

Database Management Systems

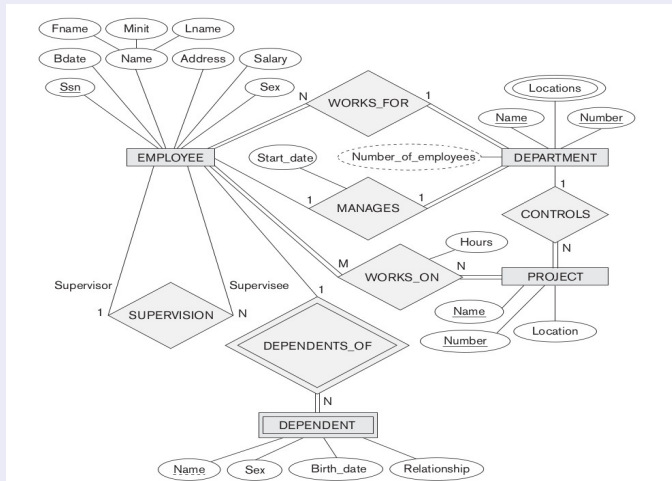
Vijaya Saradhi

IIT Guwahati













Thu, 23rd Jan 2020

COMPANY ER-Diagram

Complete ER diagram

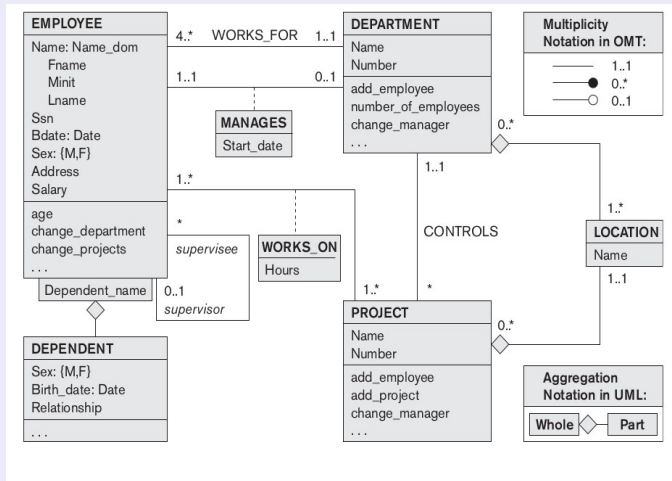


Notations

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1 : N for $E_1 : E_2$ in R
	Structural Constraint (min, max) on Participation of E in R

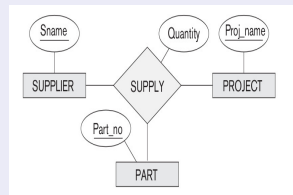
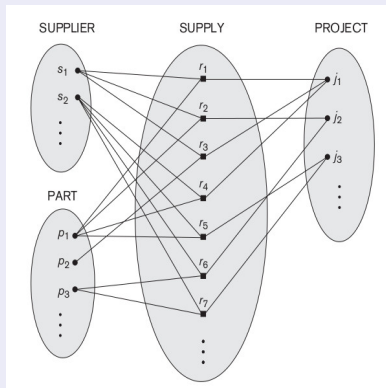
COMPANY ER-Diagram

UML Notation



Ternary Relationships

Relationship set & ER Diagram



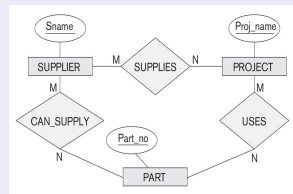
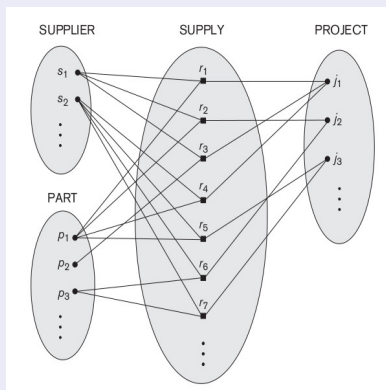
Ternary Relationships

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

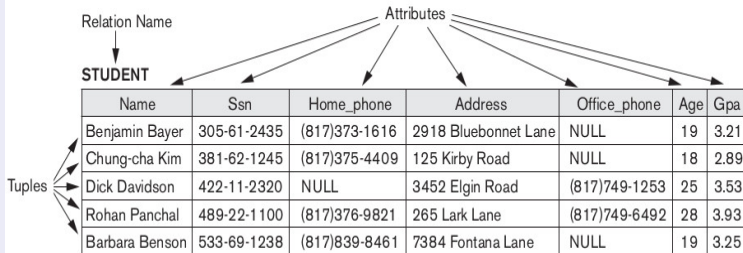
Database

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



Notations

Relational Model Terminology

Row is a tuple or a record

Column 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 occurrences 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
- Uniqueness
- NOT NULL

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

Cardinality A relation to adhere to the specified cardinality limits

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

Checks for existence of entity

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

Students				
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

Primary Key - 01

Basic Technical Criteria - A practitioner's perspective

Natural vs Surrogate keys

Uniqueness Unambiguously specify the row corresponds to a particular real world entity instance

Applicability Registration number as primary key to identify vehicles and we need to keep track of unregistered vehicles
These two are essential criteria for primary key

Minimality Should not include more attributes than are necessary for uniqueness

Stability Primary key values should not change over time

Primary Key - 02

Basic Technical Criteria - A practitioner's perspective

Natural finger print, iris, face, etc.

Surrogate IDs, row numbers, etc.

May refer to same entity Two row may be distinct but refers to same real world entity

Uniqueness

Basic Technical Criteria

- Three ways of building uniqueness

Intrinsically unique as a result of nature of the real world. Example, finger print, signature, iris etc.

Designer established A mechanism for allocation of key values.

Ensures that no value is allocated more than once - surrogate keys

Some one else established the key values. Some's elses surrogate keys became popular; limited by the present application scope

Example Vehicle registration number allocated uniquely **within a state**. If the application demands to use it across states, **State of registration** must be added to the primary key

Applicability

Basic Technical Criteria

- **Special cases** - Do you ever encounter a student without roll number (exchange student for one semester with only thesis credits?) Flights without a flight number?
 - Setting up mechanism to allocate values for these cases
 - Excluding them from entity definition
 - Rejecting the proposed primary key
- **Data Unavailability at time of entry** Example: (Customer Number, Departure Date) of travel itinerary. **Will we always know about the departure date at the first time of recording information?**
- **Broadening of Scope** You have developed application to sell books; Using ISBN as primary key. But you have decided to sell stationary items as well; ISBN is no more a primary key!

Minimality

Basic Technical Criteria

- Example, (Roll Number, Student Name) though Roll Number is sufficient
- Issues
 - Primary key appears as foreign key in `grades` file
 - Waste of space
 - Overheads involved in updating
 - Overheads involved in deletion
 - It is possible to say: (1234, rahul); (1234, amit)

Stability

Basic Technical Criteria

Key values change frequently? - Technical perspective

- Hurdle is that primary keys are used as foreign keys. When primary key value changes, so should be the foreign key value
- Another **practicle issue** is that, even if they are changed, data archived on papers, microfiche, tape are unchanged!

Stability

Basic Technical Criteria

Key values change frequently? - Reflecting meaning (identity) in real world

- Policy number as primary key
- How many new policies issued in this month
- Changing every **non-key** attribute value, the meaning is retained
- Similarly, keeping **ALL non-key** attribute values constant and changing primary key.
- Changes in some non-key attribute result in policy number change it should be bussiness driven requirement!

Surrogate Keys

Issues/Discussion

- Assume machine generates the identifiers
- If application number is generated automatically
- A candidate can apply more than once and get two or more application numbers
- Leads to violation of **singularity** - a physical real world object should have only one primary key value
- Two or more database are merged leads to violatoin of the above

Subtypes and Surrogate Keys

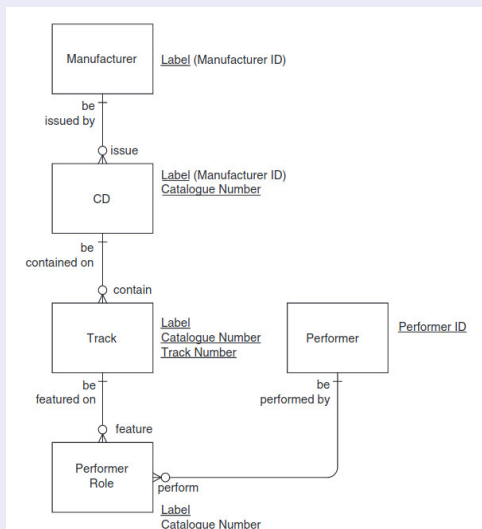
Issues/Discussion

Criminal Cases		Civil Cases	
Case No	Schedule Date	Case No	Schedule Date
000001	02-Jan-1993	000001	02-Jan-1993
000002	03-Jan-1993	000002	03-Jan-1993
000003	04-Jan-1993	000003	04-Jan-1993
000004	05-Jan-1993	000004	05-Jan-1993

Criminal Cases		Civil Cases	
Case No	Schedule Date	Case No	Schedule Date
000001	02-Jan-1993	000002	02-Jan-1993
000003	03-Jan-1993	000004	03-Jan-1993
000006	04-Jan-1993	000005	04-Jan-1993
000008	05-Jan-1993	000007	05-Jan-1993

Structured Keys

Keys made up of more than one attribute



Weak Entity Sets - 01

Definition

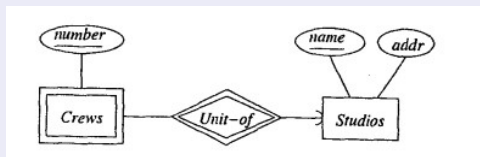
An entity set's key is composed of attributes some or all of which belong to another entity set.

Causes

- When entity sets fall into a hierarchy based on classifications unrelated to the "isa hierarchy"
- Connecting entity sets as a way to eliminate multiway (d-ary) relationships. Entity sets have no attributes of their own

Weak Entity Sets - 02

Example



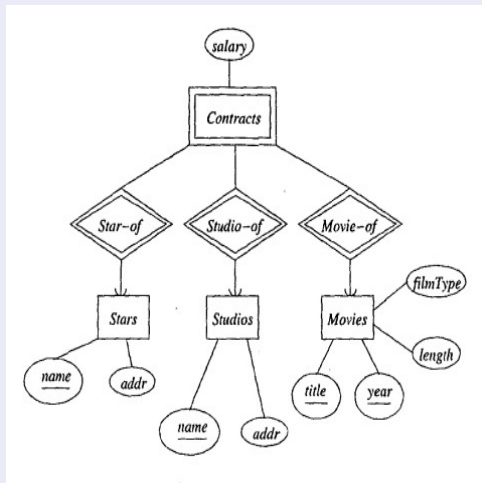
Weak Entity Sets - 03

Example

Studio		Crew
Studio name	Studio address	Number
ABC	123, HYD	1
ABC	123, HYD	2
ABC	123, HYD	3
EFG	987, BOM	1
EFG	987, BOM	2
EFG	987, BOM	3

Weak Entity Sets - 04

Example



Weak Entity Sets - 05

Requirements

- Cannot obtain key attributes for an entity set
- If E is a weak entity set then its key consists of
- Zero or more of its own attributes
- Key attributes from entity sets that are reached by certain **many-to-one** relationships from E to other entity sets.

Weak Entity Sets - 06

Requirements

- Let $R : E \rightarrow F$
- R must be a binary, many-to-one relationship from E to F
- R must have referential integrity from E to F
- Attributes of F supplies for the key of E must be key attributes of F
- If F itself is weak, then some or all of the key attributes of F supplied to E will be key attributes of one or more entity sets of G to which F is connected