21CSC205P Database Management Systems

Outline of the Presentation

- Conversion of ER model to Relational Table
- Case study: Apply conversion concept. Discussion of various design issues
- Pitfalls in Relational Database systems
- Understanding various Relational languages such as Tuple Relational calculus, Domain relational calculus, Calculus Vs Algebra, Computational capabilities.
- Case Study: Applying Relational Algebra for all the queries of application Designed.

Conversion of ER to Relational Table

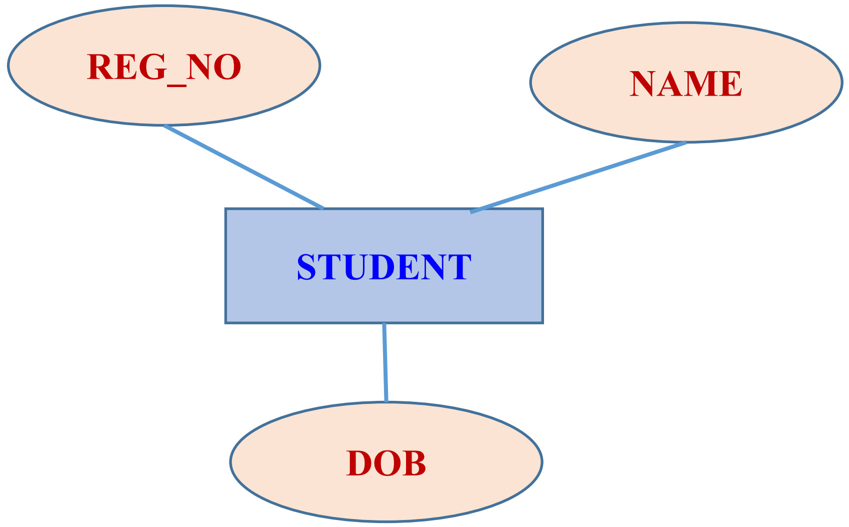
- ✔ After completing the ER diagram
- ✔ ER diagram is converted into the tables in relational model
- ✔ Relational model can be easily implemented in ORACLE, MYSQL , etc.,
- ✔ The below points to be considered for converting ER diagram into tables.
  - Strong Entity Set With Only Simple Attributes
  - Strong Entity Set With Composite Attributes
  - Strong Entity Set With Multi Valued Attributes
  - Translating Relationship Set into a Table
  - Binary Relationships With Cardinality Ratios
  - Binary Relationship With Both Cardinality Constraints and Participation Constraints
  - Binary Relationship With Weak Entity Set

Conversion of ER to Relational Table

Strong Entity Set With Only Simple Attributes

- ✔ A strong entity set with only simple attributes will require only one table in relational model.
- ✔ Attributes of the table will be the attributes of the entity set.
- ✔ The primary key of the table will be the key attribute of the entity set.

ER DIAGRAM



Er Diagram to relational table

RELATIONAL TABLE

REG_NO	NAME	DOB

Schema : student (reg_no, name, dob)

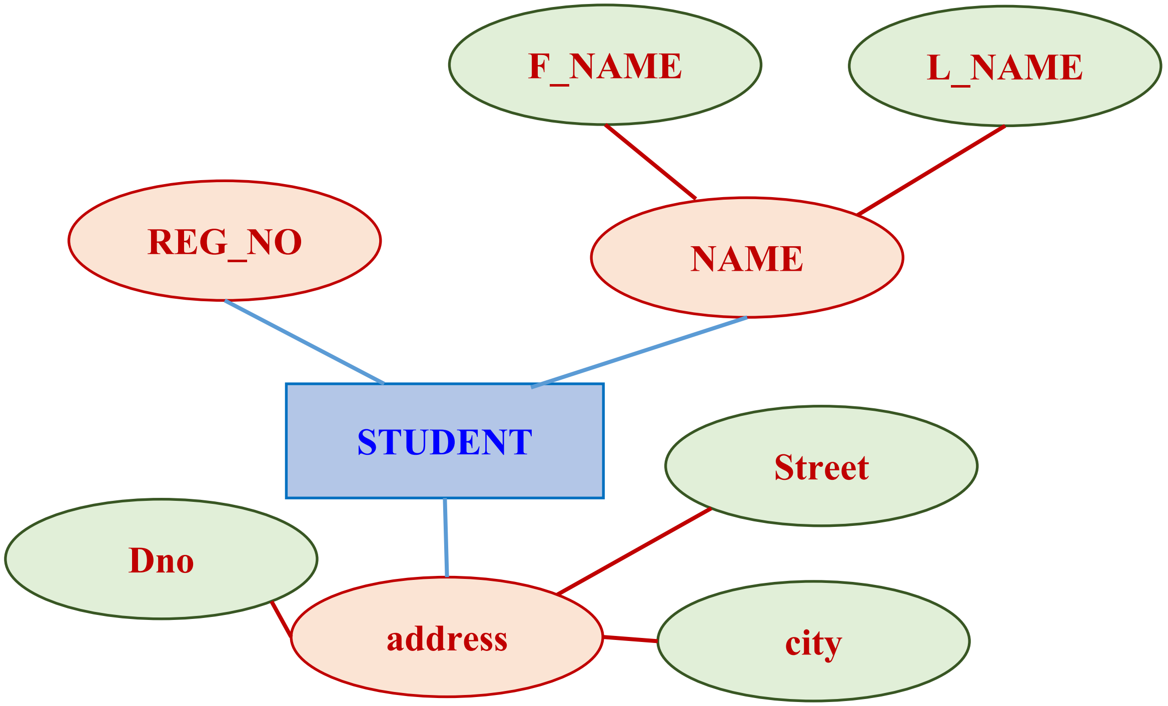
Conversion of ER to Relational Table

Strong Entity Set With Composite Attributes

- ✔ A strong entity set with any number of composite attributes will require only one table in relational model.
- ✔ While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

ER DIAGRAM

RELATIONAL TABLE



Reg_no	F_name	L_name	dno	street	city

Schema : student (reg_no, f_name, l_name, dno, street, city)

Conversion of ER to Relational Table

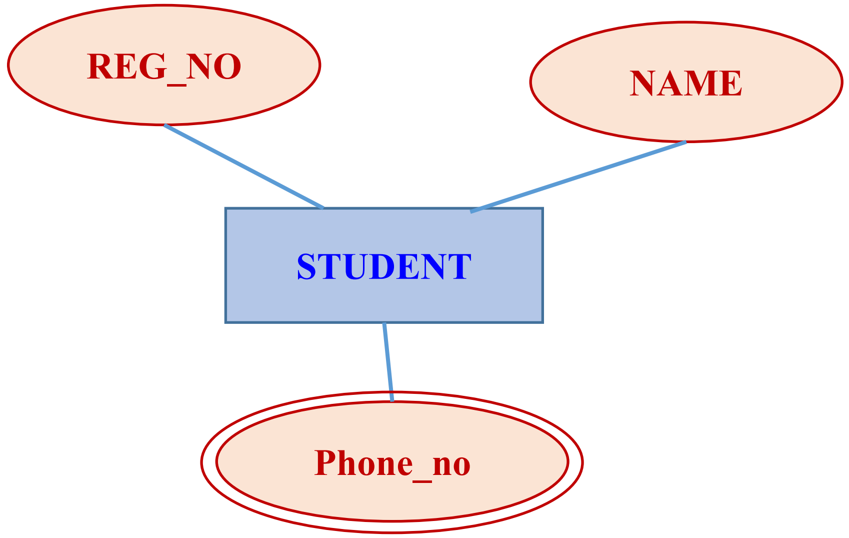
Strong Entity Set With Multi Valued Attributes

- ✔ A strong entity set with any number of multi valued attributes will require two tables in relational model.
- ✔ One table will contain all the simple attributes with the primary key.
- ✔ Other table will contain the primary key and all the multi valued attributes.

ER DIAGRAM

Er Diagram to relational table

RELATIONAL TABLE



REG_NO	NAME

Schema : student (reg_no, name)

REG_NO	Phone_no

Schema : student (reg_no, phone_no)

Conversion of ER to Relational Table

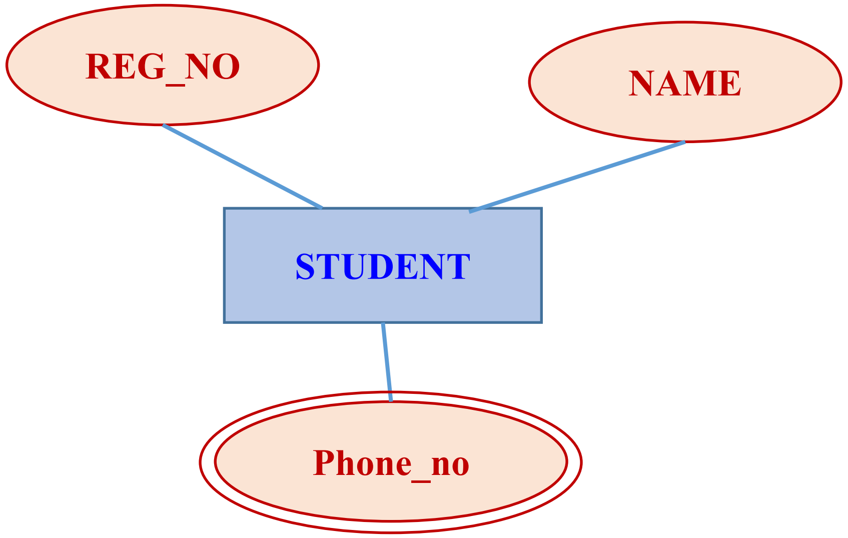
Strong Entity Set With Multi Valued Attributes

✔ Also the stronger entity set with any number of multi valued attributes may be converted as shown below

ER DIAGRAM

$\xrightarrow{\text{Er Diagram to relational table}}$

RELATIONAL TABLE



reg_no	name	phone_no1	phone_no2	Phone_no3

Schema : student (reg_no, name, phone_no1 ,phone_no2, phone_no3)

Conversion of ER to Relational Table

Translating Relationship Set into a Table

- ✔ A relationship set will require one table in the relational model.
- ✔ Attributes of the table are :
  - Primary key attributes of the participating entity sets
  - Its own descriptive attributes if any.
- ✔ Set of non-descriptive attributes will be the primary key.

NOTE

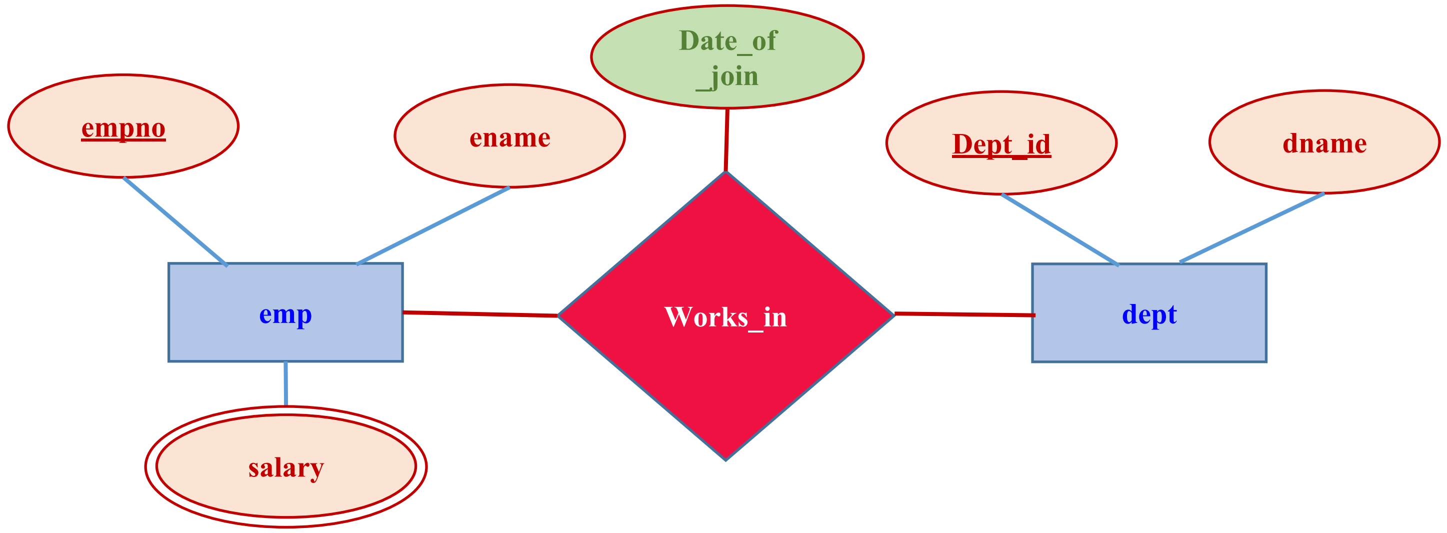
✔If we consider the overall ER diagram, three tables will be required in relational model

1. Emp
2. Dept
3. Works_in

ER DIAGRAM

Er Diagram to relational table

RELATIONAL TABLE



empno	dept_id	date_of_join

Schema : works_in (empno, dept_id, date_of_join)

Conversion of ER to Relational Table

Binary Relationships With Cardinality Ratios

✔ Four types are possible

1. Binary relationship with cardinality ratio 1:1
2. Binary relationship with cardinality ratio 1:m
3. Binary relationship with cardinality ratio m:1
4. Binary relationship with cardinality ratio m:m

Conversion of ER to Relational Table

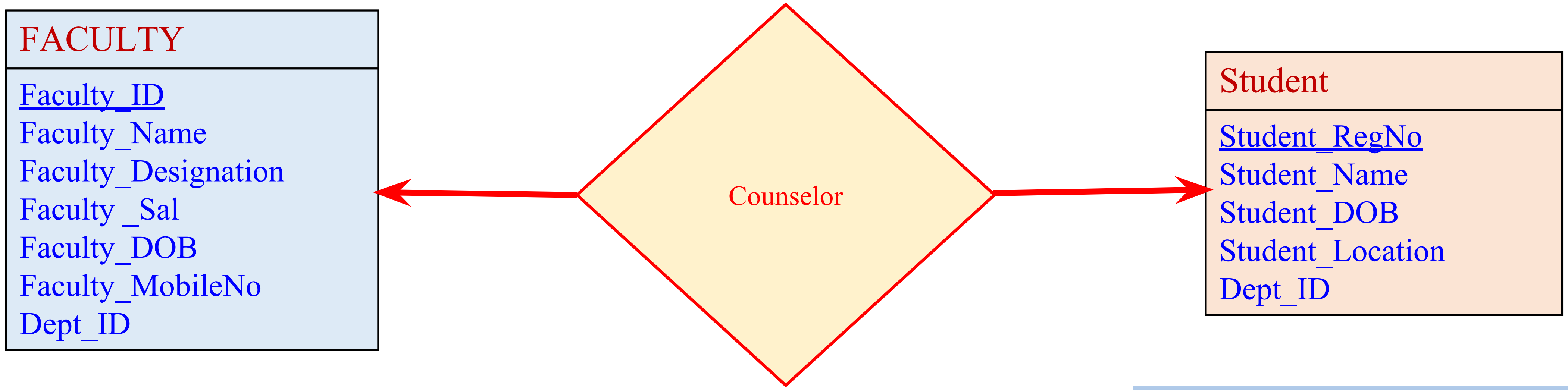
Mapping Cardinality

- ✔ The relationship set counselor, between the faculty and student entity sets may be one-to-one, one-to-many, many-to-one, or many-to-many.
- ✔ To distinguish among these types, we draw either a directed line (→) or an undirected line (—) between the relationship set and the entity.

Binary Relationships With Cardinality Ratios

- ✔ Binary relationship with cardinality ratio 1:1

  Line from the relationship set counselor to both entity sets faculty and student as given in the figure below. This indicates that a faculty may counsel at most one student, and a student may have at most one counselor.

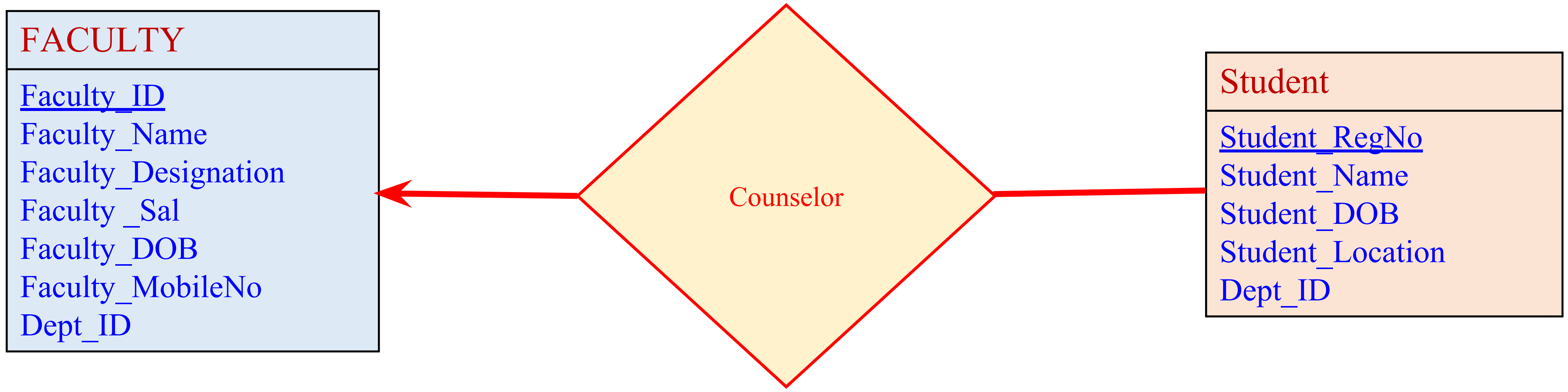


Conversion of ER to Relational Table

Binary Relationships With Cardinality Ratios

✔ Binary relationship with cardinality ratio 1:m

A directed line from the relationship set counselor to the entity set faculty and an undirected line to the entity set student as shown in the below figure, indicates that a faculty may counsel many students, but a student may have at most one counselor.

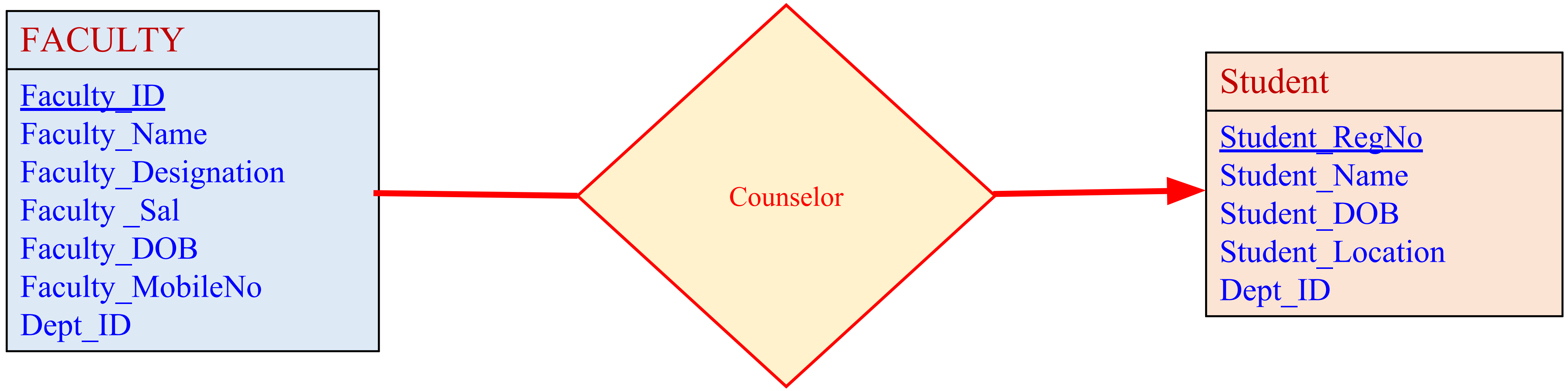


Conversion of ER to Relational Table

Binary Relationships With Cardinality Ratios

✔ Binary relationship with cardinality ratio m:1

An undirected line from the relationship set counselor to the entity set faculty and a directed line to the entity set student as shown in the below figure, indicates that a faculty may counsel at most one student, but a student may have many counselors.

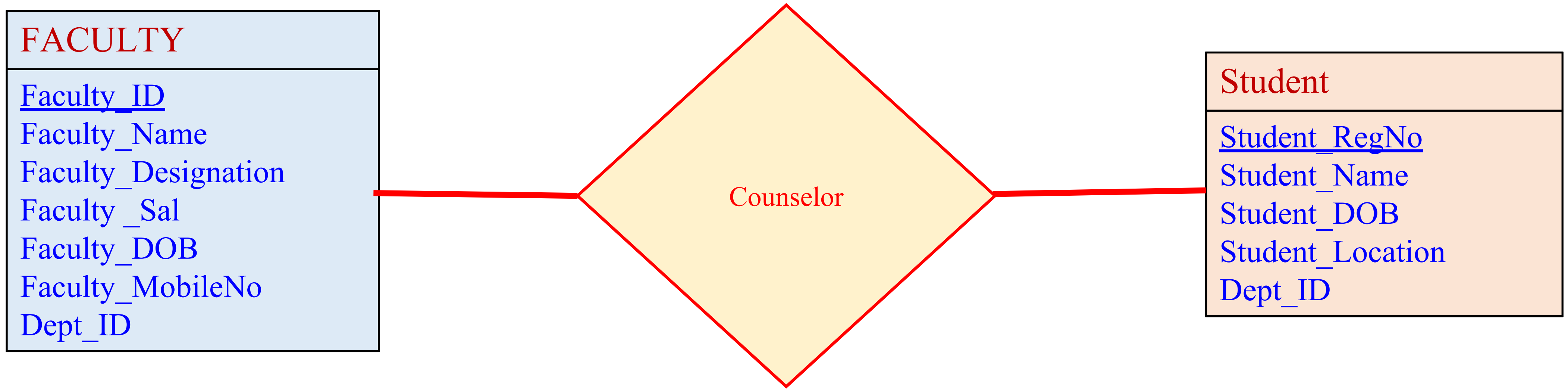


Conversion of ER to Relational Table

Binary Relationships With Cardinality Ratios

✔ Binary relationship with cardinality ratio m:m

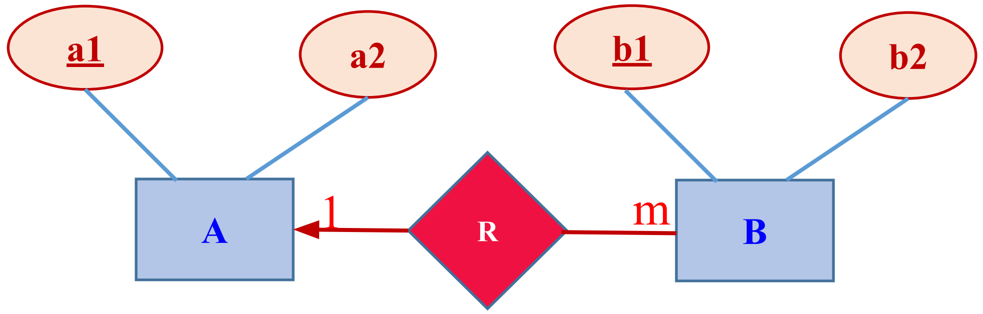
We draw an undirected line from the relationship set counselor to both entity sets faculty and student as shown in the below figure, indicates that a faculty may counsel many students, and a student may have many counselor.



Conversion of ER to Relational Table

Binary Relationship With Both Cardinality Constraints and Participation Constraints

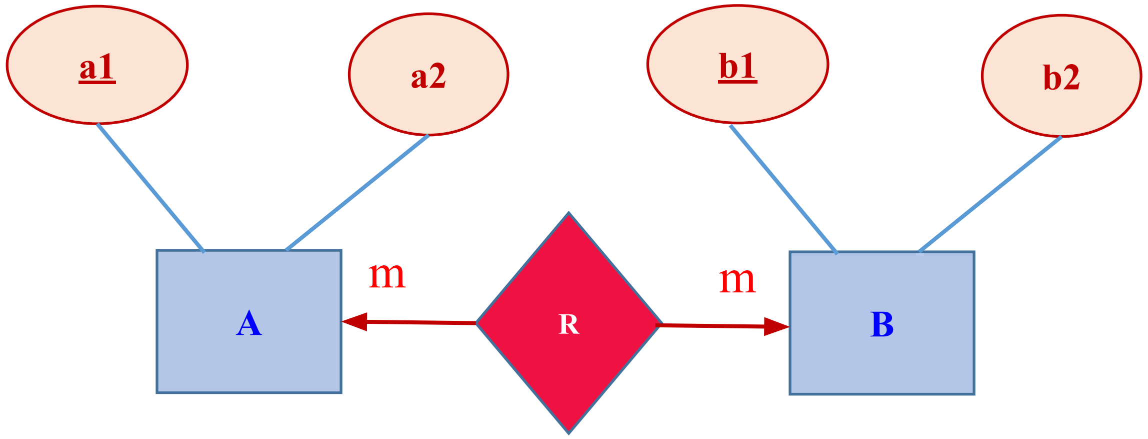
- ✔ Because of the total participation constraint, foreign key acquires NOT NULL constraint
- ✔ Now foreign key can not be null.
- ✔ Option 1: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From One Side
  - ✔ Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.
  - ✔ Then, two tables will be required-
    - A (<u>a1</u> , a2)
    - BR (a1 , <u>b1</u> , b2)
  - ✔ Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't be null now.



Conversion of ER to Relational Table

Binary Relationship With Both Cardinality Constraints and Participation Constraints

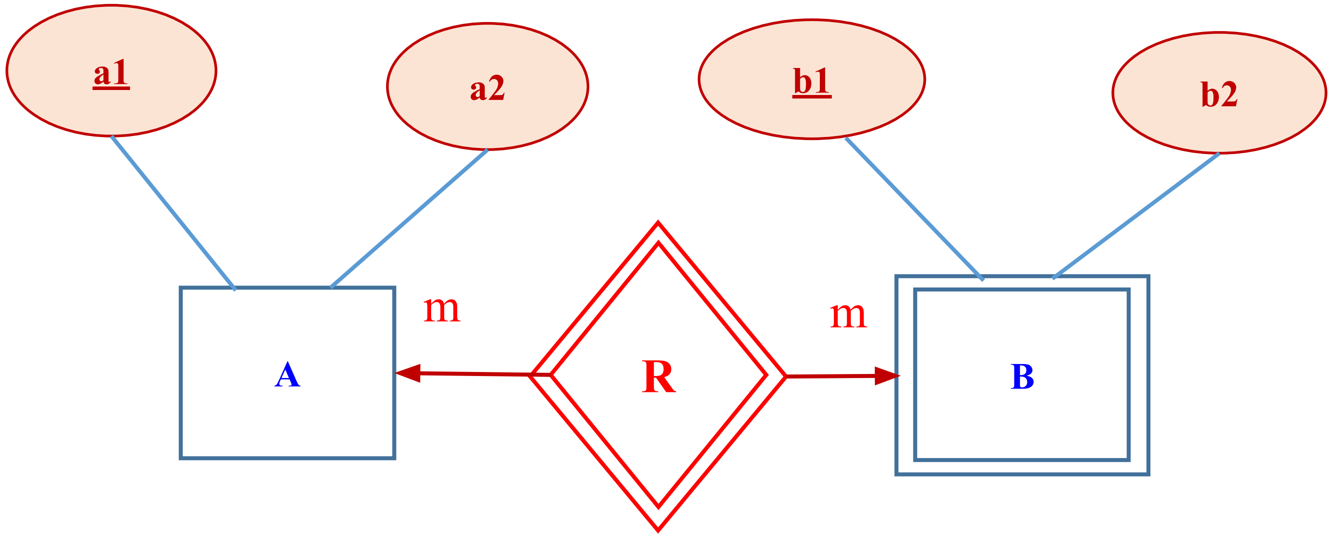
- ✔ Option 2: Binary Relationship With Cardinality Constraint and Total Participation Constraint From Both Sides
  - ✔ If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table.
    - ARB (<u>a1</u> , a2 , <u>b1</u> , b2)



Conversion of ER to Relational Table

Binary Relationship With Weak Entity Set

- ✔ Weak entity set always appears in association with identifying relationship with total participation constraint.
- ✔ Here, two tables will be required-
  - A (<u>a1</u> , a2)
  - BR (<u>a1</u> , <u>b1</u> , b2)



Pitfalls in Relational database

Pitfalls in Relational Database

- ✔ Relational database design requires that we find a “good” collection of relation schemas.
- ✔ A bad design may lead to
  - Repetition of information.
  - Inability to represent certain information.
- ✔ Design Goals:
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Facilitate the checking of updates for violation of database integrity constraints

Pitfalls in Relational database

Example

- ✔ Consider the relation schema:

  Lending-schema = (branch_name, branch_city, assets,

  customer_name, loan_number, amount)

- ✔ Redundancy:
  - Data for branch-name, branch-city, assets are repeated for each loan that a branch makes
  - Wastes space and complicates updating
- ✔ Null values
  - Cannot store information about a branch if no loans exist
  - Can use null values, but they are difficult to handle

Pitfalls in Relational database, Decomposing bad schema

Decomposition

- ✔ Repetition of information will lead to many issues, that type of schema is a bad design. To avoid such issues the designed schema is decomposed into two or more schemas.

- ✔ Consider the schema : Lending_schema

- ✔ Lending_schema is decomposed into two schemas as given below
  - Branch-customer-schema = (branch_name, branch_city, assets, customer_name)
  - Customer-loan-schema = (customer_name, loan_number, amount)

Pitfalls in Relational database, Decomposing bad schema

Decomposition

- ✔ All attributes of an original schema (R) must appear in the decomposition (R1, R2):

$$R = R1 \cup R$$

- ✔ Lossless-join decomposition. For all possible relations r on schema R.

$$r = \Pi R1\ (r) \quad \Pi R2\ (r)$$

$$\bowtie$$

Pitfalls in Relational database, Decomposing bad schema

Decomposition

Example :

Consider The relation : R = (A,B)

Decomposition of R = R1(A) and R2(B)

R

A	B
X	1
X	2
Y	1

Π_A (R)

A
X
Y

Π_B(R)

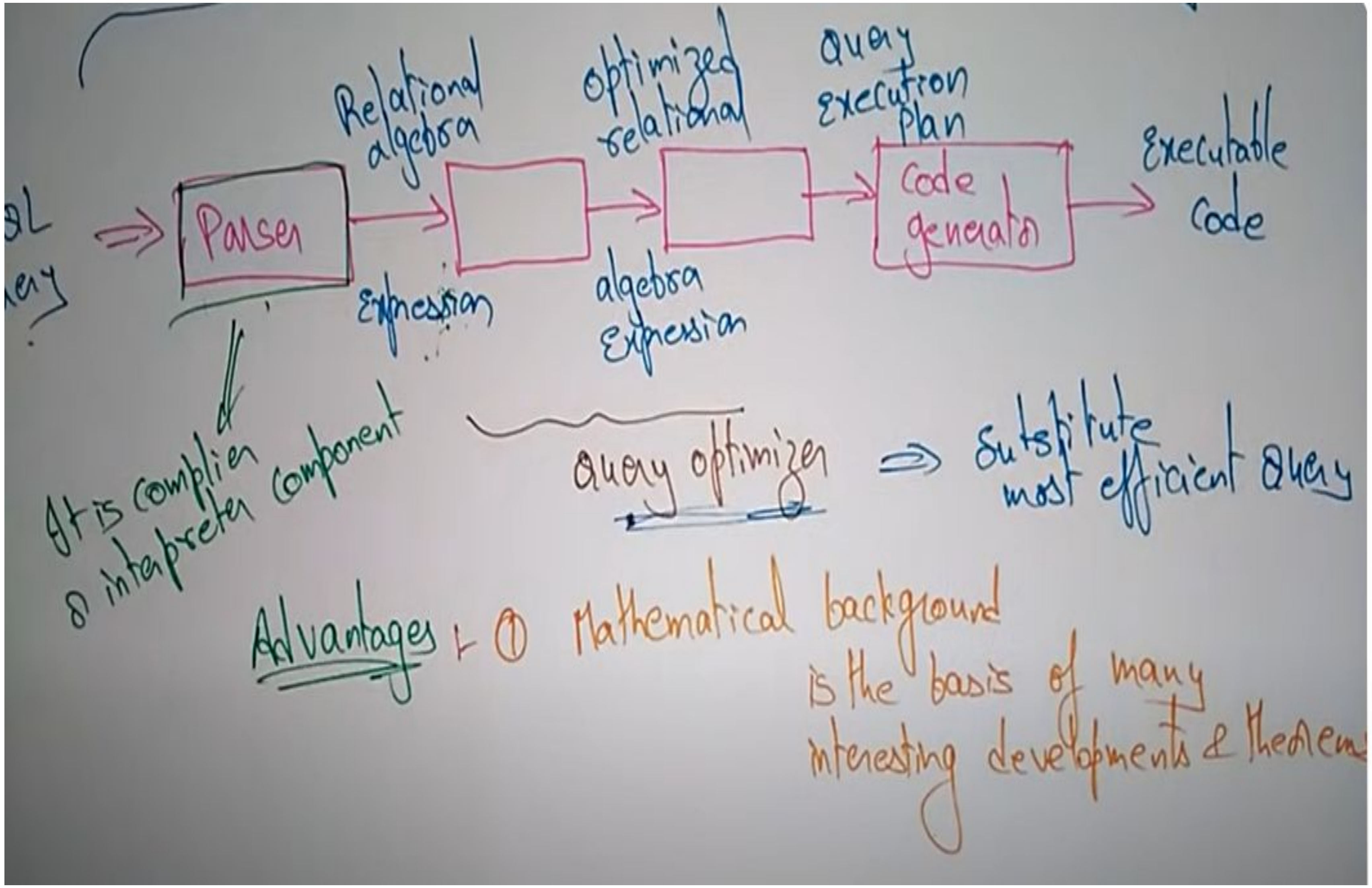
B
1
2

$\Pi A\ (R) \bowtie \Pi B(R)$

A	B
X	1
X	2
Y	1
Y	2

Relational Algebra – Fundamental Operators and syntax

- ✔ Relational algebra is a procedural query language
- ✔ An operator can be either unary or binary
- ✔ Relational algebra is performed recursively on a relation and intermediate results are also considered relations.
- ✔ Fundamental operations of Relational Algebra
  - Select
  - Project
  - Union
  - Set different
  - Cartesian product
  - Rename



QL
uery
Parser
Relational algebra expression
optimized relational algebra expression
Query execution plan
code generator
Executable code

It is complier & interpreter component

Query optimizer ⇒ substitute most efficient Query

Advantages :- ① Mathematical background is the basis of many interesting developments & theorems

Relational Algebra – Fundamental Operators and syntax

Select Operation (σ)

It retrieves the records that satisfy the given predicate from a relation

Notation : $\sigma_p(r)$

Where

σ - Selection Predicate

r – Relation

p – Prepositional logic formula

Example 1: $\sigma_{JOB = \text{“}MANAGER\text{"}}(\text{EMP})$

Output : Retrieve the records from emp where job is manager

Example 2 : $\sigma_{\text{JOB = “MANAGER" and DEPTNO = 20}}(\text{EMP})$

Output : Retrieve the records from emp where job is manger and deptno is 20.

Relational Algebra – Fundamental Operators and syntax

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation : $\prod_{A1, A2,\ldots, An} (r)$

Where

∏ - Predicate

A1,A2,…,An – Attribute names of a relation

r – Relation

Note : Because of relation is a set duplicate rows are eliminated automatically.

Example : $\prod_{ENAME, JOB} (EMP)$

Output : It selects and projects the column named as ename and job from emp table

Relational Algebra – Fundamental Operators and syntax

Union Operation (∪)

✔ It performs binary union between two given relations.

✔ It is defined as

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

Notation : r U s

Where

U is the union operator

R and S are relations

✔ The following conditions must hold from both relations to make the union operation is true

- Relations R and S should have the same number of Attributes
- The Attributes domain values (data types) must be compatible
- Duplicate records will be eliminated automatically

Relational Algebra – Fundamental Operators and syntax

Union Operation (∪)

Consider the following table

Records in table a

Sql > Select * from a

F1	F2
30	C
10	A
20	B

Records in table b

Sql > Select * from b;

ID	NAME
30	C
40	D
50	E

Example : AUB

F1	F2
30	C
10	A
20	B
40	D
50	E

Example : BUA

ID	NAME
10	A
20	B
30	C
40	D
50	E

Relational Algebra – Fundamental Operators and syntax

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation : r − s

Example : Consider the following tables

Records in table a

Sql > Select * from a

F1	F2
30	C
10	A
20	B

Records in table b

Sql > Select * from b;

ID	NAME
30	C
40	D
50	E

Example : A-B

F1	F2
10	A
20	B

Example : B-A

ID	NAME
40	D
50	E

Relational Algebra – Fundamental Operators and syntax

Cartesian Product (X)

- ✔ Combines information of two different relations into one.
- ✔ For each and every record in the first query , all the records will be executed in the second query

Notation : r X s

- ✔ Where r and s are relations and their output will be defined as :

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

Example : Consider the following tables

Records in table a

Sql > Select * from a

F1	F2
30	C
10	A
20	B

Records in table b

Sql > Select * from b;

ID	NAME
30	C
40	D
50	E

A X B

F1	F2	ID	NAME
10	A	30	C
20	B	30	C
30	C	30	C
10	A	40	D
20	B	40	D
30	C	40	D
10	A	50	E
20	B	50	E
30	C	50	E

Rename Operation (ρ)

The rename operation allows us to rename the output relation.

Notation : ρ_{x} (E)

Where the result of expression E is saved with name of x.

Relational algebra queries, Tuple relational calculus

Relational algebra queries

Example Queries 1:

- ✔ Find the empno, ename, job, salary for employess whose salary is greater than Rs.2000

  $\{t \mid t \in emp \wedge t[sal] > 2000\}$

- ✔ Suppose that we want only the empno attribute, rather than all attributes of the emp relation.
- ✔ To write this query in the tuple relational calculus, we need to write an expression for a relation on the schema (empno). We need those tuples on (empno) , such that there is a tuple in emp with the sal attribute > 2000.
- ✔ To express this request, we need the construct "there exists" from mathematical logic.
- ✔ The notation: $\exists\ t \in r\ (Q(t))$

  means "there exists a tuple t in relation r such that predicate Q(t) is true."

Relational algebra queries, Tuple relational calculus

- ✔ Using the above notation, we can write the query "Find the empno for each employee with a salary greater than Rs.2,000" as:

  $\{t \mid \exists\ s \in emp\ (t[empno] = s[empno] \wedge s[sal] > 2000)\}$

- ✔ In English, the above expression is read as

  "The set of all tuples t such that there exists a tuple s in relation emp for which the values of t and s for the empno attribute are equal, and the value of s for the sal attribute is greater than Rs.2,000."

- ✔ Tuple variable t is defined on only the empno attribute, since that is the only attribute having a condition specified for t. Thus, the result is a relation on (empno).

Relational algebra queries, Tuple relational calculus

Relational algebra queries

Example Queries 2:

- ✔ Consider the query "Find the names of all employees whose department is in the location "CHICAGO." This query is slightly more complex than the previous queries, since it involves two relations: emp and dept.
- ✔ It requires is that we have two "there exists" clauses in our tuple-relational-calculus expression, connected by and ($\wedge$).
- ✔ We write the query as follows:

$$\{t \mid \exists\ s \in \text{emp}\ (t[\text{ename}] = s[\text{ename}]$$
$$\wedge\ \exists\ u \in \text{dept}\ (u[\text{dname}] = s[\text{dname}]$$
$$\wedge\ u[\text{building}] = \text{"CHICAGO"}))\}$$

- ✔ Tuple variable u is restricted to departments that are located in the chicago, while tuple variable s is restricted to emp whose dept name matches that of tuple variable u.

Relational algebra queries,
Tuple relational calculus

Tuple Relational Calculus

- ✔ The tuple relational calculus is a nonprocedural query language.
- ✔ It describes the desired information without giving a specific procedure for obtaining that information.
- ✔ A query in the tuple relational calculus is expressed as:

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

- ✔ That is, it is the set of all tuples t such that predicate P is true for t.
- ✔ we use t[A] to denote the value of tuple t on attribute A, and we use $t \in r$ to denote that tuple t is in relation r.

Relational algebra queries, Tuple relational calculus

Tuple Relational Calculus

Formal definition

A tuple-relational-calculus expression is of the form:

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

where P is a formula.

Several tuple variables may appear in a formula. A tuple variable is said to be a free variable unless it is quantified by a $\exists$ or $\forall$.

Thus, in:

$t \in emp \wedge \exists s \in dept(t[dname] = s[dname])$

t is a free variable.

Tuple variable s is said to be a bound variable.

Relational algebra queries, Tuple relational calculus

- A tuple-relational-calculus formula is built up out of atoms. An atom has one of the following forms:
- $s \in r$, where s is a tuple variable and r is a relation (we do not allow use of the ∈/ operator).
- s[x] u[y], where s and u are tuple variables, x is an attribute on which s is defined, y is an attribute on which u is defined, and is a comparison operator ($<$, $\leq$, $=$, $=$, $>$, $\geq$); we require that attributes x and y have domains whose members can be compared by (-)
- s[x] c, where s is a tuple variable, x is an attribute on which s is defined, is a comparison operator, and c is a constant in the domain of attribute x.

Relational algebra queries, Tuple relational calculus

We build up formulae from atoms by using the following rules:

- ✔ An atom is a formula.
- ✔ If P1 is a formula, then so are ¬P1 and (P1).
- ✔ If P1 and P2 are formulae, then so are P1 $\vee$ P2, P1 $\wedge$ P2, and P1 $\Rightarrow$ P2.
- ✔ If P1(s) is a formula containing a free tuple variable s, and r is a relation, then

  $$\exists\ s \in r\ (P1(s)) \text{ and } \forall\ s \in r\ (P1(s))$$

  are also formulae.

Relational algebra queries, Tuple relational calculus

- *branch* (*branch_name, branch_city, assets*)
- *customer* (*customer_name, customer_street, customer_city*)
- *account* (*account_number, branch_name, balance*)
- *loan* (*loan_number, branch_name, amount*)
- *depositor* (*customer_name, account_number*)
- *borrower* (*customer_name, loan_number*)

Relational algebra queries, Tuple relational calculus

Banking Examples

Find the *loan_number, branch_name,* and *amount* for loans of over $1200

$$\{t \mid t \in loan \land t[amount] > 1200\}$$

Find the loan number for each loan of an amount greater than $1200

$$\{t \mid \exists\, s \in \text{loan}\ (t[loan\_number] = s[loan\_number] \land s[amount] > 1200)\}$$

Notice that a relation on schema [*loan_number*] is implicitly defined by the query

Relational algebra queries, Tuple relational calculus

Banking Examples

- Find the names of all customers having a loan, an account, or both at the bank

$$\{t \mid \exists s \in \mathit{borrower}\,(t[\mathit{customer\_name}] = s[\mathit{customer\_name}]) \lor \exists u \in \mathit{depositor}\,(t[\mathit{customer\_name}] = u[\mathit{customer\_name}])$$

Find the names of all customers who have a loan and an account at the bank

$$\{t \mid \exists\ s \in \text{loan}\,(t[\mathit{loan\_number}] = s[\mathit{loan\_number}] \land s[\mathit{amount}] > 1200)\}$$

$$\{t \mid \exists s \in \mathit{borrower}\,(t[\mathit{customer\_name}] = s[\mathit{customer\_name}]) \land \exists u \in \mathit{depositor}\,(t[\mathit{customer\_name}] = u[\mathit{customer\_name}])$$

Relational algebra queries, Tuple relational calculus

Banking Examples

- Find the names of all customers having a loan at the Perryridge branch

$\{t \mid \exists s \in borrower\ (t[customer\_name] = s[customer\_name]$
$\quad \wedge\ \exists u \in loan\ (u[branch\_name] = \text{“Perryridge”}$
$\quad\quad \wedge\ u[loan\_number] = s[loan\_number]))\}$

- Find the names of all customers who have a loan at the Perryridge branch, but no account at any branch of the bank

$\{t \mid \exists s \in borrower\ (t[customer\_name] = s[\text{customer\_name}]$
$\quad \wedge\ \exists u \in loan\ (u[branch\_name] = \text{“Perryridge”}$
$\quad\quad \wedge\ u[loan\_number] = s[\text{loan\_}number]))$
$\quad \wedge\ \textbf{not}\ \exists v \in depositor\ (v[customer\_name] =$
$\quad\quad t[customer\_name])\}$

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. $\{ t \mid t[A] = 5 \lor \textbf{true} \}$ is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in P.

Relational algebra queries,
Tuple relational calculus

Domain Relational Calculus

- ✔ A second form of relational calculus, called domain relational calculus, uses domain variables that take on values from an attributes domain, rather than values for an entire tuple. The domain relational calculus,

- ✔ However, is closely related to the tuple relational calculus.

- ✔ Domain relational calculus serves as the theoretical basis of the widely used QBE language, just as relational algebra serves as the basis for the SQL language.

Relational algebra queries, Tuple relational calculus

Domain Relational Calculus

- ✔ An expression in the domain relational calculus is of the form

$$\{< x1, x2,..., xn > \mid P(x1, x2,..., xn)\}$$

- ✔ where x1, x2,..., xn represent domain variables. P represents a formula composed of atoms, as was the case in the tuple relational calculus.

Relational algebra queries,
Tuple relational calculus

Domain Relational Calculus

- ✔ An atom in the domain relational calculus has one of the following forms:
- ✔ $< x1, x2,..., xn > \in r$, where r is a relation on n attributes and x1, x2,..., xn are domain variables or domain constants.
- ✔ x y, where x and y are domain variables and is a comparison operator (<, ≤, =, =, >, ≥). We require that attributes x and y have domains that can be compared by (-).
- ✔ x c, where x is a domain variable, is a comparison operator, and c is a constant in the domain of the attribute for which x is a domain variable.

Relational algebra queries, Tuple relational calculus

Domain Relational Calculus

Example Queries

- Find the *loan_number, branch_name,* and *amount* for loans of over $1200

  $\{< l, b, a > \mid < l, b, a > \in loan \land a > 1200\}$

- Find the names of all customers who have a loan of over $1200

  $\{< c > \mid \exists\ l, b, a\ (< c, l > \in borrower \land < l, b, a > \in loan \land a > 1200)\}$

- Find the names of all customers who have a loan from the Perryridge branch and the loan amount:
  - $\{< c, a > \mid \exists\ l\ (< c, l > \in borrower \land \exists b\ (< l, b, a > \in loan \land b = \text{“Perryridge”}))\}$
  - $\{< c, a > \mid \exists\ l\ (< c, l > \in borrower \land < l, \text{“ Perryridge”}, a > \in loan)\}$

Domain Relational Calculus

Example Queries

- Find the names of all customers having a loan, an account, or both at the Perryridge branch:

$$\{< c > \mid \exists\, l\,(< c, l > \in \mathit{borrower} \\ \wedge \exists\, b,a\,(< l, b, a > \in \mathit{loan} \wedge b = \text{“Perryridge”})) \\ \vee \exists\, a\,(< c, a > \in \mathit{depositor} \\ \wedge \exists\, b,n\,(< a, b, n > \in \mathit{account} \wedge b = \text{“Perryridge”}))\}$$

- Find the names of all customers who have an account at all branches located in Brooklyn:

$$\{< c > \mid \exists\, \text{s},n\,(< c, s, n > \in \text{customer}) \wedge \\ \forall\, x,y,z\,(< x, y, z > \in \mathit{branch} \wedge y = \text{“Brooklyn”}) \Rightarrow \\ \exists\, a,b\,(< x, y, z > \in \mathit{account} \wedge < c,a > \in \mathit{depositor})\}$$

Safety of Expressions

The expression:

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

is safe if all of the following hold:

1. All values that appear in tuples of the expression are values from *dom* (*P*) (that is, the values appear either in *P* or in a tuple of a relation mentioned in *P*).
2. For every "there exists" subformula of the form $\exists\ x\ (P_1(x))$, the subformula is true if and only if there is a value of *x* in *dom* (P_1) such that $P_1(x)$ is true.
3. For every "for all" subformula of the form $\forall_{x}\ (P_1\ (x))$, the subformula is true if and only if $P_1(x)$ is true for all values *x* from *dom* (P_1).

THANK YOU