes of Relationship types

A relationship type can have attributes:

- For example, HoursPerWeek of WORKS_ON
- Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
  - A value of HoursPerWeek depends on a particular (employee, project) combination
- Most relationship attributes are used with M:N relationships

A **relationship type** can have an **attribute** when that attribute **belongs to the association itself** — not to any of the entities individually.

Think of it as information that **only makes sense because of the relationship** between the entities.

# Attributes of Relationship types

Add attributes to a relationship type **only when**:

1. The information applies **to the link itself**, not to either entity.
2. It's **not derivable** from the entities' existing attributes.
3. It helps capture **real-world meaning** of that association.

# Attributes of Relationship types

When an ER relationship has attributes, those attributes are stored as columns in the relationship table (also called a junction or association table) that represents the relationship in the relational database.

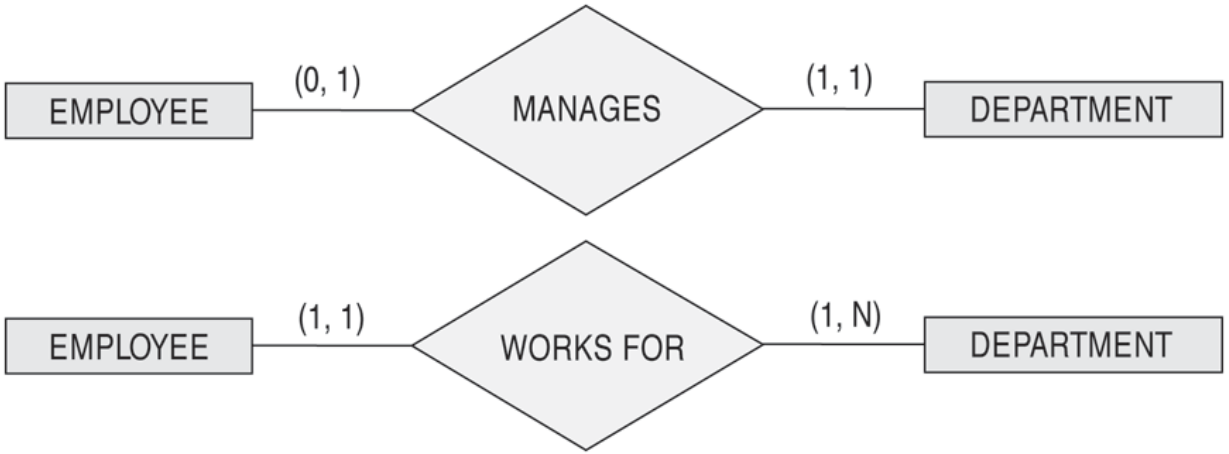
Employee(Emp_ID, Name)

Project(Proj_ID, Title)

Works_On(Emp_ID, Proj_ID, hours_per_week)

Relationship attributes become **columns in the relationship (junction) table** when the ER model is converted to the relational model.

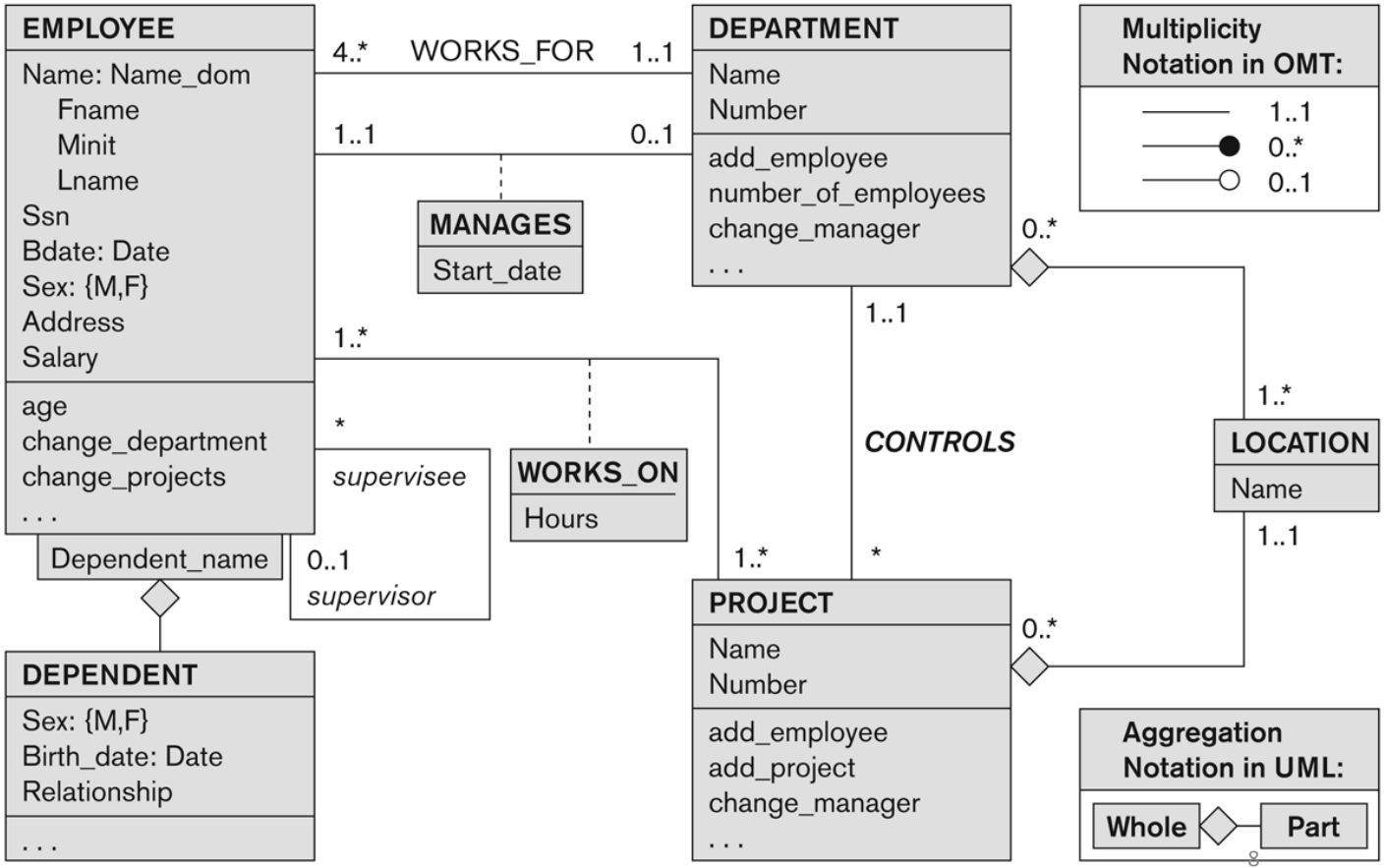
# The (min,max) notation for relationship constraints




Read the min,max numbers next to the entity type and looking **away from** the entity type

**Figure 3.16**
The COMPANY conceptual schema in UML class diagram notation.

UML class diagram for COMPANY database schema



# DBMS Programming Language Interfaces

## Programmer interfaces for embedding DML in a programming languages:

**Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)

**Procedure Call Approach**: e.g. JDBC for Java, ODBC (Open Databse Connectivity) for other programming languages as API's (application programming interfaces)

**Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components

**Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

# User-Friendly DBMS Interfaces

- Menu-based (Web-based), popular for browsing on the web
- Forms-based, designed for naïve users used to filling in entries on a form
- Graphics-based
  - Point and Click, Drag and Drop, etc.
  - Specifying a query on a schema diagram
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

# Database System Utilities

To perform certain functions such as:

- Loading data stored in files into a database. Includes data conversion tools.
- Backing up the database periodically on tape.
- Reorganizing database file structures.
- Performance monitoring utilities.
- Report generation utilities.
- Other functions, such as sorting, user monitoring, data compression, etc.

# The Relational Data Model and Relational Database Constraints

# Informal Definitions

Key of a Relation:

- Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
  - Called the *key*
- In the STUDENT table, SSN is the key

- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
  - Called *artificial key* or *surrogate key*

# Activity

Create 3 recursive relationship and show them in ER diagram

Create 3 different weak entities in 3 different mini-world and show them in ER diagram

Create 3 different sets of relationships in Infinium showing cardinality / min-max

# Formal Definitions - Schema

The **Schema** (or description) of a Relation:

- Denoted by R(A1, A2, .....An)
- R is the **name** of the relation
- The **attributes** of the relation are A1, A2, ..., An

Example:

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

- CUSTOMER is the relation name
- Defined over the four attributes: Cust-id, Cust-name, Address, Phone#

Each attribute has a **domain** or a set of valid values.

- For example, the domain of Cust-id is 6 digit numbers.

# Formal Definitions - Tuple

A **tuple** is an ordered set of values (enclosed in angled brackets ‘< … >’)

Each value is derived from an appropriate *domain*.

A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:

<632895, "John Smith", "101 Main St. Atlanta, GA  30332", "(404) 894-2000">

This is called a 4-tuple as it has 4 values

A tuple (row) in the CUSTOMER relation.

A relation is a **set** of such tuples (rows)

# Formal Definitions - Domain

A **domain** has a logical definition:

Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.

A domain also has a data-type or a format defined for it.

The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.

Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

The attribute name designates the role played by a domain in a relation:

Used to interpret the meaning of the data elements corresponding to that attribute

Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

# Definition Summary

| Informal Terms | | Formal Terms |
|---|---|---|
| Table | | Relation |
| Column Header | | Attribute |
| All possible Column Values | | Domain |
| Row | | Tuple |
| Table Definition | | Schema of a Relation |
| Populated Table | | State of the Relation |

# Characteristics Of Relations

Ordering of tuples in a relation r(R):

The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.

Ordering of attributes in a relation schema R (and of values within each tuple):

We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .

(However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes ).

Example: t= { <name, “John” >, <SSN, 123456789> }

This representation may be called as “self-describing”.

# This Lecture

# Characteristics Of Relations

Values in a tuple:

- All values are considered atomic (indivisible).
- Each value in a tuple must be from the domain of the attribute for that column
  - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
  - Then each *vi* must be a value from *dom(Ai)*

- A special **null** value is used to represent values that are unknown or not available or inapplicable in certain tuples.

# Characteristics Of Relations

Notation:

- We refer to **component values** of a tuple t by:
  - t[Ai] or t.Ai
  - This is the value vi of attribute Ai for tuple t
- Similarly, t[Au, Av, ..., Aw] refers to the subtuple of t containing the values of attributes Au, Av, ..., Aw, respectively in t

# CONSTRAINTS

Constraints determine which values are permissible and which are not in the database. They are of three main types:

1. **Inherent or Implicit Constraints**: These are based on the data model itself. (E.g., relational model does not have tuples to be duplicates)

2. **Schema-based or Explicit Constraints**: They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)

3. **Application based or semantic constraints**: These are beyond the expressive power of the model and must be specified and enforced by the application programs.

# Inherent or Implicit Constraints

Constraints that exist naturally due to the meaning of data or real-world logic, not explicitly declared in the schema

Inherent constraints are enforced by logic, not the DBMS—unless we model them

| Example | Explanation |
| --- | --- |
| A person's **date of birth cannot be in the future** | True by nature of time |
| A **student's age must be positive** | Logical and real-world rule |
| **Arrival time > Departure time** for the same flight | Temporal reality |
| A **bridge can't connect to itself** | Semantic impossibility |

# Schema-based or Explicit Constraints

Constraints that are formally declared in the database schema using SQL. These are explicitly stated in the schema and automatically checked by the DBMS.

| Type | Example | Enforced by |
|---|---|---|
| **Domain Constraint** | `age INT CHECK (age >= 18)` | Column definition |
| **Key Constraint** | `PRIMARY KEY (student_id)` | Table-level rule |
| **Entity Integrity** | Primary key cannot be NULL | DBMS automatically |
| **Referential Integrity** | `FOREIGN KEY (dept_id) REFERENCES Department(dept_id)` | Enforced by DBMS |
| **Unique Constraint** | `UNIQUE (email)` | Ensures no duplicates |

# Application based or semantic constraints

Constraints that depend on business logic, process, or context — not easily enforced in schema. These constraints live in the application layer, triggers, or stored procedures — not the schema itself.

| Example | Explanation |
| --- | --- |
| **A professor cannot teach more than 3 courses per semester** | Depends on institutional policy |
| **Discount applies only if order value > ₹5000** | Business rule enforced by application code |
| **Loan approval requires credit score > 700** | External data & logic-based |
| **Employee salary increase ≤ 20% per year** | Policy constraint, not structural |
| **Two users cannot book the same seat** | Controlled through transactions/application logic |

# Constraints summary

| Type | Defined By | Example | Where Enforced |
|---|---|---|---|
| **Inherent / Implicit** | Nature of data, logic | Age > 0, DoB < Today | Reality / Semantics |
| **Schema-based / Explicit** | SQL schema definition | CHECK, PK, FK | DBMS |
| **Application / Semantic** | Business rules, process logic | Max 3 courses / semester | Application layer |

# Relational Integrity Constraints

Constraints are **conditions** that must hold on **all** valid relation states.

There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:

- **Key** constraints
- **Entity integrity** constraints
- **Referential integrity** constraints

Another schema-based constraint is the **domain** constraint

- Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

# Key Constraints

**Superkey** of R:

Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.

Is a set of attributes SK of R with the following condition:

No two tuples in any valid relation state r(R) will have the same value for SK

That is, for any distinct tuples t1 and t2 in r(R), $t1[SK] \neq t2[SK]$

This condition must hold in *any valid state* r(R)

**Key** of R:

A "minimal" superkey

A candidate key (or simply a key) is a set of one or more attributes that uniquely identify each tuple in a relation (or table)

# Key Constraints (continued)

Example: Student{ID, First_name, Last_name, Age, Sex, Phone_no}

Two candidate keys: ID & {First_name, Last_name, Age, Sex, Phone_no}

Consider the STUDENT relation schema:

- Attribute Ssn is a key, because no 2 students tuples can have the same Ssn
- Any set of attributes that include Ssn, like {Ssn, Name, Age} is a superkey it uniquely identifies all attributes in a relation

In general:

- Any *key* is a *superkey* (but not vice versa)
- Any set of attributes that *includes a key* is a *superkey*
- A *minimal* superkey is also a key

# Example

Candidate keys?

| RollNo | Email | Aadhaar | Name | Dept |
|---|---|---|---|---|
| 101 | alex@iiith.ac.in | 1234 | Alex | CSE |
| 102 | bria@iiith.ac.in | 5678 | Bria | ECE |
| 103 | chay@iiith.ac.in | 9999 | Chay | CSE |

# Example

Candidate keys? {RollNo}, {Email}, {Aadhaar}
Why name can’t be a candidate key?

| RollNo | Email | Aadhaar | Name | Dept |
|---|---|---|---|---|
| 101 | alex@iiith.ac.in | 1234 | Alex | CSE |
| 102 | bria@iiith.ac.in | 5678 | Bria | ECE |
| 103 | chay@iiith.ac.in | 9999 | Chay | CSE |

# Candidate key

Uniqueness: No two tuples (rows) in the table can have the same key value.

Minimality: No subset of that key can uniquely identify the tuple.

| RollNo | Email | Aadhaar | Name | Dept |
|---|---|---|---|---|
| 101 | alex@iiith.ac.in | 1234 | Alex | CSE |
| 102 | bria@iiith.ac.in | 5678 | Bria | ECE |
| 103 | chay@iiith.ac.in | 9999 | Chay | CSE |
| 104 | alex2@iiith.ac.in | 4321 | Alex | ECE |

# Candidate key

| Feature | Superkey | Candidate Key |
| --- | --- | --- |
| **Definition** | Any set of attributes that uniquely identifies tuples | Minimal set of attributes that uniquely identifies tuples |
| **Uniqueness** | Must hold | Must hold |
| **Minimality** | Not required | Must hold |
| **Count per Table** | Many | Subset of superkeys |
| **Example** | `{RollNo, Name}` | `{RollNo}`, `{Email}` |

# Candidate key & Super key difference

| Super Key | Candidate Key |
| --- | --- |
| Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation. | Candidate Key is a subset of a super key. |
| All super keys can't be candidate keys. | But all candidate keys are super keys. |
| Various super keys together makes the criteria to select the candidate keys. | Various candidate keys together makes the criteria to select the primary keys. |
| In a relation, number of super keys is more than number of candidate keys. | While in a relation, number of candidate keys are less than number of super keys. |
| Super key attributes can contain NULL values. | Candidate key attributes can also contain NULL values. |

https://www.geeksforgeeks.org/difference-between-super-key-and-candidate-key/

# Key Constraints (continued)

If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.

- The primary key attributes are underlined.

Example: Consider the CAR relation schema:

- License_number & Engine_serial_number
- We chose Engine_serial_number as the primary key

The primary key value is used to *uniquely identify* each tuple in a relation

- Provides the tuple identity

Also used to *reference* the tuple from another tuple

- General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
- Not always applicable – choice is sometimes subjective

# CAR table with two candidate keys – LicenseNumber chosen as Primary Key

**CAR**

| License_number | Engine_serial_number | Make | Model | Year |
|---|---|---|---|---|
| Texas ABC-739 | A69352 | Ford | Mustang | 02 |
| Florida TVP-347 | B43696 | Oldsmobile | Cutlass | 05 |
| New York MPO-22 | X83554 | Oldsmobile | Delta | 01 |
| California 432-TFY | C43742 | Mercedes | 190-D | 99 |
| California RSK-629 | Y82935 | Toyota | Camry | 04 |
| Texas RSK-629 | U028365 | Jaguar | XJS | 04 |

**Figure 5.4**
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

# Super & Primary key difference

| S.NO | Super Key | Primary Key |
|---|---|---|
| 1. | Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation. | Primary Key is a minimal set of attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation. |
| 2. | All super keys can't be primary keys. | Primary key is a minimal super key. |
| 3. | Various super keys together makes the criteria to select the candidate keys. | We can choose any of the minimal candidate key to be a primary key. |
| 4. | In a relation, number of super keys are more than number of primary keys. | While in a relation, number of primary keys are less than number of super keys. |
| 5. | Super key's attributes can contain NULL values. | Primary key's attributes cannot contain NULL values. |

https://www.geeksforgeeks.org/difference-between-primary-key-and-super-key/

# Relational Database Schema

**Relational Database Schema:**

A set S of relation schemas that belong to the same database.

S is the name of the whole **database schema**

S = {R1, R2, ..., Rn} and a set IC of integrity constraints.

R1, R2, …, Rn are the names of the individual **relation schemas** within the database S

Following slide shows a COMPANY database schema with 6 relation schemas

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|---|---|---|

**DEPARTMENT**

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|

**DEPT_LOCATIONS**

| Dnumber | Dlocation |
|---|---|

**PROJECT**

| Pname | Pnumber | Plocation | Dnum |
|---|---|---|---|

**WORKS_ON**

| Essn | Pno | Hours |
|---|---|---|

**DEPENDENT**

| Essn | Dependent_name | Sex | Bdate | Relationship |
|---|---|---|---|---|

**Figure 5.5**
Schema diagram for the COMPANY relational database schema.

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|---|---|---|

**DEPARTMENT**

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|

**DEPT_LOCATIONS**

| Dnumber | Dlocation |
|---|---|

**PROJECT**

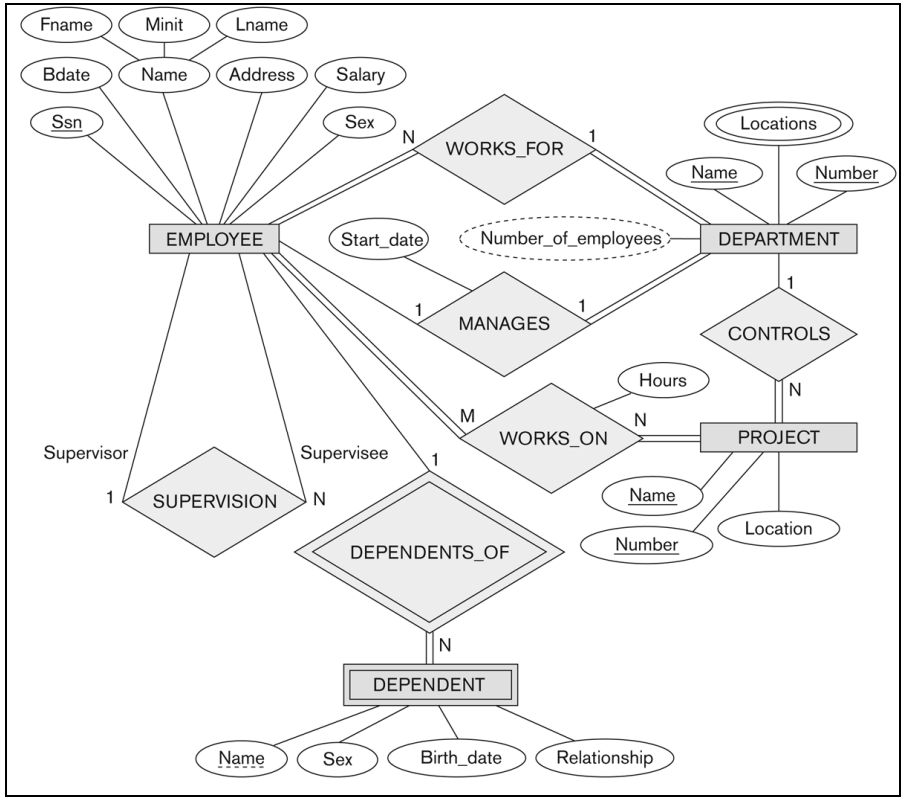
| Pname | Pnumber | Plocation | Dnum |
|---|---|---|---|

**WORKS_ON**

| Essn | Pno | Hours |
|---|---|---|

**DEPENDENT**

| Essn | Dependent_name | Sex | Bdate | Relationship |
|---|---|---|---|---|



# Relational Database State

A **relational database state** DB of $S$ is a set of relation states DB = {$r_1$, $r_2$, ..., $r_m$} such that each $r_i$ is a state of $R_i$ and such that the $r_i$ relation states satisfy the integrity constraints specified in IC.

A relational database *state* is sometimes called a relational database *snapshot* or *instance*.

# Populated database state

Each *relation* will have many tuples in its current relation state

The *relational database state* is a union of all the individual relation states

Whenever the database is changed, a new state arises

Basic operations for changing the database:

- INSERT a new tuple in a relation
- DELETE an existing tuple from a relation
- MODIFY an attribute of an existing tuple

**Figure 5.6**
One possible database state for the COMPANY relational database schema.

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|---|---|---|
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 30000 | 333445555 | 5 |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 |
| Alicia | J | Zelaya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 |

**DEPARTMENT**

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

**DEPT_LOCATIONS**

| Dnumber | Dlocation |
|---|---|
| 1 | Houston |
| 4 | Stafford |
| 5 | Bellaire |
| 5 | Sugarland |
| 5 | Houston |

**WORKS_ON**

| Essn | Pno | Hours |
|---|---|---|
| 123456789 | 1 | 32.5 |
| 123456789 | 2 | 7.5 |
| 666884444 | 3 | 40.0 |
| 453453453 | 1 | 20.0 |
| 453453453 | 2 | 20.0 |
| 333445555 | 2 | 10.0 |
| 333445555 | 3 | 10.0 |
| 333445555 | 10 | 10.0 |
| 333445555 | 20 | 10.0 |
| 999887777 | 30 | 30.0 |
| 999887777 | 10 | 10.0 |
| 987987987 | 10 | 35.0 |
| 987987987 | 30 | 5.0 |
| 987654321 | 30 | 20.0 |
| 987654321 | 20 | 15.0 |
| 888665555 | 20 | NULL |

**PROJECT**

| Pname | Pnumber | Plocation | Dnum |
|---|---|---|---|
| ProductX | 1 | Bellaire | 5 |
| ProductY | 2 | Sugarland | 5 |
| ProductZ | 3 | Houston | 5 |
| Computerization | 10 | Stafford | 4 |
| Reorganization | 20 | Houston | 1 |
| Newbenefits | 30 | Stafford | 4 |

**DEPENDENT**

| Essn | Dependent_name | Sex | Bdate | Relationship |
|---|---|---|---|---|
| 333445555 | Alice | F | 1986-04-05 | Daughter |
| 333445555 | Theodore | M | 1983-10-25 | Son |
| 333445555 | Joy | F | 1958-05-03 | Spouse |
| 987654321 | Abner | M | 1942-02-28 | Spouse |
| 123456789 | Michael | M | 1988-01-04 | Son |
| 123456789 | Alice | F | 1988-12-30 | Daughter |
| 123456789 | Elizabeth | F | 1967-05-05 | Spouse |

# Populated DB state for COMPANY

# Entity Integrity

**Entity Integrity:**

The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).

- This is because primary key values are used to *identify* the individual tuples.
- t[PK] $\neq$ null for any tuple t in r(R)
- If PK has several attributes, null is not allowed in any of these attributes

Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

# Referential Integrity

Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.

A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].

A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.PK

# Referential Integrity (or foreign key) Constraint

Statement of the constraint

The value in the foreign key column (or columns) FK of the **referencing relation** R1 can be **either**:

(1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or

(2) a **null**.

In case (2), the FK in R1 should **not** be a part of its own primary key.

# Displaying a relational database schema and its constraints

Each relation schema can be displayed as a row of attribute names

The name of the relation is written above the attribute names

The primary key attribute (or attributes) will be underlined

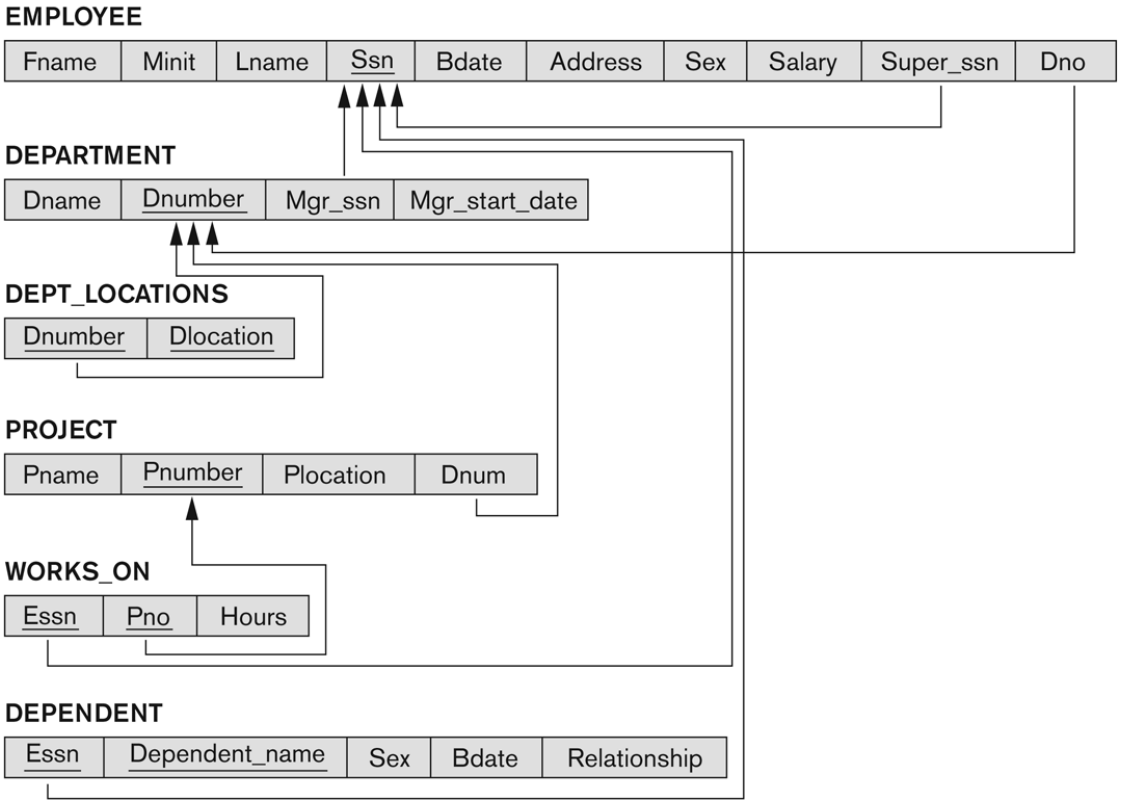
A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table

- Can also point the the primary key of the referenced relation for clarity

Next slide shows the COMPANY **relational schema diagram with referential integrity constraints**

# Referential integrity constraints for Company

**Figure 5.7**
Referential integrity constraints displayed on the COMPANY relational database schema.



# Activity: Infinium mini-world

Write down 3 examples for Inherent constraints

Write down 3 examples for Explicit constraints

Write down 3 examples for Application constraints

Generate a Relation, Super keys, Candidate keys, Primary keys

Write down a at least 2 integrity constraints, and 2 referential integrity constraints

With multiple entities show at lest 2 referential integrity constraints through relational schema

# Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in