Database Management Systems

Vijaya Saradhi

IIT Guwahati

Fri, 17th Jan 2020

Quantifiers

Details

Manifests in many forms

Count vehicle count, employee count, student count etc.

Dimension Answer questions like How long?, how high?, how wide?, how heavy?. A unit must follow the number

Currency amount answers questions of the form How much? and specifies an amount of money (Unit price, payment amount, etc)

Factor Dimensionless quantity: Interest rate, hourly rate, etc.

Specific Time Point answers questions of the form when?: Order date, arrival date etc.

Recurrent Time Point : RD deposit date, subscription renewal time, fee payment date etc

Interval answers questions like: loan repayment period, EMI installments periods etc.

Location answers questions like: Where?

Domain

Definition

A domain D is a set of atomic values. Atomic means each value in the domain is indivisible

- A domain is associated with three things
- A name
- Data type
- Data format

Examples

Example

- CPI - a real value between 0.00 and 10.00

  Name : cpi
  Data type : floating point between (0.00 and 10.00)
  Format `ff.ff`

- Employee age

  Name : `emp_age`
  Data type : integer value between (15 and 80)
  Format `dd`

- Deparmtnet

  Name : `dept_name`
  Data type : From a set of values { CSE, ECE, ME, CE, DD . . . }
  Format `internal format`

Attribute Domains

Details

- Each simple attribute is associated with a domain of values
- Example: age attribute - between 16 and 70
- Example: Name string formed using alphabet {a, b, c, ..., z, SPACE}
- Domain is not represented in ER diagram

Attribute Domains

Mathematical definition

An attribute A of an entity set E whose domain is V is defined as function: $A : E \rightarrow P(V)$ where $P(V)$ is the power set

Attribute Domains

Discussion

- Value of attribute A for entity e is referred as $A(e)$
- Definition of domain covers single-valued and multivalued attributes
- NULL is represented by empty set
- Composite attribute A, the domain of V is the power set of Cartesian product of $P(V_1), P(V_2), \cdots, P(V_n)$;
  $V = P(P(V_1) \times P(V_2) \times \cdots \times P(V_n))$

Entity Types, Entity Sets

Entity Type - Definition

A collection of entities that have same attributes. It describes schema
Example: {Employee, Company}

Entity Types, Entity Sets

Entity Type - Definition

A collection of entities that have same attributes. It describes schema
Example: {Employee, Company}

Entity Set - Definition

The collection of entities of a particular entity type
Example: $\{e_1, e_2, \cdots ,\}$

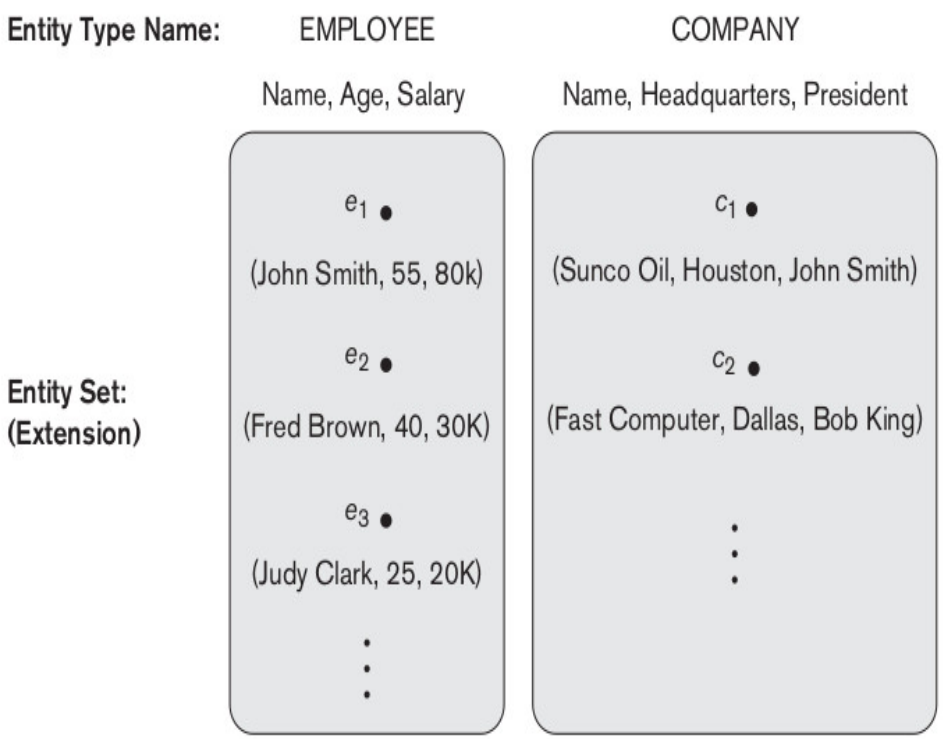
Entity Types, Entity Sets

Entity Type - Definition

A collection of entities that have same attributes. It describes schema
Example: {Employee, Company}

Entity Set - Definition

The collection of entities of a particular entity type
Example: $\{e_1, e_2, \cdots,\}$



Key Attributes

Definition

An entity type having minimal set of attributes whose values are distinct for each individual entity in an entity set.

Key Attributes

Definition

An entity type having minimal set of attributes whose values are distinct for each individual entity in an entity set.

- It stems from practicle considerations
- No two students posses identical roll numbers
- No two employees are assigned identical employee number
- Note that one or more attributes together form a key
- For example, student registers for a course, roll number and course number becomes a key
- This constraint prohibits any two entities having same value for the key attribute at the same time.

Introduction

About

- Relationships are connections among two or more entity sets
- Example: Movies and Stars two entities
- They are *Related* by Starts-in that connects Movies and Stars
- Students and Courses are two entities
- They are *Related* by Credits that connects Students and Courses

Notation

Introduction

- Entity sets are represented by rectangles
- Attributes are represented by ovals
- Relationships are represented by diamonds

Relationship Types

Definition

A relationship type R among n entity types $E_1, E_2, \cdots, E_n$ defines a relationship set among these entity types

Relationship Types

Definition

A relationship type R among n entity types $E_1, E_2, \cdots, E_n$ defines a relationship set among these entity types

Mathematical

Relationship set $R = \{r_i = (e_1, e_2, \cdots, e_n) | e_i \in E_i \ \forall \ i = 1, 2, \cdots, n\}$

Relationship Types

Definition

A relationship type R among n entity types $E_1, E_2, \cdots, E_n$ defines a relationship set among these entity types

Mathematical

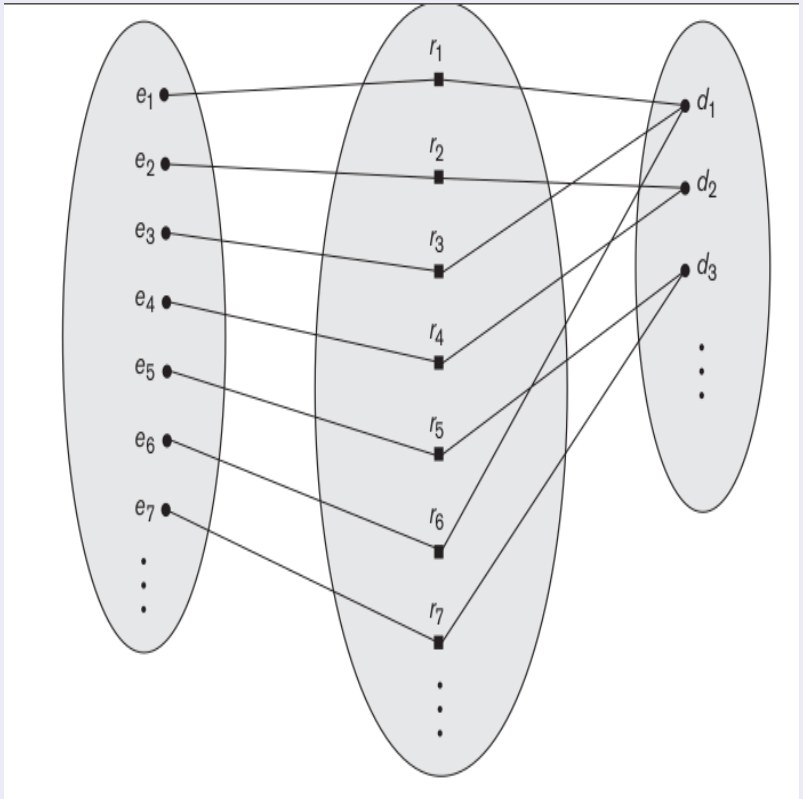
Relationship set $R = \{r_i = (e_1, e_2, \cdots, e_n) | e_i \in E_i \; \forall \; i = 1, 2, \cdots, n\}$

Alternate Definition

A relationship type $R \subseteq E_1 \times E_2 \times \cdots \times E_n$

Example

WORKS FOR



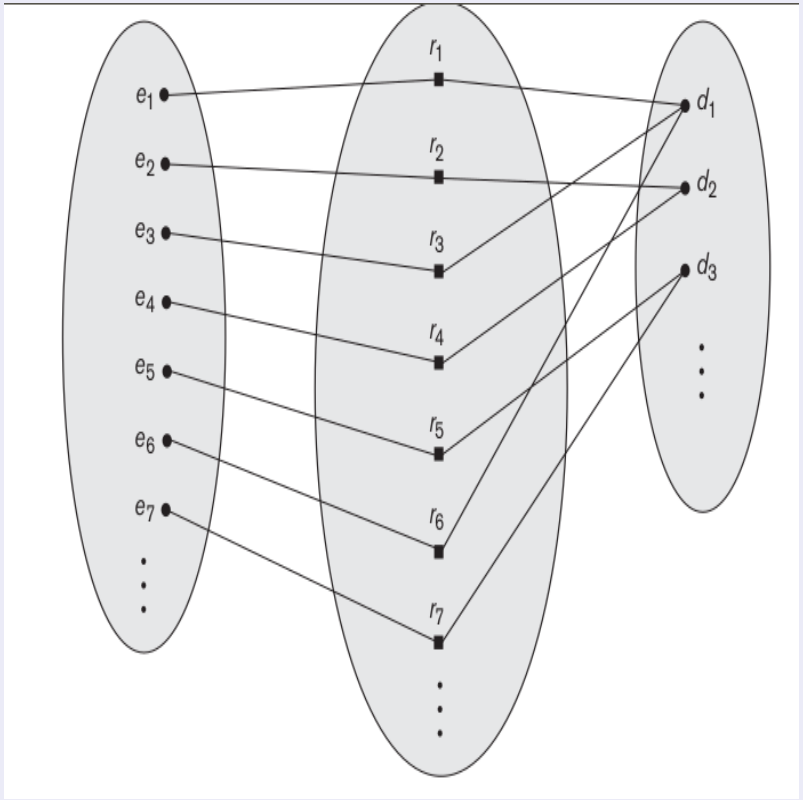
Relationship Degree

Definition

The degree of a relationship type is the number of participating entity types

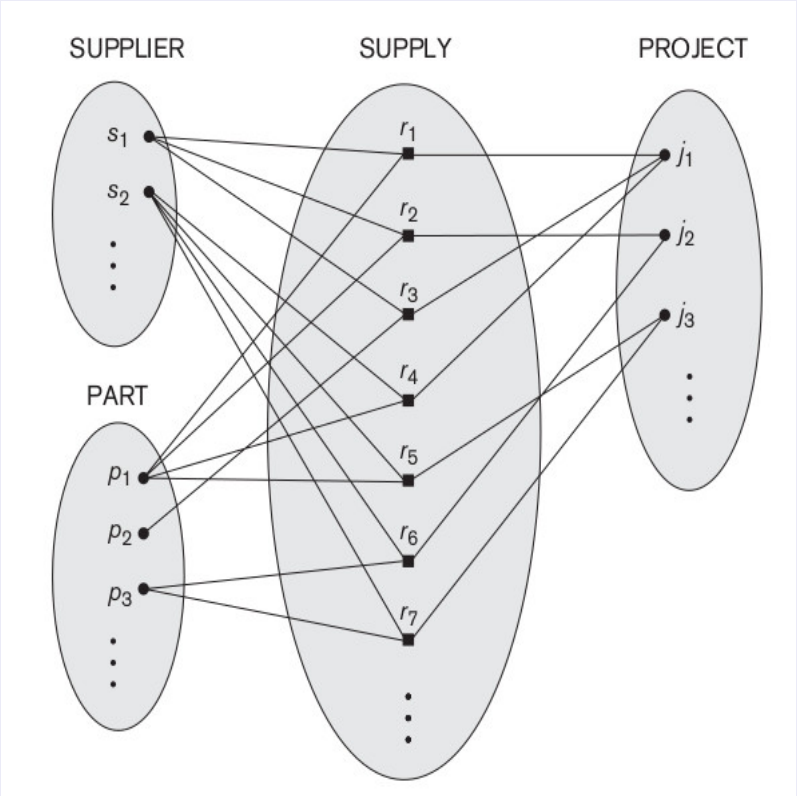
Binary Relationship

Example



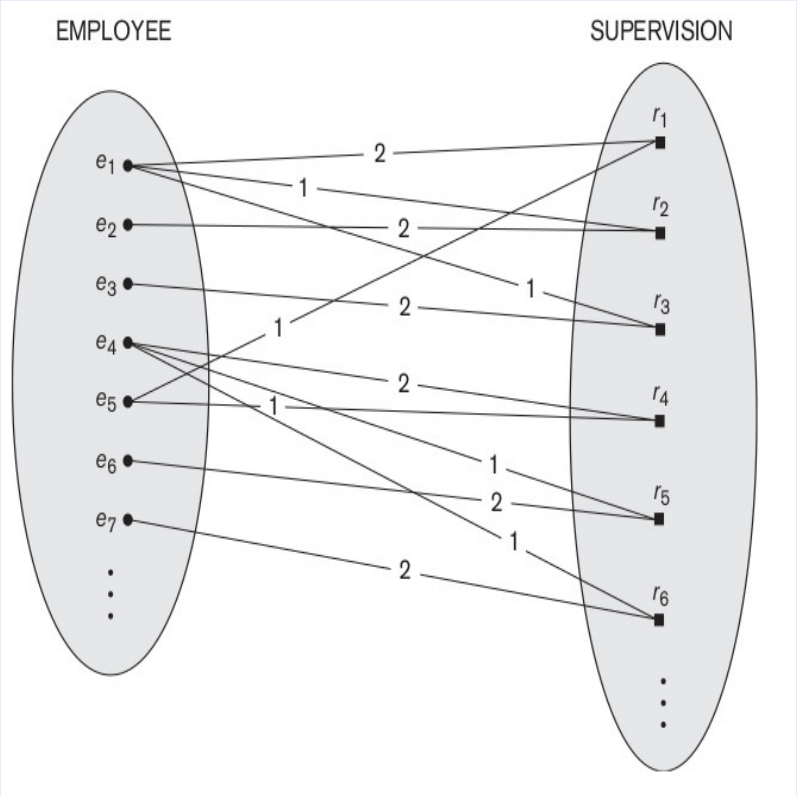
Ternary Relationship

Example



Recursive Relationship

Example



Cardinalities

Examples

One-to-One One student must belong to one department; one student One department must have one HoD;

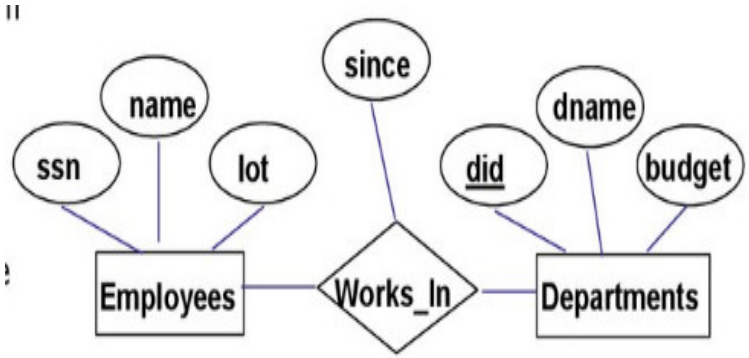
One-to-Many One department has many employees; One course can have many registered students...

Many-to-One Many transactions relate to one account number; Many employees works for one department

Many-to-Many One faculty can offer many courses; one course can be offered by many faculty;

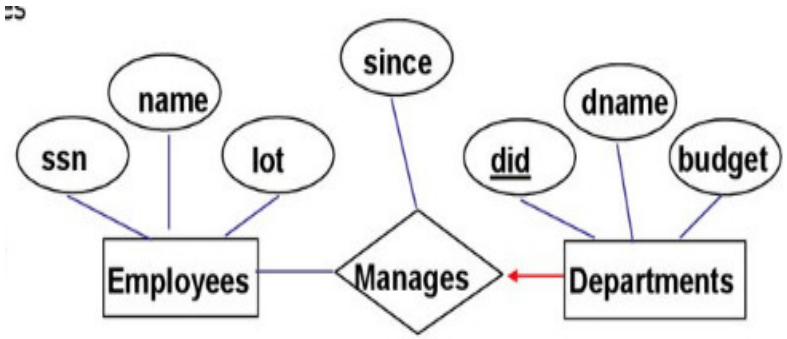
Cardinalities

Many-to-Many Many employees works for one department; one department can have many employees



Cardinalities

Many-to-One One employee manages multiple departments; Every department must have one manager.



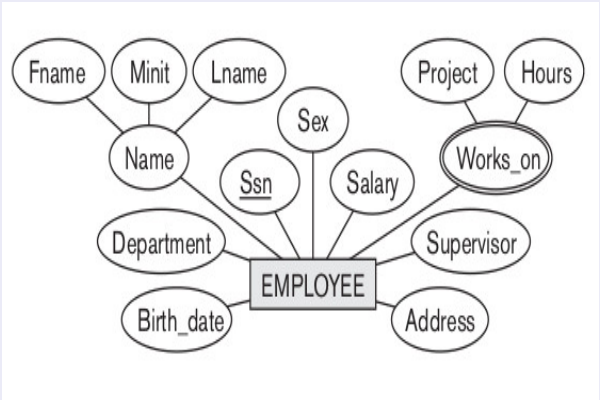
ER Design for COMPANY Database

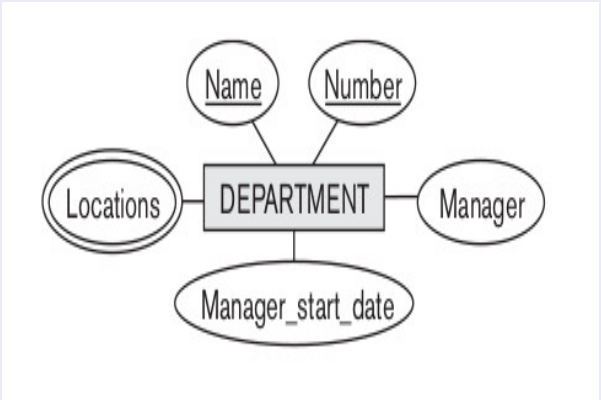
Entities

- Employee
- Department
- Project
- Dependent

Entity Attributes

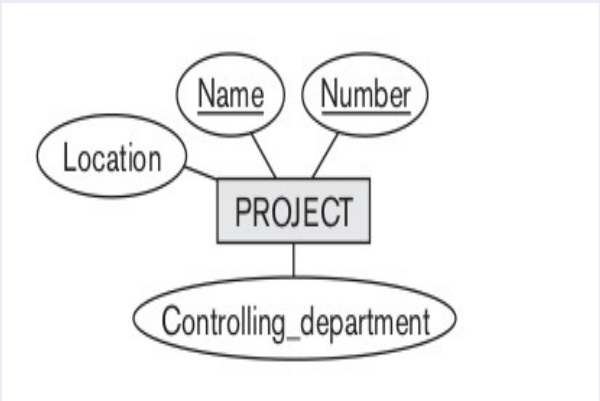
Department and Employee

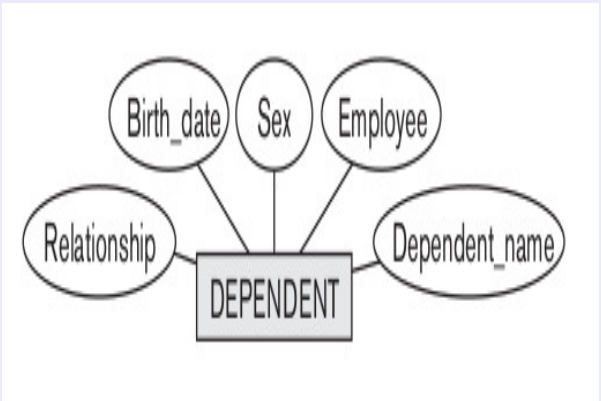




Entity Attributes

Dependent and Project





ER Design for COMPANY Database

Relations

- MANAGES: One-to-one relationship between EMPLOYEE and DEPARTMENT

ER Design for COMPANY Database

Relations

- MANAGES: One-to-one relationship between EMPLOYEE and DEPARTMENT
- WORKS_FOR: One-to-many relationship between DEPARTMENT and EMPLOYEE

ER Design for COMPANY Database

Relations

- MANAGES: One-to-one relationship between EMPLOYEE and DEPARTMENT
- WORKS_FOR: One-to-many relationship between DEPARTMENT and EMPLOYEE
- CONTROLS: One-to-many relationship between DEPARTMENT and PROJECT

ER Design for COMPANY Database

Relations

- MANAGES: One-to-one relationship between EMPLOYEE and DEPARTMENT
- WORKS_FOR: One-to-many relationship between DEPARTMENT and EMPLOYEE
- CONTROLS: One-to-many relationship between DEPARTMENT and PROJECT
- SUPERVISION: One-to-many relationship between EMPLOYEE (supervisor role) and EMPLOYEE (supervisee role)

ER Design for COMPANY Database

Relations

- MANAGES: One-to-one relationship between EMPLOYEE and DEPARTMENT
- WORKS_FOR: One-to-many relationship between DEPARTMENT and EMPLOYEE
- CONTROLS: One-to-many relationship between DEPARTMENT and PROJECT
- SUPERVISION: One-to-many relationship between EMPLOYEE (supervisor role) and EMPLOYEE (supervisee role)
- WORKS_ON: Many-to-many relationship between EMPLOYEE and PROJECT

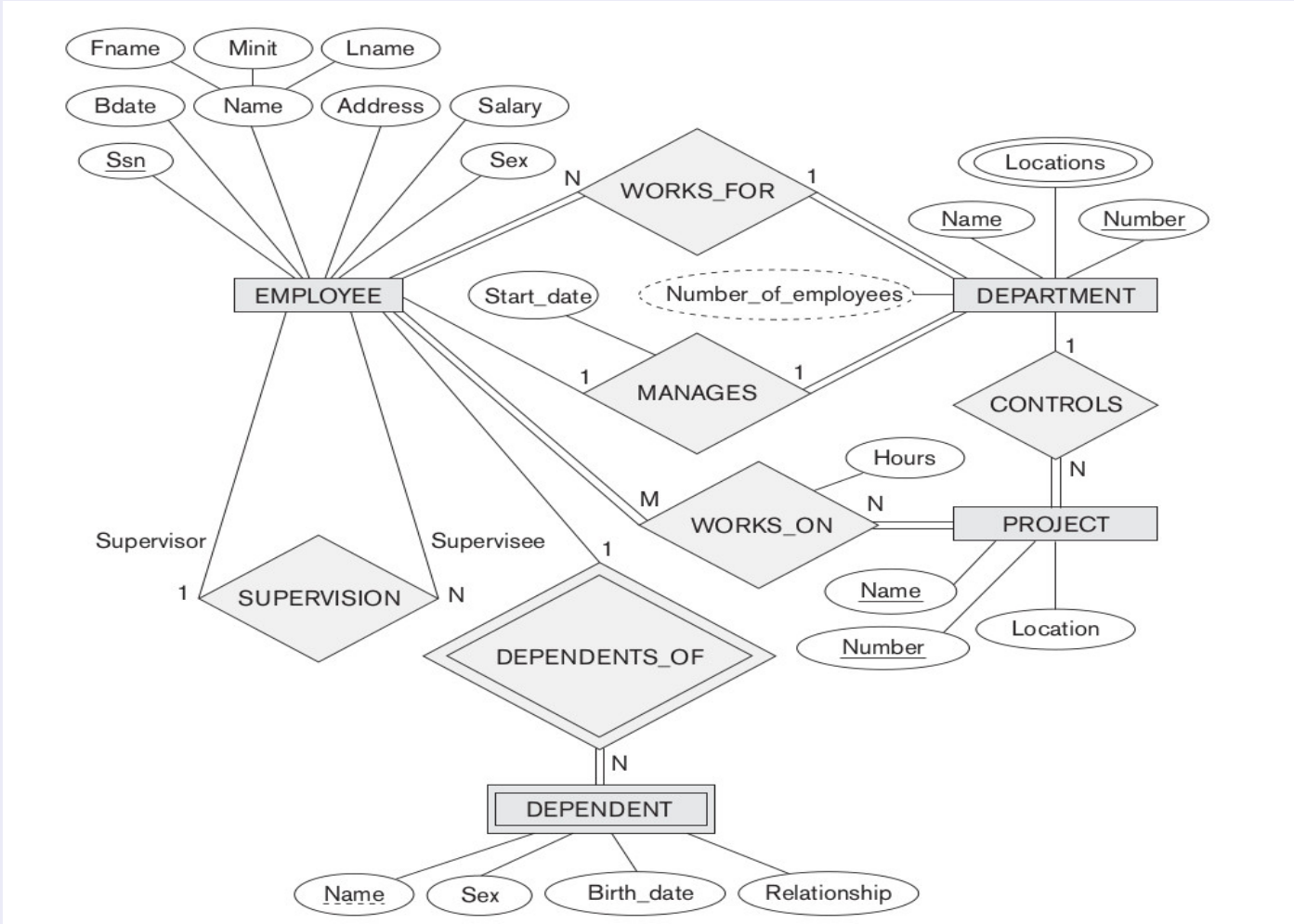
ER Design for COMPANY Database

Relations

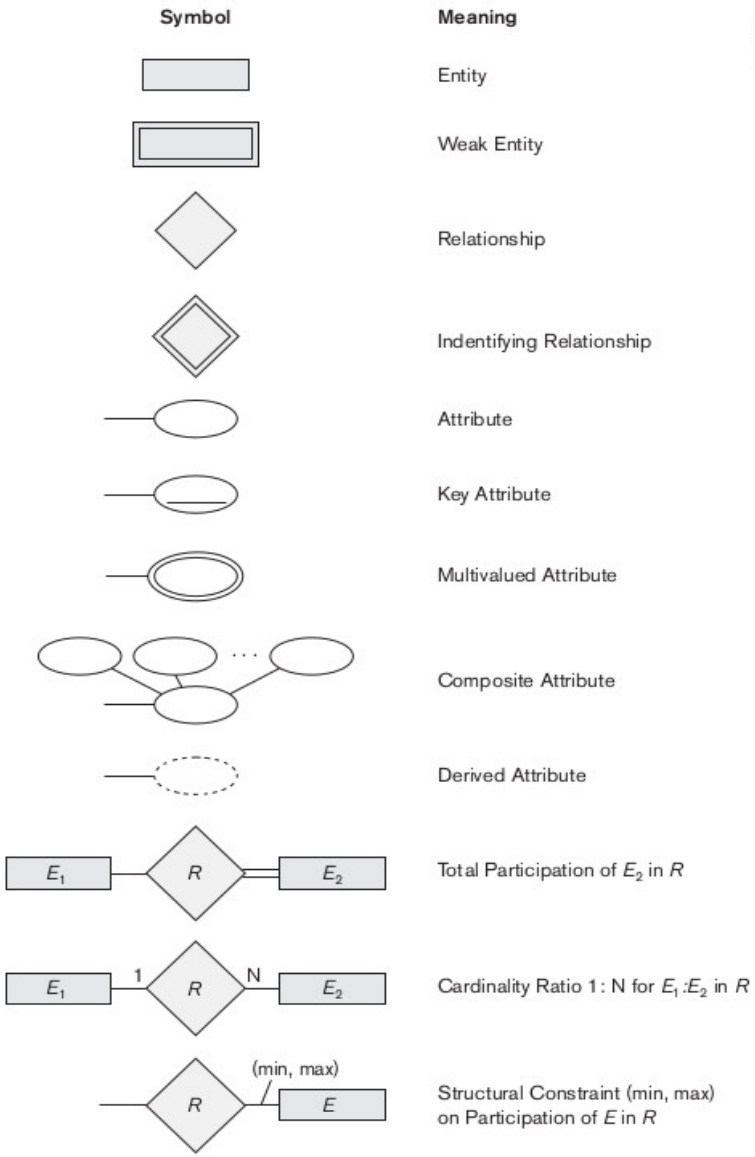
- MANAGES: One-to-one relationship between EMPLOYEE and DEPARTMENT
- WORKS_FOR: One-to-many relationship between DEPARTMENT and EMPLOYEE
- CONTROLS: One-to-many relationship between DEPARTMENT and PROJECT
- SUPERVISION: One-to-many relationship between EMPLOYEE (supervisor role) and EMPLOYEE (supervisee role)
- WORKS_ON: Many-to-many relationship between EMPLOYEE and PROJECT
- DEPENDENTS_OF One-to-many relationship between EMPLOYEE and DEPENDENT

COMPANY ER-Diagram

Complete ER diagram

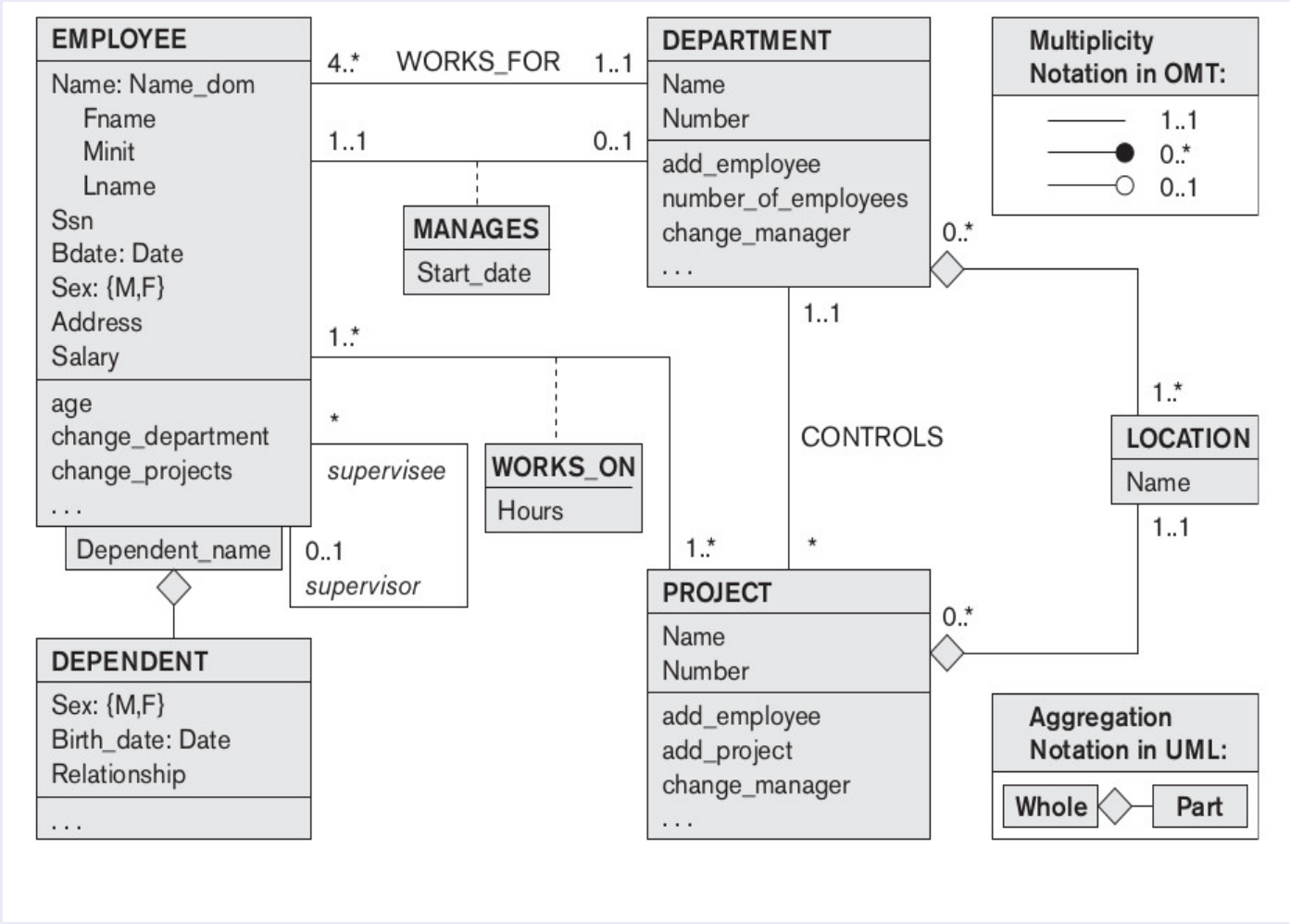


Notations


Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E1
R
E2
Total Participation of E2 in R
1
N
Cardinality Ratio 1 : N for E1 :E2 in R
(min, max)
Structural Constraint (min, max)
on Participation of E in R

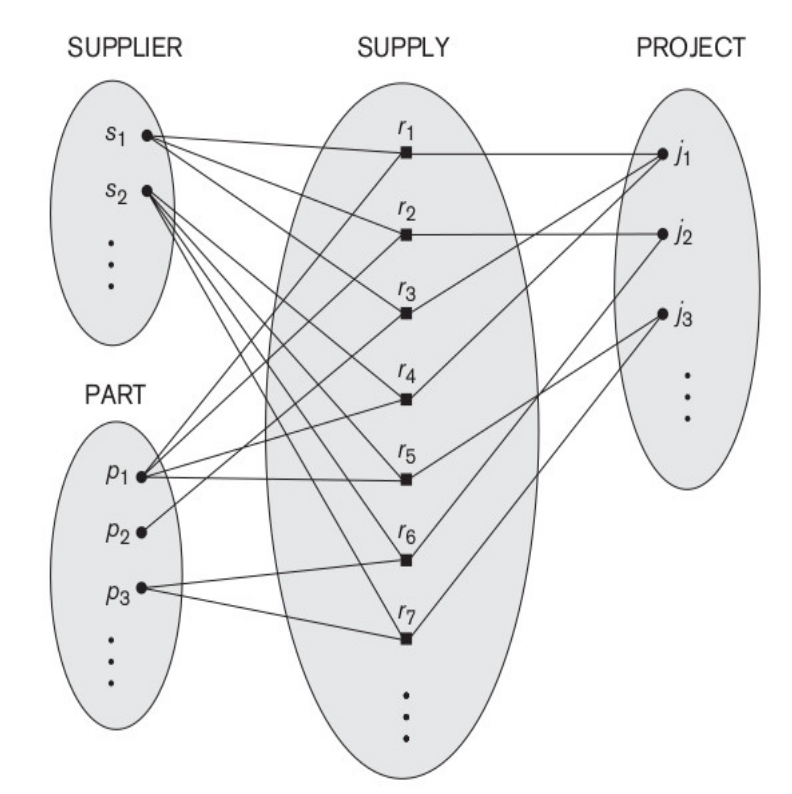
COMPANY ER-Diagram

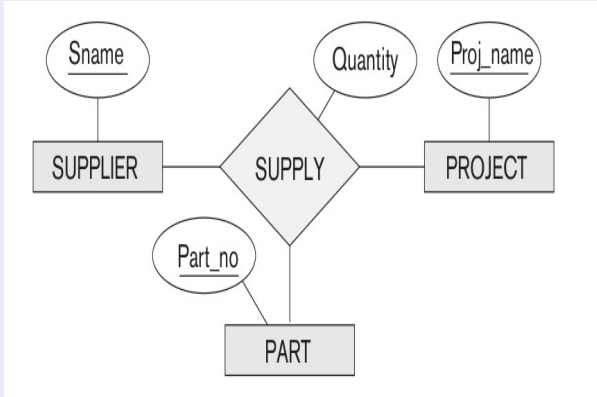
UML Notation



Ternary Relationships

Relationship set & ER Diagram





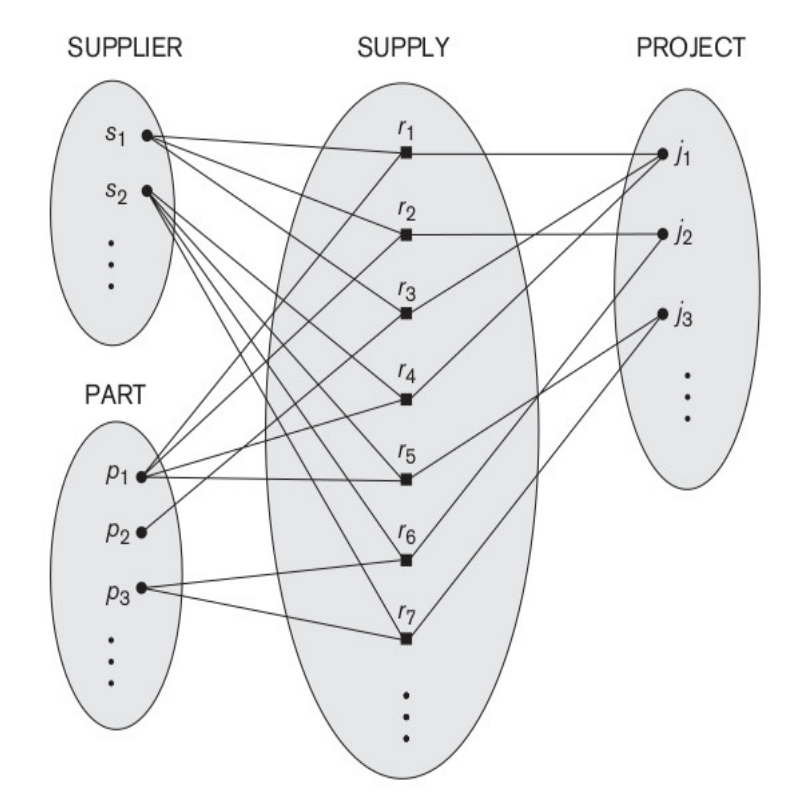
Ternary Relationships

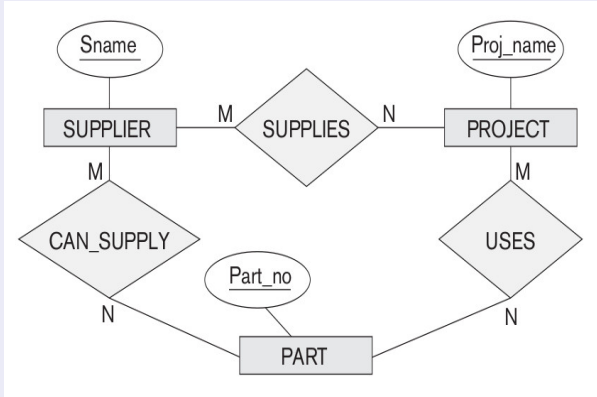
Close Look

- SUPPLY relationship set contains instances (s, j, p)
- s is a SUPPLIER
- Who SUPPLIES part p
- To PROJECT j

Ternary Relationships

Relationship set & ER Diagram





Three Binary Relationships

Close Look

- CAN_SUPPLY relationship set contains instances (s, p)
  - s is a SUPPLIER
  - Who SUPPLIES part p
- USES relationship set contains instances (j, p)
  - Project j uses part p
- SUPPLIES relationship set contains instances (s, j)
  - Supplier s supplies (*some part*) to project j
- Existence of $(s, p), (s, j), (j, p)$ do not necessarily imply that instance (s, j, p) existence in ternary relationship

Ternary Relationships and Three Binary Relationships

Constraints

- Binary relationship cannot replace ternary relationship
- Ternary relationships may be replaced with binary relationships under certain additional constraints
- Example: A particular project-part combination only one supplier will be used
- In this case any relationship instance (s, j, p) is uniquely identified by (j, p) combination

Characteristics of Relations

Characteristics

Tuples Ordering

- Relation is a set of tuples
- Elements of a set have no order among them
- Relation is not sensitive to the ordering of the tuples
- Tuple ordering is not part of relation definition

Within Tuples Ordering of values

- Relation of $n - tuple$ is an ordered list of n values
- So the ordering of values in a tuple and hence of attributes in a schema is important
- However, the order of attributes and their values is not that important
- As long as the correspondence between attributes and values is maintained

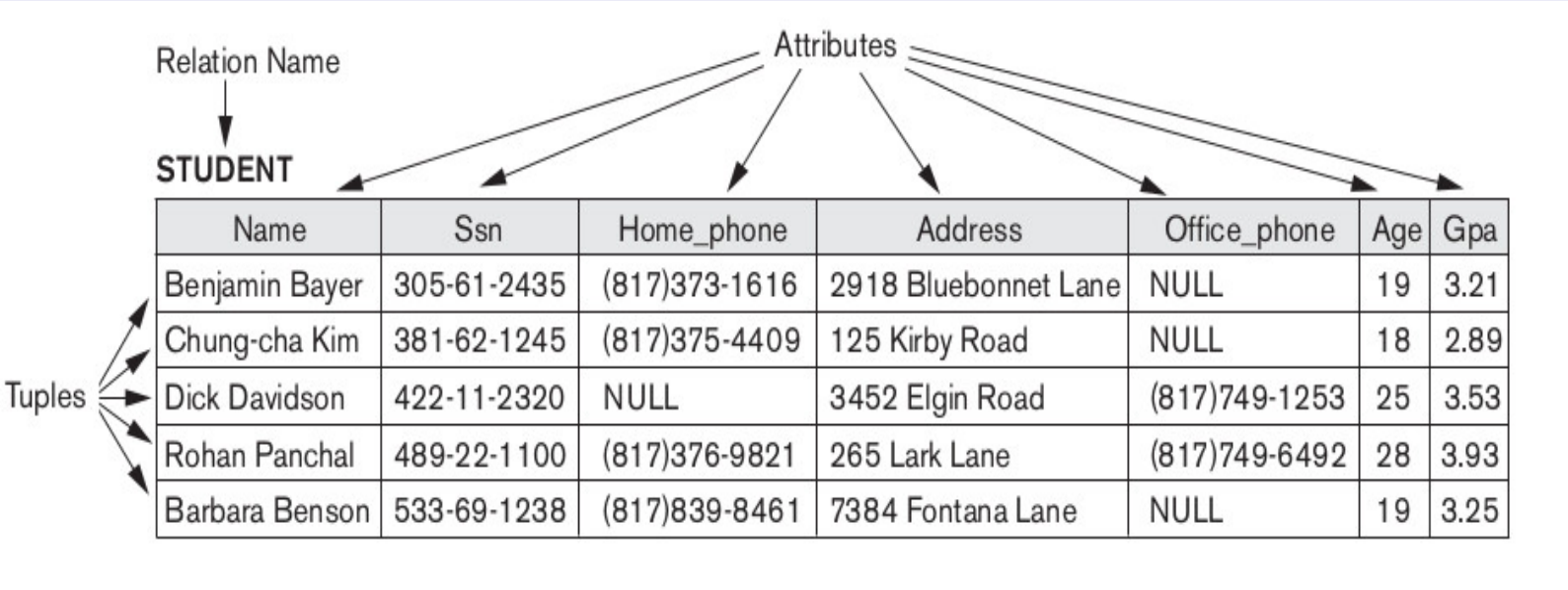
Database

Informal definitions

- Database is represented as collection of relations (or entities)
- Each relation resembles a table
- Table contains rows and columns
- Each row represent a collection of related data values
- Every column stand for attributes of the entities
- A row represents a fact that correspond to a real-world entity or a relationship

Relation

Illustrative Figure



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Notations

Relational Model Terminology

Row	is a tuple or a record
Colum header	is an attribute
Table	is a relation or an entity
Data type	corresponding to each column - is the domain

Schemas, Instances, & Database State

About

- Description of database is different from database itself
- The description of a database is called database schema
- Schema is specified during database design
- Schema's do not change frequently as opposed to data

Schema - Example

Examples

Student(| Name | Student_number | Class | Major |)

Schema - Example

Examples

Student(| Name | Student_number | Class | Major |)

Examples

Course(| Name | Number | Credit_hours | Department |)

Schema - Example

Examples

Student(| Name | Student_number | Class | Major |)

Examples

Course(| Name | Number | Credit_hours | Department |)

Examples

Prerequisite(| Course_number | Prerequisite_numbe |)

Schema - Example

Examples

Student(| Name | Student_number | Class | Major |)

Examples

Course(| Name | Number | Credit_hours | Department |)

Examples

Prerequisite(| Course_number | Prerequisite_numbe |)

Examples

Section(| section_id | course_number | semester | year | instructor |)

Schema - Example

Examples

Student(| Name | Student_number | Class | Major |)

Examples

Course(| Name | Number | Credit_hours | Department |)

Examples

Prerequisite(| Course_number | Prerequisite_numbe |)

Examples

Section(| section_id | course_number | semester | year | instructor |)

Examples

Grade_Report(| student_number | section_id | grade |)

Schema - Four Entities

EMPLOYEE(id: integer, dob: date, fname: string, minit: string, lname: string, address: string, salary: float, gender: string)

Schema - Four Entities

EMPLOYEE(id: integer, dob: date, fname: string, minit: string, lname: string, address: string, salary: float, gender: string)

DEPARTMENT(name: string, number: integer, location: string, no_of_employees: int)

Schema - Four Entities

EMPLOYEE(id: integer, dob: date, fname: string, minit: string, lname: string, address: string, salary: float, gender: string)
DEPARTMENT(name: string, number: integer, location: string, no_of_employees: int)
PROJECT(name: string, number: integer, location: string)

Schema - Four Entities

EMPLOYEE(id: integer, dob: date, fname: string, minit: string, lname: string, address: string, salary: float, gender: string)
DEPARTMENT(name: string, number: integer, location: string, no_of_employees: int)
PROJECT(name: string, number: integer, location: string)
DEPENDENTS(name: string, gender: string, dob: date, relationship: string)

Schema - Six Relations

MANAGES(id: string, name: string, number: integer, start_date: date)

Schema - Six Relations

MANAGES(id: string, name: string, number: integer, start_date: date)

WORKS_FOR(id: string, name: string, number: integer)

Schema - Six Relations

MANAGES(id: string, name: string, number: integer, start_date: date)

WORKS_FOR(id: string, name: string, number: integer)

CONTROLS(dname: string, dnumber: integer, pname: string, pnumber: integer)

Schema - Six Relations

MANAGES(id: string, name: string, number: integer, start_date: date)

WORKS_FOR(id: string, name: string, number: integer)

CONTROLS(dname: string, dnumber: integer, pname: string, pnumber: integer)

WORKS_ON(id: integer, pname: string, pnumber: integer, hours: integer)

Schema - Six Relations

MANAGES(id: string, name: string, number: integer, start_date: date)

WORKS_FOR(id: string, name: string, number: integer)

CONTROLS(dname: string, dnumber: integer, pname: string, pnumber: integer)

WORKS_ON(id: integer, pname: string, pnumber: integer, hours: integer)

SUPERVISION(supervisor_id: integer, supervisee_id: integer)

Schema - Six Relations

MANAGES(id: string, name: string, number: integer, start_date: date)
WORKS_FOR(id: string, name: string, number: integer)
CONTROLS(dname: string, dnumber: integer, pname: string, pnumber: integer)
WORKS_ON(id: integer, pname: string, pnumber: integer, hours: integer)
SUPERVISION(supervisor_id: integer, supervisee_id: integer)
DEPENDENTS_OF(id: integer, dependent_name: string)

Schemas and databases

About

- When we define a new database, we specify its database schema only to the DBMS
- Immediately after this database state is empty
- We get to initial state when database is populated with some data.
- Every update operation leads to a different database state
- At any point database has a *current state*
- DBMS is responsible for ensuring that every state is a valid state

Database - at a particular moment of time

database state or snapshot or set of occurances or instances

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Classification of Constraints

Classification

Key attribute or set of attributes that uniquely identify an entity within its entity set.

- Candidate key
- Primary key

Single value a requirement that the value in a certain context be unique

Referential integrity requirements that a value referred to by some object actually exists

Domain require the value of an attribute must be draw from a specific set of values

Default require the value to take a predefined value

Covering determine whether the entities in the subclass include all of the entities in the superclass.

Participation entity participation in a relationship

Key Constraints

Key Definition

A minimal set of attributes whose values identify an entity in the set.

Key Constraints

Key Definition

A minimal set of attributes whose values identify an entity in the set.

Candidate Keys

- An entity may have more than one set of attributes that satisfy the key definition.
- All such sets qualify for the key
- One of them is preferred or chosen as primary key

Key Constraints

Key Definition

A minimal set of attributes whose values identify an entity in the set.

Candidate Keys

- An entity may have more than one set of attributes that satisfy the key definition.
- All such sets qualify for the key
- One of them is preferred or chosen as primary key

Candidate Keys

Student: Roll number, phone number

Department: Name, department id

Single Value Constraints

Single Values

- Attribute requirement that the value in a certain context be unique
- Key constraint satisfies this requirement
- In addition, Relationships many-to-one or one-to-many also are the source of generation of this requirement

Referential Integrity Constraint

cid	grade	student_id
101	AB	53831
203	BC	53832
112	AB	53650
105	BB	53666

sid	name	login	age	spi
50000	Dave	dave	19	7.3
53666	Jones	jones	18	7.4
53688	Smith	smith	18	7.7
53650	Smith	smith	19	7.8
53831	Madan	madan	18	7.5
53832	Gaurav	gaurav	18	7.6

Referential Integrity Constraint

Example

- `student_id` attribute refers to sid of student table
- This means only students who are present in the student table are allowed to be present in the grade table
- Student who is not present in the student table is not allowed to be part of grade table
- Example: Inserting: (55555, 104, AA) into grade table is not permitted
- As 55555 is not present in the student table

Referential Integrity Constraint

Deletion

- Deletion of student with id 53666 not only affects student table but also grades table
- Possible scenarios of deletion
  - Delete all from grade which references 53666
  - Disallow the deletion of the student row 53666
  - Set the `student_id` to some default value
  - Set the `student_id` to null value.
  - However primary keys cannot assume null values and hence this is not a feasible option

Referential Integrity Constraint

Update

- Update of student with id 53666 to 53777 affects student table but also grades table
- Possible scenarios of update are identical to that of deletion
- Referential integrity or foreign key performs existential check
- Whether the entity exists in the entity set or not

Domain Constraint

Domain

- Some attributes do not assume all permissible domain values
- Example: age attribute specified as integer
- Do not take all values of integer domain
- Assumes some values between [0, 100] or [20, 70] depending on application
- Constraining the values that an attribute takes is the domain constraint