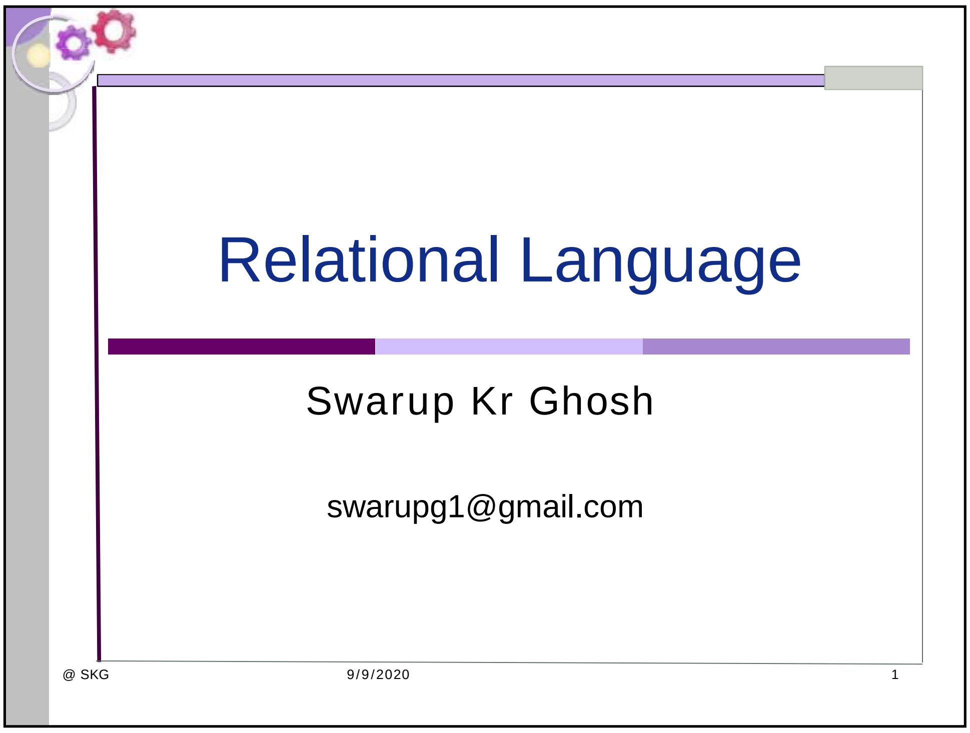
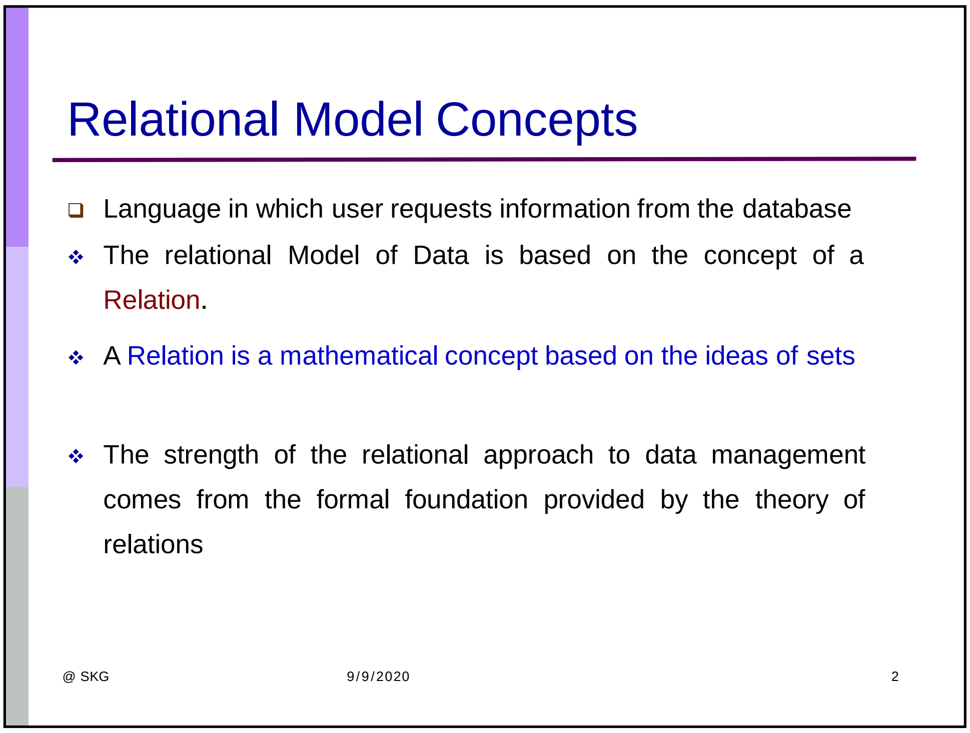


Relational Language

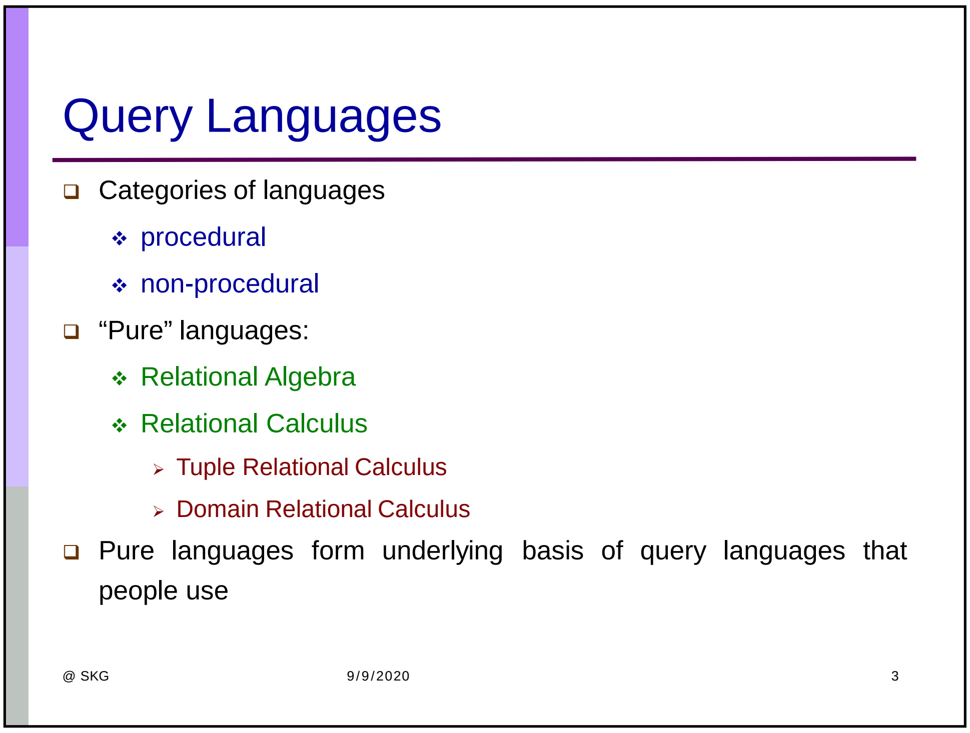
Swarup Kr Ghosh

swarupg1@gmail.com



Relational Model Concepts

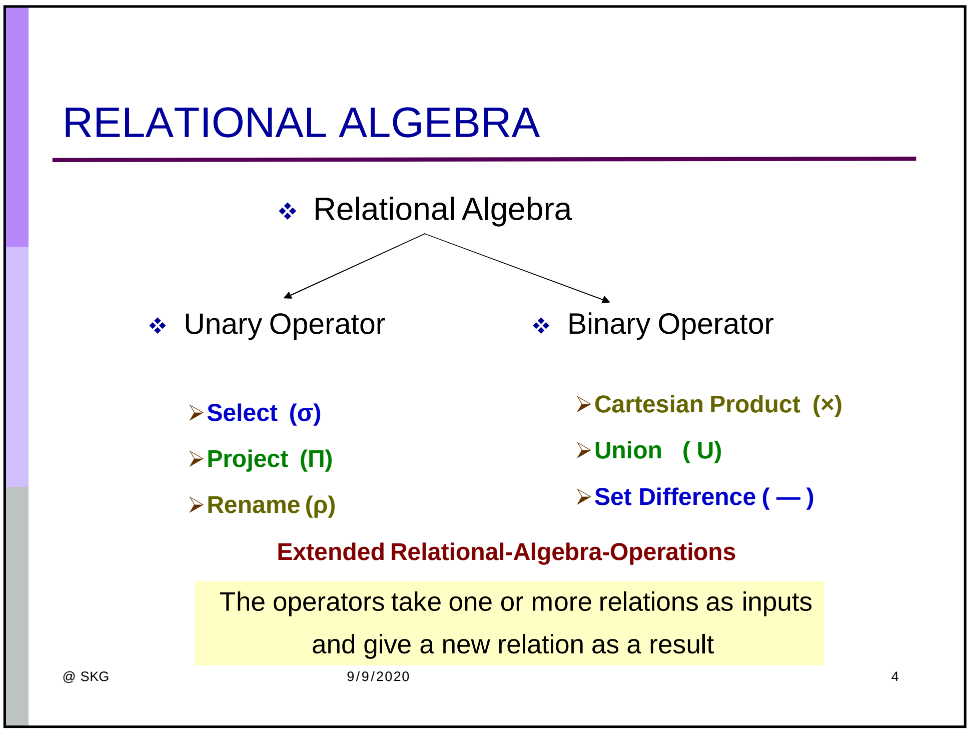
- Language in which user requests information from the database
- The relational Model of Data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations



Query Languages

- Categories of languages
  - procedural
  - non-procedural
- "Pure" languages:
  - Relational Algebra
  - Relational Calculus
    - Tuple Relational Calculus
    - Domain Relational Calculus
- Pure languages form underlying basis of query languages that people use

@ SKG 9/9/2020 3



RELATIONAL ALGEBRA

- Relational Algebra
  - Unary Operator
    - **Select (σ)**
    - **Project (Π)**
    - **Rename (ρ)**
  - Binary Operator
    - **Cartesian Product (×)**
    - **Union (U)**
    - **Set Difference (—)**

Extended Relational-Algebra-Operations

The operators take one or more relations as inputs and give a new relation as a result

@ SKG 9/9/2020 4

Example of Relation

emp

eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

Select (σ)

emp

eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

$\sigma_{<\text{search condition}>}$ (Relation Name)

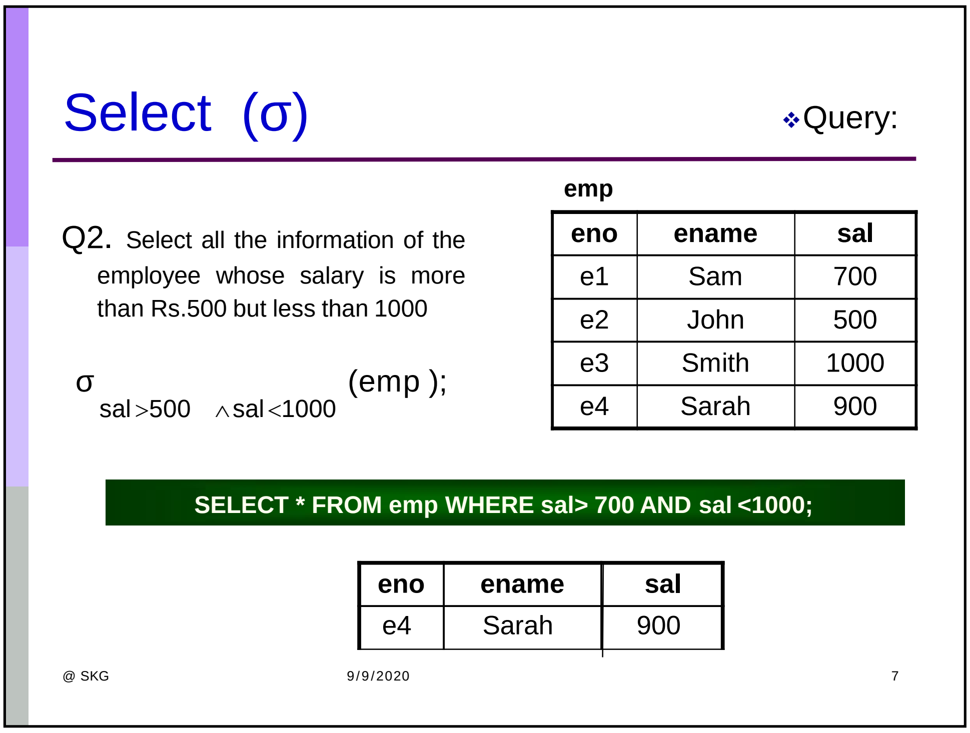
❖Query:

Q1. Select all the information of the employee whose salary is more than Rs.700

$\sigma_{sal>700}(emp);$

SELECT * FROM emp WHERE sal> 700;

eno	ename	sal
e3	Smith	1000
e4	Sarah	900



Select (σ)

❖Query:

Q2. Select all the information of the employee whose salary is more than Rs.500 but less than 1000

$\sigma_{sal>500 \wedge sal<1000}$ (emp);

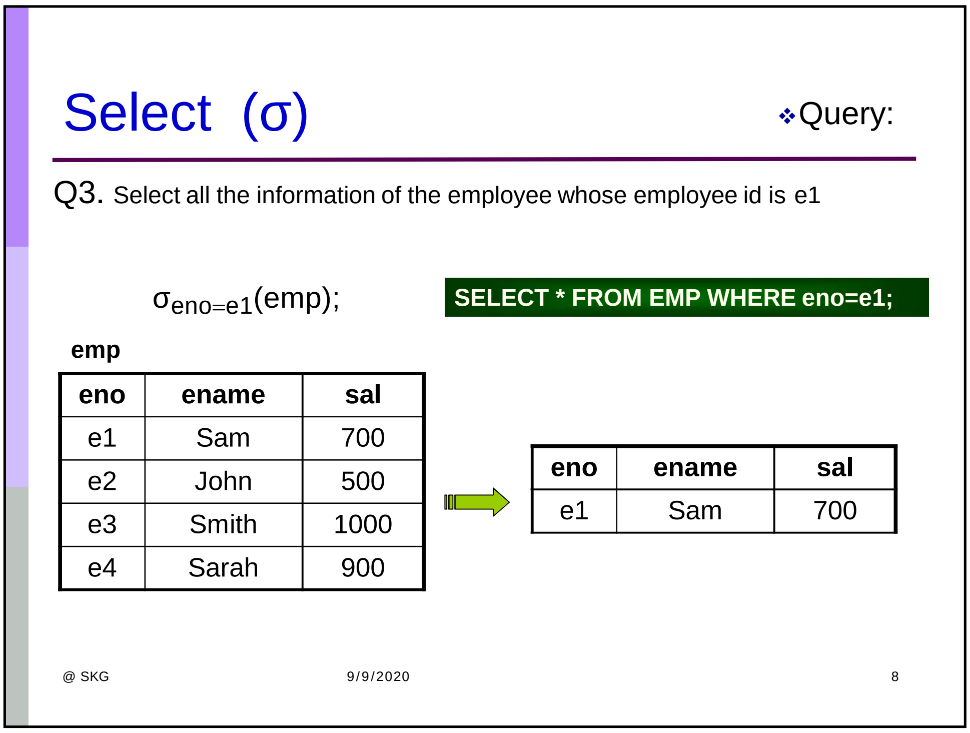
emp

eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

SELECT * FROM emp WHERE sal> 700 AND sal <1000;

eno	ename	sal
e4	Sarah	900

@ SKG 9/9/2020 7



Select (σ)

❖Query:

Q3. Select all the information of the employee whose employee id is e1

$\sigma_{eno=e1}$(emp);

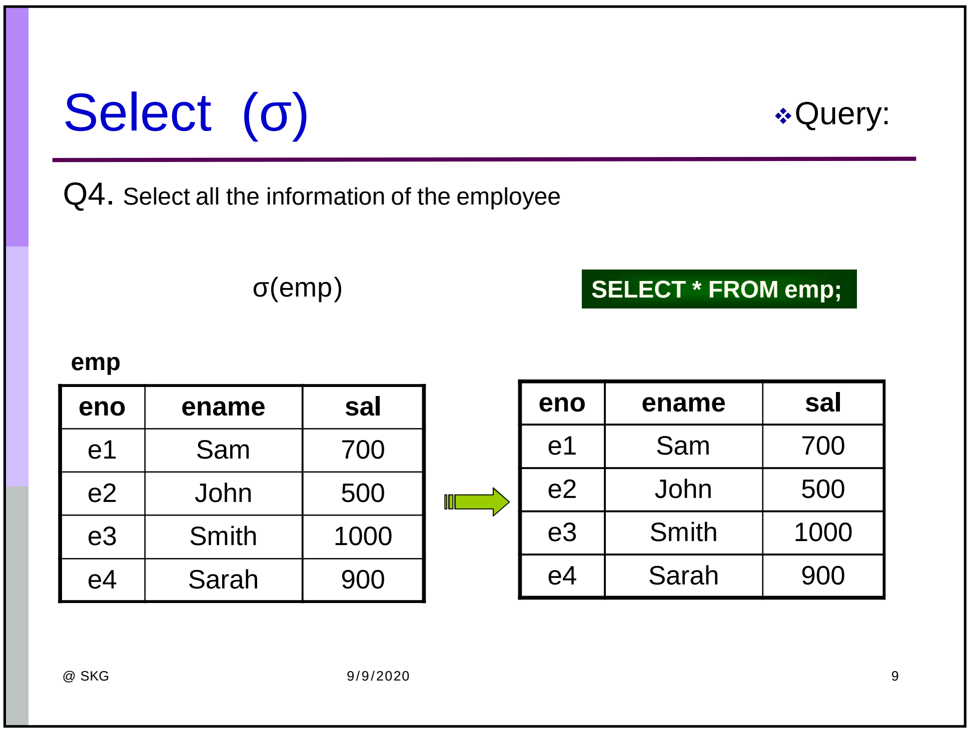
SELECT * FROM EMP WHERE eno=e1;

emp

eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

eno	ename	sal
e1	Sam	700

@ SKG 9/9/2020 8



Select (σ)

❖Query:

Q4. Select all the information of the employee

σ(emp)

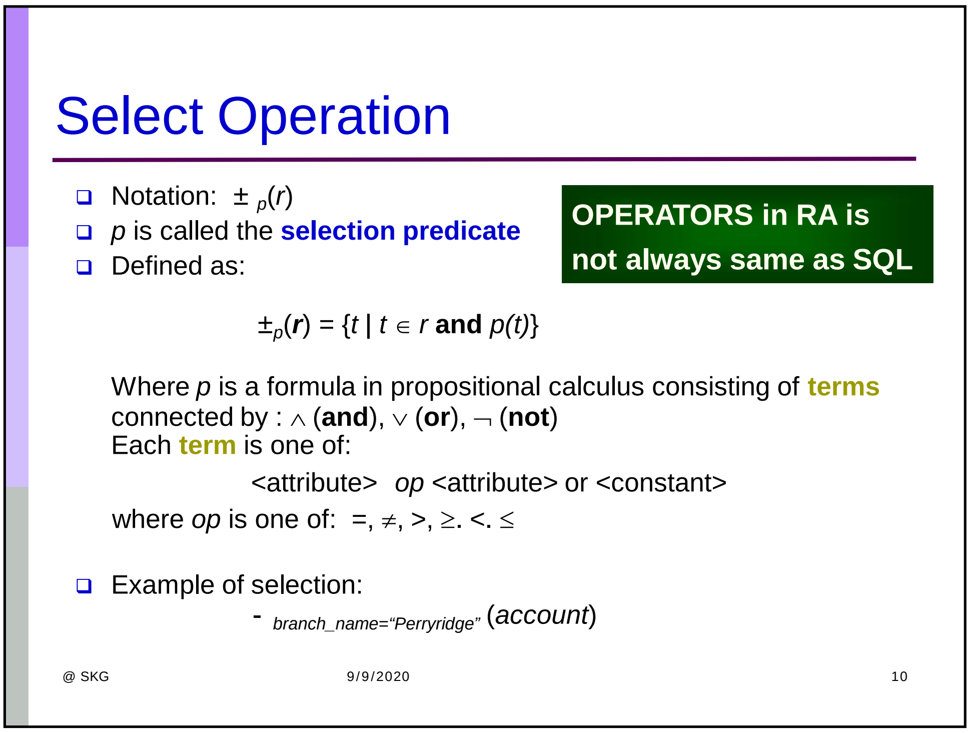
SELECT * FROM emp;

emp

eno	**ename**	**sal**
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

eno	**ename**	**sal**
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

@ SKG 9/9/2020 9



Select Operation

- Notation: $\pm_{p}(r)$
- *p* is called the **selection predicate**
- Defined as:

OPERATORS in RA is not always same as SQL

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

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

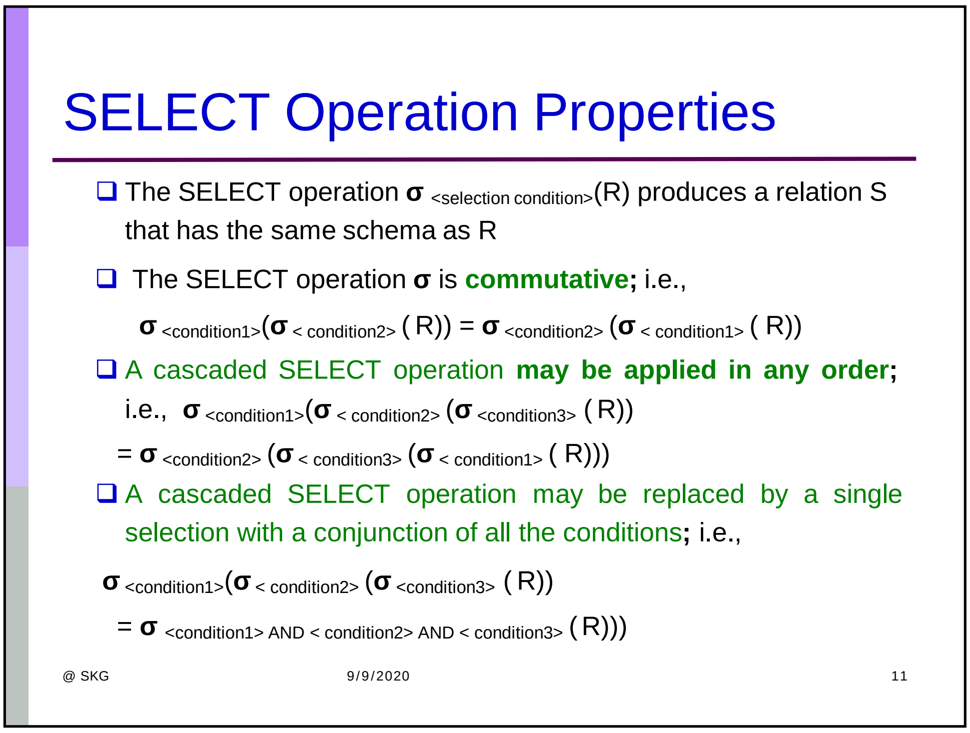
<attribute> *op* <attribute> or <constant>

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

- Example of selection:

- $_{\textit{branch\_name="Perryridge"}}$ (*account*)

@ SKG 9/9/2020 10



SELECT Operation Properties

- The SELECT operation $\sigma_{\text{<selection condition>}}(R)$ produces a relation S that has the same schema as R
- The SELECT operation σ is **commutative**; i.e.,

  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$
- A cascaded SELECT operation **may be applied in any order**;

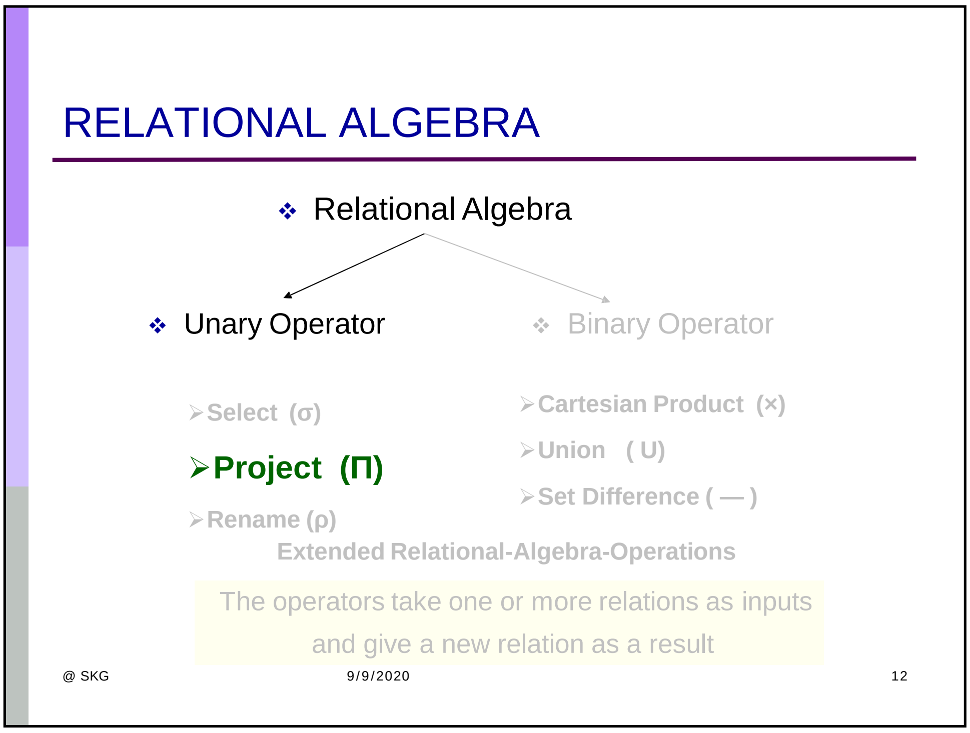
  i.e., $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$

  $= \sigma_{\text{<condition2>}}(\sigma_{\text{< condition3>}}(\sigma_{\text{< condition1>}}(R)))$
- A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions; i.e.,

  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$

  $= \sigma_{\text{<condition1> AND < condition2> AND < condition3>}}(R)))$

@ SKG 9/9/2020 11



RELATIONAL ALGEBRA

- Relational Algebra
  - Unary Operator
    - Select (σ)
    - **Project (Π)**
    - Rename (ρ)
  - Binary Operator
    - Cartesian Product (×)
    - Union (U)
    - Set Difference (—)

Extended Relational-Algebra-Operations

The operators take one or more relations as inputs and give a new relation as a result

@ SKG 9/9/2020 12

Project (Π)

$\pi_{<\text{attribute name}>}$ (Relation Name)

$\pi_{<attr_1, attr_2, ... >}$ (Relation Name)

❖Query:

Q5. Select the employee id of the employee.

`SELECT eno FROM emp;`

emp

eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

$\pi_{eno}(emp)$

eno
e1
e2
e3
e4

@ SKG 9/9/2020 13

Project (Π)

❖Query:

Q6. Select the employee id and the name of the employee

$\pi_{eno,ename}(emp)$

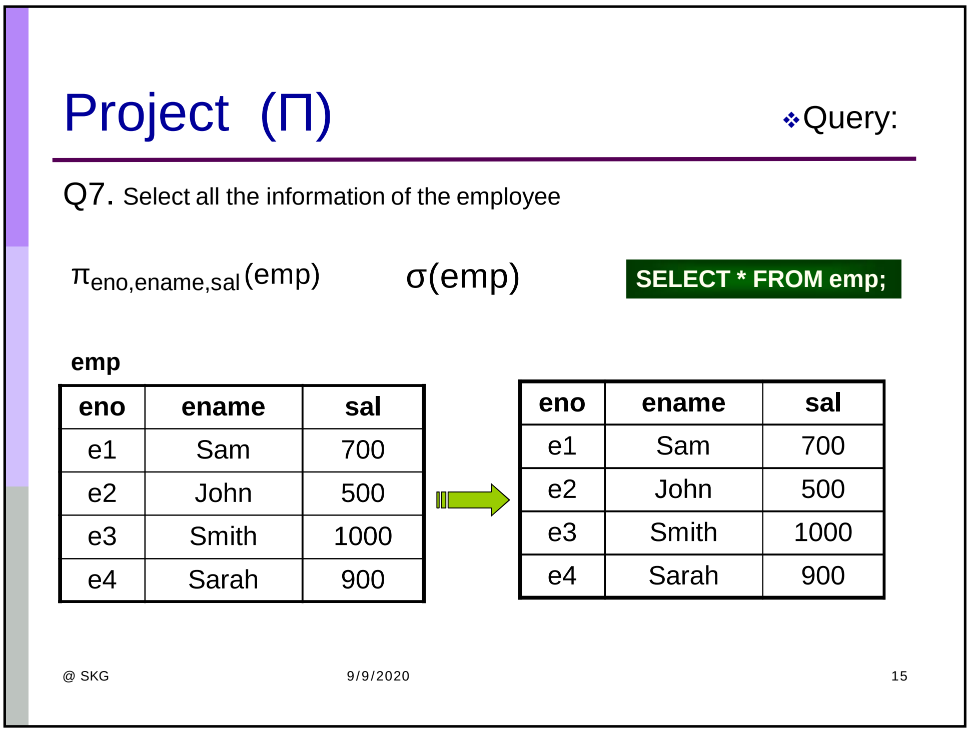
`SELECT eno, ename FROM emp;`

emp

eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

eno	ename
e1	Sam
e2	John
e3	Smith
e4	Sarah

@ SKG 9/9/2020 14



Project (Π)

❖Query:

Q7. Select all the information of the employee

$\pi_{eno,ename,sal}(emp)$ $\sigma(emp)$ **SELECT * FROM emp;**

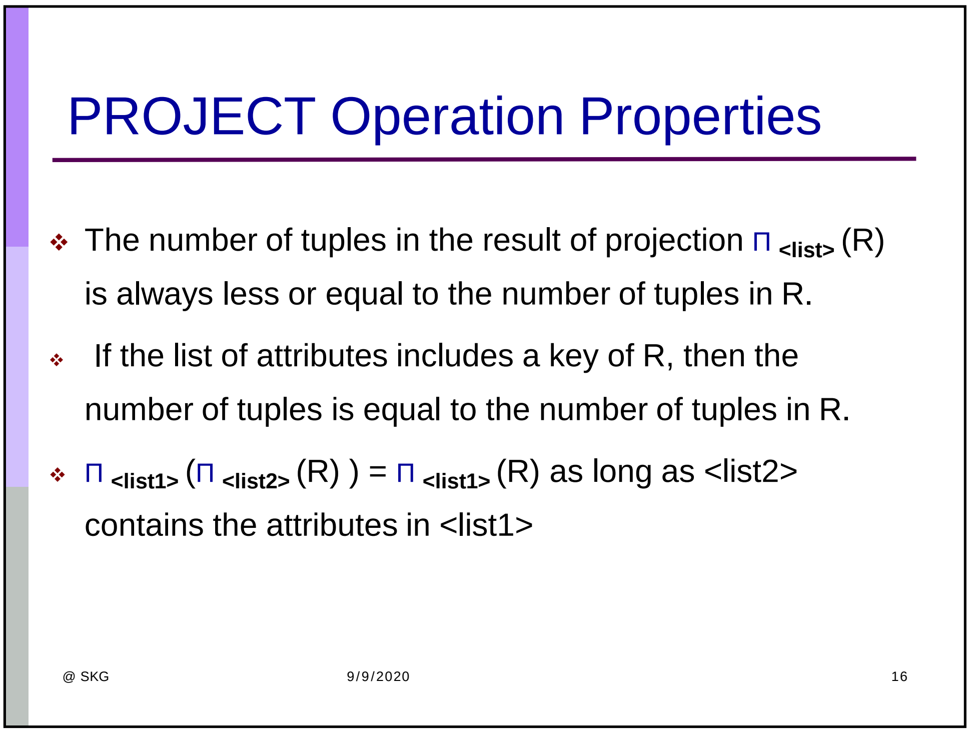
emp

eno	**ename**	**sal**
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

$\Rightarrow$

eno	**ename**	**sal**
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

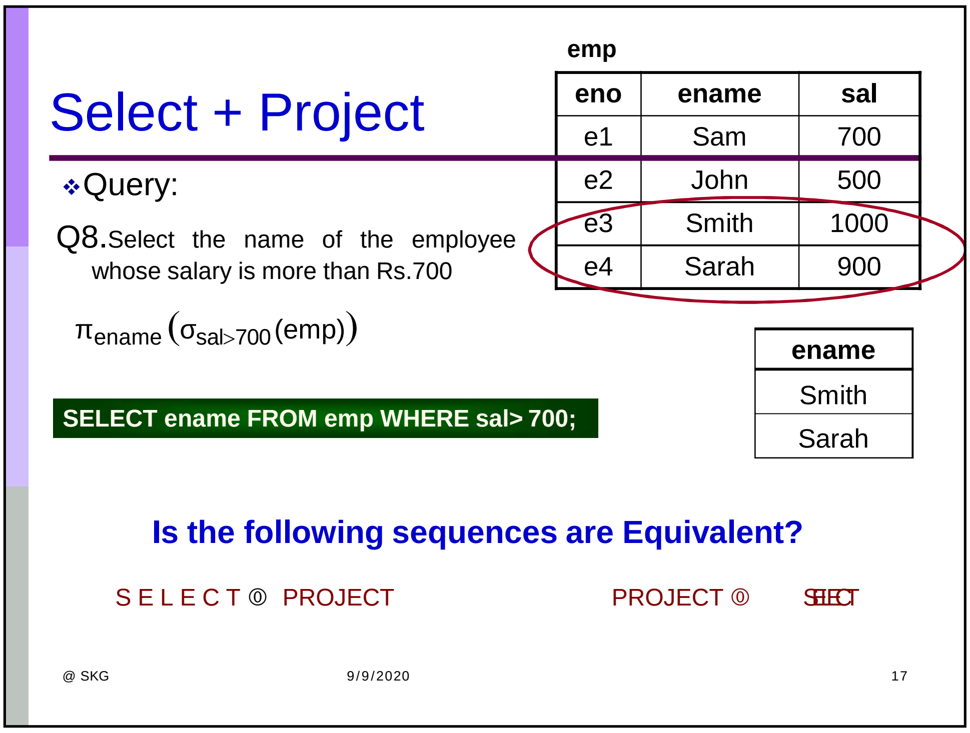
@ SKG 9/9/2020 15



PROJECT Operation Properties

- ❖ The number of tuples in the result of projection $\Pi_{<list>}(R)$ is always less or equal to the number of tuples in R.
- ❖ If the list of attributes includes a key of R, then the number of tuples is equal to the number of tuples in R.
- ❖ $\Pi_{<list1>}(\Pi_{<list2>}(R)) = \Pi_{<list1>}(R)$ as long as <list2> contains the attributes in <list1>

@ SKG 9/9/2020 16



Select + Project

❖Query:

Q8.Select the name of the employee whose salary is more than Rs.700

$\pi_{ename}(\sigma_{sal>700}(emp))$

SELECT ename FROM emp WHERE sal> 700;

emp

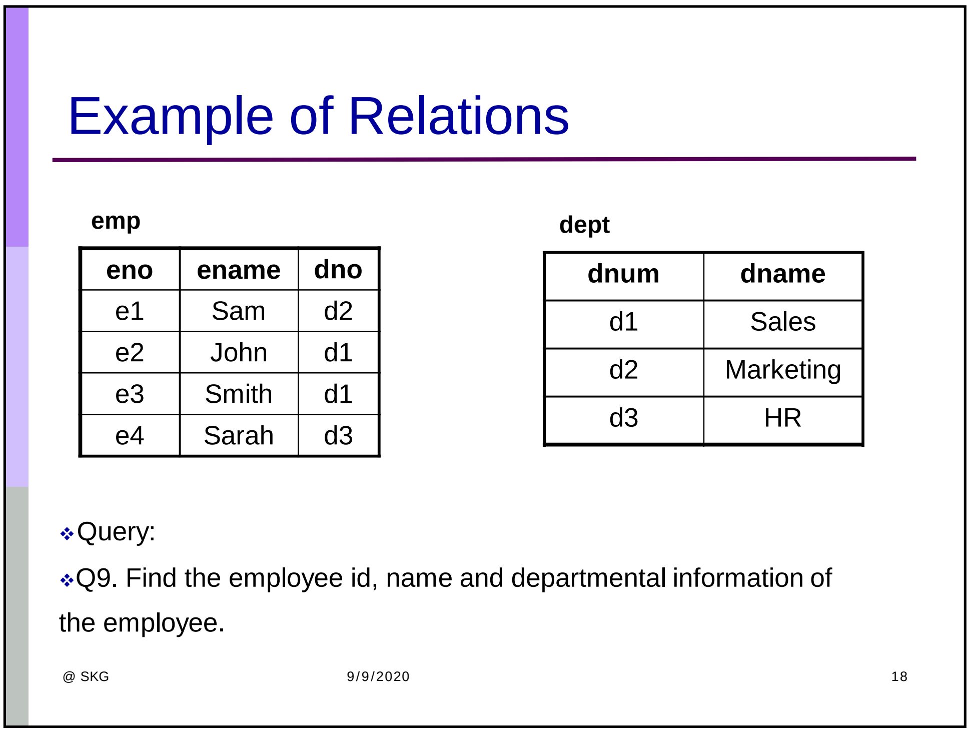
eno	ename	sal
e1	Sam	700
e2	John	500
e3	Smith	1000
e4	Sarah	900

ename
Smith
Sarah

Is the following sequences are Equivalent?

S E L E C T ⓪ PROJECT PROJECT ⓪ SELECT

@ SKG 9/9/2020 17



Example of Relations

emp

eno	ename	dno
e1	Sam	d2
e2	John	d1
e3	Smith	d1
e4	Sarah	d3

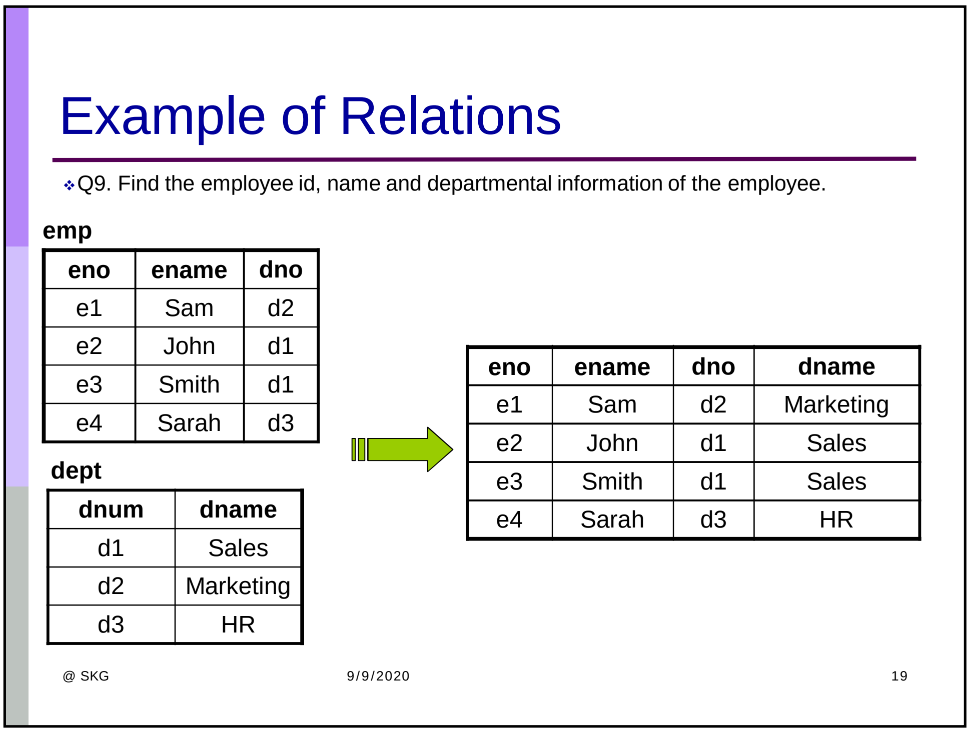
dept

dnum	dname
d1	Sales
d2	Marketing
d3	HR

❖Query:

❖Q9. Find the employee id, name and departmental information of the employee.

@ SKG 9/9/2020 18



Example of Relations

❖Q9. Find the employee id, name and departmental information of the employee.

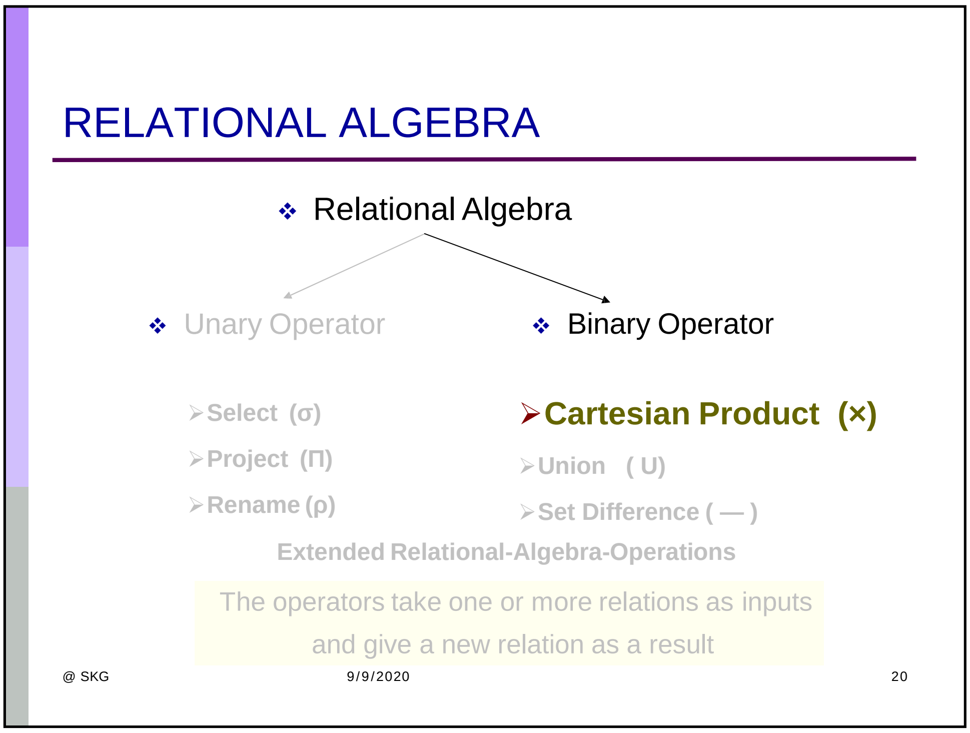
emp

eno	ename	dno
e1	Sam	d2
e2	John	d1
e3	Smith	d1
e4	Sarah	d3

dept

dnum	dname
d1	Sales
d2	Marketing
d3	HR

eno	ename	dno	dname
e1	Sam	d2	Marketing
e2	John	d1	Sales
e3	Smith	d1	Sales
e4	Sarah	d3	HR



RELATIONAL ALGEBRA

❖ Relational Algebra

❖ Unary Operator

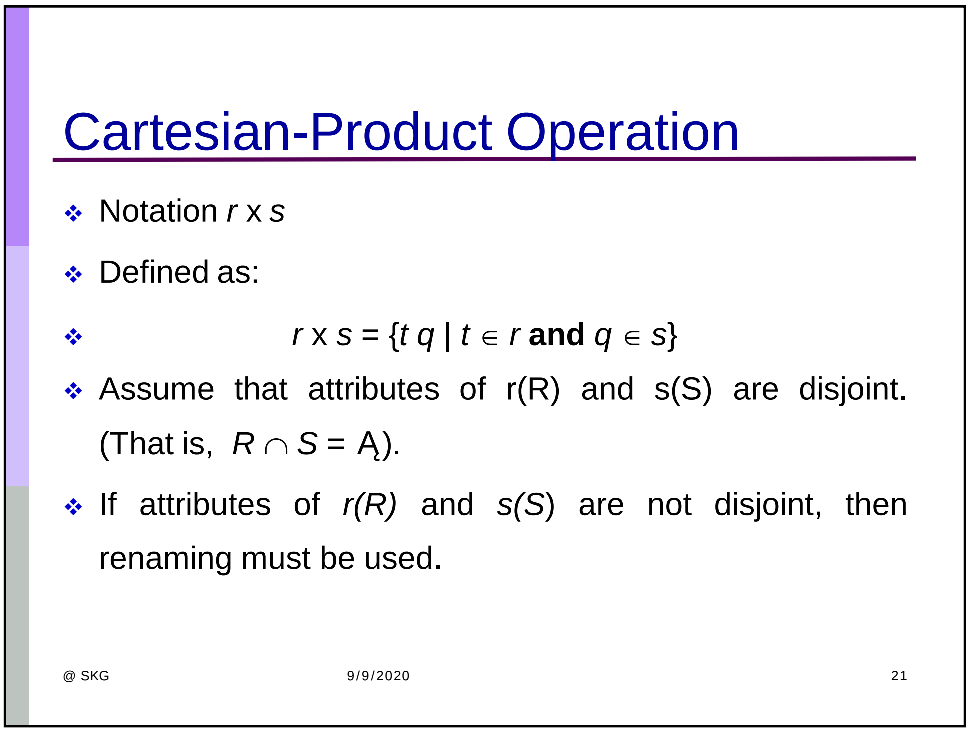
- Select (σ)
- Project (Π)
- Rename (ρ)

❖ Binary Operator

- **Cartesian Product (×)**
- Union (U)
- Set Difference (—)

Extended Relational-Algebra-Operations

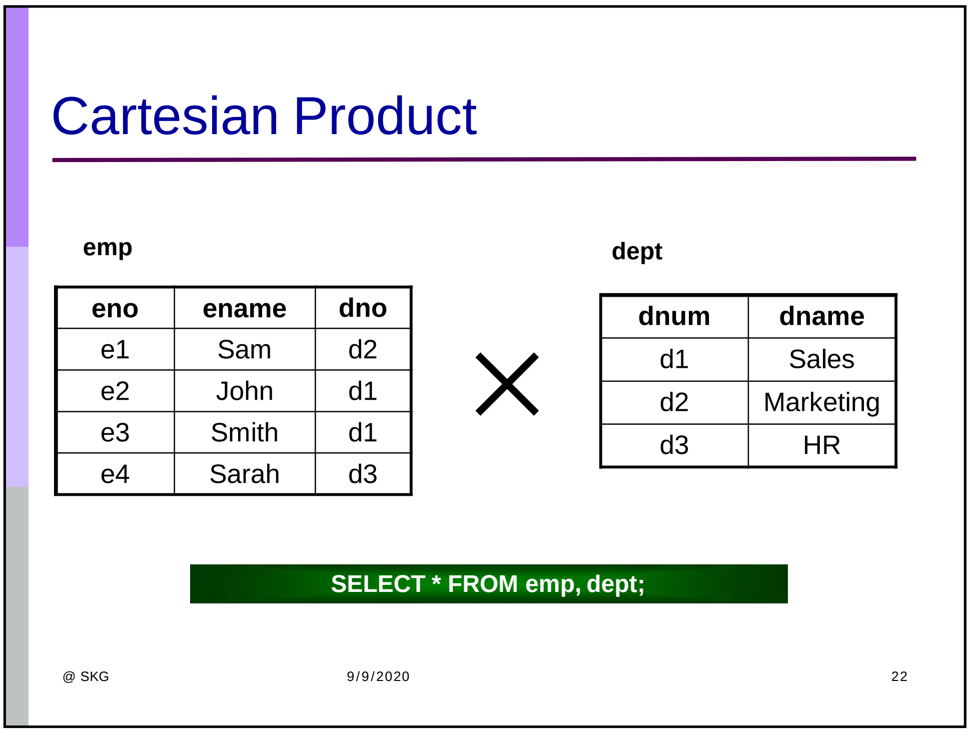
The operators take one or more relations as inputs and give a new relation as a result



Cartesian-Product Operation

- Notation *r* x *s*
- Defined as:
- $r \times s = \{t\, q \mid t \in r$ **and** $q \in s\}$
- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S$ = Ą).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

@ SKG 9/9/2020 21



Cartesian Product

emp

eno	**ename**	**dno**
e1	Sam	d2
e2	John	d1
e3	Smith	d1
e4	Sarah	d3

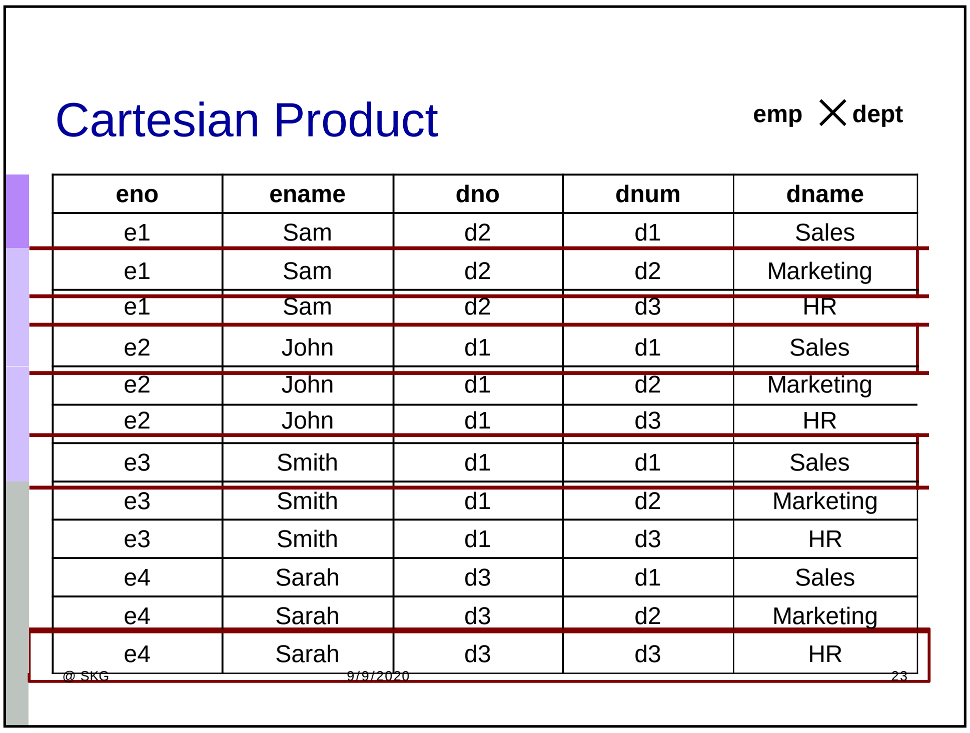
$\times$

dept

dnum	**dname**
d1	Sales
d2	Marketing
d3	HR

SELECT * FROM emp, dept;

@ SKG 9/9/2020 22

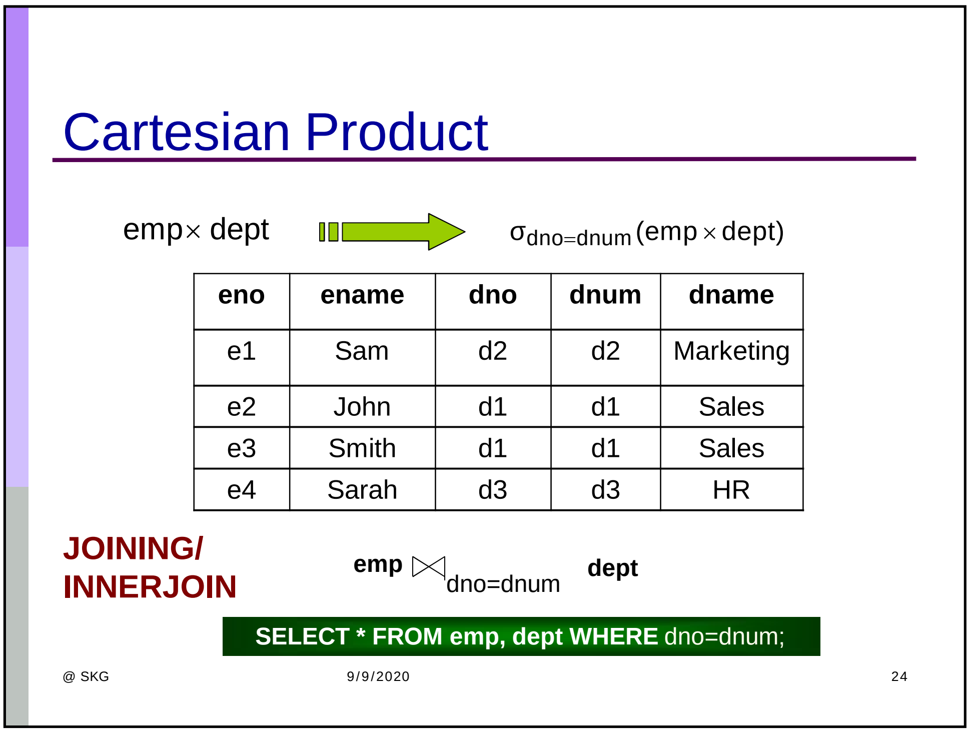


Cartesian Product

emp $\times$ **dept**

eno	ename	dno	dnum	dname
e1	Sam	d2	d1	Sales
e1	Sam	d2	d2	Marketing
e1	Sam	d2	d3	HR
e2	John	d1	d1	Sales
e2	John	d1	d2	Marketing
e2	John	d1	d3	HR
e3	Smith	d1	d1	Sales
e3	Smith	d1	d2	Marketing
e3	Smith	d1	d3	HR
e4	Sarah	d3	d1	Sales
e4	Sarah	d3	d2	Marketing
e4	Sarah	d3	d3	HR

@ SKG 9/9/2020 23



Cartesian Product

emp $\times$ dept $\Rightarrow$ $\sigma_{dno=dnum}(emp \times dept)$

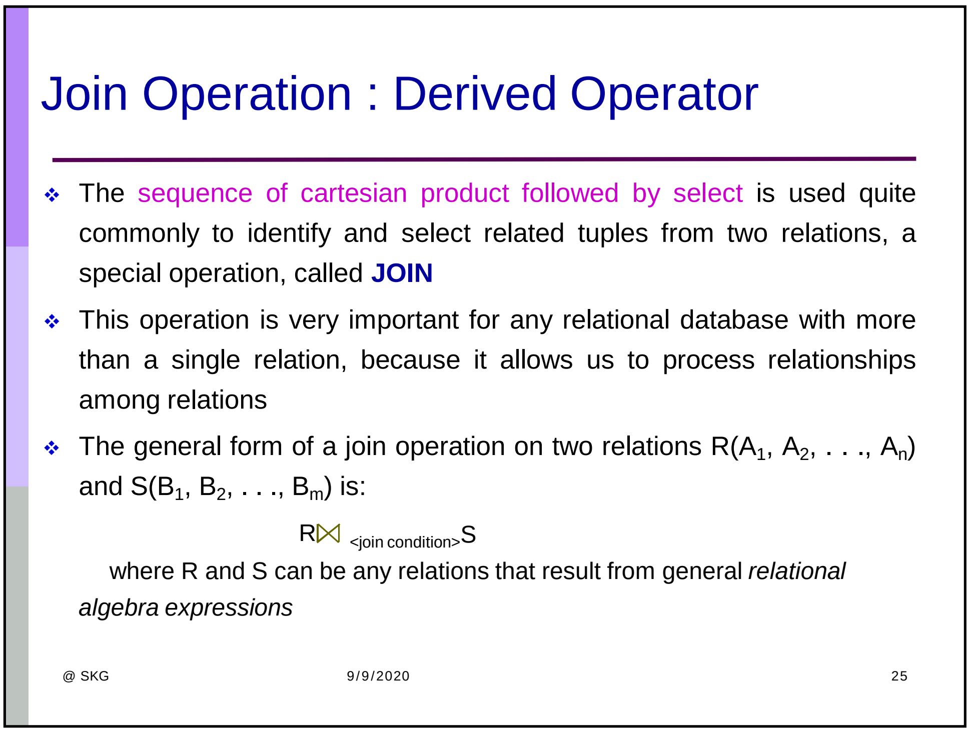
eno	ename	dno	dnum	dname
e1	Sam	d2	d2	Marketing
e2	John	d1	d1	Sales
e3	Smith	d1	d1	Sales
e4	Sarah	d3	d3	HR

JOINING/ INNERJOIN

emp $\bowtie_{dno=dnum}$ **dept**

SELECT * FROM emp, dept WHERE dno=dnum;

@ SKG 9/9/2020 24



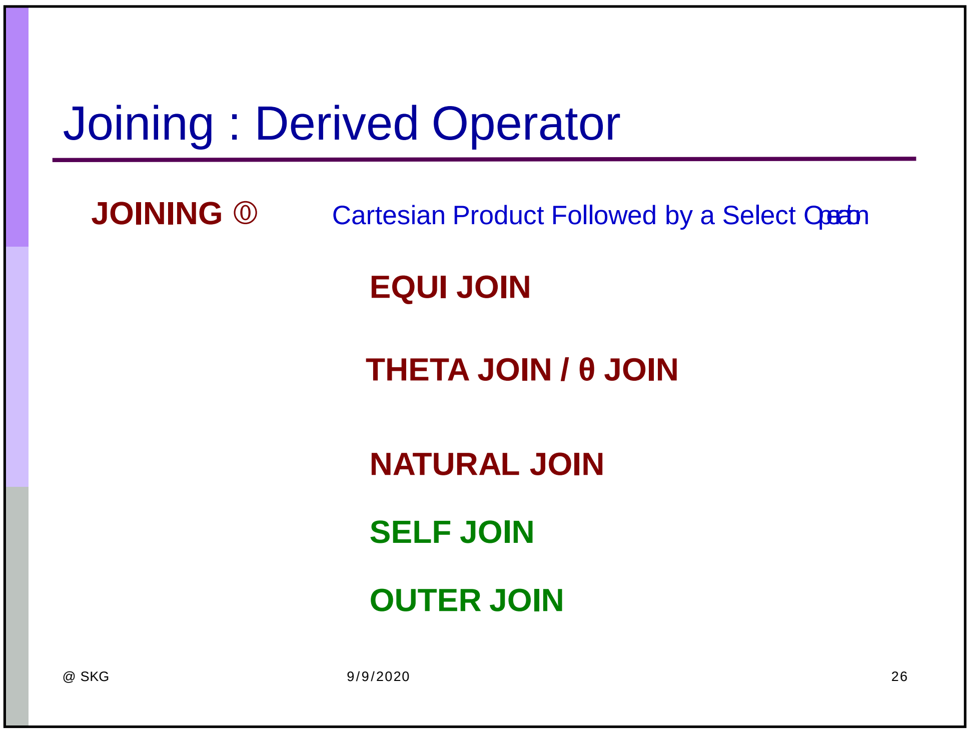
Join Operation : Derived Operator

- The sequence of cartesian product followed by select is used quite commonly to identify and select related tuples from two relations, a special operation, called **JOIN**
- This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations
- The general form of a join operation on two relations $R(A_1, A_2, . . ., A_n)$ and $S(B_1, B_2, . . ., B_m)$ is:

$$R \bowtie_{<join\ condition>} S$$

where R and S can be any relations that result from general *relational algebra expressions*

@ SKG 9/9/2020 25



Joining : Derived Operator

JOINING ⓪ Cartesian Product Followed by a Select Operation

EQUI JOIN

THETA JOIN / θ JOIN

NATURAL JOIN

SELF JOIN

OUTER JOIN

@ SKG 9/9/2020 26

Joining : Derived Operator

EQUI JOIN

$$\textbf{emp} \bowtie_{\text{dno=dnum}} \textbf{dept}$$

THETA JOIN / θ JOIN

$$\textbf{emp} \bowtie_{\text{dno=dnum} \wedge \text{sal> 700}} \textbf{dept}$$

$$\textbf{R} \bowtie_{\theta} \textbf{S}$$

Cartesian Product

THETA JOIN

$$\textbf{emp} \bowtie_{\text{dno=dnum}} \textbf{dept}$$

emp

eno	ename	sal	dno
e1	Sam	100	d2
e2	John	800	d1
e3	Smith	1000	d1
e4	Sarah	400	d3

dept

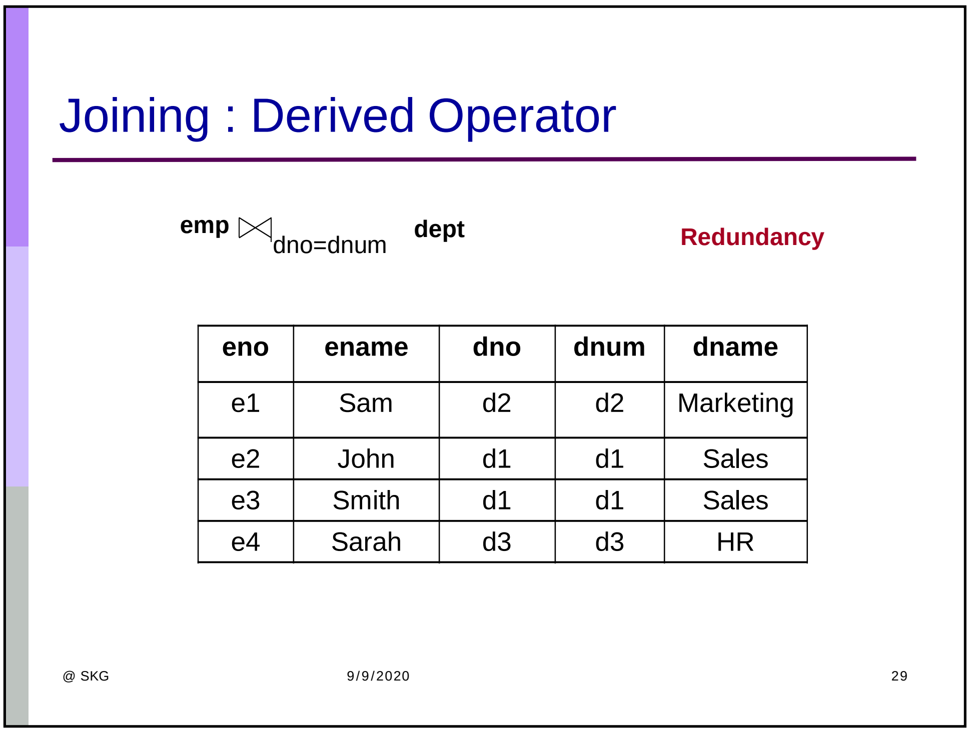
dnum	dname
d1	Sales
d2	Marketing
d3	HR

eno	ename	sal	dno	dnum	dname
e1	Sam	100	d2	d2	Marketing
e2	John	800	d1	d1	Sales
e3	Smith	1000	d1	d1	Sales
e4	Sarah	400	d3	d3	HR

$$\textbf{emp} \bowtie_{\text{dno=dnum} \wedge \text{sal> 700}} \textbf{dept}$$

SELECT * FROM emp, dept WHERE dno=dnum AND sal>700;

eno	ename	sal	dno	dnum	dname
e2	John	800	d1	d1	Sales
e3	Smith	1000	d1	d1	Sales



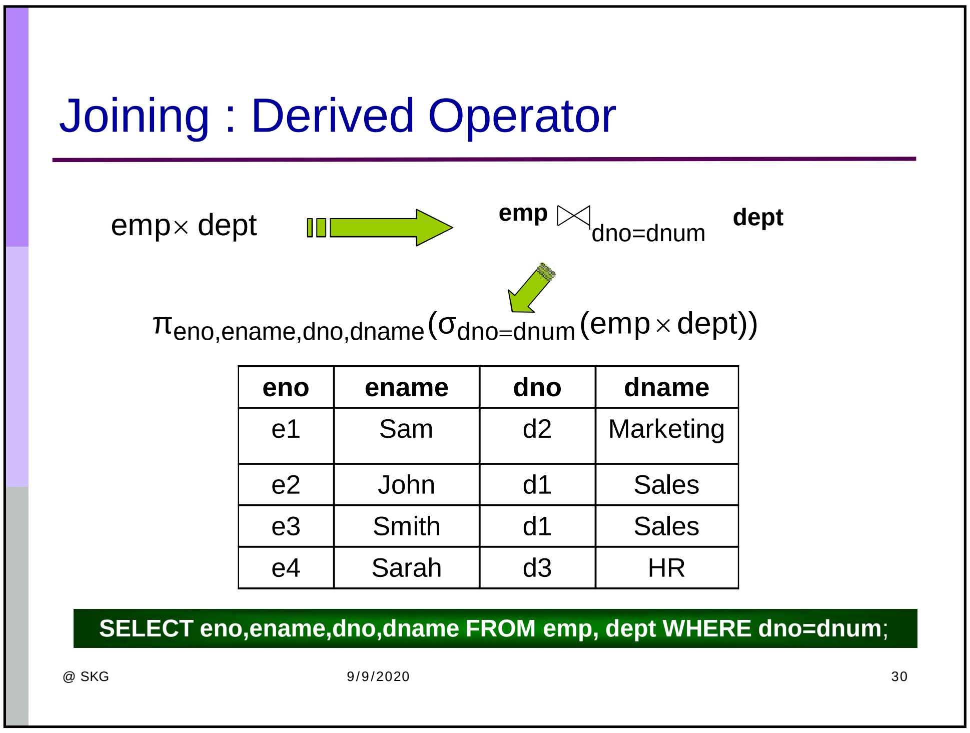
Joining : Derived Operator

emp $\bowtie_{dno=dnum}$ **dept**

Redundancy

eno	ename	dno	dnum	dname
e1	Sam	d2	d2	Marketing
e2	John	d1	d1	Sales
e3	Smith	d1	d1	Sales
e4	Sarah	d3	d3	HR

@ SKG 9/9/2020 29



Joining : Derived Operator

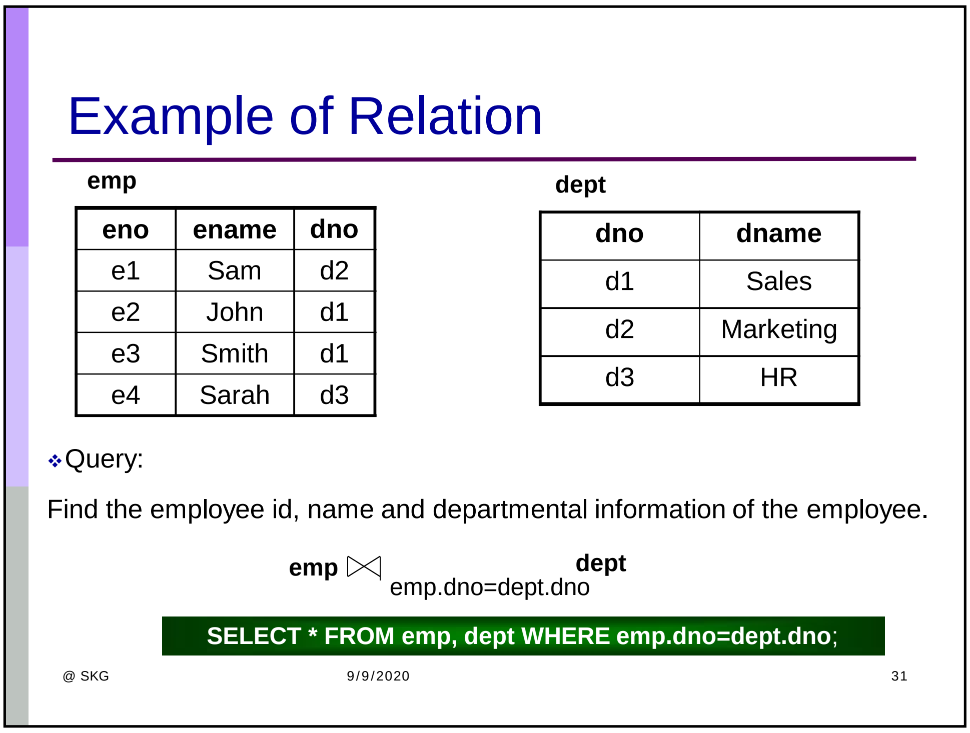
$\text{emp} \times \text{dept}$ $\Longrightarrow$ **emp** $\bowtie_{dno=dnum}$ **dept**

$$\pi_{eno,ename,dno,dname}(\sigma_{dno=dnum}(\text{emp} \times \text{dept}))$$

eno	ename	dno	dname
e1	Sam	d2	Marketing
e2	John	d1	Sales
e3	Smith	d1	Sales
e4	Sarah	d3	HR

SELECT eno,ename,dno,dname FROM emp, dept WHERE dno=dnum;

@ SKG 9/9/2020 30



Example of Relation

emp

eno	ename	dno
e1	Sam	d2
e2	John	d1
e3	Smith	d1
e4	Sarah	d3

dept

dno	dname
d1	Sales
d2	Marketing
d3	HR

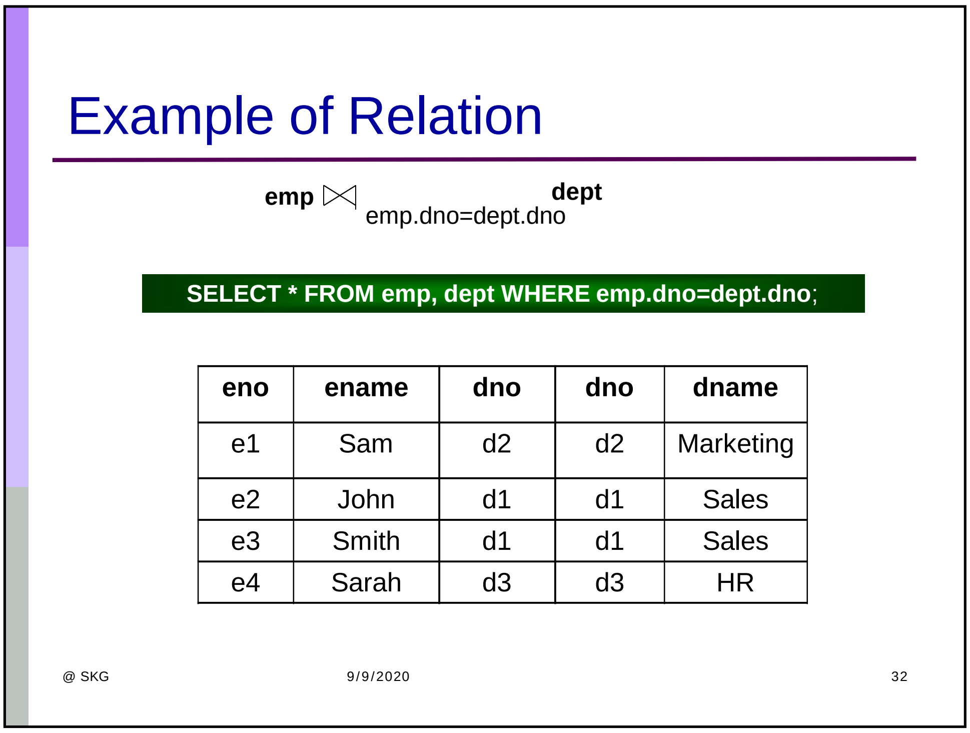
- Query:

Find the employee id, name and departmental information of the employee.

$$\textbf{emp} \bowtie_{\text{emp.dno=dept.dno}} \textbf{dept}$$

```
SELECT * FROM emp, dept WHERE emp.dno=dept.dno;
```

@ SKG 9/9/2020 31



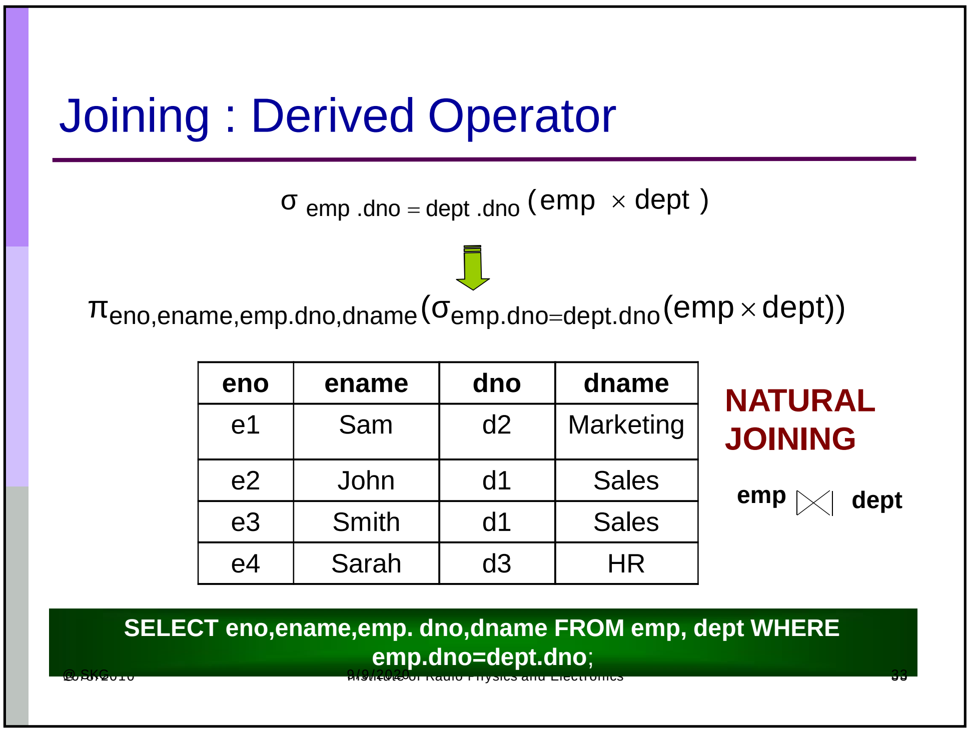
Example of Relation

$$\textbf{emp} \bowtie_{\text{emp.dno=dept.dno}} \textbf{dept}$$

```
SELECT * FROM emp, dept WHERE emp.dno=dept.dno;
```

eno	ename	dno	dno	dname
e1	Sam	d2	d2	Marketing
e2	John	d1	d1	Sales
e3	Smith	d1	d1	Sales
e4	Sarah	d3	d3	HR

@ SKG 9/9/2020 32



Joining : Derived Operator

$$\sigma_{emp.dno = dept.dno}(emp \times dept)$$

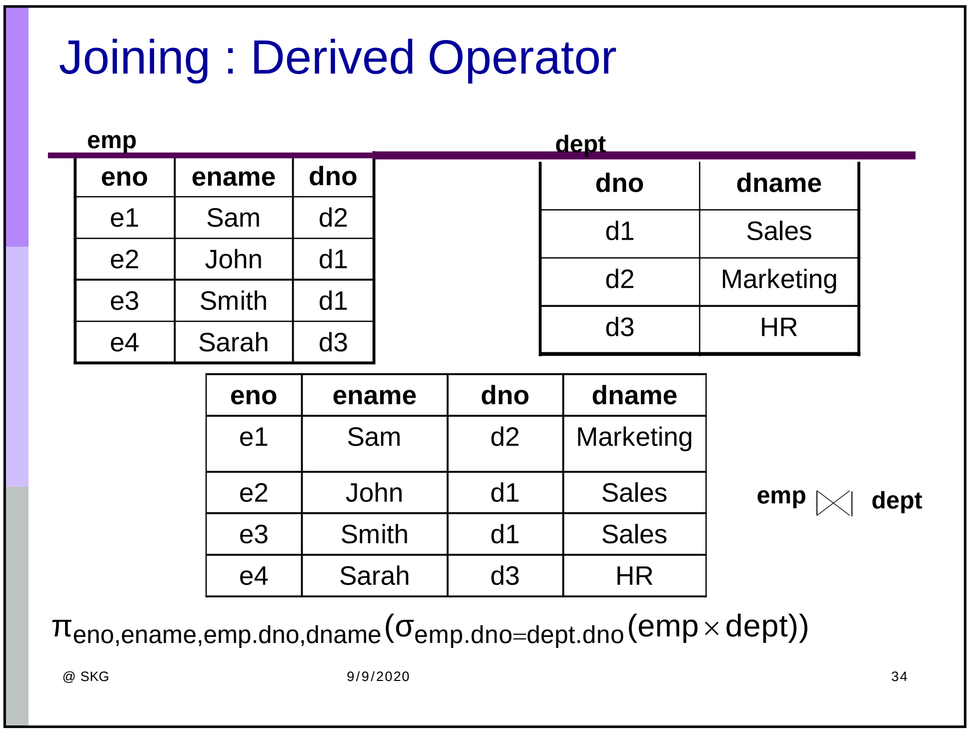
$$\pi_{eno,ename,emp.dno,dname}(\sigma_{emp.dno=dept.dno}(emp \times dept))$$

eno	ename	dno	dname
e1	Sam	d2	Marketing
e2	John	d1	Sales
e3	Smith	d1	Sales
e4	Sarah	d3	HR

NATURAL JOINING

emp $\bowtie$ **dept**

SELECT eno,ename,emp. dno,dname FROM emp, dept WHERE emp.dno=dept.dno;



Joining : Derived Operator

emp

eno	ename	dno
e1	Sam	d2
e2	John	d1
e3	Smith	d1
e4	Sarah	d3

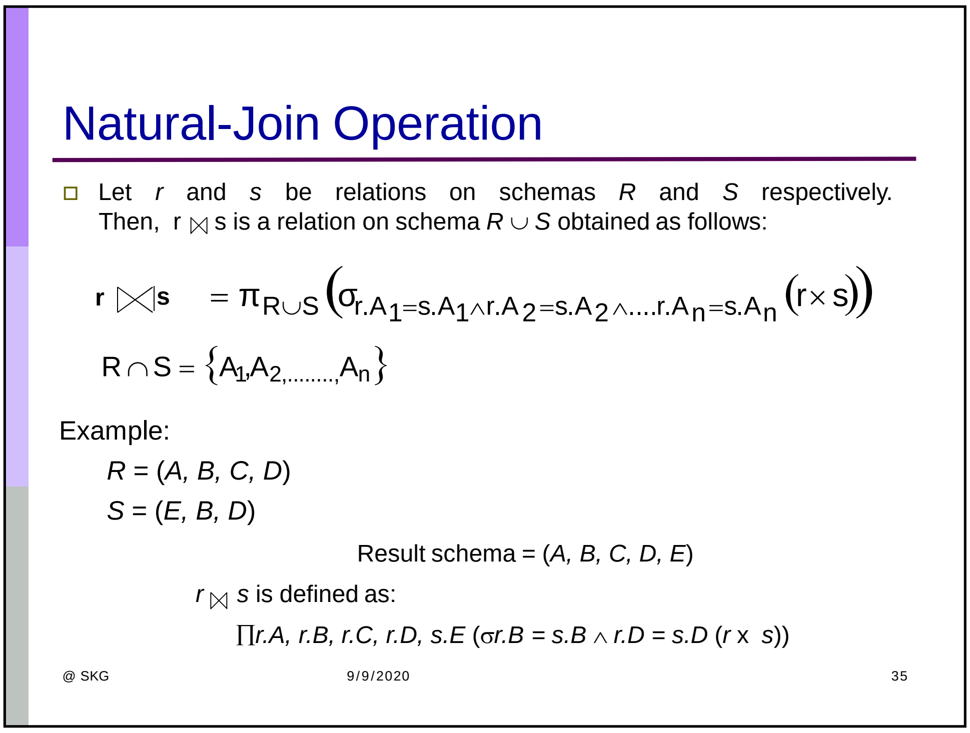
dept

dno	dname
d1	Sales
d2	Marketing
d3	HR

eno	ename	dno	dname
e1	Sam	d2	Marketing
e2	John	d1	Sales
e3	Smith	d1	Sales
e4	Sarah	d3	HR

emp $\bowtie$ **dept**

$$\pi_{eno,ename,emp.dno,dname}(\sigma_{emp.dno=dept.dno}(emp \times dept))$$



Natural-Join Operation

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, r ⋈ s is a relation on schema *R* ∪ *S* obtained as follows:

$$r \bowtie s = \pi_{R \cup S}\left(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots r.A_n = s.A_n}(r \times s)\right)$$

$$R \cap S = \{A_1, A_2, \ldots\ldots\ldots, A_n\}$$

Example:

R = (*A, B, C, D*)

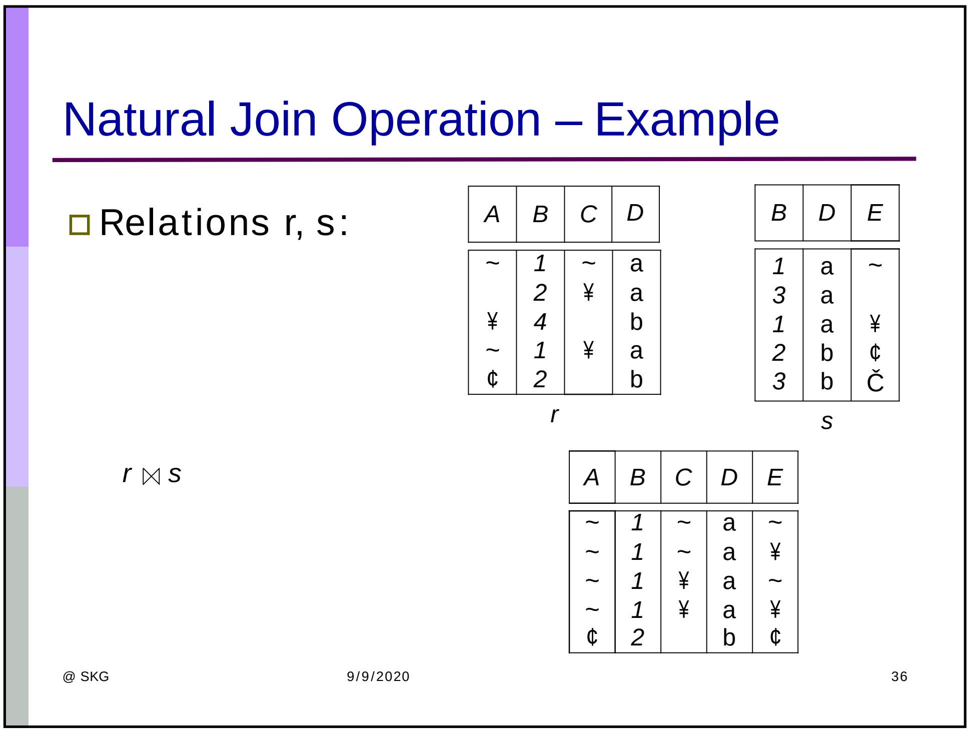
S = (*E, B, D*)

Result schema = (*A, B, C, D, E*)

r ⋈ *s* is defined as:

$\prod r.A, r.B, r.C, r.D, s.E\ (\sigma r.B = s.B \wedge r.D = s.D\ (r \times s))$

@ SKG 9/9/2020 35



Natural Join Operation – Example

- Relations r, s:

A	*B*	*C*	*D*
~	*1*	~	a
	2	¥	a
¥	*4*		b
~	*1*	¥	a
¢	*2*		b

r

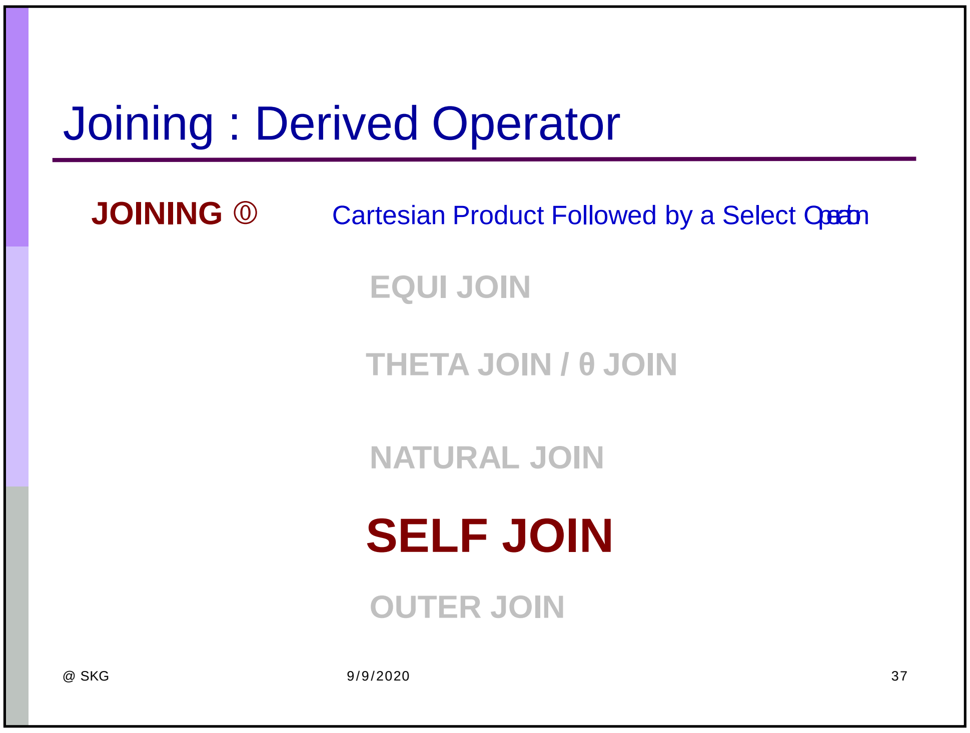
B	*D*	*E*
1	a	~
3	a	
1	a	¥
2	b	¢
3	b	Č

s

r ⋈ *s*

A	*B*	*C*	*D*	*E*
~	*1*	~	a	~
~	*1*	~	a	¥
~	*1*	¥	a	~
~	*1*	¥	a	¥
¢	*2*		b	¢

@ SKG 9/9/2020 36



Joining : Derived Operator

JOINING ⓪ Cartesian Product Followed by a Select Operation

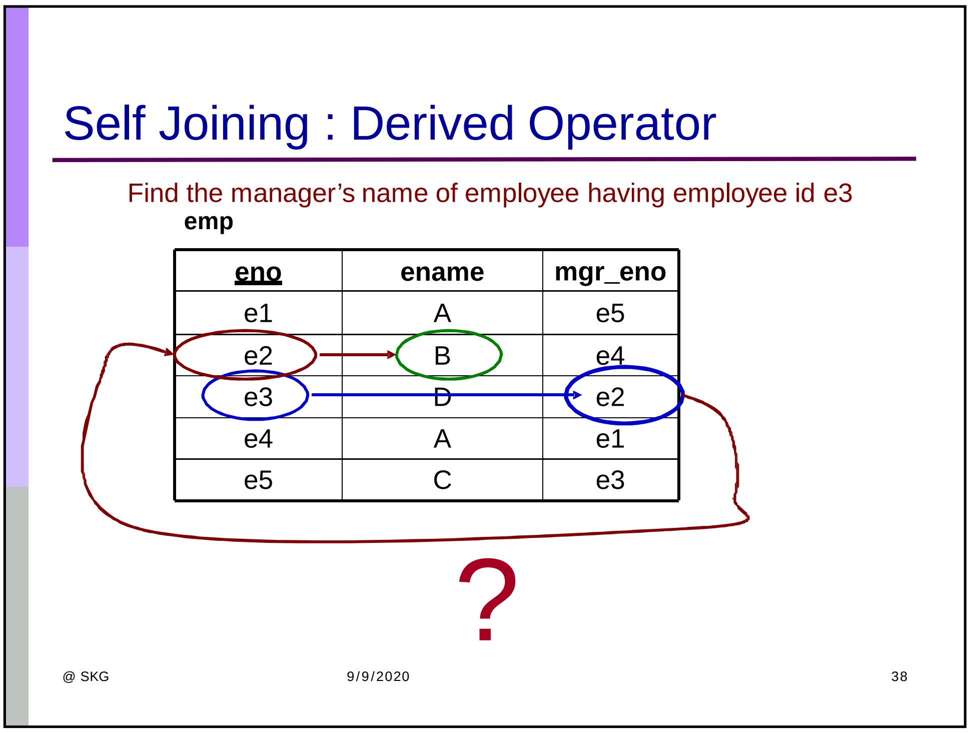
EQUI JOIN

THETA JOIN / θ JOIN

NATURAL JOIN

SELF JOIN

OUTER JOIN



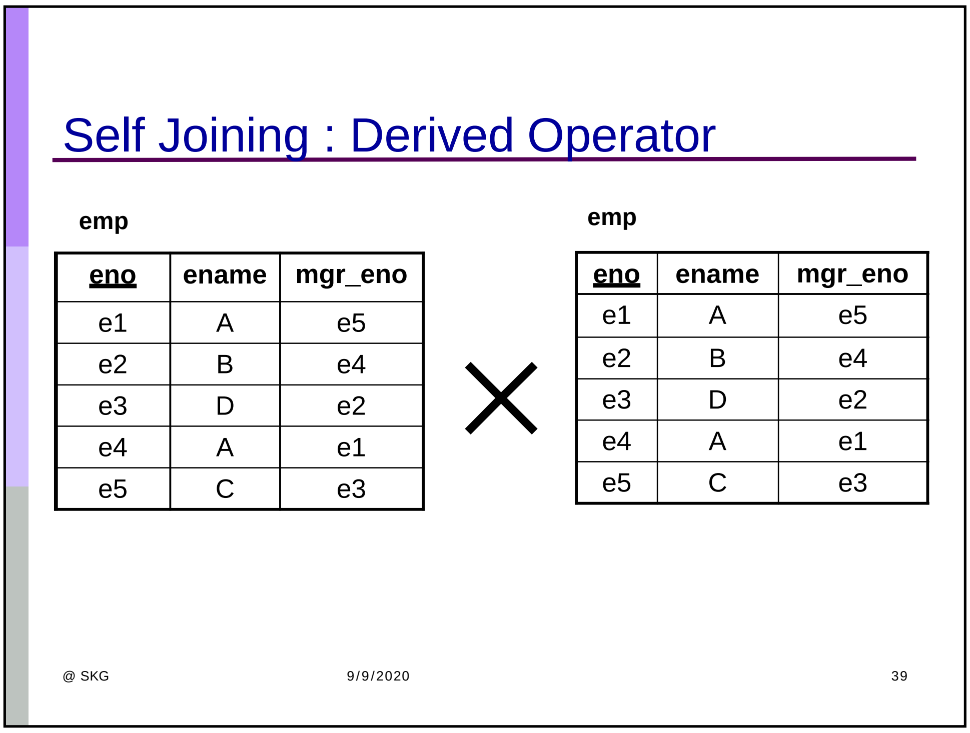
Self Joining : Derived Operator

Find the manager’s name of employee having employee id e3

emp

eno	**ename**	**mgr_eno**
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

?



Self Joining : Derived Operator

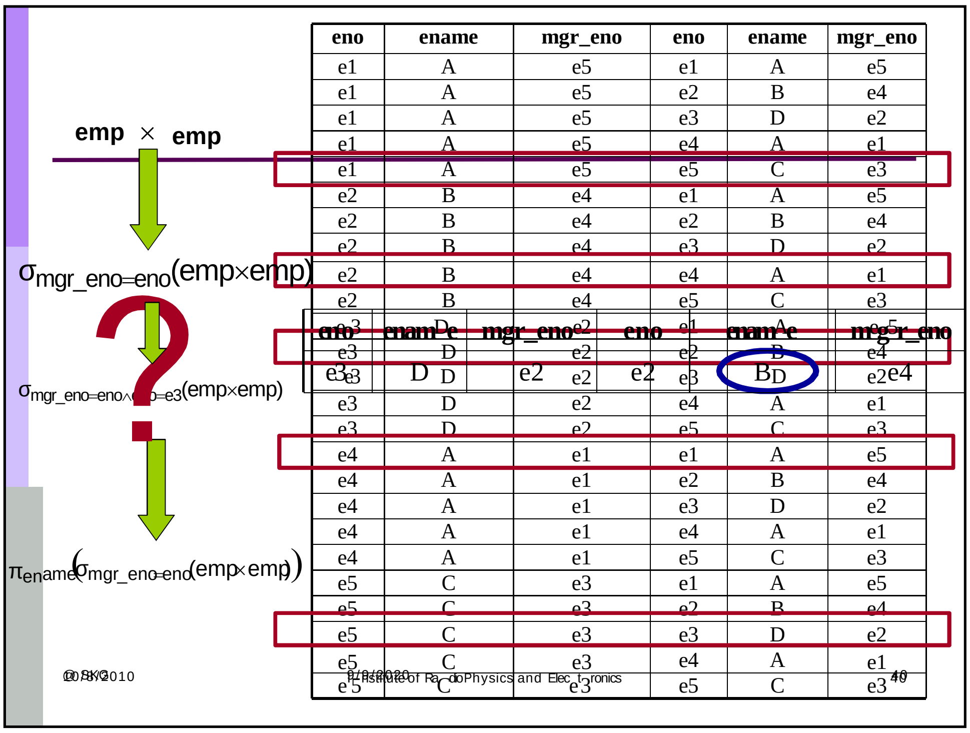
emp

<u>eno</u>	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

$\times$

emp

<u>eno</u>	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3



emp $\times$ emp

$\sigma_{mgr\_eno=eno}(emp\times emp)$

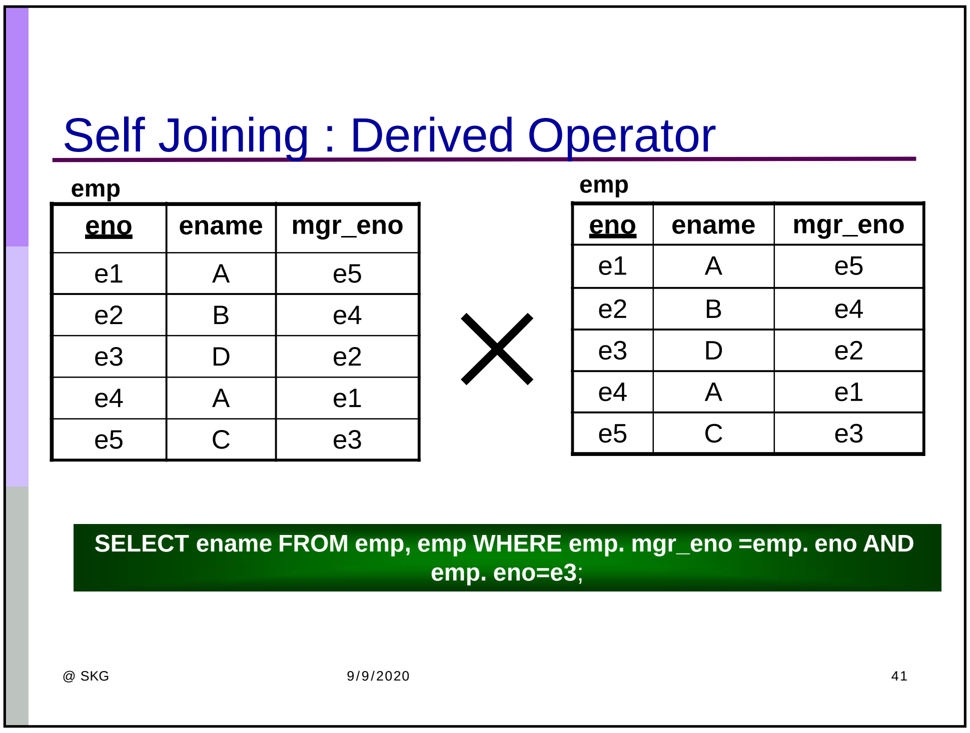
?

$\sigma_{mgr\_eno=eno\wedge eno=e3}(emp\times emp)$

$\pi_{ename}(\sigma_{mgr\_eno=eno}(emp\times emp))$

eno	ename	mgr_eno	eno	ename	mgr_eno
e1	A	e5	e1	A	e5
e1	A	e5	e2	B	e4
e1	A	e5	e3	D	e2
e1	A	e5	e4	A	e1
e1	A	e5	e5	C	e3
e2	B	e4	e1	A	e5
e2	B	e4	e2	B	e4
e2	B	e4	e3	D	e2
e2	B	e4	e4	A	e1
e2	B	e4	e5	C	e3
e3	D	e2	e1	A	e5
e3	D	e2	e2	B	e4
e3	D	e2	e3	D	e2
e3	D	e2	e4	A	e1
e3	D	e2	e5	C	e3
e4	A	e1	e1	A	e5
e4	A	e1	e2	B	e4
e4	A	e1	e3	D	e2
e4	A	e1	e4	A	e1
e4	A	e1	e5	C	e3
e5	C	e3	e1	A	e5
e5	C	e3	e2	B	e4
e5	C	e3	e3	D	e2
e5	C	e3	e4	A	e1
e5	C	e3	e5	C	e3

eno	ename	mgr_eno	eno	ename	mgr_eno
e3	D	e2	e2	B	e4



Self Joining : Derived Operator

emp

eno	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

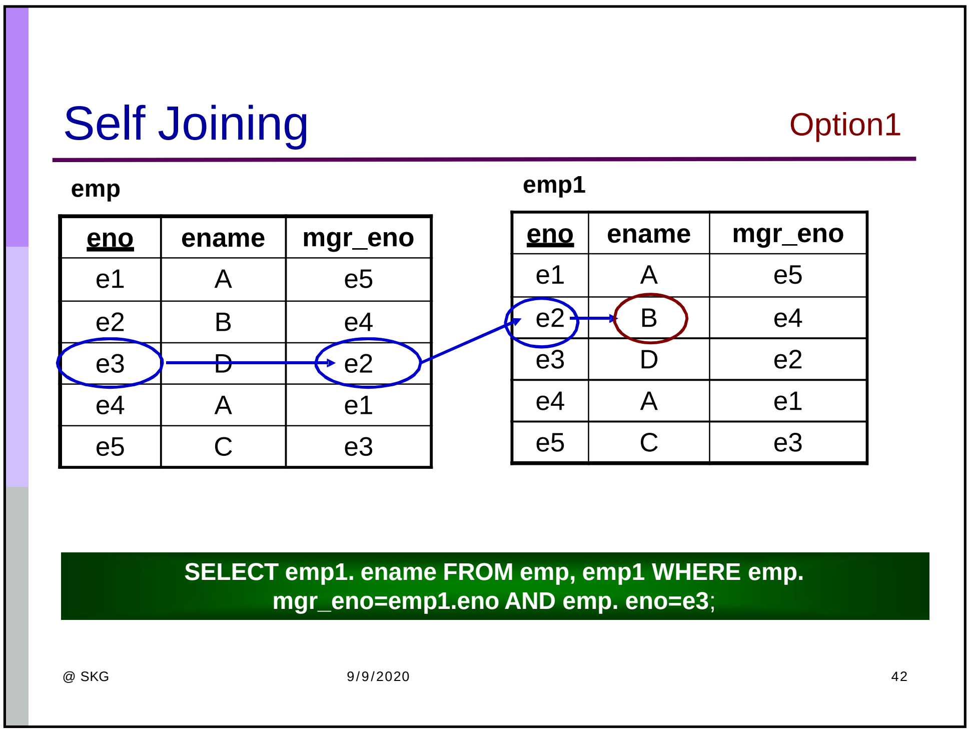
$\times$

emp

eno	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

SELECT ename FROM emp, emp WHERE emp. mgr_eno =emp. eno AND emp. eno=e3;

@ SKG 9/9/2020 41



Self Joining

Option1

emp

eno	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

emp1

eno	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

SELECT emp1. ename FROM emp, emp1 WHERE emp. mgr_eno=emp1.eno AND emp. eno=e3;

@ SKG 9/9/2020 42

Self Joining

Option2

RENAME

emp

<u>eno</u>	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e2
e4	A	e1
e5	C	e3

RELATIONAL ALGEBRA

❖ Relational Algebra

❖ Unary Operator

- ➢Select (σ)
- ➢Project (Π)
- ➢**Rename (ρ)**

❖ Binary Operator

- ➢Cartesian Product (×)
- ➢Union (U)
- ➢Set Difference (—)

Extended Relational-Algebra-Operations

The operators take one or more relations as inputs and give a new relation as a result

Rename Operator

❖Refer a relation by more than one name. $\rho_X(E);$

returns the expression *E* under the name *X*

SELECT * FROM emp employee;

❖Refer the relation and the attribute of the relation by more than one name

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

SELECT eno eid, ename name, mgr_eno supervisor_id FROM emp employee;

Self Joining

Option2

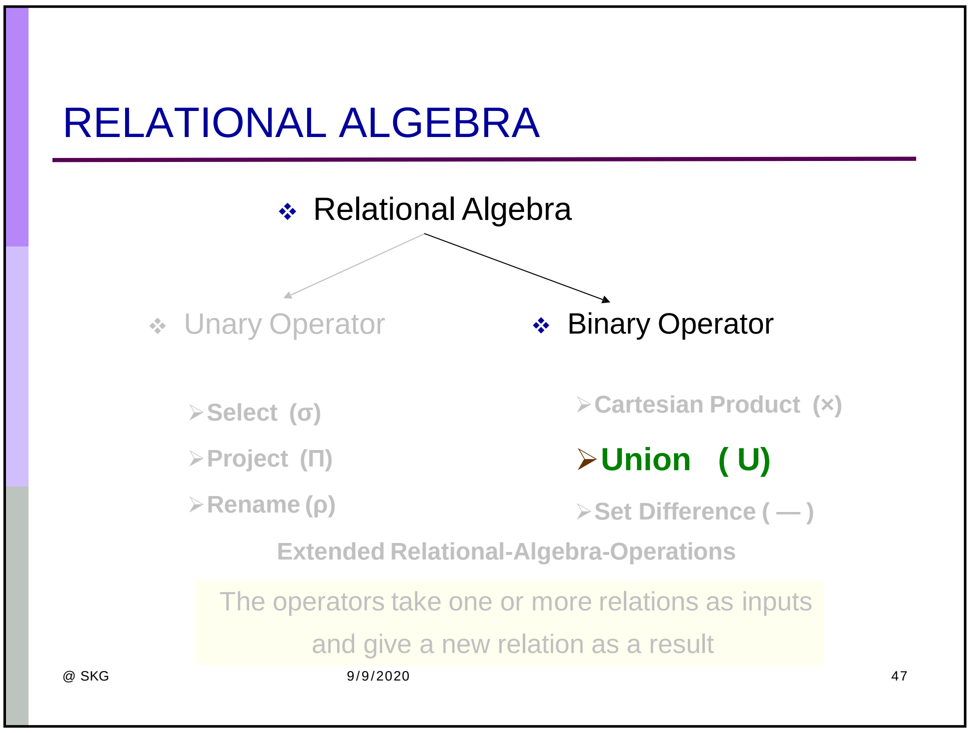
RENAME

emp

eno	ename	mgr_eno
e1	A	e5
e2	B	e4
e3	D	e3
e4	A	e1
e5	C	e2

SELECT emp1. ename FROM emp, emp1 WHERE emp. mgr_eno=emp1.eno AND emp. eno=e3;

SELECT b. ename FROM emp a, emp b WHERE a. mgr_eno=b.eno AND a. eno=e3;



RELATIONAL ALGEBRA

❖ Relational Algebra

❖ Unary Operator

- ➢Select (σ)
- ➢Project (Π)
- ➢Rename (ρ)

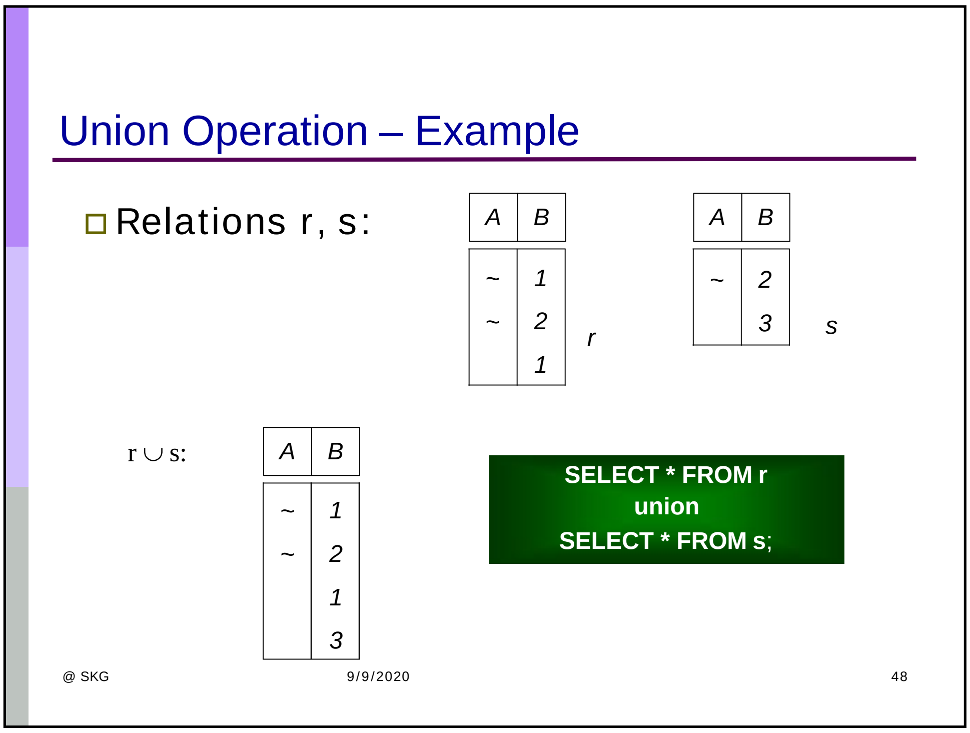
❖ Binary Operator

- ➢Cartesian Product (×)
- ➢**Union (U)**
- ➢Set Difference (—)

Extended Relational-Algebra-Operations

The operators take one or more relations as inputs and give a new relation as a result

@ SKG 9/9/2020 47



Union Operation – Example

□ Relations r, s:

r

A	B
~	1
~	2
	1

s

A	B
~	2
	3

r ∪ s:

A	B
~	1
~	2
	1
	3

```
SELECT * FROM r
union
SELECT * FROM s;
```

@ SKG 9/9/2020 48

Union Operation

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

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

- For $r \cup s$ to be valid

1. *r, s* must have the same number of attributes

2.The attribute domains must be *compatible* (e.g., 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

3.The sequence of the attributes in the both the relation must be same

$$n(rUs)=n(r)+n(s)-n(r\cap s)$$

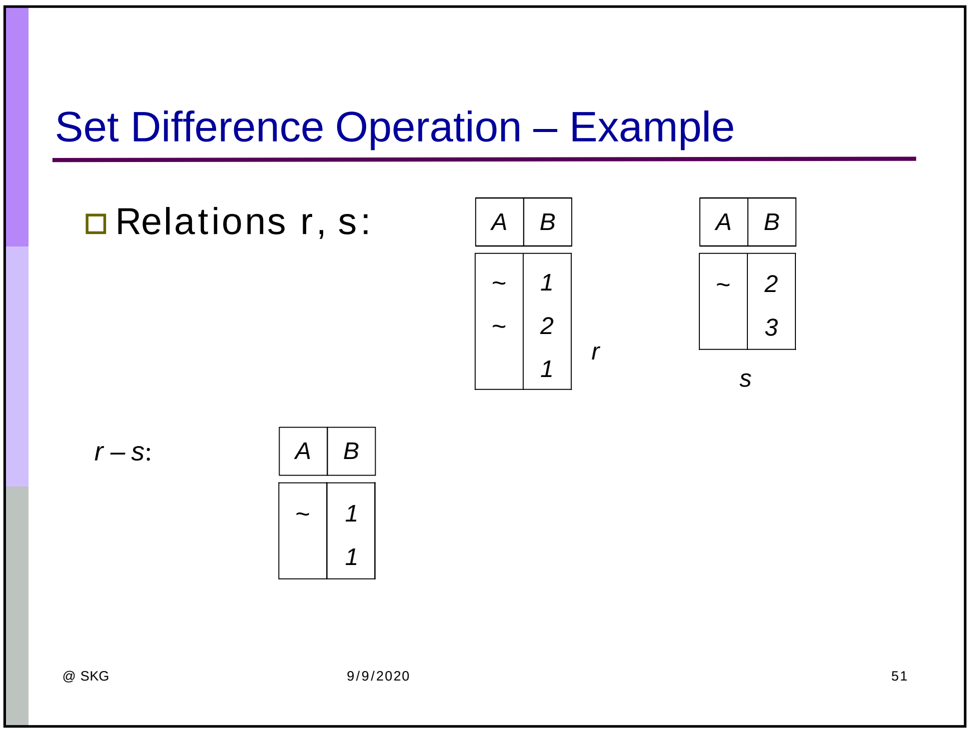
RELATIONAL ALGEBRA

$$\begin{tikzcd}
 & \text{Relational Algebra} \arrow[dl] \arrow[dr] & \\
\text{Unary Operator} & & \text{Binary Operator}
\end{tikzcd}$$

Unary Operator	Binary Operator
➢Select (σ)	➢Cartesian Product (×)
➢Project (Π)	➢Union (U)
➢Rename (ρ)	➢**Set Difference (—)**

Extended Relational-Algebra-Operations

The operators take one or more relations as inputs and give a new relation as a result



Set Difference Operation – Example

- Relations r, s:

A	B
~	1
~	2
	1

r

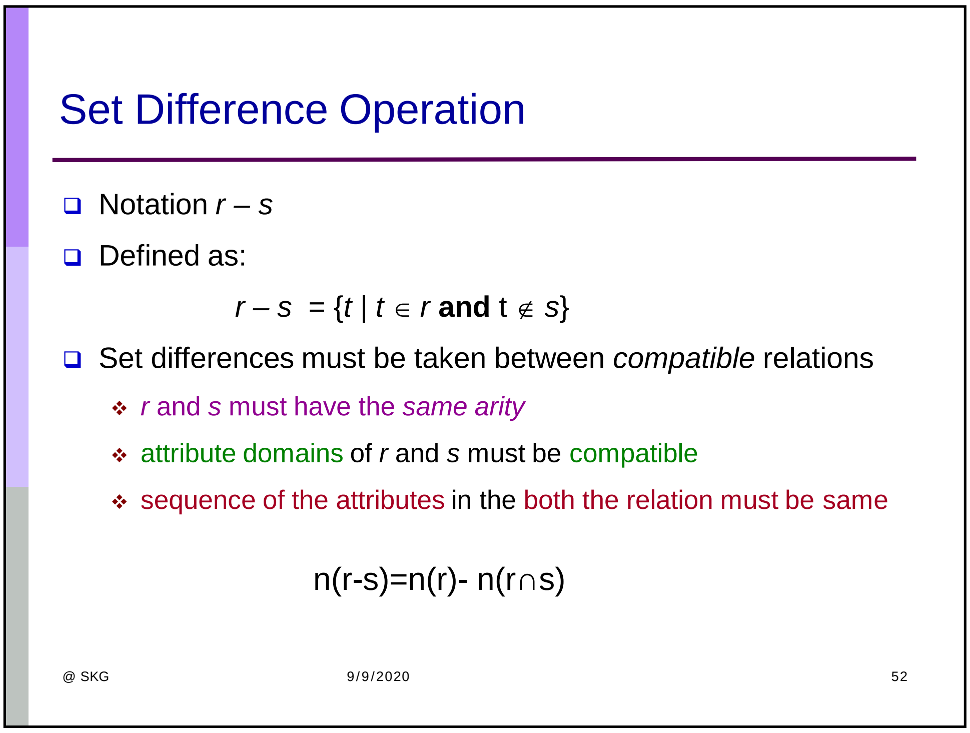
A	B
~	2
	3

s

r – *s*:

A	B
~	1
	1

@ SKG 9/9/2020 51



Set Difference Operation

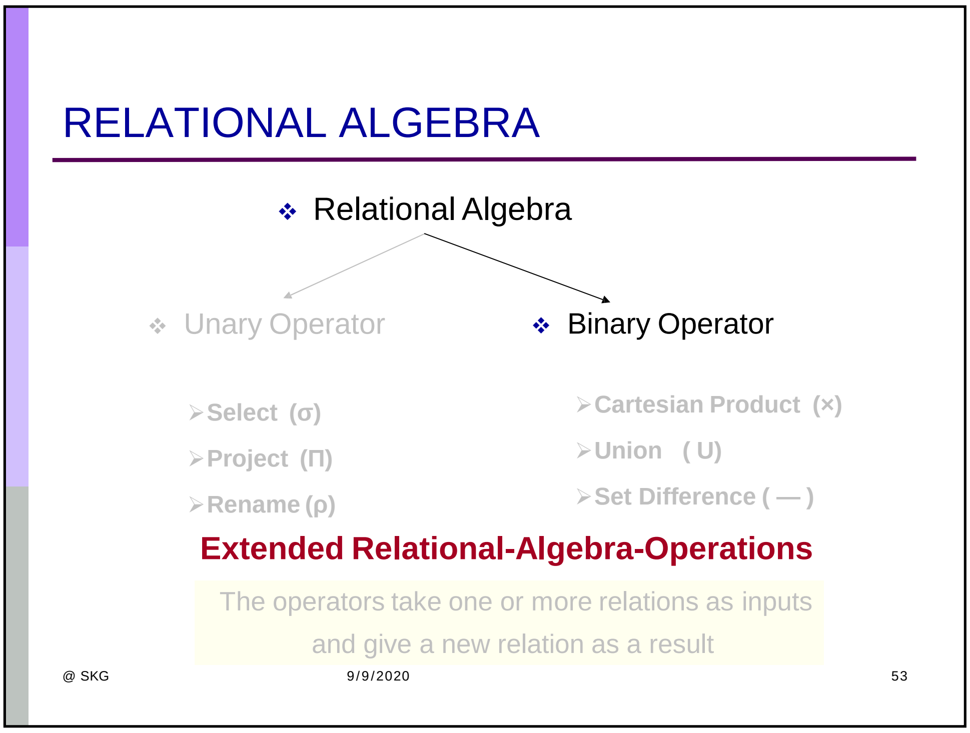
- Notation *r* – *s*
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- 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
  - sequence of the attributes in the both the relation must be same

$$n(r\text{-}s)=n(r)\text{-}\ n(r\cap s)$$

@ SKG 9/9/2020 52



RELATIONAL ALGEBRA

❖ Relational Algebra

❖ Unary Operator

- ➢Select (σ)
- ➢Project (Π)
- ➢Rename (ρ)

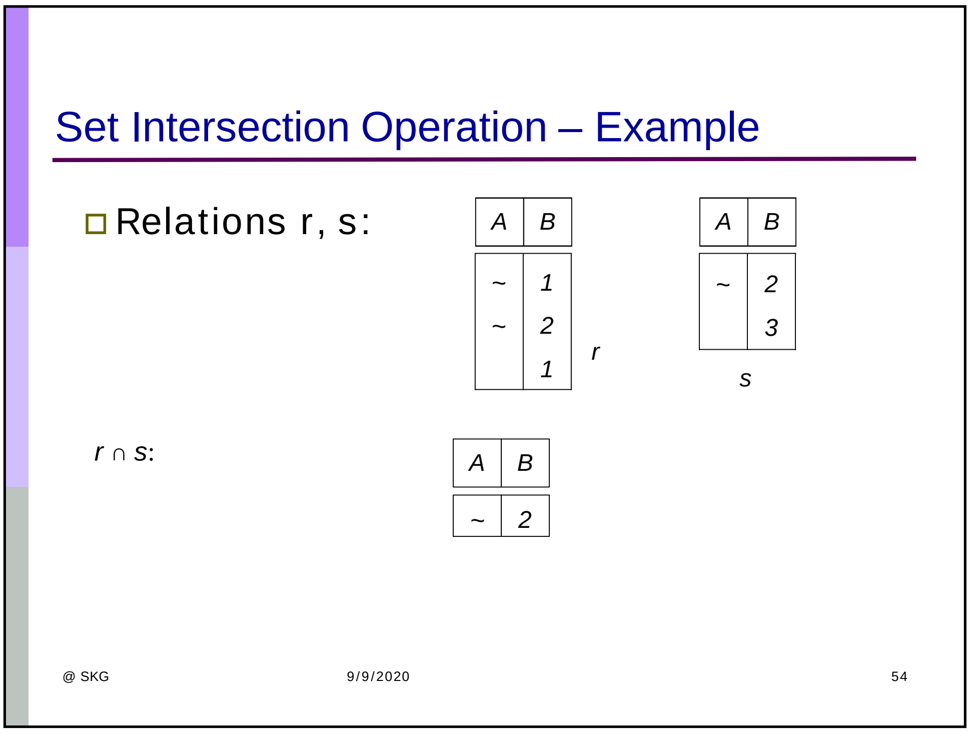
❖ Binary Operator

- ➢Cartesian Product (×)
- ➢Union (U)
- ➢Set Difference (—)

Extended Relational-Algebra-Operations

The operators take one or more relations as inputs and give a new relation as a result

@ SKG 9/9/2020 53



Set Intersection Operation – Example

□ Relations r, s:

A	*B*
~	*1*
~	*2*
	1

r

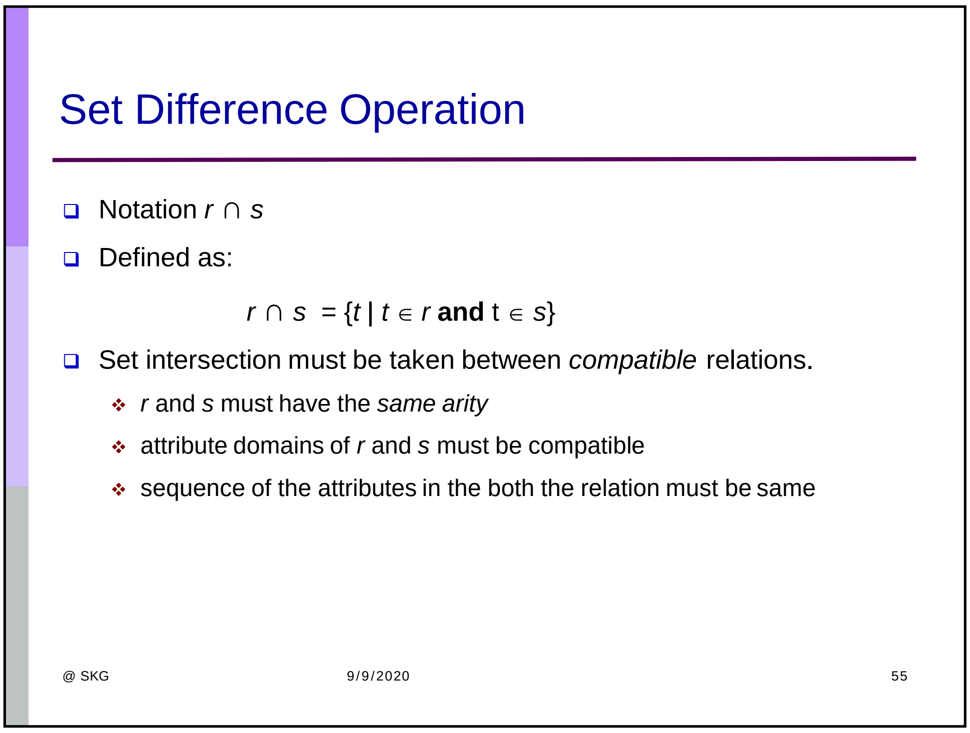
A	*B*
~	*2*
	3

s

$r \cap s$:

A	*B*
~	*2*

@ SKG 9/9/2020 54

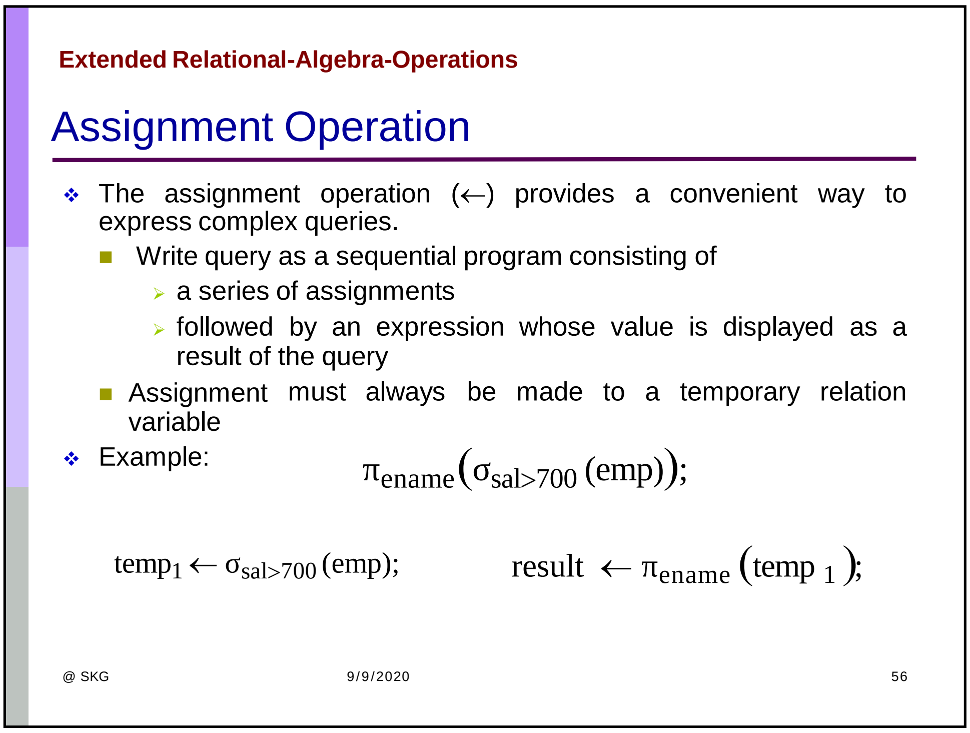


Set Difference Operation

- Notation *r* ∩ *s*
- Defined as:

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

- Set intersection must be taken between *compatible* relations.
  - *r* and *s* must have the *same arity*
  - attribute domains of *r* and *s* must be compatible
  - sequence of the attributes in the both the relation must be same

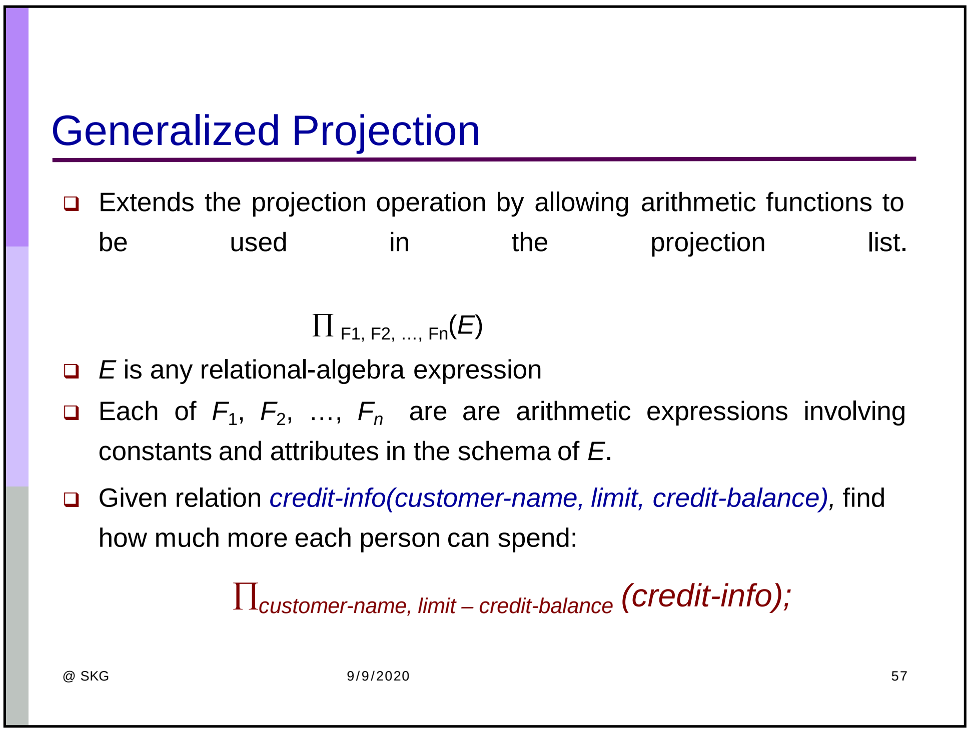


Extended Relational-Algebra-Operations

Assignment Operation

- The assignment operation (←) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query
  - Assignment must always be made to a temporary relation variable
- Example: $\pi_{ename}\left(\sigma_{sal>700}\,(emp)\right);$

$temp_1 \leftarrow \sigma_{sal>700}\,(emp);$ $result \leftarrow \pi_{ename}\left(temp_1\right);$



Generalized Projection

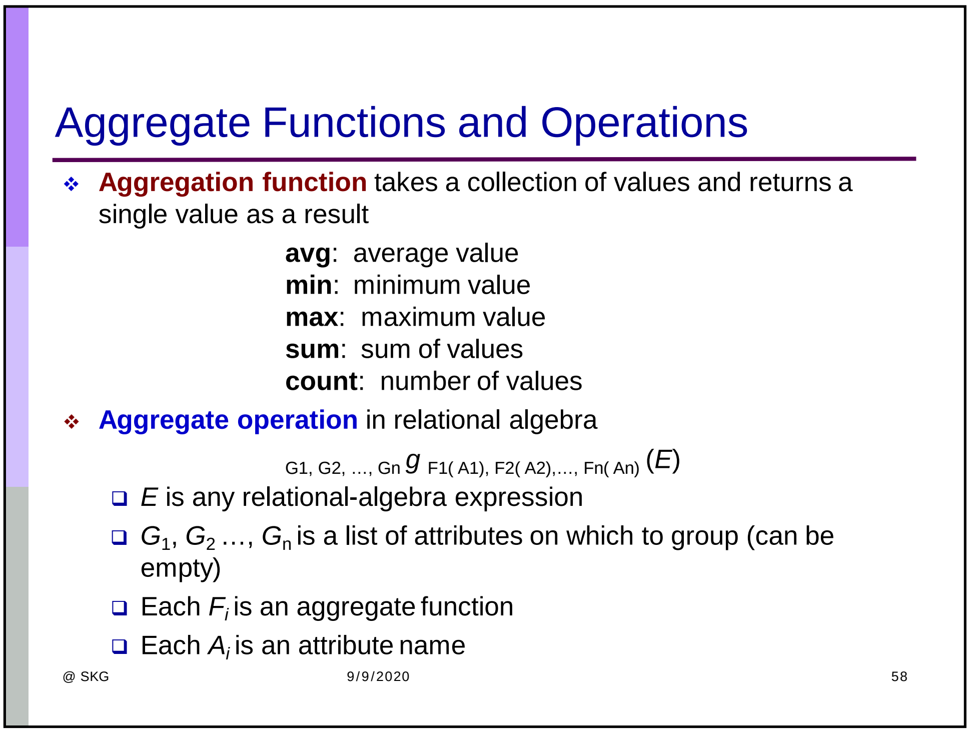
- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\Pi_{F1,\ F2,\ \ldots,\ Fn}(E)$$

- *E* is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of *E*.
- Given relation *credit-info(customer-name, limit, credit-balance)*, find how much more each person can spend:

$$\Pi_{\text{customer-name, limit – credit-balance}}\ (\textit{credit-info});$$

@ SKG 9/9/2020 57



Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result

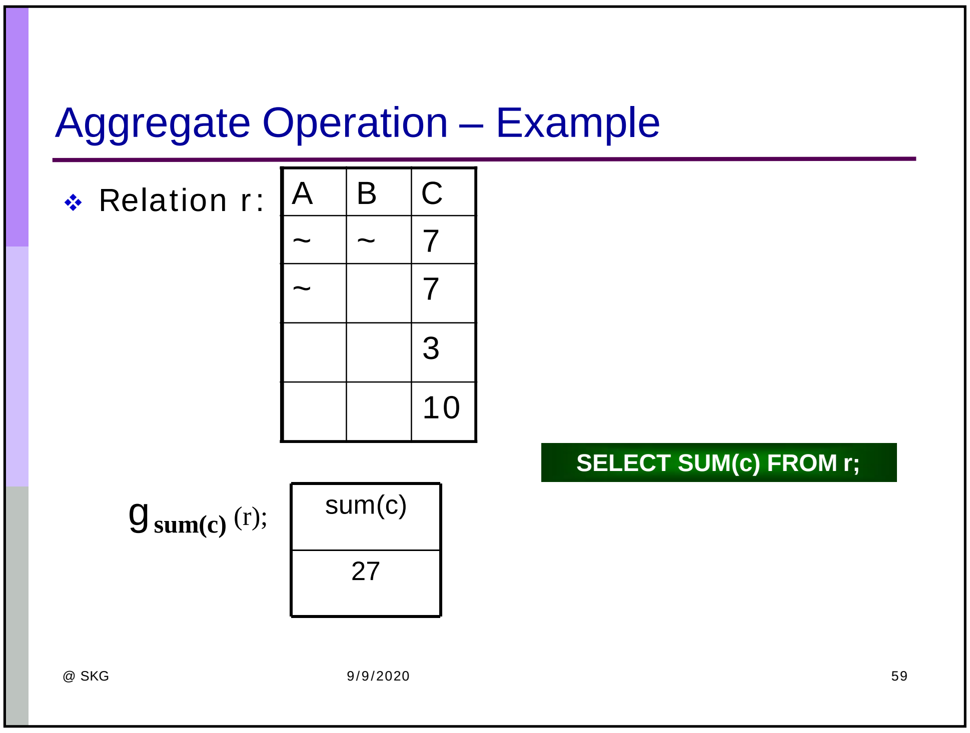
  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$_{G1,\ G2,\ \ldots,\ Gn}\,\mathcal{g}_{\ F1(A1),\ F2(A2),\ldots,\ Fn(An)}\ (E)$$

  - *E* is any relational-algebra expression
  - G_1, G_2 …, G_n is a list of attributes on which to group (can be empty)
  - Each F_i is an aggregate function
  - Each A_i is an attribute name

@ SKG 9/9/2020 58



Aggregate Operation – Example

❖ Relation r:

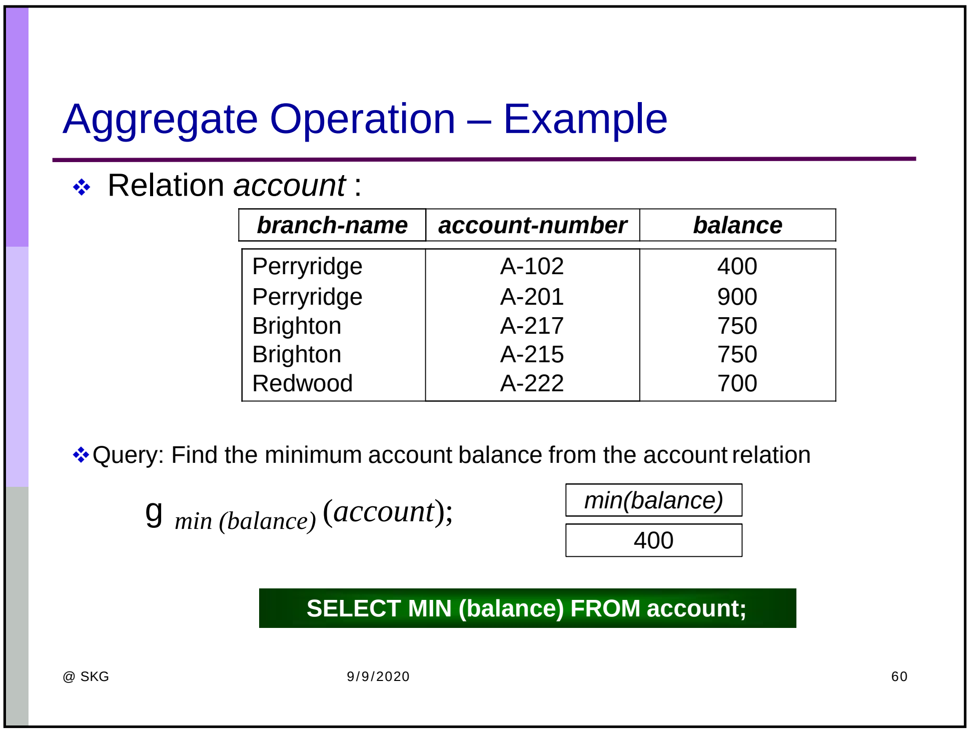
A	B	C
~	~	7
~		7
		3
		10

SELECT SUM(c) FROM r;

$g_{\mathbf{sum(c)}}(r)$;

sum(c)
27

@ SKG 9/9/2020 59



Aggregate Operation – Example

❖ Relation *account* :

branch-name	***account-number***	***balance***
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

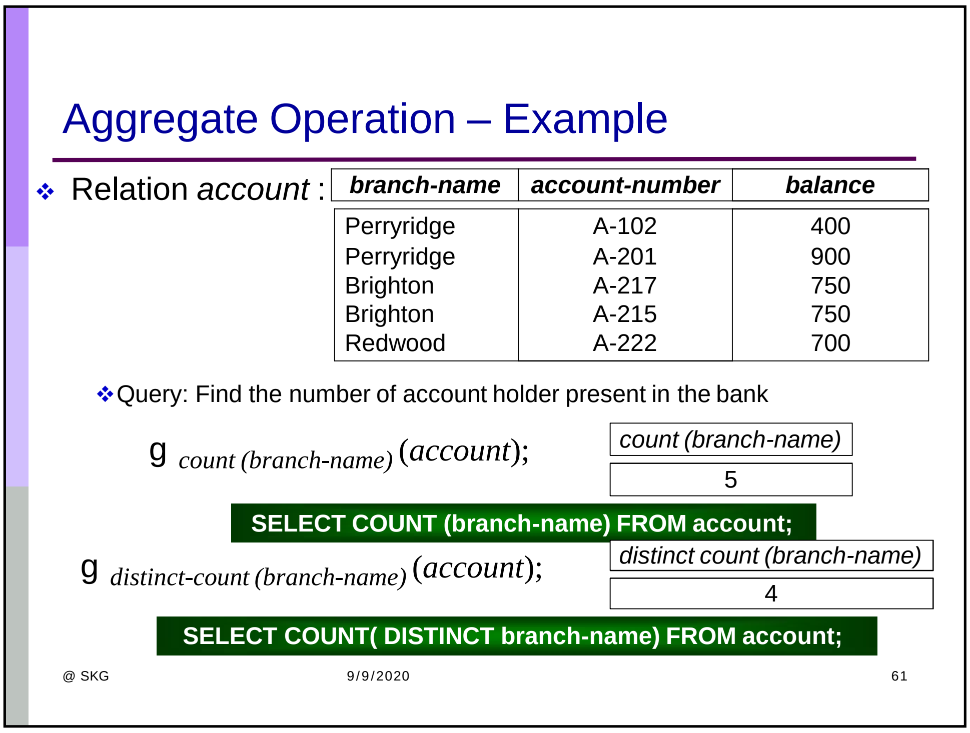
❖Query: Find the minimum account balance from the account relation

$g_{\textit{min (balance)}}(\textit{account})$;

min(balance)
400

SELECT MIN (balance) FROM account;

@ SKG 9/9/2020 60



Aggregate Operation – Example

❖ Relation *account* :

branch-name	***account-number***	***balance***
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

❖Query: Find the number of account holder present in the bank

$g_{\ count\ (branch\text{-}name)}\ (account);$

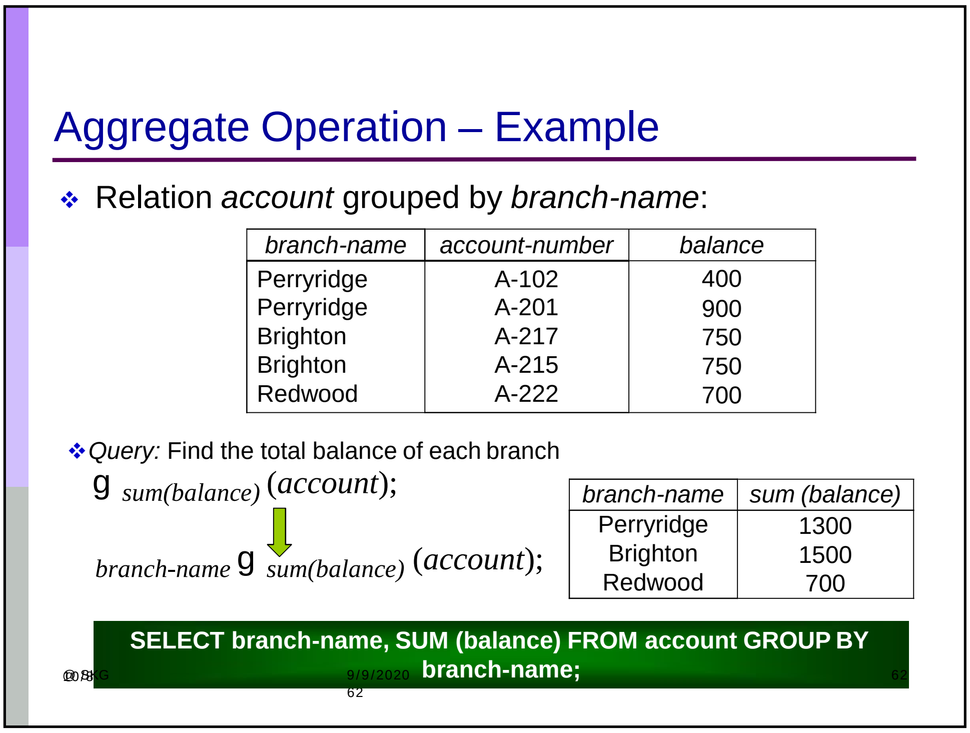
count (branch-name)
5

SELECT COUNT (branch-name) FROM account;

$g_{\ distinct\text{-}count\ (branch\text{-}name)}\ (account);$

distinct count (branch-name)
4

SELECT COUNT(DISTINCT branch-name) FROM account;



Aggregate Operation – Example

❖ Relation *account* grouped by *branch-name*:

branch-name	*account-number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

❖*Query:* Find the total balance of each branch

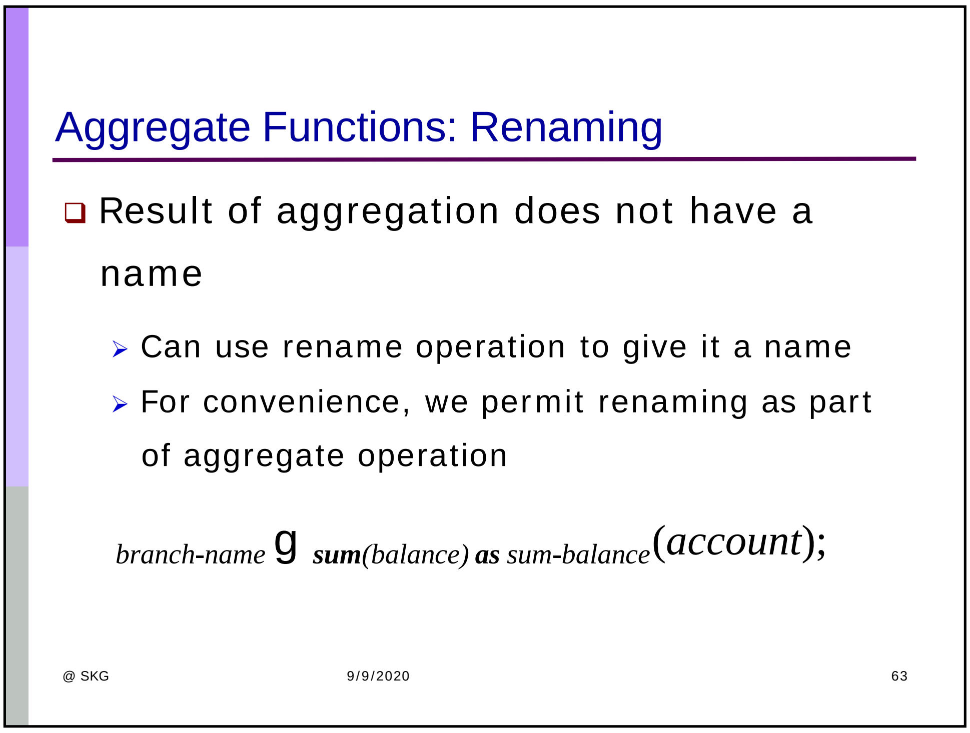
$g_{\ sum(balance)}\ (account);$

$\Downarrow$

${}_{branch\text{-}name}\,g_{\ sum(balance)}\ (account);$

branch-name	*sum (balance)*
Perryridge	1300
Brighton	1500
Redwood	700

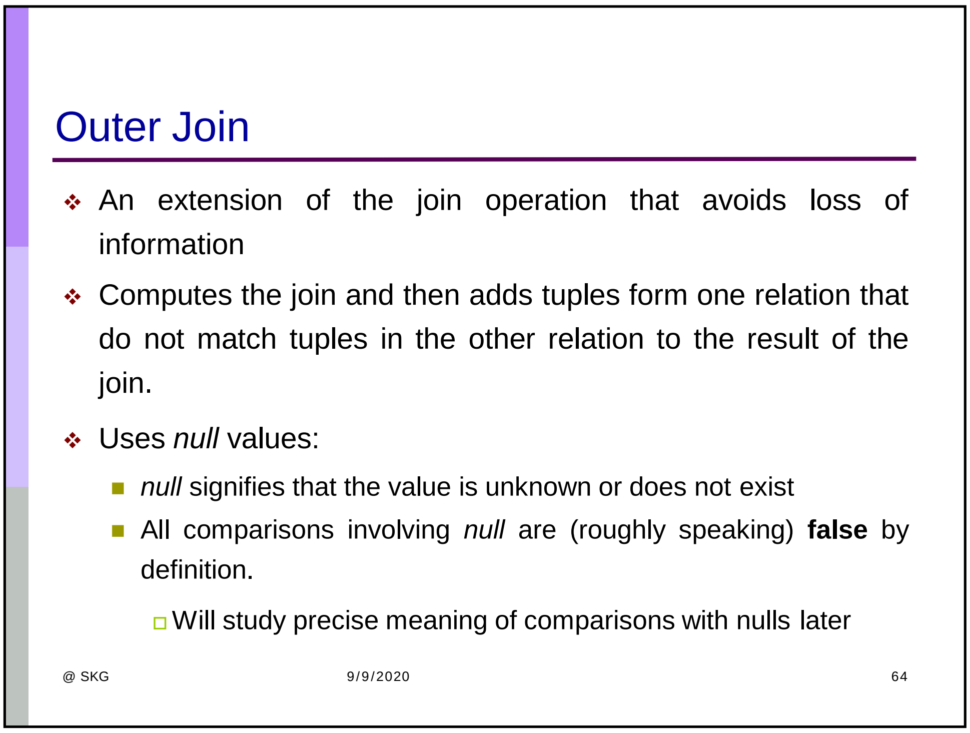
SELECT branch-name, SUM (balance) FROM account GROUP BY branch-name;



Aggregate Functions: Renaming

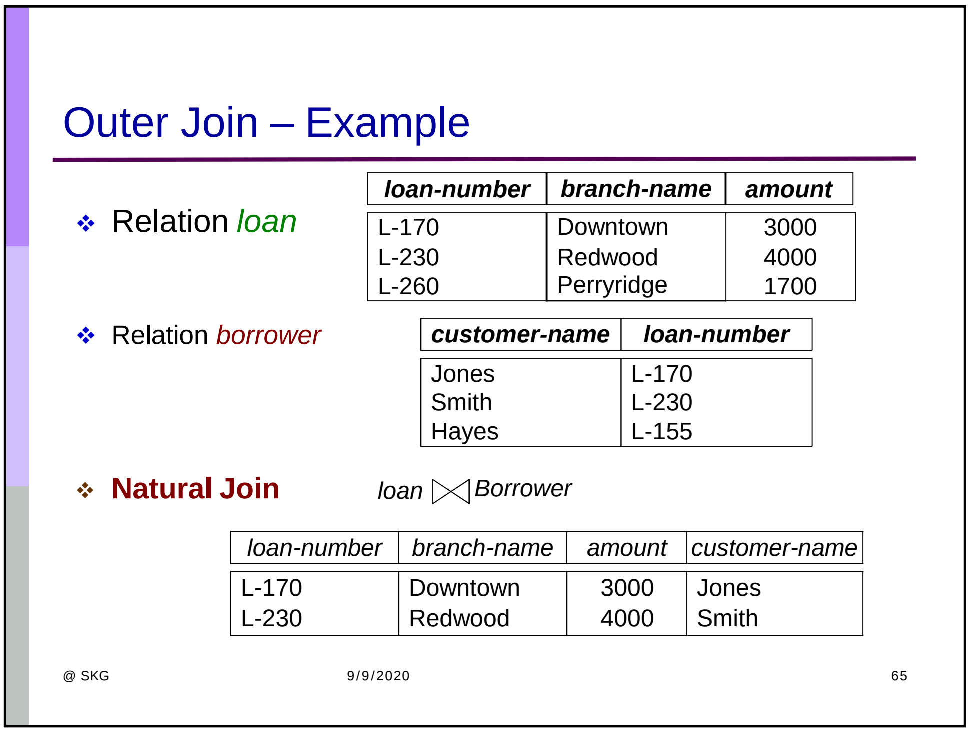
- Result of aggregation does not have a name
  - Can use rename operation to give it a name
  - For convenience, we permit renaming as part of aggregate operation

$${}_{\textit{branch-name}}\ g\ {}_{\textbf{sum}(\textit{balance})\ \textbf{as}\ \textit{sum-balance}}(\textit{account});$$



Outer Join

- An extension of the join operation that avoids loss of information
- Computes the join and then adds tuples form one relation that do not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.
    - Will study precise meaning of comparisons with nulls later



Outer Join – Example

- Relation *loan*

loan-number	***branch-name***	***amount***
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

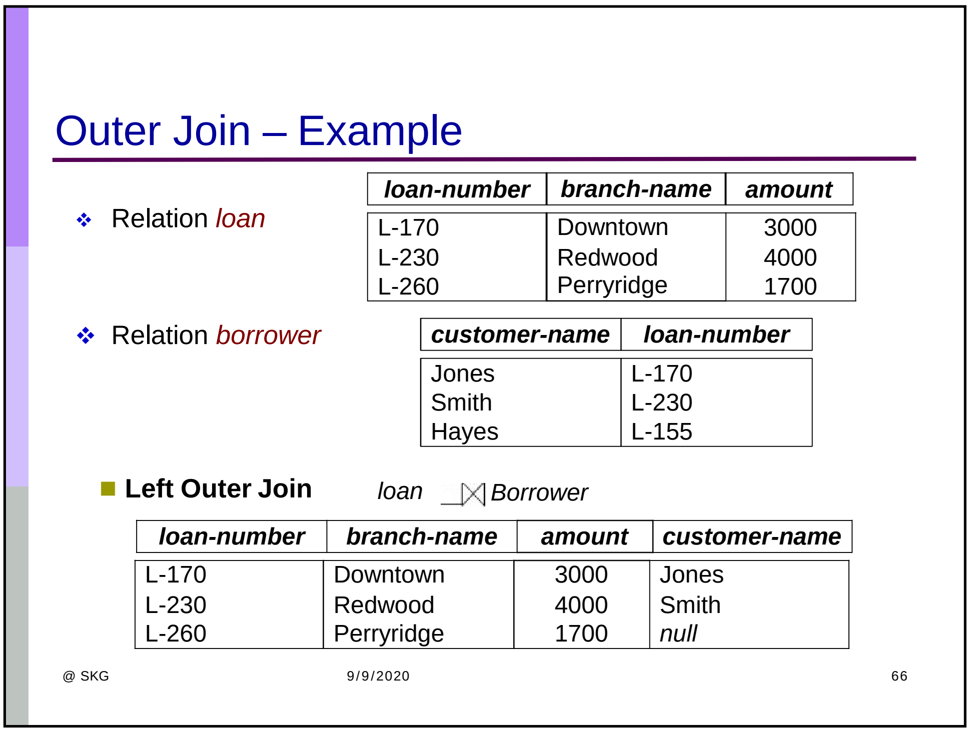
- Relation *borrower*

customer-name	***loan-number***
Jones	L-170
Smith	L-230
Hayes	L-155

- **Natural Join** *loan* $\bowtie$ *Borrower*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

@ SKG 9/9/2020 65



Outer Join – Example

- Relation *loan*

loan-number	***branch-name***	***amount***
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

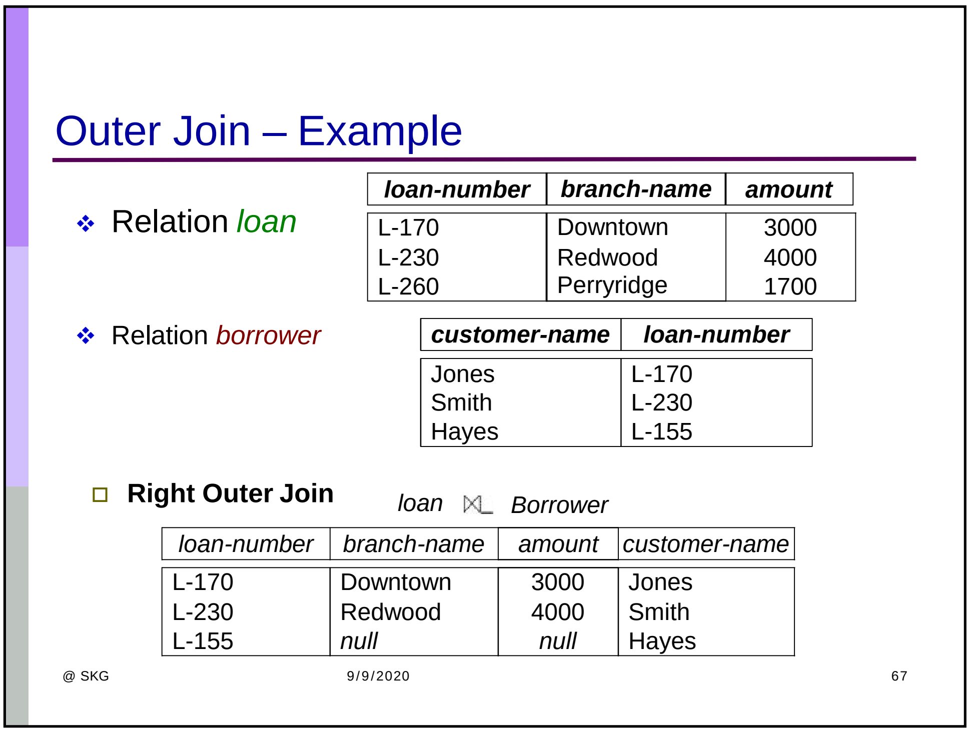
- Relation *borrower*

customer-name	***loan-number***
Jones	L-170
Smith	L-230
Hayes	L-155

- **Left Outer Join** *loan* ⟕ *Borrower*

loan-number	***branch-name***	***amount***	***customer-name***
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

@ SKG 9/9/2020 66



Outer Join – Example

- Relation *loan*

loan-number	***branch-name***	***amount***
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

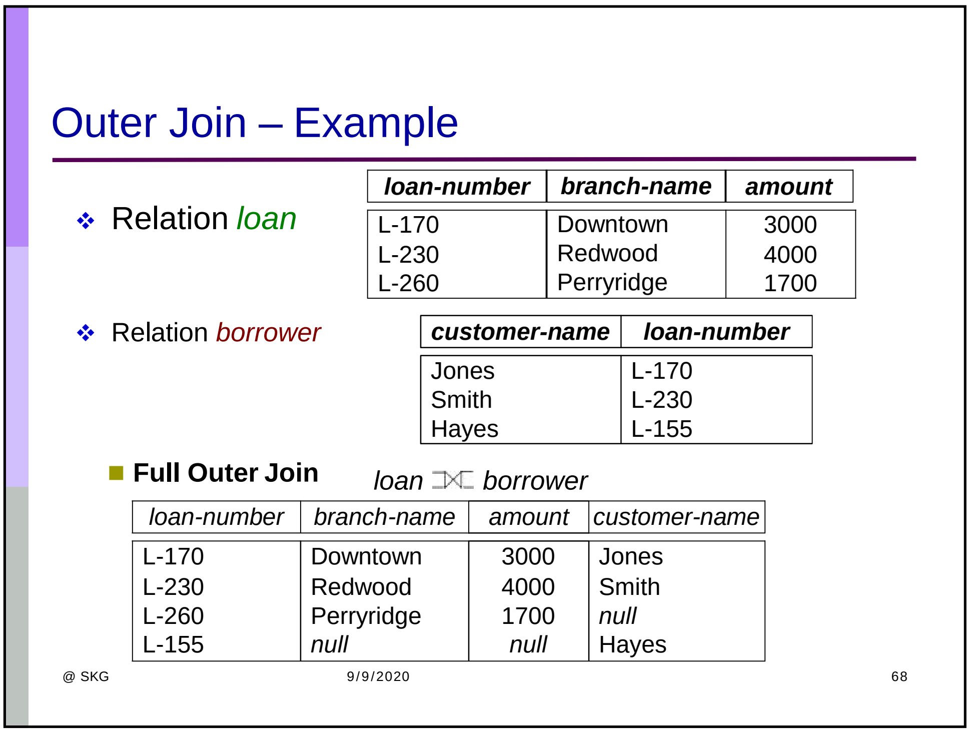
- Relation *borrower*

customer-name	***loan-number***
Jones	L-170
Smith	L-230
Hayes	L-155

- **Right Outer Join** *loan* ⟖ *Borrower*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

@ SKG 9/9/2020 67



Outer Join – Example

- Relation *loan*

loan-number	***branch-name***	***amount***
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

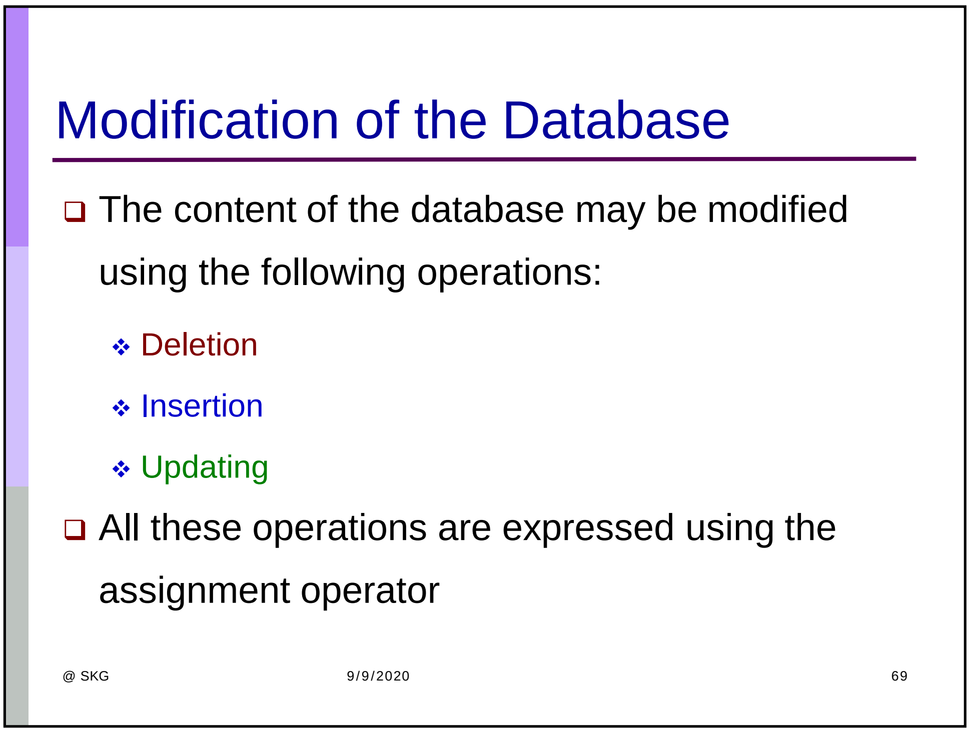
- Relation *borrower*

customer-name	***loan-number***
Jones	L-170
Smith	L-230
Hayes	L-155

- **Full Outer Join** *loan* ⟗ *borrower*

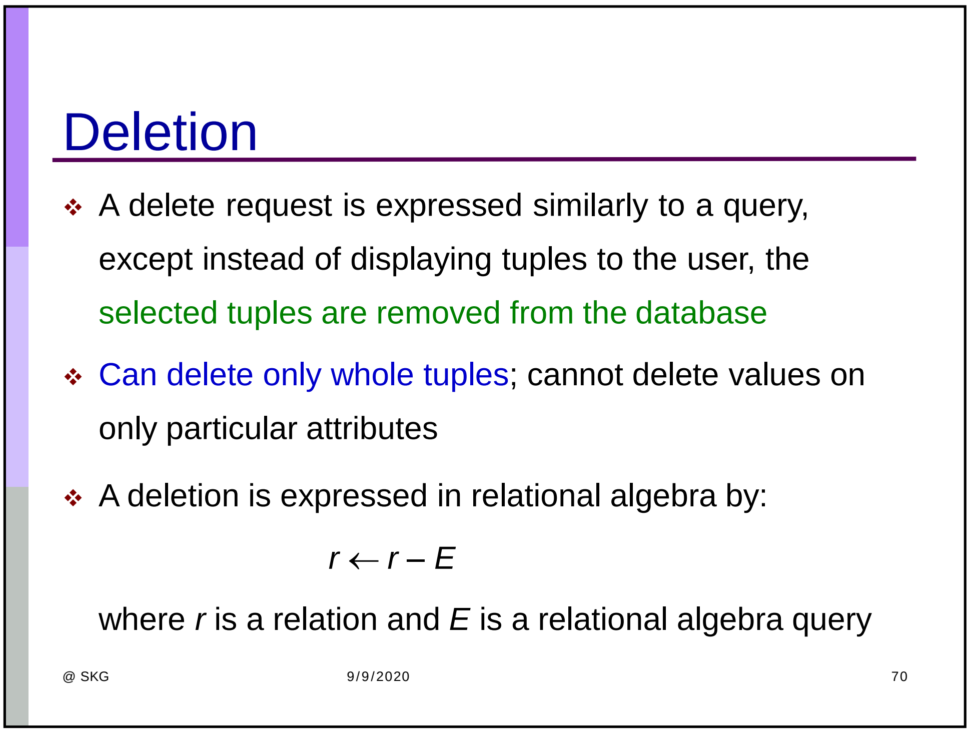
loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

@ SKG 9/9/2020 68



Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations are expressed using the assignment operator

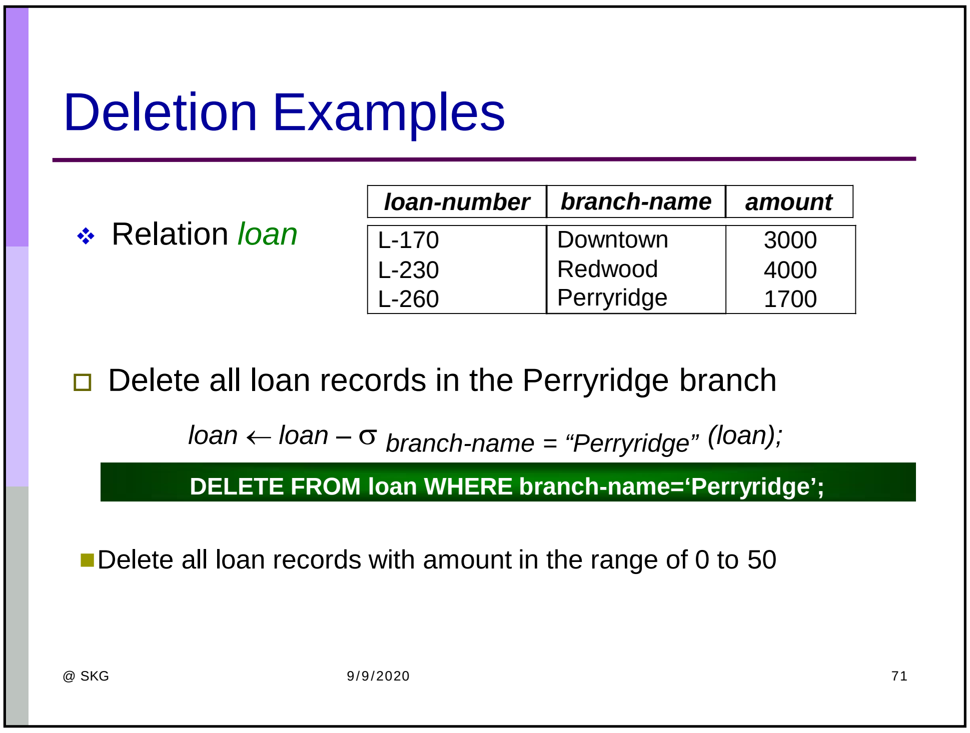


Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where *r* is a relation and *E* is a relational algebra query



Deletion Examples

- Relation *loan*

loan-number	*branch-name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

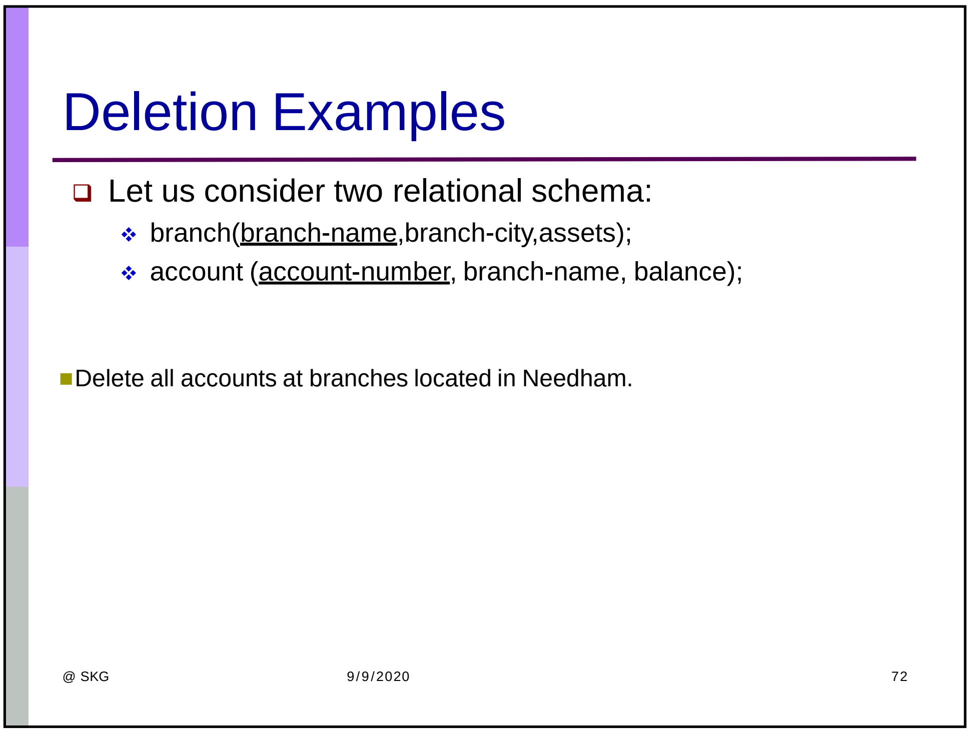
- Delete all loan records in the Perryridge branch

$loan \leftarrow loan - \sigma_{\textit{branch-name} = \text{“Perryridge”}} (loan);$

```
DELETE FROM loan WHERE branch-name='Perryridge';
```

- Delete all loan records with amount in the range of 0 to 50

@ SKG 9/9/2020 71



Deletion Examples

- Let us consider two relational schema:
  - branch(<u>branch-name</u>,branch-city,assets);
  - account (<u>account-number</u>, branch-name, balance);

- Delete all accounts at branches located in Needham.

@ SKG 9/9/2020 72

Insertion

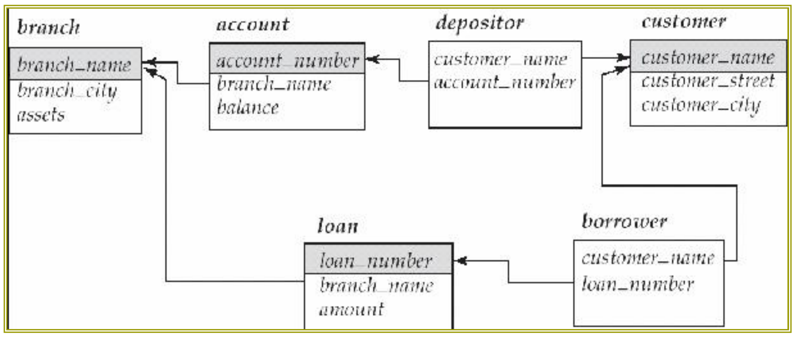
- To insert data into a relation, we either:
  - specify a tuple to be inserted
  - write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

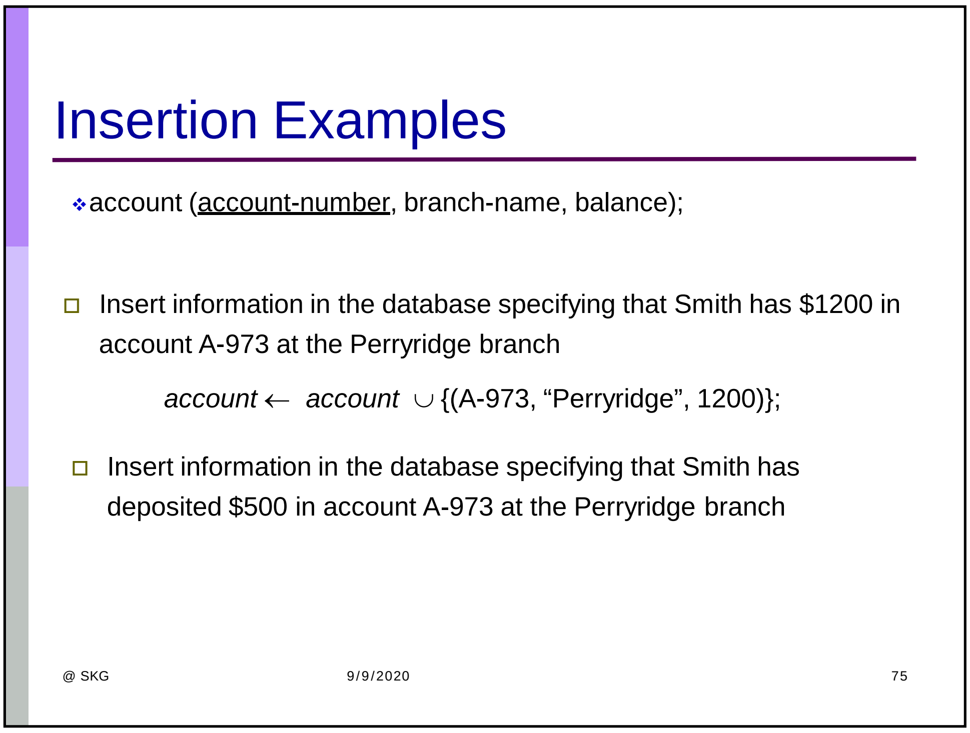
$$r \leftarrow r \cup E$$

where *r* is a relation and *E* is a relational algebra expression.

- The insertion of a single tuple is expressed by letting *E* be a constant relation containing one tuple.

- Let us consider this schema diagram





Insertion Examples

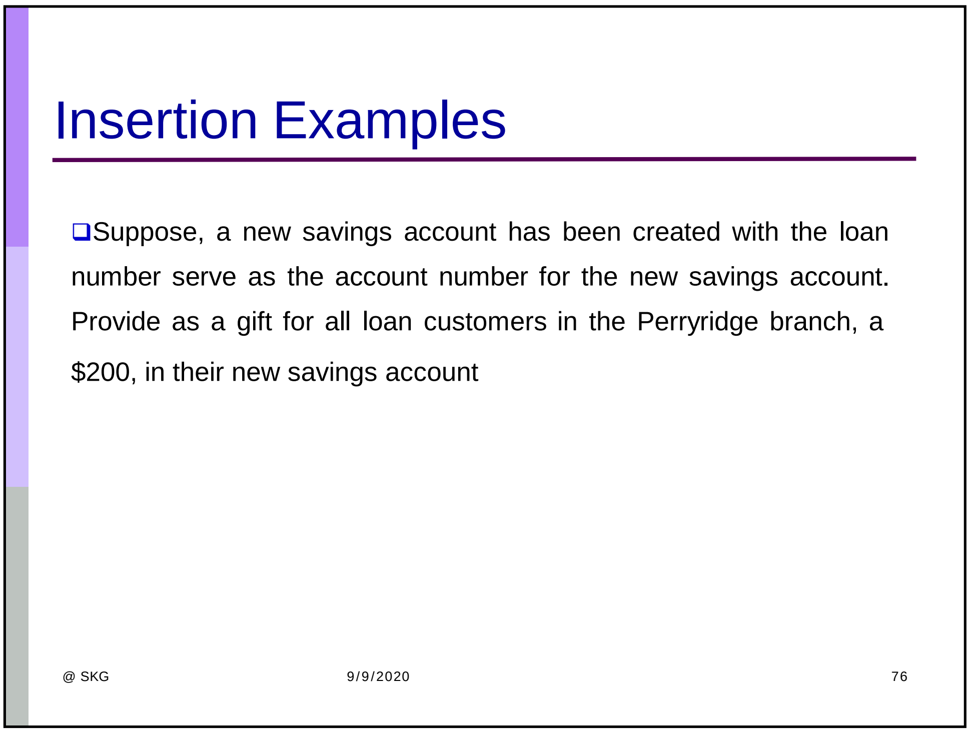
❖account (<u>account-number</u>, branch-name, balance);

- ◻ Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch

  account ← *account* ∪ {(A-973, "Perryridge", 1200)};

- ◻ Insert information in the database specifying that Smith has deposited $500 in account A-973 at the Perryridge branch

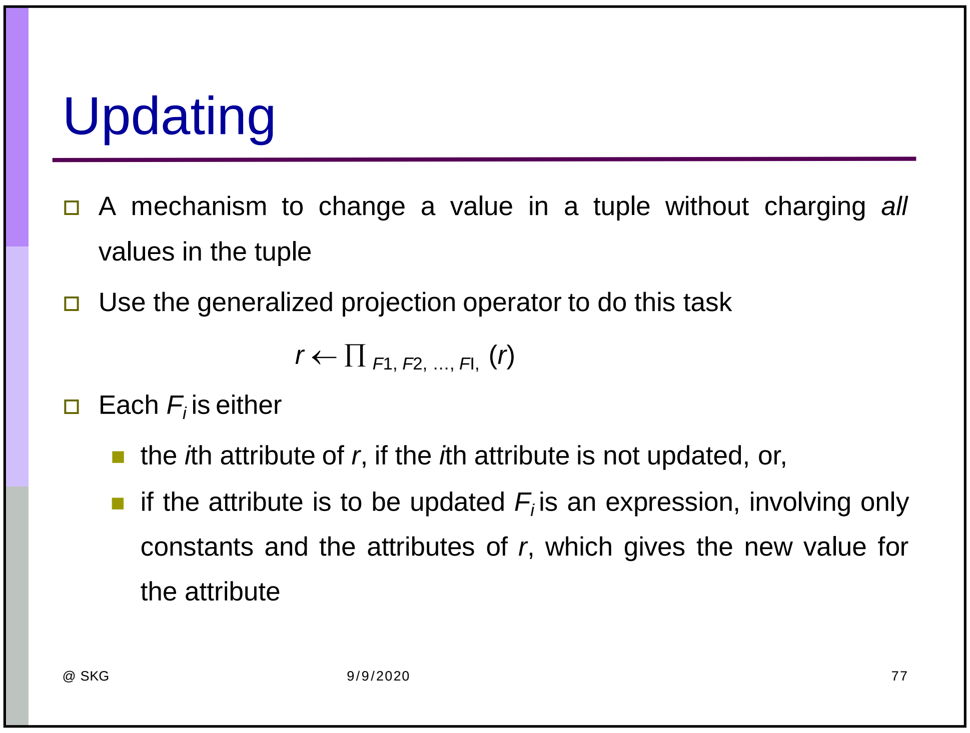
@ SKG 9/9/2020 75



Insertion Examples

❑Suppose, a new savings account has been created with the loan number serve as the account number for the new savings account. Provide as a gift for all loan customers in the Perryridge branch, a $200, in their new savings account

@ SKG 9/9/2020 76



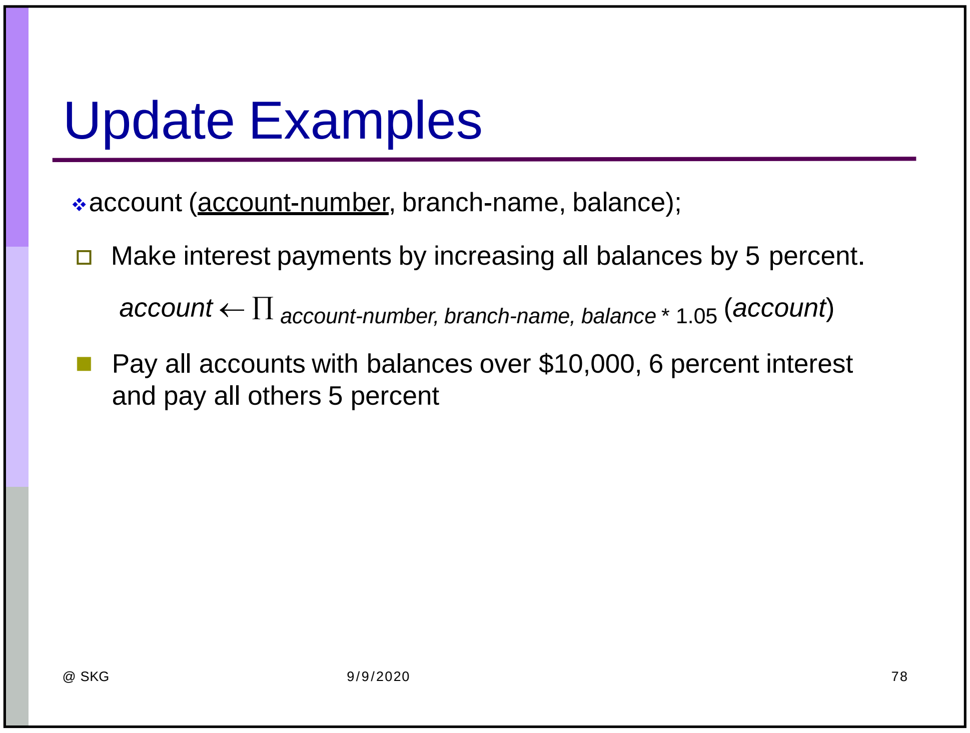
Updating

- A mechanism to change a value in a tuple without charging *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \Pi_{F1,\ F2,\ \ldots,\ Fl,}\ (r)$$

- Each F_i is either
  - the *i*th attribute of *r*, if the *i*th attribute is not updated, or,
  - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of *r*, which gives the new value for the attribute

@ SKG 9/9/2020 77



Update Examples

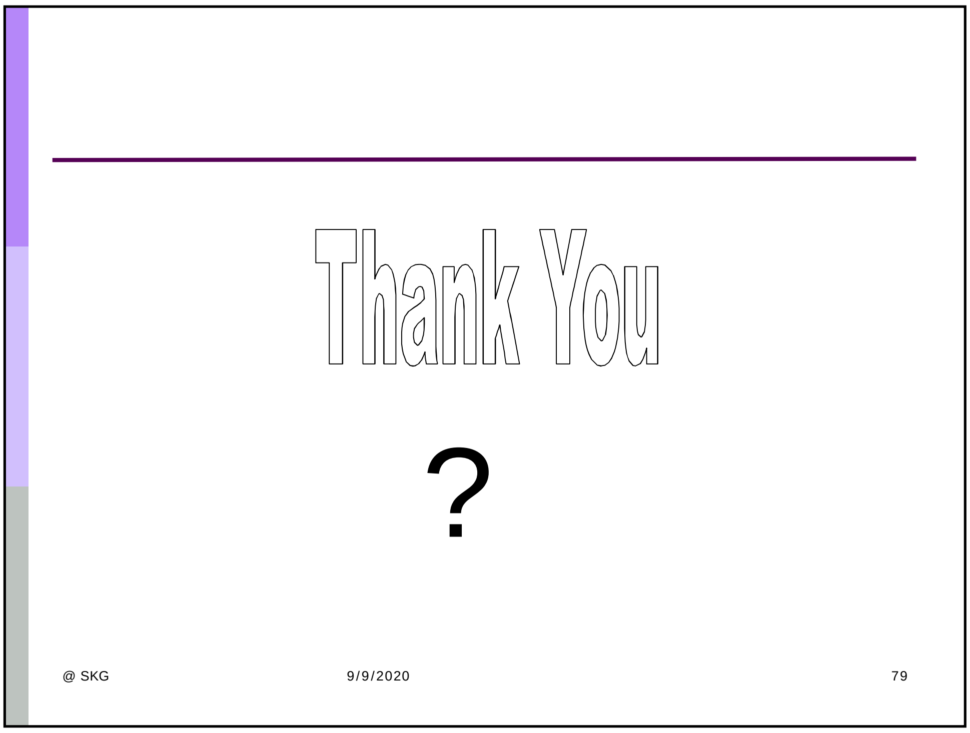
❖account (<u>account-number</u>, branch-name, balance);

- Make interest payments by increasing all balances by 5 percent.

$$account \leftarrow \Pi_{account\text{-}number,\ branch\text{-}name,\ balance\ *\ 1.05}\ (account)$$

- Pay all accounts with balances over $10,000, 6 percent interest and pay all others 5 percent

@ SKG 9/9/2020 78



Thank You

?