CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Relational Algebra

LECTURE 5

Dr. Philipp Leitner

philipp.leitner@chalmers.se

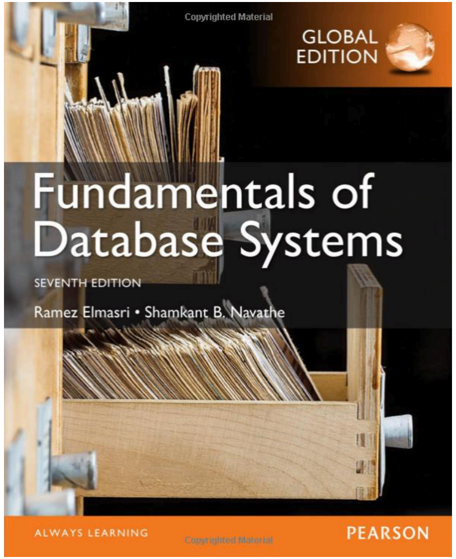
@xLeitix

LECTURE 5

Covers …

Chapter 8

Please read this up until next lecture!



What we will be covering

Basics of Relational Algebra

Unary Relational Operations

Standard Set Theory Relational Operations

Binary Relational Operations (i.e., JOINS)

Reminder - Formal Definitions

Define a relation as:

`R(A1, A2, .........., An)`

`r(R) ⊆ dom (A1) X dom (A2) X … X dom(An)`

`r(R)`:

a specific state (or "value" or “population”) of R (set of tuples)

`r(R) = {t1, t2, …, tn}` where each `ti` is an n-tuple

`ti = <v1, v2, …, vn>` where each `vj` element-of `dom(Aj)`

Reminder - Formal Definitions

Denoting attribute values - for a tuple `t`:

`t[1]` … value of first attribute in the list (remember - ordered!)

`t[a]` … value of attribute a

`t.a` … same as `t[a]`

`t[X]`, with `X` a subset of attributes

… values of all attributes in `X`

Reminder - Formal Definitions

Database state is collection of the state of all relations

`DB = {r1, r2, ..., rm}`

Database is in a valid state if all `ri` are valid

Referential integrity

Example:

In the EMPLOYEE and DEPARTMENT relationships, the attribute that specifies which department an employee works in needs to refer to a department that actually exists.

A constraint **involving two relations**

The previous constraints involved only a single relation

Used to specify a **relationship among tuples** in two relations:

AKA referencing relation and the referenced relation

Foreign keys `FK`

Tuples in the referencing relation `R1` have attributes `FK` (the **foreign key**) that reference the primary key attributes `PK` of the referenced relation `R2`.

A tuple `t1` in `R1` is said to reference a tuple `t2` in `R2` if

`t1[FK] = t2[PK]`

Foreign keys `FK`

Note that (in 1:N or N:M relationships), **foreign keys are not keys themselves**!

- E.g., multiple employees work in the same department, hence have the same value in their department FK

FKs may also be allowed to be NULL

- If relationship is optional, e.g., employees may not be assigned to a department at all

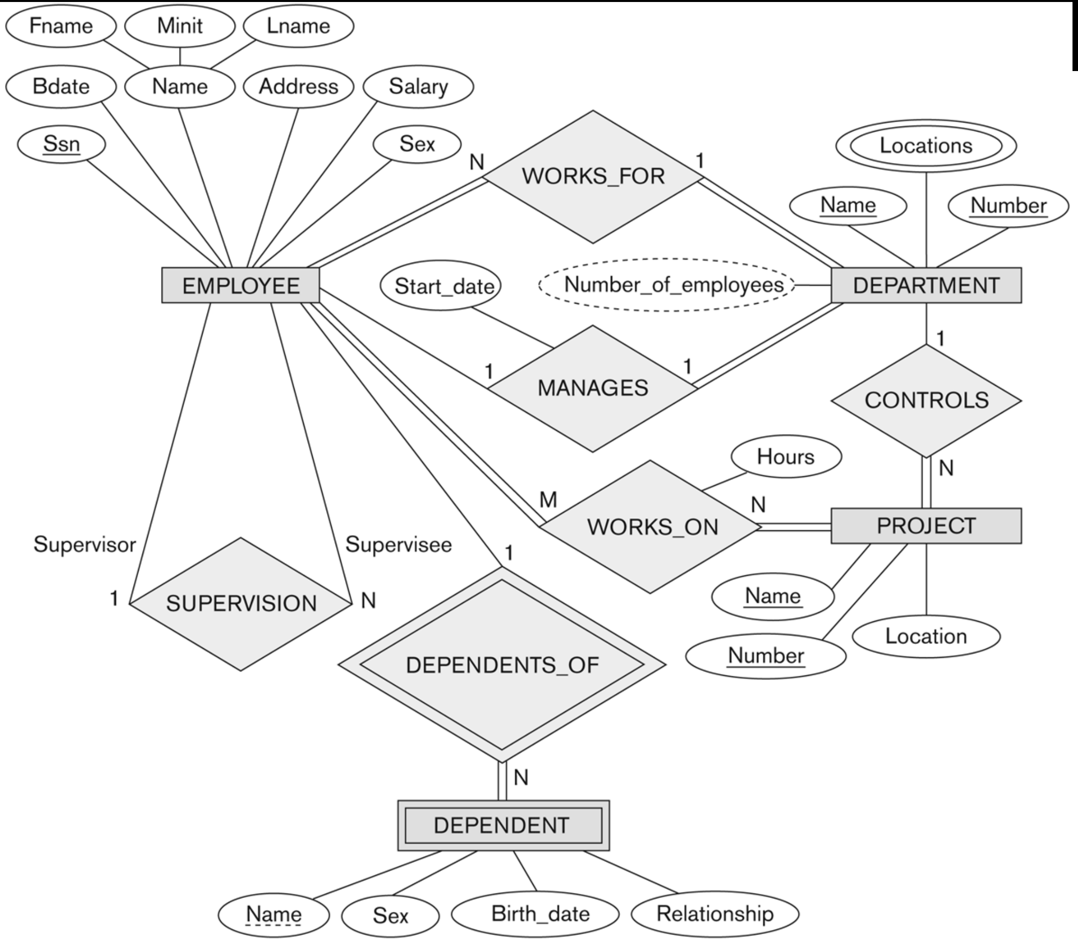
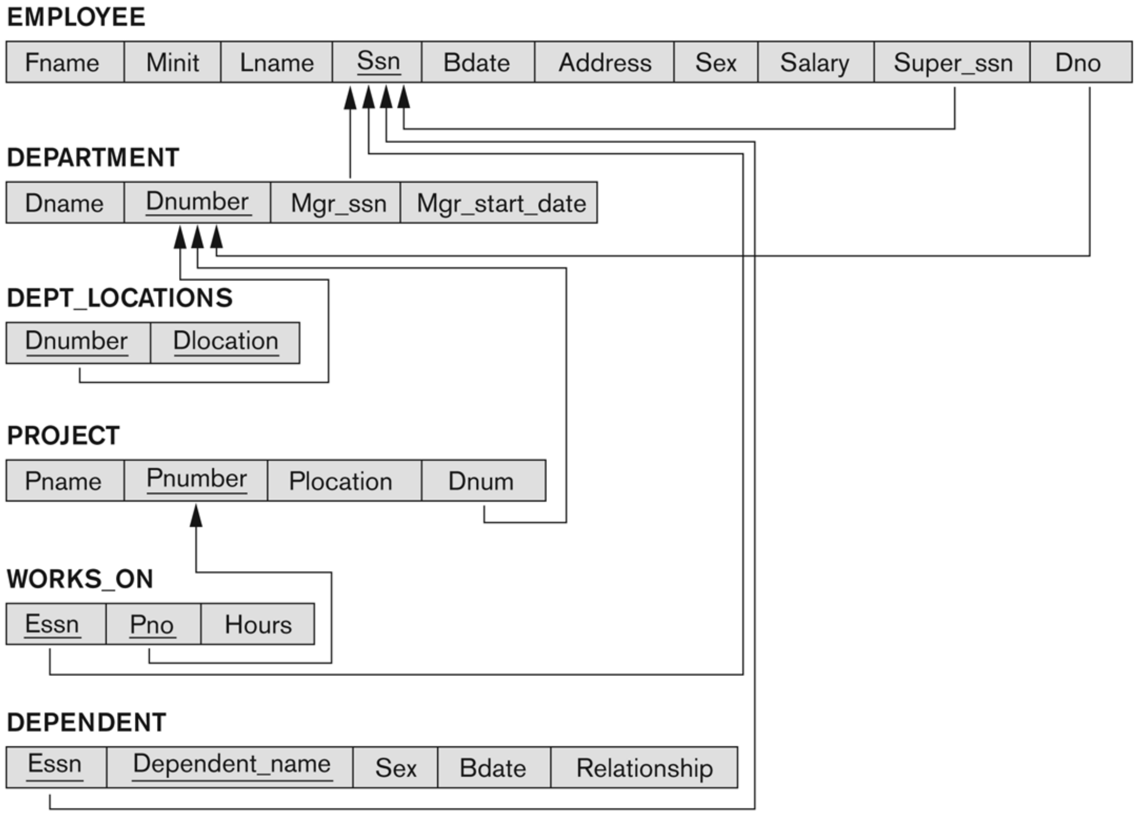


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Partial mapping of company example



Relational Algebra

A formalism to define **queries** over relational models

A **mathematical language** that we can use to “ask questions” about our data model

- What is the Ssn of the employee “Philipp Leitner”?
- How many employees are there?
- Who is the supervisor of “Philipp Leitner”?
- …

Directly implemented in the query language of SQL

Aside on algebras

Algebra is *"the study of mathematical symbols and the rules for manipulating these symbols"* [Wikipedia]

We call those rules **operators**.

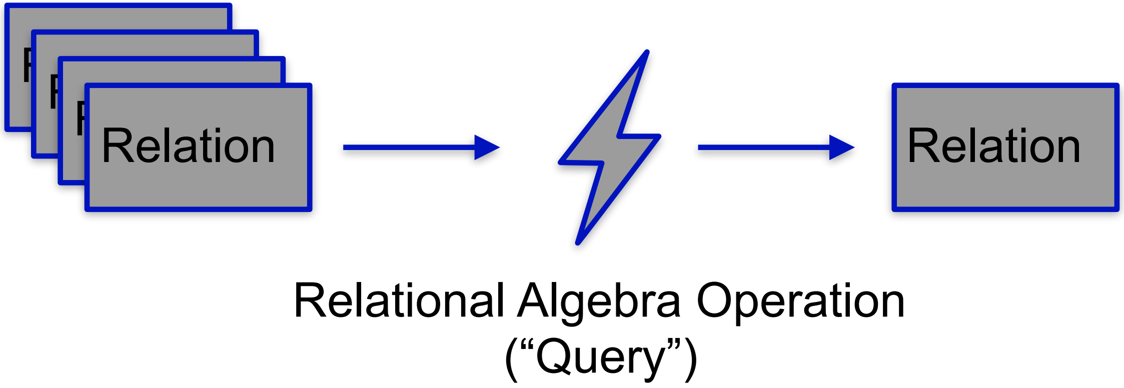
You know operators from math and from programming:

```
4x + 17 = e
int variable = 17 * other_var;
```

Relational algebra follows similar rules:

$$\text{TEMP} \leftarrow \boldsymbol{\sigma}_{\text{Ssn}=125763}(\text{EMPLOYEE})$$

Basic Structure



Query results lead to new relations

Makes relational algebra a **closed** algebra

Overview

Unary Relational Operations

SELECT (symbol: **σ** (sigma))

PROJECT (symbol: π (pi))

RENAME (symbol: ρ (rho), or simply ←)

Relational Algebra Operations From Set Theory

UNION (∪), INTERSECTION (∩), DIFFERENCE (or MINUS, –)

CARTESIAN PRODUCT (x)

Overview

Binary Relational Operations

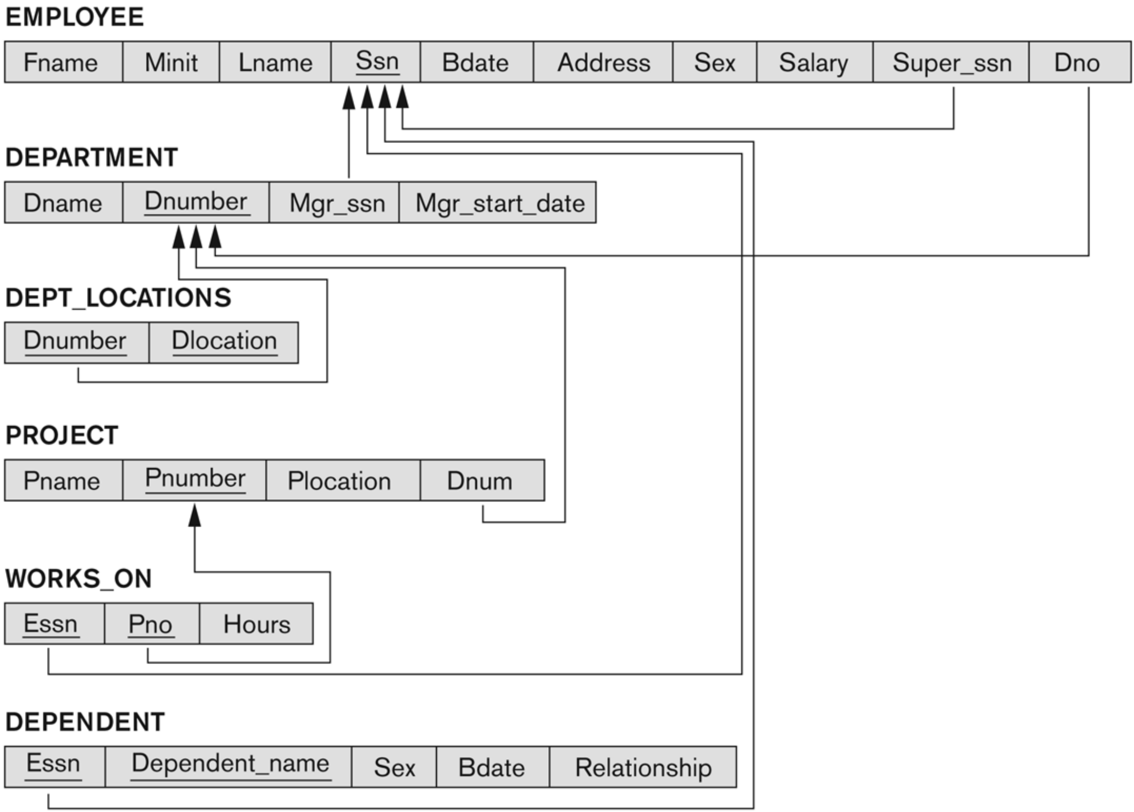
- JOIN (several variations)
- DIVISION

Additional Relational Operations

- OUTER JOINS, OUTER UNION
- AGGREGATE FUNCTIONS
  - SUM, COUNT, AVG, MIN, MAX

We’ll be using this example relational model

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Unary Relational Operations: SELECT

SELECT operation (**σ** -sigma) is used to select a subset of the tuples from a relation based on a **selection condition**

Selection condition acts as a filter

Keeps only those tuples that satisfy the qualifying condition

Examples:

Select the EMPLOYEE tuples whose department number is 4:

$\sigma_{DNO=4}(EMPLOYEE)$

Select the employee tuples whose salary is greater than $30,000:

$\sigma_{SALARY>30000}(EMPLOYEE)$

Unary Relational Operations: SELECT

General format:

$$\sigma_{\texttt{<condition>}}(\texttt{R})$$

Result is a new relation `R’` containing only those tuples for which `condition` evaluates to `true`.

`R’` has the **same schema** as `R`

All attributes are the same

Mathematical Properties of SELECT

SELECT σ is **commutative**:

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

Because of commutativity property, a cascade (sequence) of SELECT operations may be applied **in any order:**

$$\sigma_{<cond1>}(\sigma_{<cond2>}(\sigma_{<cond3>}(R)) = \sigma_{<cond2>}(\sigma_{<cond3>}(\sigma_{<cond1>}(R)))$$

Mathematical Properties of SELECT

A cascade of SELECT operations **may be replaced by a single selection** with a conjunction of all the conditions:

$$\sigma_{\text{<cond1>}}(\sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(R)) = \sigma_{\text{<cond1> AND <cond2> AND <cond3>}}(R)))$$

The number of tuples in the result of a SELECT is **less than (or equal to)** the number of tuples in the input relation `R`

Unary Relational Operations: PROJECT

- Denoted by π (pi)
  - Keeps certain attributes from a relation and discards the others
  - **Vertical partitioning**

- Example:
  - List each employee's first and last name and salary:
    - $\pi_{\mathrm{LNAME,FNAME,SALARY}}(\mathrm{EMPLOYEE})$

Figure 5.6
One possible database state for the COMPANY relational database schema.

σ (SELECT)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	333445555	5
Ahmad	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	87654321	4
James	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ULL	1

$\sigma_{\text{Fname} = (\text{Alicia OR Jennifer OR Ramesh})}(\text{EMPLOYEE})$

π (PROJECT)

$\pi_{\text{Ssn, Bdate}}(\text{EMPLOYEE})$

Figure 5.6
relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	31 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	38 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	91 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	75 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	80 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	50 Stone, Houston, TX	M	55000	NULL	1

Unary Relational Operations: PROJECT

General format:

$$\pi_{\texttt{<attribute\_list>}}(\texttt{R})$$

Result is a new relation `R'` containing **some or all** tuples from `R`, but with a **different schema**

Namely, the new schema is exactly `<attribute_list>`

(attributes are in listed order)

Unary Relational Operations: PROJECT

Why would there ever be less tuples in the result relation?

The project operation removes any duplicate tuples

Remember that `R'` is a mathematical set, which cannot have duplicates

Example:

π_{FNAME}(EMPLOYEE)

… returns a list of all first names

… removes tuples where employees had the same first name

In general, tuples may be lost if projecting on non-key attributes.

Mathematical Properties of PROJECT

PROJECT is **not** commutative, but:

$$p_{\langle list1 \rangle}(p_{\langle list2 \rangle}(R)) = p_{\langle list1 \rangle}(R)$$

iff <list2> contains all the attributes in <list1>

(returns an empty relation otherwise)

Generalized PROJECT

If we allow the parameters of π to be **functions** instead of simple attribute names we arrive at a more powerful version of PROJECT

Report yearly salary for all employees:

$$\pi_{\mathrm{LNAME,FNAME,12\ *\ SALARY}}(\mathrm{EMPLOYEE})$$

Unary Relational Operations: Assignment

General format:

`E` $\leftarrow$ <expression>

Used to **“store”** an intermediary result from a complex relational algebra operation

We could always write it in a single operation, but it is often more convenient / clear to use an assignment

Example Assignment

In one operation:

$$\pi_{\mathtt{LNAME,FNAME,SALARY}}(\sigma_{\mathtt{DNO=5}}(\mathtt{EMPLOYEE}))$$

Split up using assignment:

$$\mathtt{DEP5\_EMPS} \leftarrow \sigma_{\mathtt{DNO=5}}(\mathtt{EMPLOYEE})$$

$$\pi_{\mathtt{LNAME,FNAME,SALARY}}(\mathtt{DEP5\_EMPS})$$

More general version of assignment: RENAME

General format:

$$\rho_{\texttt{S(B1, B2, …, Bn)}}(\texttt{R})$$

This does 2 things:

(1) Renames the relation to `S` (this is the part that's identical to the assignment operator)

(2) Renames the attributes

Note that the **degree of the relation remains unchanged**

(RENAME is not a superset of PROJECT)

RENAME is very useful together with generalized PROJECT

Remember that we can use the generalized PROJECT function to have functions in attribute lists.

Problem: these derived attributes **don’t have a natural name**

We can use RENAME to give them one:

$$\text{YEMPLOYEE} \leftarrow \rho_{(\text{LNAME, FNAME, YSALARY})}(\pi_{\text{LNAME,FNAME,}12 * \text{SALARY}}(\text{EMPLOYEE}))$$

Summary Unary Operators

$\sigma_{\texttt{DNO=4 AND Salary > 30000}}(\texttt{EMPLOYEE})$

Partition the relation horizontally (filter)

$\pi_{\texttt{LNAME,FNAME}}((\texttt{EMPLOYEE}))$

Partition the relation vertically (slice attributes)

$\texttt{NAMES} \leftarrow \pi_{\texttt{LNAME,FNAME}}((\texttt{EMPLOYEE}))$

Assign a name to results

$\rho_{\texttt{(LNAME, FNAME, YSALARY)}}(\pi_{\texttt{LNAME,FNAME,12 * SALARY}}(\texttt{EMPLOYEE}))$

Rename derived attributes

Relational Algebra Operations from Set Theory

There are three operations that relational algebra "inherits" from set theory:

UNION ($\cup$)

INTERSECTION ($\cap$)

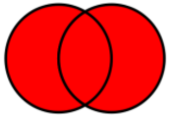
SET DIFFERENCE (-)

These are easy to understand when you keep in mind that **relations are mathematically sets of tuples**

So clearly they can do everything that other sets can do

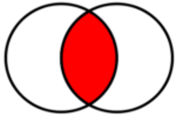
Relational Algebra Operations from Set Theory

R1 ∪ R2



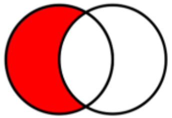
Resulting relation has all tuples from R1 and R2

R1 ∩ R2



Resulting relation has only those tuples that are in R1 and R2

R1 - R2



Resulting relation has only those tuples that are in R1 but not in R2

Type Compatibility

Important caveat:

`R1` and `R2` must be **type-compatible**

`R1(A1, A2, … An)` is type-compatible to `R2(B1, B2, … Bn)` *iff*

(1) `R1` and `R2` have the same number of attributes

(2) the domains of all attributes are the same:

`dom(Ai)=dom(Bi) for i=1, 2, ..., n`

Note that **attribute names don't matter** for type compatibility (only the domains!)

CARTESIAN PRODUCT

Another “general” set theoretical operation is the **cartesian product** (also called cross product, cross join) denoted by x

`R(A1, A2, …, An) x S(B1, B2, …, Bn)`

Result of the cartesian product is a new relation which contains **all possible combinations** of both source relations.

- New relation has all attributes of the source relations
- And `x * y` tuples, where `x` and `y` are the number of tuples in the source relations
- Type-compatiblity not required

CARTESIAN PRODUCT – Intuition

Ranks = {A, K, Q, J, 10, 9, 8, 7, 6 ,5, 4, 3, 2}

Suits = {♠, ♥, ♣, ♦}

Ranks x Suits = {

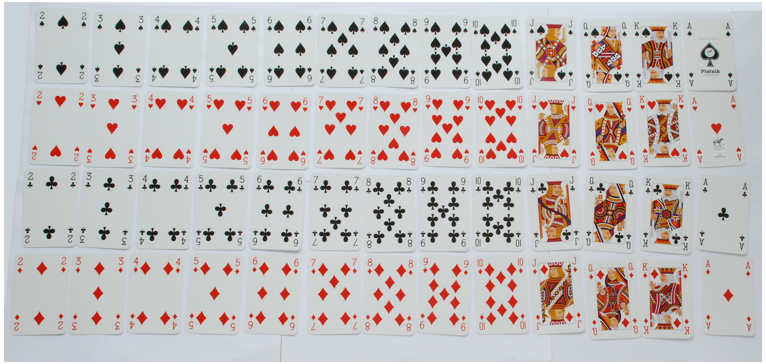
(A, ♠), (A, ♥), (A, ♣), (A, ♦),

(K, ♠), (K, ♥), (K, ♣), (K, ♦),

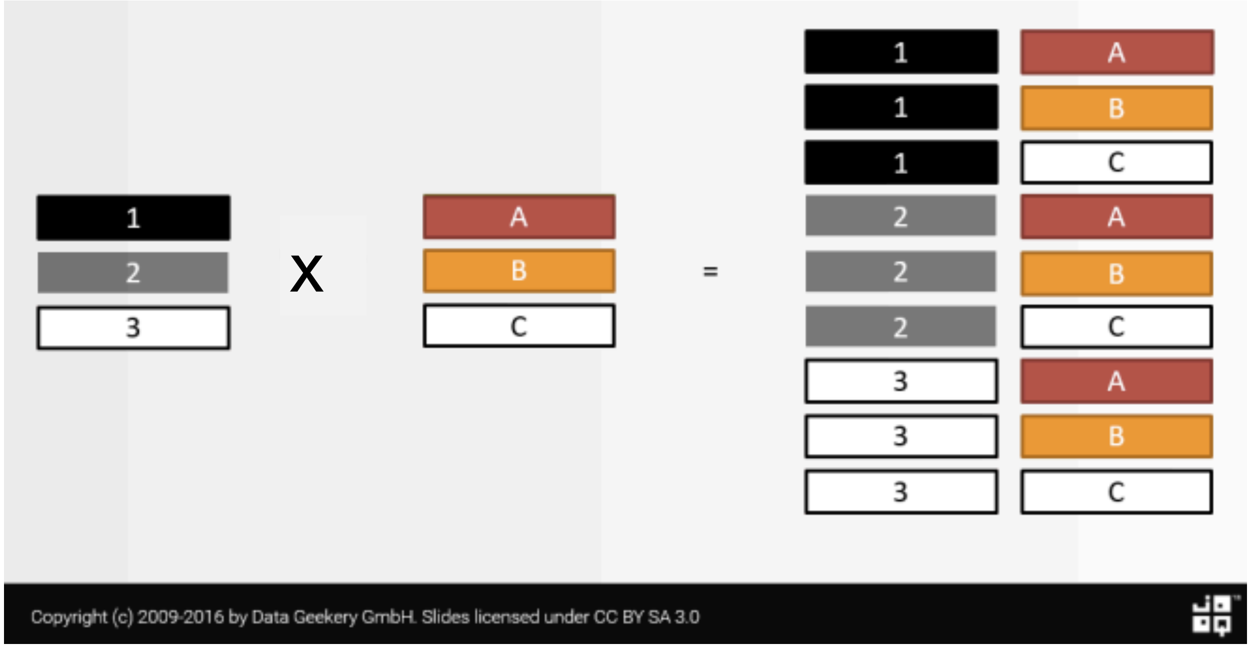
. . . ,

(2 , ♠), (2 , ♥), (2 , ♣), (2 , ♦),

}



CARTESIAN PRODUCT – Example



CARTESIAN PRODUCT

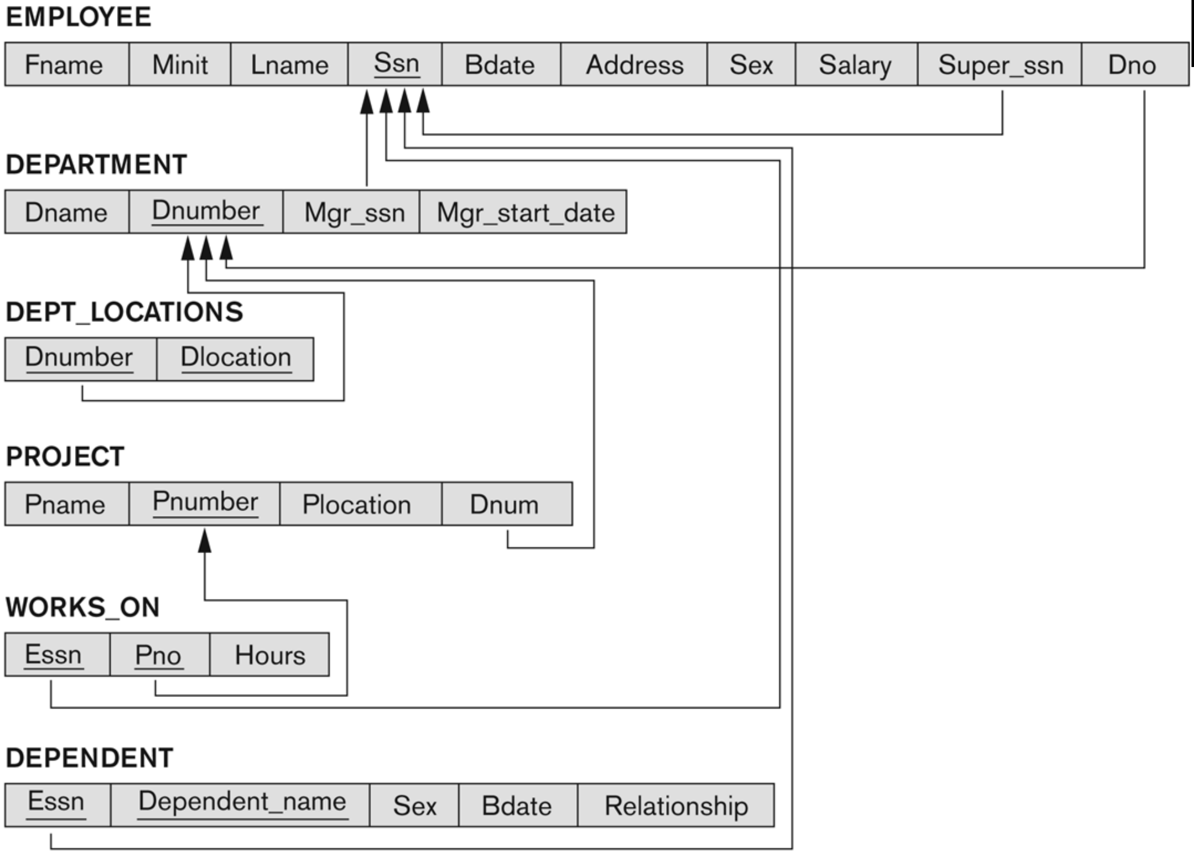
In general, the result of applying the cartesian product arbitrarily to two relations delivers garbage

Example:

EMPLOYEE x DEPENDENT

… result is a relation with 15 attributes and many, many tuples, one for every combination of employee and dependent

(not only for the employees **and their** dependents)


EMPLOYEE
Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno
DEPARTMENT
Dname | Dnumber | Mgr_ssn | Mgr_start_date
DEPT_LOCATIONS
Dnumber | Dlocation
PROJECT
Pname | Pnumber | Plocation | Dnum
WORKS_ON
Essn | Pno | Hours
DEPENDENT
Essn | Dependent_name | Sex | Bdate | Relationship

CARTESIAN PRODUCT

Usually, need to combine cartesian product with selection to make it meaningful

But then it becomes a powerful tool for **joining** tables

Example:

FEMALE_EMPS ← $\sigma_{SEX=F}$(EMPLOYEE)

EMPNAMES ← $\pi_{(FNAME,\ LNAME,\ SSN)}$(FEMALE_EMPS)

EMP_DEPENDENTS ← EMPNAMES x DEPENDENT

ACTUAL_DEPS ← $\sigma_{SSN=ESSN}$(EMP_DEPENDENTS)

RESULT ← $\pi_{(FNAME,\ LNAME,\ DEPENDENT\_NAME)}$(ACTUAL_DEPS)

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

$$\text{EMPNAMES} \leftarrow \pi_{\text{FNAME, LNAME, SSN}}(\text{FEMALE\_EMPS})$$

EMP_DEPENDENTS ← EMPNAMES x DEPENDENT

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

$$\text{ACTUAL\_DEPS} \leftarrow \sigma_{\text{SSN=ESSN}}(\text{EMP\_DEPENDENTS})$$

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

$$\text{RESULT} \leftarrow \pi_{(\text{FNAME, LNAME, DEPENDENT\_NAME})}(\text{ACTUAL\_DEPS})$$

JOIN Operator

Given how useful (and cumbersome) this joining via a combination of cartesian product and selection is, there are a number of **common shortcut operators**

The most fundamental one of those is the the **THETA JOIN**

Denoted by:

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

Example

Suppose that we want to retrieve the name of the **manager** of each **department**.

To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.

$$\mathrm{DEPT\_MGR} \leftarrow \mathrm{DEPARTMENT} \bowtie_{\mathrm{MGRSSN=SSN}} \mathrm{EMPLOYEE}$$

(MGRSSN=SSN is the join condition, or **theta**)

Example

DEPT_MGR

Dname	Dnumber	Mgr_ssn	. . .	Fname	Minit	Lname	Ssn	. . .
Research	5	333445555	. . .	Franklin	T	Wong	333445555	. . .
Administration	4	987654321	. . .	Jennifer	S	Wallace	987654321	. . .
Headquarters	1	888665555	. . .	James	E	Borg	888665555	. . .

Note that result relation contains **all attributes** from both source relations, **including join attributes**

Often desired to combine JOIN with a projection

Resolving ambiguity

When joining, attribute names are **not necessarily unique**

Especially, but not only, join attributes are often called the same in both source relations

Solution is to **prefix with relation name**:

$$\mathrm{CAR} \bowtie_{\mathrm{CAR.MANUFACTURER\_ID=MANUFACTURER.MANUFACTURER\_ID}} \mathrm{MANUFACTURER}$$

This can be done with **any** attribute name, not just ambiguous ones

Joins and Foreign Keys

In practice the single most important task of joins is to connect two relations based on a **primary key - foreign key** connection

Think about the join as creating a single relation from two separate input relations, where each tuple is “glued together” based on the equality of foreign keys in one relation to the primary keys in the other.

More specialized JOINS

EQUIJOIN

Any JOIN where the theta consists **only of a single equality comparison**

In theory thetas can be arbitrary conditional statements, but in practice most joins will be over the equality of two attributes

No special syntax:

$R \bowtie_{R.a = S.b} S$

More specialized JOINS

NATURAL JOINS

If the join attributes are **named the same** in both relations, we can use a simpler syntax, the NATURAL JOIN

R * S

(this is shorthand for $R \bowtie_{R.a\ =\ S.a} S$)

More specialized JOINS

NATURAL JOINS

If there is more than one pair of same-named attributes, **all of them** need to be equal

`R(A,B,C,D) * S(C,D,E)` is equivalent to:

$R(A,B,C,D) \bowtie_{(R.C=S.C \text{ AND } R.D=S.D)} S(C,D,E)$

JOIN Chains

In real databases we often need to join across a large number of relations:

$$\pi_{\mathtt{A.a,D.d}}(\mathtt{A * B * C * D})$$

("find the name of the department where the manager's dependent lives in …")

PROJ_DEPT ← PROJECT * DEPT

DEPT_LOCS ←
DEPARTMENT * DEPT_LOCATIONS

Figure 8.7 Results of two natural join operations. (a) proj_dept ← project * dept. (b) dept_locs ← department * dept_locations.

(a)

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b)

DEPT_LOCS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}),(\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}),(\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Key Takeaways

A lot to take in from this week …

But most important are:

Understanding the basic concepts of RA

Projection, Selection, Renaming

Joining and the relationship of joins to foreign keys