Formal Relational Query Language

- Relational Algebra
  - Procedural and Algebra based
- Tuple Relational Calculus
  - Non-Procedural and Predicate Calculus based
- Domain Relational Calculus
  - Non-Procedural and Predicate Calculus based

Relational Algebra

- Created by Edgar F Codd at IBM in 1970
- Procedural language
- Six basic operators
  - select: σ
  - project: Π
  - union: $\cup$
  - set difference: $-$
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result

Select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

Each terms is one of:

$$< attribute > op < attribute > or < constant >$$

where op is one of: $=, \neq, >, \geq . < . \leq$

- Example of selection:

$$\sigma_{dept\_name = 'Physics'}(instructor)$$

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

A	B	C	D
α	α	1	7
β	β	23	10

$\sigma_{A=B \wedge D > 5}(r)$

Project Operation

- Notation: $\Pi_{A_1, A_2, \ldots A_k}$ (r)
  where A_1, A_2 are attribute names and r is a relation
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

$$\Pi_{ID, name, salary}(instructor)$$

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Union Operation

- Notation: $r \cup s$
- Defined as: $r \cup s = \{t | t \in r \text{ or } t \in s\}$
- For $r \cup s$ to be valid.
  a) r, s must have the *same* arity (same number of attributes)
  b) The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)
  c) Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course\_id}(\sigma_{semester=\text{“}Fall\text{”} \wedge year=2009}(section)) \cup \Pi_{course\_id}(\sigma_{semester=\text{“}Spring\text{”} \wedge year=2010}(section))$$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

r ∪ s

Difference Operation

- Notation $r - s$
- Defined as: $r - s = \{t | t \in r \text{ and } t \notin s\}$
- Set differences must be taken between **compatible** relations
  - r and s must have the same arity
  - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \land year=2009}(section)) -$$
$$\Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \land year=2010}(section))$$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

$r - s$

Intersection Operation

- Notation: $r \cap s$
- Defined as:

$$r \cap s = \{t \mid t \in r \text{ and } t \in s\}$$

- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s \;=\; r - (r - s)$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

$r \cap s$

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t\ q \mid t \in r \text{ and } q \in s\}$$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \phi$)
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used

$\bowtie_\theta (r, s)$

$= \sigma_\theta (r \times s)$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

r x s

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

  $\rho_X(E)$

  returns the expression E under the name X
- If a relational-algebra expression E has arity n, then

  $$\rho_{X(A_1,A_2,\cdots,A_n)}(E)$$

  returns the result of expression E under the name X, and with the attributes renamed to

  $$A_1, A_2, \ldots, A_n$$

  .

Division Operation

- The division operation is applied to two relations

X ⊆ Z

- R(Z) $\div$ S(X), where X subset Z. Let Y = Z - X (and hence Z = X $\cup$ Y); that is, let Y be the set of attributes of R that are not attributes of S
- The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with
  - $t_R[X] = t_s$ for every tuple t_s in S.
- For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with every tuple in S
- Division is a derived operation and can be expressed in terms of other operations
- $r \div s \equiv \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

remember jo common attrib hai vo NAHI aayega result mein

A	B
a1	b1
a4	b4

B
b2
b3

r ÷ s1

A
a1
a4

Division Examples

- R

Lecturer	Module
Brown	Compilers
Brown	Databases
Green	Prolog
Green	Databases
Lewis	Prolog
Smith	Databases

S

Subject
Prolog

R | S

Lecturer
Green
Lewis

Division Examples (2)

- R

Lecturer	Module
Brown	Compilers
Brown	Databases
Green	Prolog
Green	Databases
Lewis	Prolog
Smith	Databases

S

Subject
Databases
Prolog

R | S

Lecturer
Green

Division Example (4)

- Relations r, s:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

r

B
1
2

s

- $r \div s$:

A
α
β

e.g. A is customer name
B is branch-name
1 and 2 here show two specific branch-names
(Find customers who have an account in all branches of the bank)

Division Example (5)

- Relations r, s:

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

s

- $r \div s$:

A	B	C
α	a	γ
γ	a	γ

e.g. Students who have taken both "a" and "b" courses, with instructor "1"

(Find students who have taken all courses given by instructor 1)

Source: db.fcngroup.nl/silberslides/Divsion

Predicate

- Consider the statement, "x is greater than 3". It has two parts. The first part, the variable x, is the subject of the statement. The second part, "is greater than 3", is the predicate. It refers to a property that the subject of the statement can have.
- The statement "x is greater than 3" can be denoted by $P(x)$ where P denotes the predicate "is greater than 3" and x is the variable.
- The predicate P can be considered as a function. It tells the truth value of the statement $P(x)$ at x. Once a value has been assigned to the variable x, the statement $P(x)$ becomes a proposition and has a *truth* or *false* value.
- In general, a statement involving n variables $x_1, x_2, x_3, \cdots, x_n$ can be denoted by $P(x_1, x_2, x_3, \cdots, x_n)$. Here P is also referred to as *n*-place predicate or a *n*-ary predicate.

Quantifiers

In predicate logic, predicates are used alongside quantifiers to express the extent to which a predicate is true over a range of elements. Using *quantifiers* to create such propositions is called *quantification*. There are two types of quantifiers:

- **Universal Quantifier**
- **Existential Quantifier**

Universal Quantifier

Universal Quantification: Mathematical statements sometimes assert that a property is true for all the values of a variable in a particular domain, called the **domain of discourse**

- Such a statement is expressed using universal quantification.
- The universal quantification of $P(x)$ for a particular domain is the proposition that asserts that $P(x)$ is *true* for all values of x in this domain
- The domain is very important here since it decides the possible values of x
- Formally, The universal quantification of $P(x)$ is the statement "$P(x)$ for all values of x in the domain".
- The notation $\forall P(x)$ denotes the universal quantification of $P(x)$. Here $\forall$ is called the universal quantifier. $\forall P(x)$ is read as "for all x $P(x)$".
- Example: Let $P(x)$ be the statement "$x + 2 > x$". What is the truth value of the statement $\forall x\ P(x)$?
  Solution: As $x + 2$ is greater than x for any real number, so $P(x) \equiv T$ for all x or $\forall x\ P(x) \equiv T$

Existential Quantifier

Existential Quantification: Some mathematical statements assert that there is an element with a certain property. Such statements are expressed by existential quantification. Existential quantification can be used to form a proposition that is true if and only if $P(x)$ is *true* for at least one value of x in the domain.

- Formally, the existential quantification of $P(x)$ is the statement "There exists an element x in the domain such that $P(x)$".
- The notation $\exists P(x)$ denotes the existential quantification of $P(x)$. Here $\exists$ is called the existential quantifier. $\exists P(x)$ is read as "There is atleast one such x such that $P(x)$"
- Example: Let $P(x)$ be the statement "$x > 5$". What is the truth value of the statement $\exists x P(x)$?
  Solution: $P(x)$ is *true* for all real numbers greater than 5and *false* for all real numbers less than 5. So $\exists x\ P(x) \equiv T$

Tuple Relational Calculus

TRC is a non-procedural query language, where each query is of the form

$$\{t \mid P(t)\}$$

where t = resulting tuples,
P(t) = known as predicate and these are the conditions that are used to fetch t.
P(t) may have various conditions logically combined with OR ($\vee$), AND ($\wedge$), NOT($\neg$).

It also uses quantifiers:
$\exists t \in r(Q(t))$ = "there exists" a tuple in t in relation r such that predicate Q(t) is true.
$\forall t \in r(Q(t))$ = Q(t) is true "for all" tuples in relation r.

- $\{P \mid \exists S \in Students\ and\ (S.CGPA > 8 \wedge P.name = S.sname \wedge P.age = S.age)\}$: returns the name and age of students with a CGPA above 8.

Predicate Calculus Formula

a) Set of attributes and constants
b) Set of comparison operators: $(e.g., <, \leq, =, \neq, >, \geq)$
c) Set of connectives: and $(\wedge)$, or $(\vee)$, not $(\neg)$
d) Implication $(\Rightarrow)$: $x \Rightarrow y$, if x if true, then y is true
   $x \Rightarrow y \equiv \neg x \vee y$
e) Set of quantifiers:
   - $\exists t \in r(Q(t)) \equiv$ "there exists" a tuple in t in relation r such that predicate $Q(t)$ is true
   - $\forall t \in r(Q(t)) \equiv Q$ is true "for all" tuples t in relation r

TRC Example

Student

Fname	Lname	Age	Course
David	Sharma	27	DBMS
Aaron	Lilly	17	JAVA
Sahil	Khan	19	Python
Sachin	Rao	20	DBMS
Varun	George	23	JAVA
Simi	Verma	22	JAVA

Q.1 Obtain the first name of students whose age is greater than 21.

Solution: all 3 are correct

$$\{t.Fname \mid Student(t) \wedge t.age > 21\}$$

$$\{t.Fname \mid t \in Student \wedge t.age > 21\}$$

$$\{t \mid \exists s \in Student(s.age > 21 \wedge t.Fname = s.Fname)\}$$

Fname
David
Varun
Simi

TRC Example (2)

Consider the relational schema
student($\underline{rollNo}$, $name$, $year$, $courseId$)
course($\underline{courseId}$, $cname$, $teacher$)

Q.2 Find out the names of all students who have taken the course name 'DBMS'.

- $\{t \mid \exists s \in student\ \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'} \wedge t.name = s.name)\}$
- $\{s.name \mid s \in student \wedge \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'})\}$

Q.3 Find out the names of all students and their rollNo who have taken the course name 'DBMS'.

- $\{s.name, s.rollNo \mid s \in student \wedge \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'})\}$
- $\{t \mid \exists s \in student\ \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'}$
$\wedge t.name = s.name \wedge t.rollNo = s.rollNo)\}$

TRC Example (3)

Consider the following relations:
Flights(*flno*, *from*, *to*, *distance*, *departs*, *arrives*)
Aircraft(*aid*, *aname*, *cruisingrange*)
Certified(*eid*, *aid*)
Employees(*eid*, *ename*, *salary*)

Q.4. Find the eids of pilots certified for Boeing aircraft.

RA

natural

$$\Pi_{eid}(\sigma_{aname='Boeing'}(Aircraft \bowtie Certified))$$

TRC

- $\{C.eid \mid C \in Certified \wedge \exists A \in Aircraft(A.aid = C.aid \wedge A.aname = \text{'Boeing'})\}$
- $\{T \mid \exists C \in Certified \exists A \in Aircraft(A.aid = C.aid \wedge A.aname = \text{'Boeing'}$
  $\wedge T.eid = C.eid)\}$

Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations
- For example, $\{t \mid \neg t \in r\}$ results in an infinite relation if the domain of any attribute of relation r is infinite
- To guard against the problem, we restrict the set of allowable expressions to safe expressions
- An expression $\{t \mid P(t)\}$ in the tuple relational calculus is *safe* if every component of t appears in one of the relations, tuples, or constants that appear in P.
  - NOTE: this is more than just a syntax condition
  - E.g. $\{t \mid t[A] = 5 \vee true\}$ is not safe — it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in P

Domain Relational Calculus

- A non-procedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{< x_1, x_2, \ldots, x_n > | P(x_1, x_2, \ldots, x_n)\}$$

  - $x_1, x_2, \ldots, x_n$ represent domain variables
  - P represents a formula similar to that of the predicate calculus

Equivalence of RA, TRC and DRC

Select Operation

R = (A, B)

Relational Algebra: $\sigma_{B=17}\ (r)$

Tuple Calculus: $\{t \mid t \in r \wedge B = 17\}$

Domain Calculus: $\{\langle a, b\rangle \mid \langle a, b\rangle \in r \wedge b = 17\}$

Source: http://www.cs.sfu.ca/CourseCentral/354/louie/Equiv_Notations.pdf

Equivalence of RA, TRC and DRC

Project Operation

R = (A, B)

Relational Algebra: $\Pi_A(r)$

Tuple Calculus: $\{t \mid \exists\, p \in r\ (t[A] = p[A])\}$

Domain Calculus: $\{\langle a \rangle \mid \exists\, b\ (\langle a, b \rangle \in r)\}$

Source: http://www.cs.sfu.ca/CourseCentral/354/louie/Equiv_Notations.pdf

Role of Abstraction

- *Disorganized Complexity* results from
  - *Storage (STM) limitations* of human brain – an individual can simultaneously comprehend of the order of seven, plus or minus two chunks of information
  - *Speed limitations* of human brain – it takes the mind about five seconds to accept a new chunk of information
- **Abstraction** provides the major tool to handle Disorganized Complexity by *chunking information*
- Ignore inessential details, deal only with the generalized, idealized model of the world

Consider: A binary number 110010101001

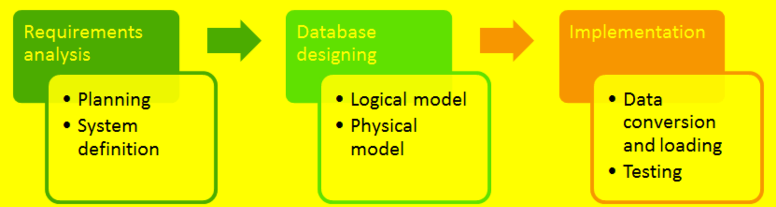
Hard to remembers. Right?

Try the octal form: (110)(010)(101)(001) $\Rightarrow$ **6251**

Or the hex form: (1100)(1010)(1001) $\Rightarrow$ **CA9**

Design Approach

- **Requirement Analysis**: Analyse the data needs of the prospective database users
  - Planning
  - System Definition
- **Database Designing**: Use a modeling framework to create abstraction of the real world
  - Logical Model
  - Physical Model
- **Implementation**
  - Data Conversion and Loading
  - Testing



Design Approach (3): Database Designing: Logical Model

- **Entity Relationship Model**
  - Models an enterprise as a collection of entities and relationships
    - *Entity*: A distinguishable "thing" or "object" in the enterprise
      - Described by a set of attributes
    - *Relationship*: An association among multiple entities
  - Represented by an *Entity-Relationship or ER Diagram*
- **Database Normalization** (Chapter 8)
  - Formalize what designs are bad, and test for them

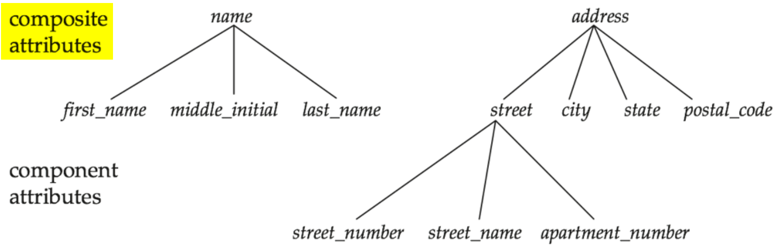
ER Model: Database Modeling

- The ER data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database
- The ER model is useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema
- The ER data model employs three basic concepts:
  - Attributes
  - Entity sets
  - Relationship sets
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically

Attributes

- An **Attribute** is a property associated with and entity / entity set. Based on the values of certain attributes, an entity can be identified uniquely
- Attribute types:
  - **Simple** and **Composite** attributes
  - **Single-valued** and **Multivalued** attributes
    - Example: Multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth
- **Domain**: Set of permitted values for each attribute

Attributes (2): Composite



Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.
  - Example:
    $instructor = (\underline{ID}, name, street, city, salary\ )$
    $course = (\underline{course\_id}, title, credits)$
- A subset of the attributes form a **primary key** of the entity set; that is, uniquely identifying each member of the set.
  - Primary key of an entity set is represented by underlining it

Relationship Sets

- A **relationship** is an association among several entities
  Example:

44553 (Peltier)	*advisor*	22222 (Einstein)
student entity	relationship set	*instructor* entity

- A **relationship set** is a mathematical relation among n $\geq$ 2 entities, each taken from entity sets

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

  where $(e_1, e_2, \ldots e_n)$ is a relationship.
  - Example: $(44553, 22222) \in \textit{advisor}$

Relationship Set (4): Degree

- Binary relationship
  - involves two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary
  - Example: *students* work on research projects under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor*, *student*, and *project*

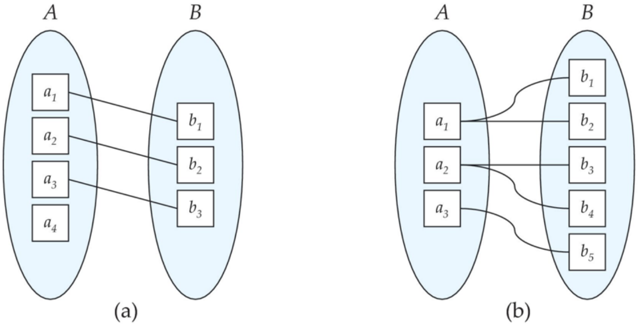
Attributes (3): Redundant

- Suppose we have entity sets:
  - *instructor*, with attributes: *ID, name, dept_name, salary*
  - *department*, with attributes: *dept_name, building, budget*
- We model the fact that each instructor has an associated department using a relationship set *inst_dept*
- The attribute *dept_name* appears in both entity sets. Since it is the primary key for the entity set *department*, it replicates information present in the relationship and is therefore redundant in the entity set *instructor* and needs to be removed
- BUT: When converting back to tables, in some cases the attribute gets reintroduced, as we will see later

Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities

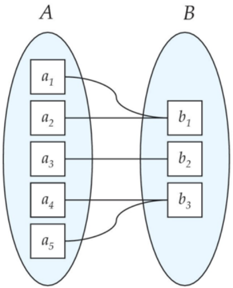


One to one

One to many

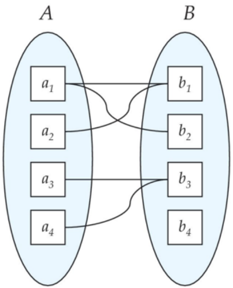
Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping Cardinalities



(a)

Many to one



(b)

Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set

Weak Entity Sets

An entity set may be of two types:

- Strong entity set
  - A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities
  - In other words, *a primary key exists for a strong entity set*
  - Primary key of a strong entity set is represented by underlining it
- Weak entity set
  - A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities
  - In other words, *a primary key does not exist for a weak entity set*
  - However, it contains a partial key called as a **discriminator**
  - Discriminator can identify a group of entities from the entity set
  - Discriminator is represented by underlining with a dashed line

Weak Entity Sets (2)

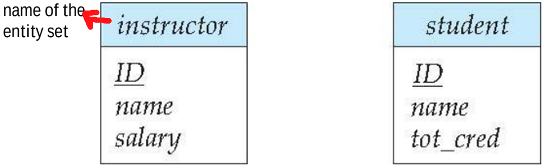
- Since a weak entity set does not have primary key, it cannot independently exist in the ER Model
- It features in the model in relationship with a strong entity set. This is called the **identifying relationship**
- Primary Key of Weak Entity Set
  - The combination of discriminator and primary key of the strong entity set makes it possible to uniquely identify all entities of the weak entity set
  - Thus, this combination serves as a primary key for the weak entity set.
  - Clearly, this primary key is not formed by the weak entity set completely.
  - **Primary Key of Weak Entity Set = Its own discriminator + Primary Key of Strong Entity Set**
- Weak entity set must have **total participation** in the identifying relationship. That is all its entities must feature in the relationship

Weak Entity Sets (3): Example

- **Strong Entity Set**: *Building*(building_no, building_name, address). building_no is its primary key
- **Weak Entity Set**: *Apartment*(door_no, floor). door_no is its discriminator as door_no alone can not identify an apartment uniquely. There may be several other buildings having the same door number
- **Relationship**: *BA* between *Building* and *Apartment*
- By **total participation** in *BA*, each apartment must be present in at least one building
- In contrast, *Building* has **partial participation** in *BA* only as there might exist some buildings which has no apartment
- **Primary Key**: To uniquely identify any apartment
    - First, building_no is required to identify the particular building
    - Second, door_no of the apartment is required to uniquely identify the apartment
- Primary key of Apartment = Primary key of Building + Its own discriminator = building_no + door_no

Entity Sets

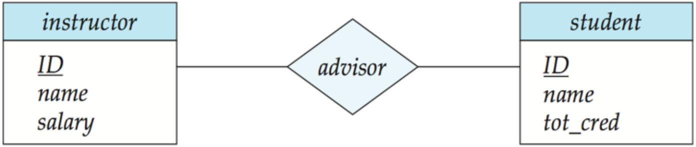
- Entities can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes are listed inside entity rectangle.
  - Underline indicates primary key attributes.



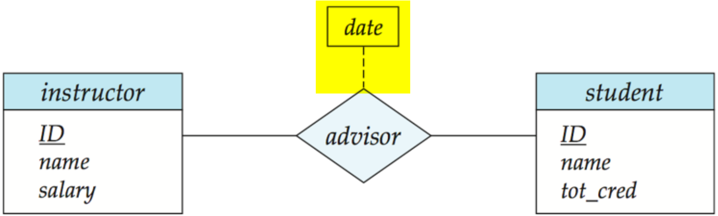
Relationship Sets

- Diamonds represent relationship sets.

(unless otherwise specified relation is b/w primary key of both)

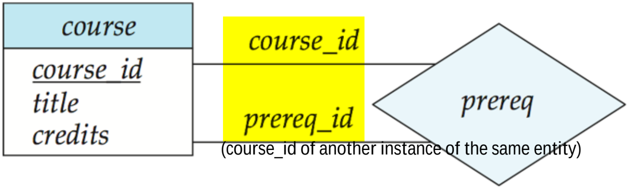


Relationship Sets with Attributes



Roles

- Entity sets of a relationship need not be distinct Each occurrence of an entity set plays a "role" in the relationship
- The labels *"course_id"* and *"prereq_id"* are called **roles**.

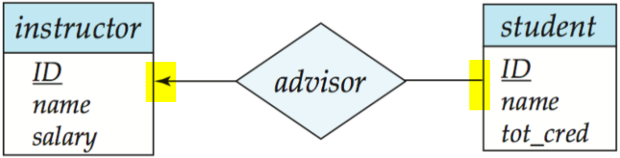


Cardinality Constraints

- We express cardinality constraints by drawing either a directed line ($\rightarrow$), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one instructor via the relationship *advisor*
  - An instructor is associated with at most one student via the relationship *advisor*

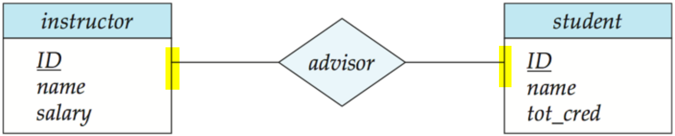
One-to-Many Relationship

- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via advisor
  - a student is associated with at most one instructor via *advisor*



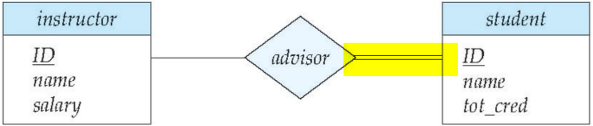
Many-to-Many Relationship

- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*



Total and Partial Participation

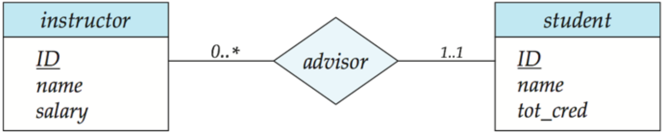
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



  - participation of *student* in *advisor* relation is total
    - every *student* must have an associated instructor
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

Notation for Expressing More Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where **l** is the minimum and **h** the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.



Instructor can advise 0 or more students.
A student must have 1 advisor; cannot have multiple advisors

Notation to Express Entity with Complex Attributes

instructor

<u>*ID*</u>
name
    first_name
    middle_initial
    last_name
address
    street
        street_number
        street_name
        apt_number
    city
    state
    zip
{ phone_number }
date_of_birth
age ( )

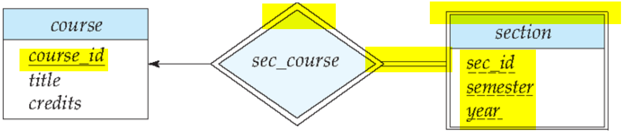
write with tab space

{} means multi valued attribute

function, needs to be computed
(here since it is derived attribute from DOB)

Expressing Weak Entity Sets

- In ER diagrams, a weak entity set is depicted via a double rectangle
- We underline the discriminator of a weak entity set with a dashed line
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond
- Primary key for *section* – *(course_id, sec_id, semester, year)*



Reduction to Relation Schema

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database
- A database which conforms to an ER diagram can be represented by a collection of schemas
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set
- Each schema has a number of columns (generally corresponding to attributes), which have unique names

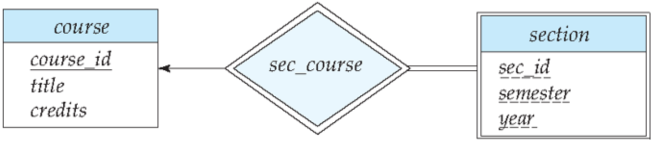
Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes

  student(ID, name, tot_cred)

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

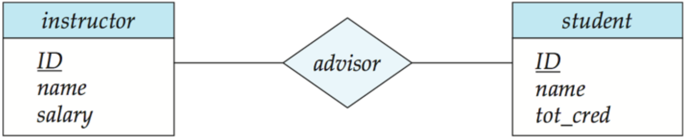
  section (course_id, sec_id, sem, year)



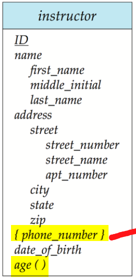
Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

$advisor = (\underline{s\_id, i\_id})$



Representation of Entity Sets with Composite Attributes



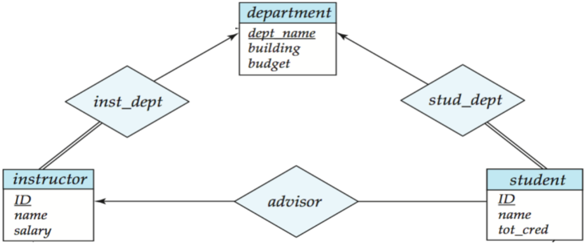
- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set **instructor** with composite attribute **name** with component attributes **first_name** and **last_name** the schema corresponding to the entity set has two attributes **name_first_name** and **name_last_name**
    - Prefix omitted if there is no ambiguity (**name_first_name** could be **first_name**)
- Ignoring multivalued attributes, extended instructor schema is
  - **instructor(ID, first_name, middle_initial, last_name, street_number, street_name, apt_number, city, state, zip_code, date_of_birth)**

Representation of Entity Sets with Multivalued Attributes

- A multivalued attribute M of an entity E is represented by a separate schema EM
- Schema EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
- Example: Multivalued attribute phone_number of *instructor* is represented by a schema:
  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema EM
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples: (22222, 456-7890) and (22222, 123-4567)

Redundancy of Schema

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*



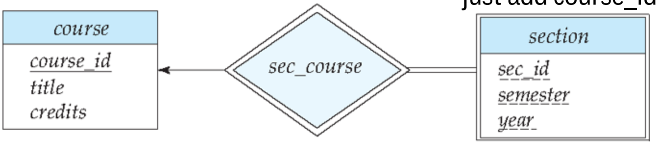
Redundancy of Schema (2)

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, an extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values

Redundancy of Schema (3)

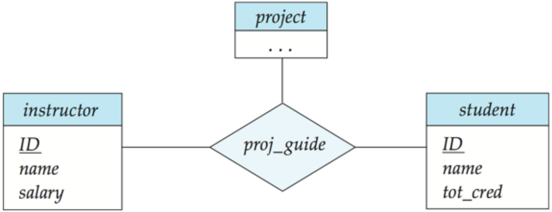
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
- Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema

just add course_id here



Non-binary Relationship Sets

- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary
- ER Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

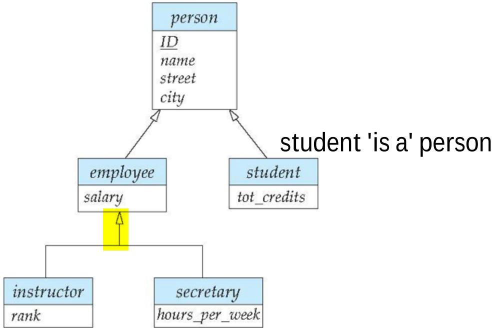
- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - For example, a ternary relationship R between A, B and C with arrows to B and C could mean
    a) Each A entity is associated with a unique entity from B and C or
    b) Each pair of entities from (A, B) is associated with a unique C entity, and each pair (A, C) is associated with a unique B
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

Specialization: ISA

- **Top-down design process**: We designate sub-groupings within an entity set that are distinctive from other entities in the set
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set
- Depicted by a *triangle* component labeled ISA (e.g., *instructor* "is a" *person*)
- **Attribute inheritance**: A lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

Specialization: ISA (2)

- **Overlapping**: *employee* and *student*
- **Disjoint**: *instructor* and *secretary*
- Total and Partial



Representing Specialization via Schema

- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary

  - Drawback: Getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

Representing Specialization as Schema (2)

- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

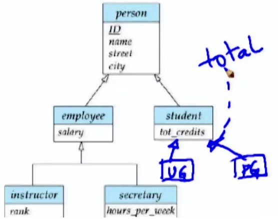
  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

Generalization

- **Bottom-up design process**: Combine a number of entity sets that share the same features into a higher-level entity set
- Specialization and generalization are simple inversions of each other; they are represented in an ER diagram in the same way
- The terms specialization and generalization are used interchangeably

Design Constraints on a Specialization / Generalization

- **Completeness constraint**: Specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets
- Partial generalization is the default. We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- The *student* generalization is total. All student entities must be either graduate or undergraduate. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total.

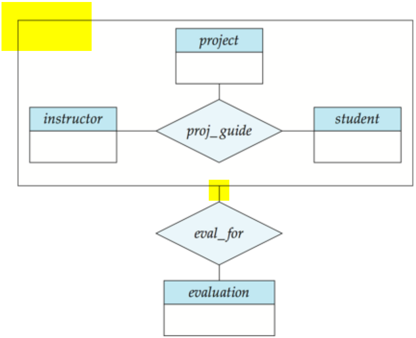


Aggregation (2)

- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - ▷ So we cannot discard the *proj_guide* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

Aggregation (3)

- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



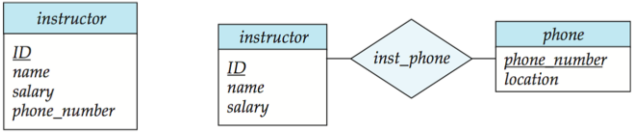
Representing Aggregation via Schema

- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema eval_for is:

    eval_for (s_ID, project_id, i_ID, evaluation_id)
  - The schema *proj_guide* is redundant

Entities vs. Attributes

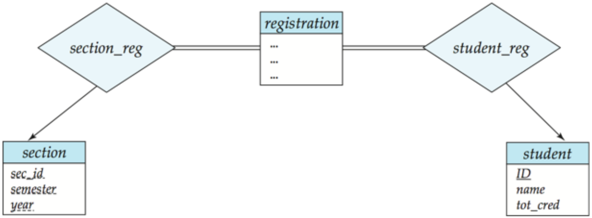
- Use of entity sets vs. attributes



- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Entities vs Relationship Sets

- **Use of entity sets vs. relationship sets**
  Possible guideline is to designate a relationship set to describe an action that occurs between entities



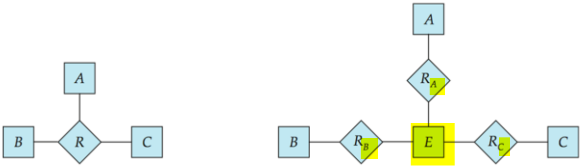
- **Placement of relationship attributes**
  For example, attribute date as attribute of advisor or as attribute of student

Binary vs Non-Binary Relationships

- Although it is possible to replace any non-binary (n-ary, for n $>$ 2) relationship set by a number of distinct binary relationship sets, a n-ary relationship set shows more clearly that several entities participate in a single relationship
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

Binary vs Non-Binary Relationships (2): Conversion

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace R between entity sets A, B and C by an entity set E, and three relationship sets:
    1. R_A, relating E and A
    2. R_B, relating E and B
    3. R_C, relating E and C
  - Create an identifying attribute for E and add any attributes of R to E
  - For each relationship (a_i, b_i, c_i) in R, create
    a) a new entity e_i in the entity set E
    b) add (e_i, a_i) to R_A
    c) add (e_i, b_i) to R_B
    d) add (e_i, c_i) to R_C

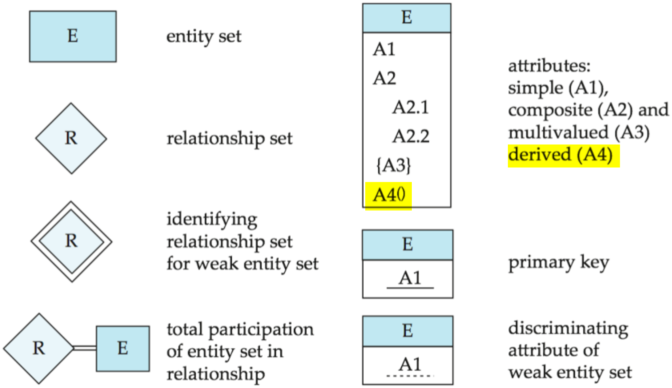


(a) (b)

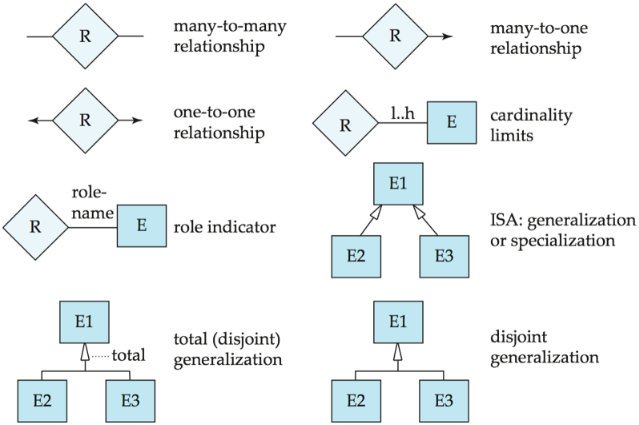
Binary vs Non-Binary Relationships (3): Conversion

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of R.
    - Exercise: *add constraints to the relationships* R_A, R_B *and* R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets —A, B and C
  - We can avoid creating an identifying attribute by making E, a weak entity set (described shortly) identified by the three relationship sets

Symbols Used in ER Notation



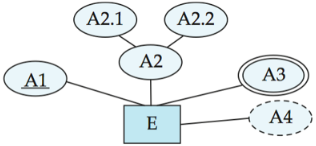
Symbols Used in ER Notation (2)



Symbols Used in ER Notation (3): Alternate

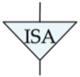
- Chen, IDE1FX,. . .

entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1

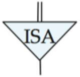


weak entity set

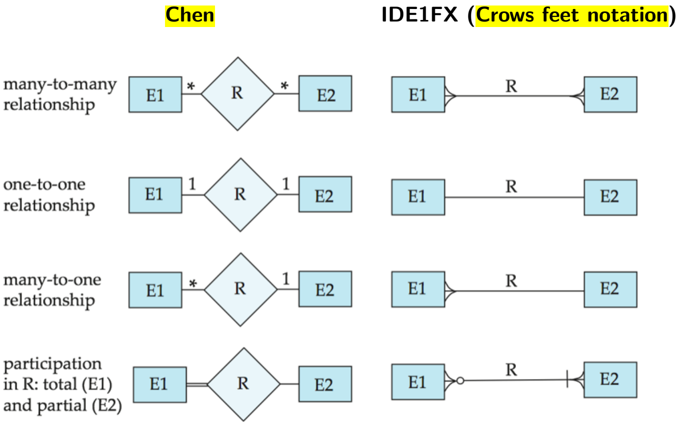
generalization



total generalization



Symbols Used in ER Notation (4): Alternates



Q2: Division Operation

IIT Madras
BSc Degree

- Find the names of employees who are working in all the departments located in 'Campus2'.

department	location
Physics	Campus1
Chemistry	Campus1
Chemistry	Campus2
Mathematics	Campus2
Biology	Campus3
Physics	Campus2
Chemistry	Campus3

DepartmentLocation

employee	department
Ishaan	Physics
Jairaj	Chemistry
Ananya	Chemistry
Barkha	Mathematics
Onkar	Mathematics
Onkar	Biology
Ayesha	Mathematics
Aditi	Chemistry
Ananya	Mathematics
Lohit	Mathematics
Ananya	Physics
Barkha	Physics
Jairaj	Physics
Onkar	Chemistry
Jairaj	Mathematics

EmployeeDepartment

Figure: Tables **DepartmentLocation** and **EmployeeDepartment**

Solution:

$T_1 \leftarrow \Pi_{department}(\sigma_{location='Campus2'}(DepartmentLocation))$

$T_2 \leftarrow EmployeeDepartment \div T_1$

Introduction

TRC is a nonprocedural query language, where each query is of the form
gives info on what data to find out but not how

$$\{t \mid P(t)\}$$

where t = resulting tuples,
P(t) = known as predicate and these are the conditions that are used to fetch t.
P(t) may have various conditions logically combined with OR ($\vee$), AND ($\wedge$), NOT($\neg$).

It also uses quantifiers:
$\exists t \in r(Q(t))$ = "there exists" a tuple in t in relation r such that predicate Q(t) is true.
$\forall t \in r(Q(t))$ = Q(t) is true "for all" tuples in relation r.

- $\{P \mid \exists S \in Students(S.CGPA > 8 \wedge P.name = S.sname \wedge P.age = S.age)\}$: returns the name and age of students with a CGPA above 8.

TRC - Example

Student

Fname	Lname	Age	Course
David	Sharma	27	DBMS
Aaron	Lilly	17	JAVA
Sahil	Khan	19	Python
Sachin	Rao	20	DBMS
Varun	George	23	JAVA
Simi	Verma	22	JAVA

Q.1 Obtain the first name of students whose age is greater than 21.

Solution:

$$\{t.Fname \mid Student(t) \land t.age > 21\}$$

$$\{t.Fname \mid t \in Student \land t.age > 21\}$$

$$\{t \mid \exists s \in Student(s.age > 21 \land t.Fname = s.Fname)\}$$

Fname
David
Varun
Simi

TRC- Example

Consider the relational schema
student($\underline{rollNo}$, $name$, $year$, $courseId$)
course($\underline{courseId}$, $cname$, $teacher$)

Q.2 Find out the names of all students who have taken the course name 'DBMS'.

required for natural join

- $\{t \mid \exists s \in student\ \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'} \wedge t.name = s.name)\}$
- $\{s.name \mid s \in student \wedge \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'})\}$

Q.3 Find out the names of all students and their rollNo who have taken the course name 'DBMS'.

- $\{s.name, s.rollNo \mid s \in student \wedge \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'})\}$
- $\{t \mid \exists s \in student\ \exists c \in course(s.courseId = c.courseId \wedge c.cname = \text{'DBMS'}$
$\wedge t.name = s.name \wedge t.rollNo = s.rollNo)\}$

TRC - Example(Cont..)

Consider the following relations:
Flights(*<u>flno</u>, from, to, distance, departs, arrives*)
Aircraft(*<u>aid</u>, aname, cruisingrange*)
Certified(*<u>eid, aid</u>*)
Employees(*<u>eid</u>, ename, salary*)

Q.6 Identify the flights that can be piloted by every pilot whose salary is more than \$100,000.
(Hint: The pilot must be certified for at least one plane with a sufficiently large cruising range.)

- $\{F.flno \mid F \in Flights \wedge \exists A \in Aircraft \exists C \in Certified \exists E \in Employees(A.cruisingrange > F.distance \wedge A.aid = C.aid \wedge E.salary > 100,000 \wedge E.eid = C.eid)\}$

Source: Database Management System by Raghu Ramakrishnan.

Strong Entity Set

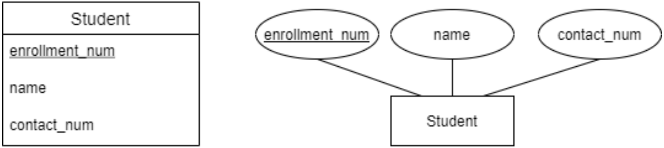


Figure: Strong entity with simple attributes

Student{*enrollment_num*, *name*, *contact_num*}

enrollment_num	**name**	**contact_num**
101	RAJ KUMAR MISHRA	222-222
102	SANAT K ROY	333-333

Figure: Table **Student**

Strong Entity Set with Composite Key

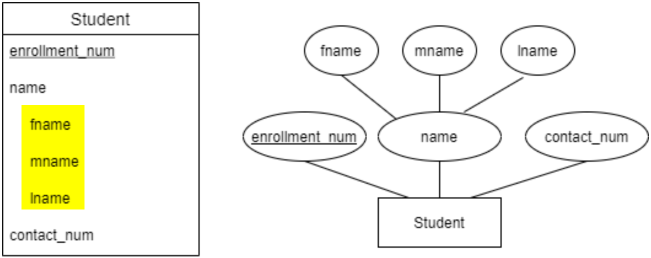


Figure: Entity set **Student** with simple and composite attributes

Student{*enrollment_num*, *fname, mname, lname*, *contact_num*}

enrollment_num	fname	mname	lname	contact_num
101	RAJ	KUMAR	MISHRA	222-222
102	SANAT	K	ROY	333-333

Figure: Table **Student**

Strong Entity Set
with Composite Key + Multivalued Attribute

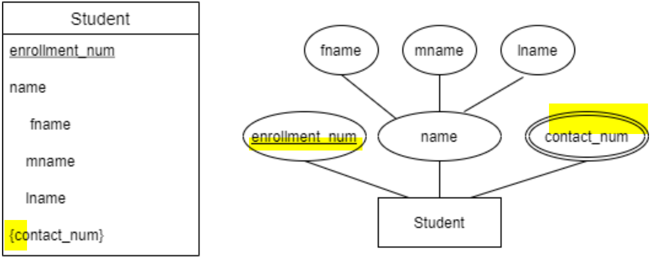


Figure: Entity set **Student** with simple, composite, and multivalued attributes

Student{*enrollment_num*, *fname*, *mname*, *lname*, *contact_num*}

enrollment_num	fname	mname	lname	contact_num
101	RAJ	KUMAR	MISHRA	222-222, 777-777
102	SANAT	K	ROY	333-333, 999-999, 666-666

Figure: Table **Student**

causes problems when we our query has WHERE

SOLN: decompose the table into multiple tables

Strong Entity Set
with Composite Key + Multivalued Attribute

Student{*enrollment_num*, *fname*, *mname*, *lname*, *contact_num*}

enrollment_num	fname	mname	lname	contact_num
101	RAJ	KUMAR	MISHRA	222-222
101	RAJ	KUMAR	MISHRA	777-777
102	SANAT	K	ROY	333-333
102	SANAT	K	ROY	999-999
102	SANAT	K	ROY	666-666

Figure: Table **Student**

OR

Student{*enrollment_num*, *fname*, *mname*, *lname* }

enrollment_num	fname	mname	lname
101	RAJ	KUMAR	MISHRA
102	SANAT	K	ROY

Figure: Table **Student**

Contacts{*enrollment_num*, *contact_num*}

enrollment_num	contact_num
101	222-222
101	777-777
102	333-333
102	999-999
102	666-666

Figure: Table **Student**

Strong Entity Set with Composite Key + Multivalued Attribute + Derived Attribute

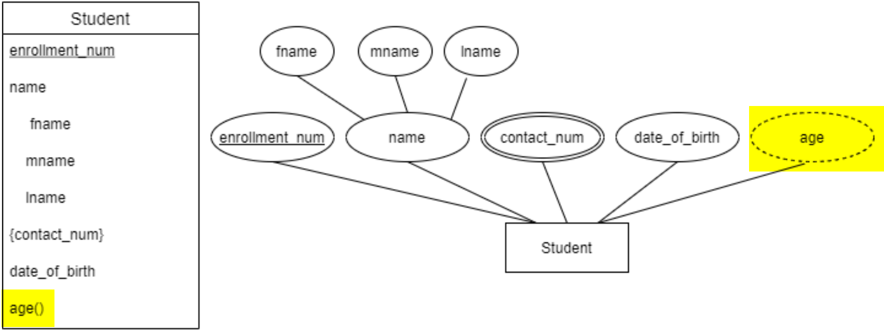


Figure: Entity set **Student** with simple, composite, multivalued attributes, and derived attribute

Student{*enrollment_num*, *fname*, *mname*, *lname*, *data_of_birth* }

Contacts{*enrollment_num*, *contact_num*}

Relationship: Cardinality Constraint (many-to-many)

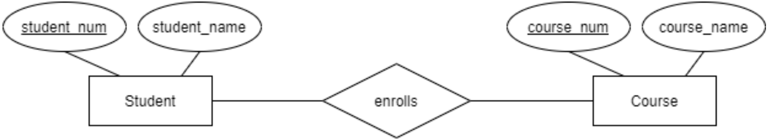


Figure: A many-to-many relationship between **Student** and **Course**

- **Student**{*student_num*, *student_name*}
- **Course**{*course_num*, *course_name*}
- **enrolls**{ *student_num*, *course_num* }

student_num	student_name
101	RAJ
102	SANAT

Student

student_num	course_num
101	CS101
102	CS101
102	MT110

enrolls

course_num	course_name
CS101	Computer Science
MT110	Mathematics

Course

Figure: Table: **Student**, **Course** and **enrolls**

Relationship: Cardinality Constraint (many-to-many) with Descriptive Attributes

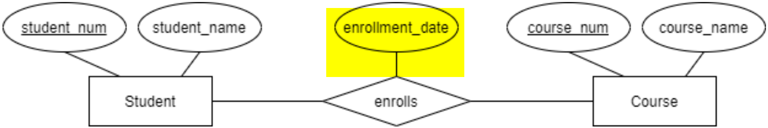


Figure: A many-to-many relationship between **Student** and **Course**

- **Student**{*student_num*, *student_name*}
- **Course**{*course_num*, *course_name*}
- **enrolls**{ *student_num*, *course_num*, *enrollment_date* }

Relationship: Cardinality Constraint (many-to-one)

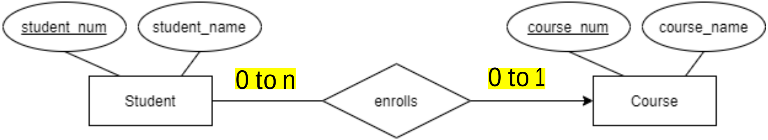


Figure: An many-to-one relationship between **Student** and **Course**

primary of one becomes attribute of many

- **Student**{*student_num*, *student_name*, *course_num*}
  must be nullable
- **Course**{*course_num*, *course_name*}

Relationship: Cardinality Constraint (one-to-one)

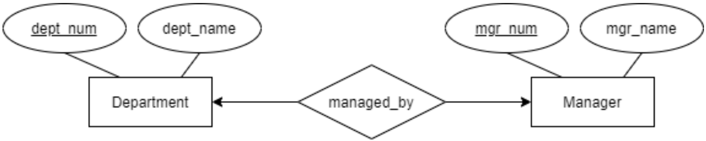


Figure: An one-to-one relationship between **Department** and **Manager**

- **Department**{*dept_num*, *dept_name*}
- **Manager**{*mgr_num*, *mgr_name*, *dept_num*}

OR

- **Department**{*dept_num*, *dept_name*, *mgr_num*}
- **Manager**{*mgr_num*, *mgr_name* }

Relationship: Cardinality Constraint (many-to-one) with Participation Constraint

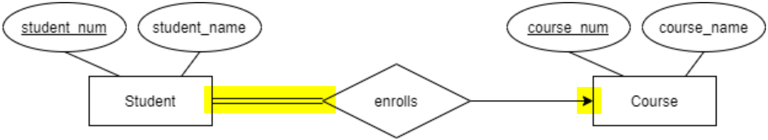


Figure: An many-to-one relationship between **Student** and **Course**

not null

- **Student**{*student_num*, *student_name*, *course_num*}
- **Course**{*course_num*, *course_name*}

Relationship: Cardinality Constraint (one-to-one) with Participation Constraint

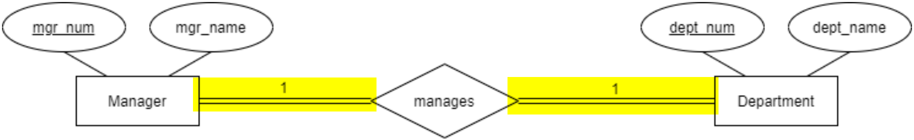


Figure: A one-to-one relationship between **Manager** and **Department**

- **Mgr_Dept**{*mgr_num, dept_num*, *mgr_name*, *dept_name* }

Weak Entity Set

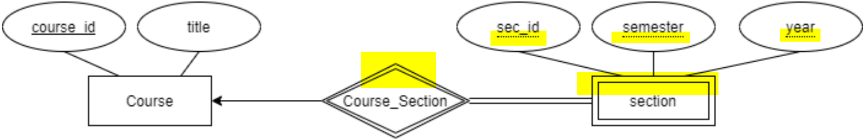


Figure: A relationship between **Course** and **Section**

- **Course**{<u>*course_id*</u>, *title* }
- **Section**{<u>*course_id, sec_id, semester, year*</u> }

Ternary Relationship

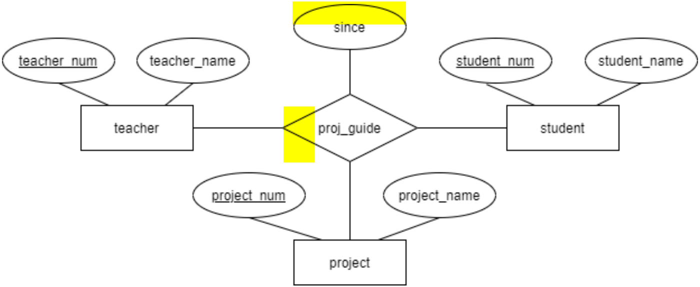
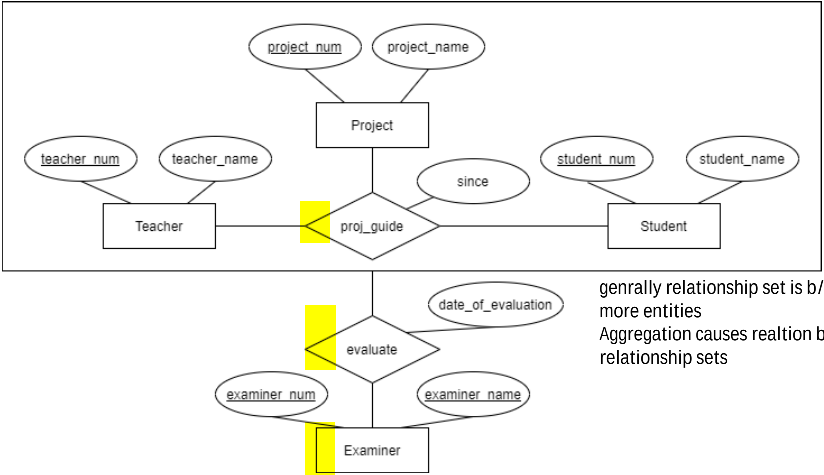


Figure: Example of ternary relationship

- **teacher**{<u>*teacher_num*</u>, *teacher_name* }
- **student**{<u>*student_num*</u>, *student_name* }
- **project**{<u>*project_num*</u>, *project_name* }
- **proj_guide**{<u>*teacher_num*, *student_num*, *project_num*</u>, *since* }

E-R Diagram with Aggregation



genrally relationship set is b/w 2 or more entities
Aggregation causes realtion b/w relationship sets

Figure: Example of Aggregation

- **proj_guide**{*teacher_num*, *student_num*, *project_num*, *since* }
- **Examiner**{*examiner_num*, *examiner_name* }
- **evaluate**{*teacher_num*, *student_num*, *project_num*, *examiner_num*, *date_of_evaluation* }

Representation of Specialization

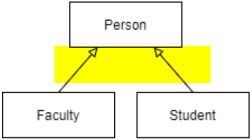


Figure: Overlapping and Partial

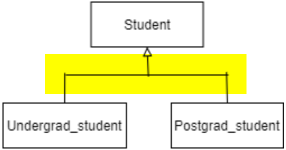


Figure: Disjoint and Partial

Solution:

- **Person**{*ID*, *name* }
- **Faculty**{*ID*, *salary* }
- **Student**{*ID*, *grade* }

- **Person**{*ID*, *name* }
- **Faculty**{*ID*, *name*, *salary* }
- **Student**{*ID*, *name*, *grade* }

- **Student**{*ID*, *name* }
- **undergrad_student**{*ID*, *project_marks* }
- **postgrad_student**{*ID*, *thesis_marks* }

- **Student**{*ID*, *name* }
- **undergrad_student**{*ID*, *name*, *project_marks* }
- **postgrad_student**{*ID*, *name*, *thesis_marks* }

Representation of Specialization (cont.)

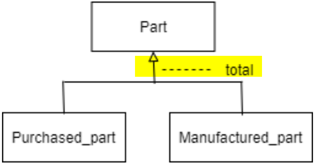


Figure: Disjoint and Complete

cannot be just 'Part' type, has to be either

Solution:

- **Purchased_part**{<u>*part_num*</u>, *name*, *price*, *vendor* }
- **Manufactured_part**{<u>*part_num*</u>, *name*, *grade*, *department* }

E-R diagrams symbols