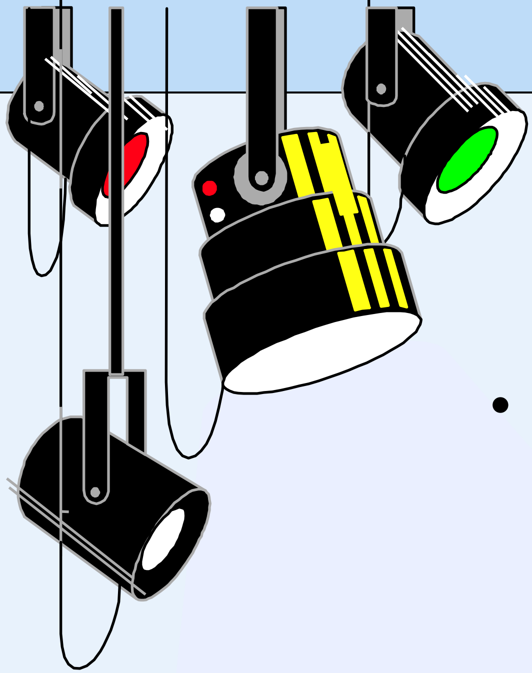
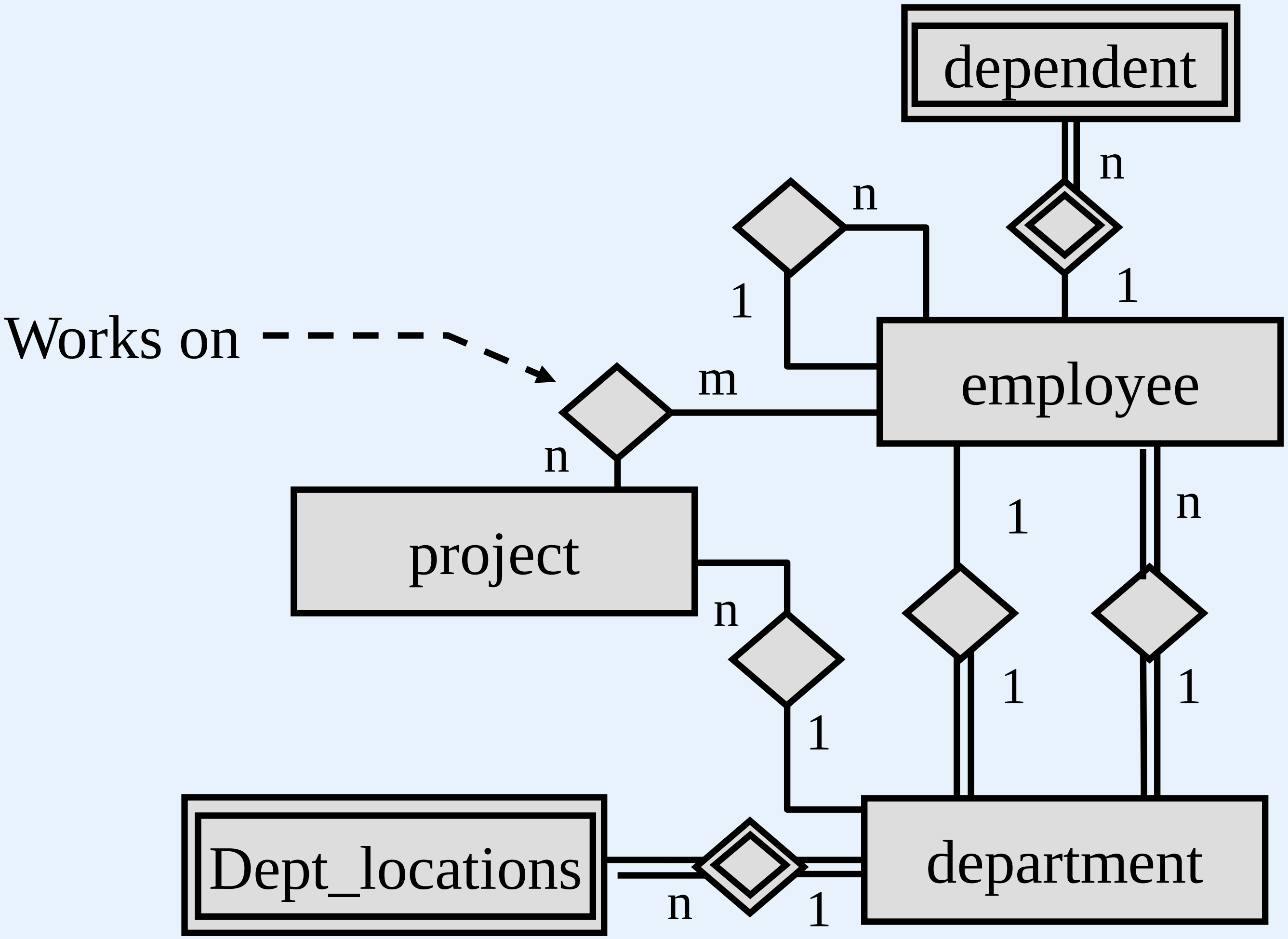
Relational Data Model



Outline: Relational Data Model

- Relational Data Model
  - relation schema, relations
  - database schema, database state
  - integrity constraints and updating
- Relational algebra
  - select, project, join, cartesian product division
  - set operations:
    union, intersection, difference

Relational Data Model



Relational Data Model

First introduced in 1970 by Ted Codd (IBM)

A **relation schema** R, denoted by $R(A_1, \ldots, A_n)$, is made up of a relation name R and a list of attributes $A_1, \ldots, A_n$.

A **relation** r(R) is a mathematical relation of degree n on the domains $dom(A_1)$, $dom(A_2)$, … $dom(A_n)$, which is a subset of the Cartesian product of the domains that define R:

$$r(R) \subseteq (dom(A_1) \times (dom(A_2) \times \ldots \times (dom(A_n))$$

formal terms	informal
relation	table
tuple	row
attribute	column header
domain	data type describing column values

Cartesian product

Emp(SSN, name, sex)

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} J \\ D \end{pmatrix} \begin{pmatrix} m \\ f \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \times \begin{pmatrix} J \\ D \end{pmatrix} = \{(1, J), (1, D), (2, J), (2, D), (3, J), (3, D)\}$$

Relational Data Model

Cartesian product

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \times \begin{bmatrix} J \\ D \end{bmatrix} \times \begin{bmatrix} m \\ f \end{bmatrix}$$

= {(<u>1, J, m</u>), (1, D, m), (2, J, m), (2, D, m), (3, J, m), (3, D, m),
(1, J, f), (1, D, f), (2, J, f), (<u>2, D, f</u>), (3, J, f), (3, D, f)}

Emp(SSN, name, sex)

1	J	m
2	D	f

Domain

A domain is a set of atomic values from which values can be drawn

- Examples
  - social insurance numbers: set of valid 9-digit social insurance numbers
  - names: set of names of persons
  - grade point average: possible values of computed grade point averages; each must be a real number between 0 and 4.5.

Domain

In many systems one specifies a data type (e.g. integer, date, string(20), …) and writes supporting application code to enforce any specific constraints (e.g. a SIN must be a 9-digit number).

Attribute

An attribute A_i is a name given to the role a domain plays in a relation schema R.

Relation (or Relation State)

A relation, or relation state, r of the relation schema $R(A_1, A_2, \ldots A_n)$ is a set of n-tuples $r=\{t_1, t_2, \ldots t_m\}$, where each n-tuple is an ordered list of n values $t_i=< v_1, v_2, \ldots v_n >$ $(i = 1, \ldots, m)$.

Relational Data Model

Relation Schema example

EMPLOYEE(Name, SSN, HomePhone, Address, OfficePhone, …)

EMPLOYEE Relation example:

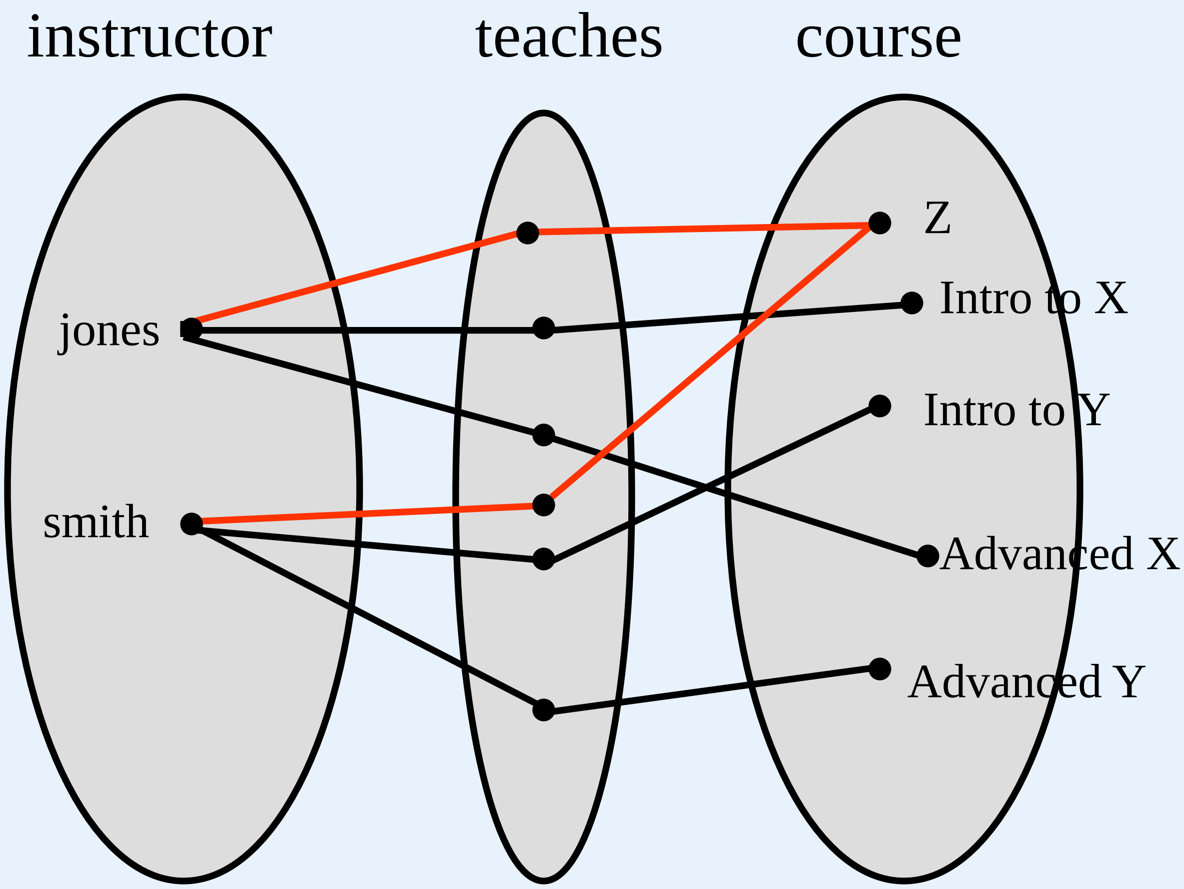
EMPLOYEE	**Name**	**SSN**	**HomePhone**	**Address**	**• •**
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	...
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	...
	Dick Davidson	422-11-2320	*null*	3452 Elgin Road	...

See Figure 6.1

Some characteristics of relations

- no ordering of tuples
- each value in a tuple is atomic
  - no composite values
  - separate relation tuples for multivalued attributes
- some attributes may be null
  - no value
  - value missing/unknown
- a relation is an assertion
  - e.g. an employee entity has a Name, SSN, HomePhone, etc
  - each tuple is a fact or a particular instance
  - some relations store facts about relationships

Relational Data Model



jones	z
jones	Intro to X
jones	Advanced X
smith	z
smith	Intro to Y
smith	Advanced Y

Relational Database

- a relational database schema S is a set of relation schemas $S = \{R_1, R_2, ...\}$ and a set of integrity constraints IC.
- A relational database state DB of S is a set of relation states $DB=\{r(R_1), r(R_2), ...\}$ such that ...
- Figure 7.5 - a schema
- Figure 7.6 - a possible relational database state
- Figure 7.7 - RI constraints

Relational Data Model

EMPLOYEE

fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno

DEPARTMENT

Dname, dnumber, mgrssn, mgrstartdate

DEPT _LOCATIONS

Dnumber, dlocation

PROJECT

Pname, pnumber, plocation, dnum

WORKS ON

Essn pno, hours

DEPENDENT

Essn, dependentname, sex, bdate, relationship

A database schema:

Relational Data Model

EMPLOYEE

fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno									
John	B	Smith	123489	1965-01-09	731 Fondren	M	40000	343488	5
Franklin	T	Wong	239979	1955-01-10	638 Voss	M	50000	343488	5

r(EMPLOYEE)

DEPARTMENT

Dname, dnumber, mgrssn, mgrstartdate			
Research	5	343488	1988-05-22

r(DEPARTMENT)

DEPT _LOCATIONS

Dnumber, dlocation	
5	Houston
6	Stafford

r(DEPT_LOCATION)

A database state:

• • •

Integrity Constraints

- any database will have some number of constraints that must be applied to ensure correct data (valid states)

  1. domain constraints
     - a domain is a restriction on the set of valid values
     - domain constraints specify that the value of each attribute A must be an atomic value from the domain dom(A).
  2. key constraints
     - a superkey is any combination of attributes that uniquely identify a tuple: $t_1[\text{superkey}] \neq t_2[\text{superkey}]$.
       - Example: <Name, SSN> (in **Employee**)
     - a key is superkey that has a minimal set of attributes
       - Example: <SSN> (in **Employee**)

Integrity Constraints

- If a relation schema has more than one key, each of them is called a candidate key.
- one candidate key is chosen as the primary key (PK)
- foreign key (FK) is defined as follows:

i) Consider two relation schemas R_1 and R_2;

ii) The attributes in FK in R_1 have the same domain(s) as the primary key attributes PK in R_2; the attributes FK are said to reference or refer to the relation R_2;

iii) A value of FK in a tuple t_1 of the current state $r(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r(R_2)$ or is null. In the former case, we have $t_1[FK] = t_2[PK]$, and we say that the tuple t_1 references or refers to the tuple t_2.

Example:

FK   FK

Employee(SSN, …, Dno)   Dept(Dno, …, MGRSSN)

Integrity Constraints

3. entity integrity
   - no part of a PK can be null

4. referential integrity
   - domain of FK must be same as domain of PK
   - FK must be null or have a value that appears as a PK value

5. semantic integrity
   - other rules that the application domain requires:
     - state constraint: gross salary > net income
     - transition constraint: *Widowed* can only follow *Married*; salary of an employee cannot decrease

Relational Data Model

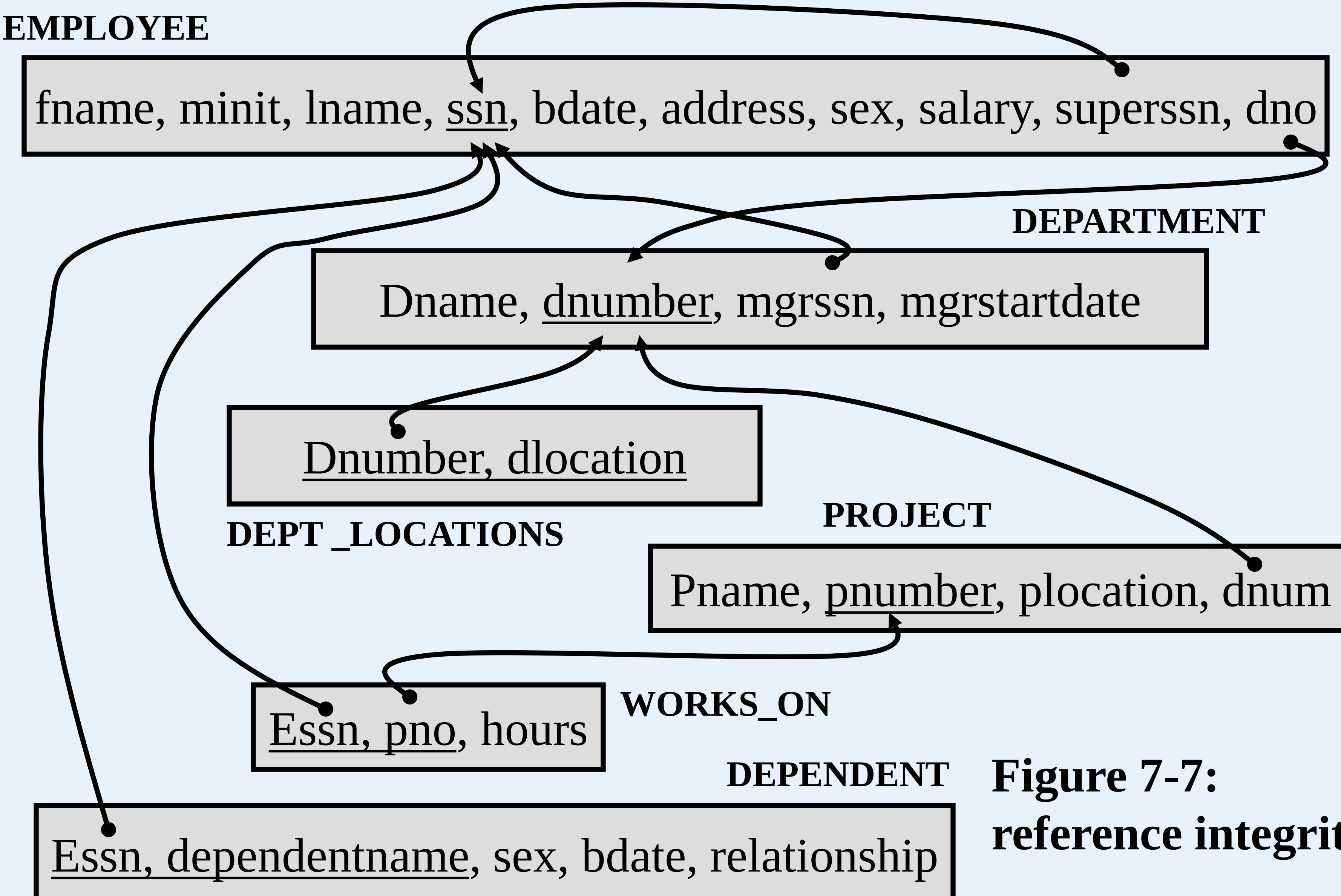
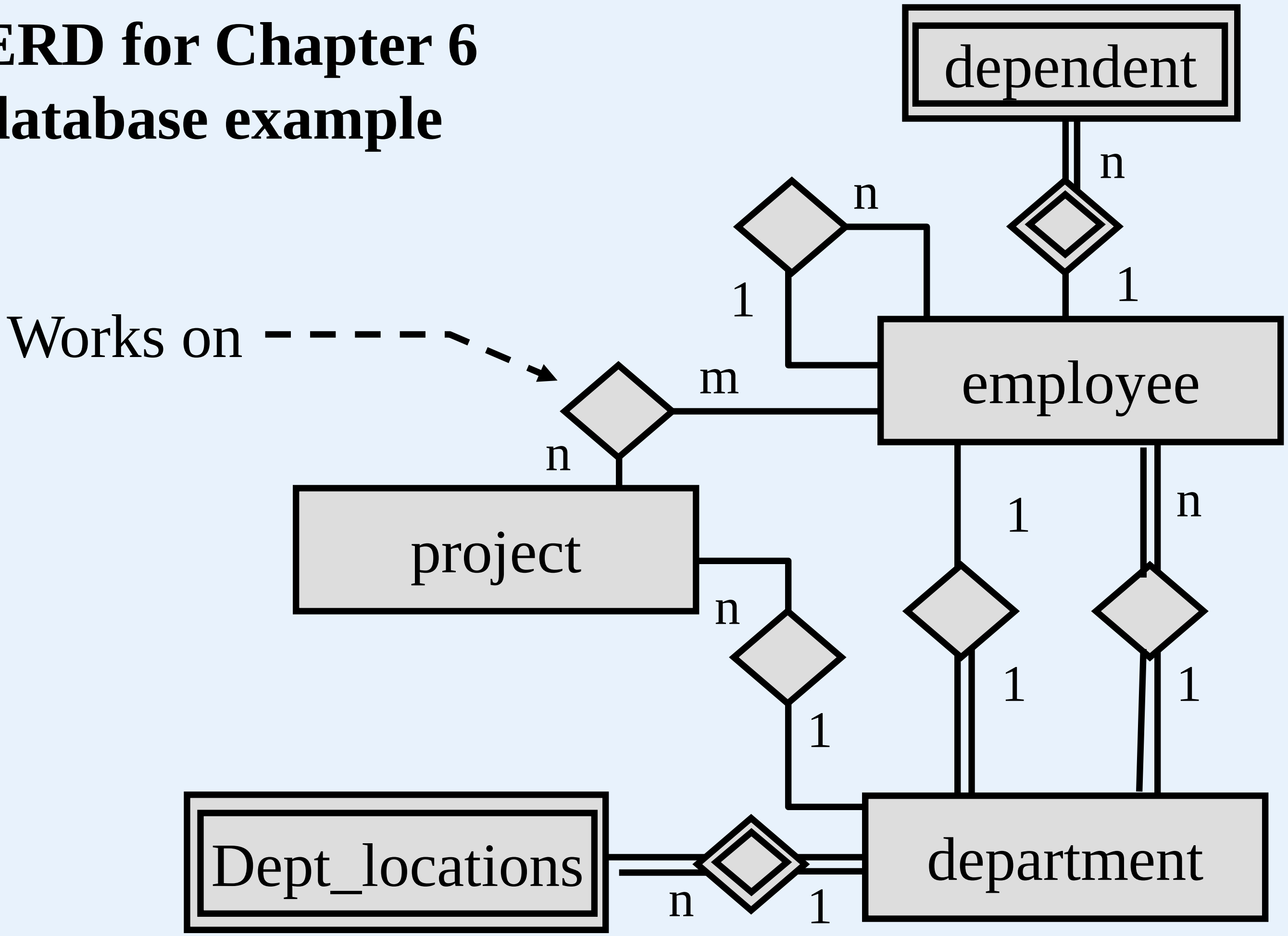


Figure 7-7: reference integrity

ERD for Chapter 6 database example



Updating and constraints

insert

- Insert the following tuple into EMPLOYEE:

<‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 40000, null, 4>

- When inserting, the integrity constraints should be checked: domain, key, entity, referential, semantic integrity

update

- Update the SALARY of the EMPLOYEE tuple with ssn = ‘999887777’ to 30000.
- When updating, the integrity constraints should be checked: domain, key, entity, referential, semantic integrity

Updating and constraints

delete

- Delete the WORK_ON tuple with Essn = ‘999887777’ and pno = 10.
- When deleting, the referential constraint will be checked.
  - The following deletion is not acceptable:

    Delete the EMPLOYEE tuple with ssn = ‘999887777’

    - reject, cascade, modify

Relational Data Model

cascade – a strategy to enforce referential integrity

Employee

ssn	…	
123456789		
…		

delete

Works-on

Essn	Pno
123456789	5
…	…

delete

cascade – a strategy to enforce referential integrity

Employee

<u>ssn</u>	… …	supervisor
123456789		234589710
… …		
234589710		null

delete

Employee

<u>ssn</u>	… …	supervisor
123456789		234589710
… …		
234589710		null

not reasonable

delete

delete

Modify – a strategy to enforce referential integrity

Employee

ssn	…	
123456789		
…		

delete

Works-on

Essn	Pno
123456789	5
…	…

Works-on

Essn	Pno
null	5
…	…

This violates the entity constraint.

Relational Data Model

Modify – a strategy to enforce referential integrity

Employee

<u>ssn</u>	…	
123456789		
…		

delete

Department

<u>Dno</u>	…	chairman
5		123456789
…		

Department

<u>Dno</u>	…	chairman
5		null
…		

This does not violate the entity constraint.

Relational Algebra

- a set of relations
- a set of operations
  - relation specific
    - select
    - project
    - join
    - division
  - set operations
    - union
    - intersection
    - difference
    - cartesian product

Relational algebra

select

- horizontal subset

project

- vertical subset

join (equijoin, natural join, inner, outer)

- combine multiple relations

cartesian product

union, intersection, difference

division

Relational algebra - Select

- horizontal subset
- symbol: **σ**
- boolean condition for row filter
- e.g. employees earning more than 30,000
  - $\sigma_{salary>30000}$(Employee)

fname	minit	…	salary	...
Franklin	T	…	40000	...
Jennifer	S	…	43000	...
James	E	…	55000	...

Every column of Employee appears in the result

Relational algebra - Project

- vertical subset
- symbol: π
- e.g. names of employees
- $\pi_{\text{fname, minit, lname}}$(Employee)

fname	minit	lname
John	B	Sarah
Franklin	T	Wong
Alicia	J	Zalaya
Jennifer	S	Wallace
Ramesh	K	Narayan
Joyce	A	English
Ahmad	V	Jabbar
James	E	Borg

Relational algebra - Join

- join or combine tuples from two relations into single tuples
- symbol: $\bowtie$
- boolean condition specifies the join condition
- e.g. to report on employees and their dependents
  - Employee $\bowtie_{\text{ssn=essn}}$ Dependent

fname	minit	…	essn	dependent_name	…

All attributes of both employee and dependent will appear

Relational algebra - Join

- Employee $\bowtie_{ssn=essn}$ Dependent

fname	minit	...	ssn
Franklin	T	...	333445555
Jennifer	S	...	987654321
John	B	...	123456789

Essn	dependent_name	...
333445555	Alice	
333445555	Theodore	
333445555	Joy	
987654321	Abner	
123456789	Michael	
123456789	Alice	
123456789	Elizabeth	

$$\text{Employee} \bowtie_{ssn=essn} \text{Dependent}$$

fname	minit	ssn	essn	dependent_name	...
Franklin	T	333445555	333445555	Alice	
Franklin	T	...	333445555	Theodore	
Franklin	T	...	333445555	Joy	
Jennifer	S	...	987654321	Abner	
John	B	...	123456789	Michael	
John	B	...	123456789	Alice	
John	B	...	123456789	Elizabeth	

Relational algebra - Join

- what is the result of
  - Employee ⋈ Dependent ?
  - Note there is no join condition

This is the

Cartesian Product

fname	minit	...	essn	dependent_name	...

If "Employee" contains 7 rows and "Dependent" contains 8 rows, there would be 8 times 7 = 56 rows in the result

Relational algebra

- e.g. to report on employees and their dependents
  - R1 $\longleftarrow$ Employee $\bowtie$ Dependent
  - R2 $\longleftarrow$ $\sigma_{\text{ssn=essn}}$ (R1)
  - Result $\longleftarrow$ $\pi_{\text{fname, minit, lname, dependent\_name}}$ (R2)

fname	minit	lname	dependent_name
Franklin	T	Wong	Alice
Franklin	T	Wong	Theodore
Franklin	T	Wong	Joy
Jennifer	S	Wallace	Abner
John	B	Smith	Michael
John	B	Smith	Alice
John	B	Smith	Elizabeth

Relational algebra - Join

- **equijoin** - one condition and the = operator
- **natural** join - an equijoin with removable of superfluous attribute(s).
- **inner** join - only tuples (in one relation) that *join* with at least one tuple (in the other relation) are included. This is what we have exhibited so far.
- **outer** join - full outer join, left outer join, right outer join

Relational algebra - Natural join

- **natural** join - an equijoin with removable of superfluous attribute(s). E.g. to list employees and their dependents:

  - employee $*$ dependent

    has all attributes of employee, and all attributes of dependent minus essn, in the result

  - if there is ambiguity regarding which attributes are involved, use a list notation like:

    employee $*_{\{ssn,\ essn\}}$ dependent

Outer Joins

R ⋈ S

- join - only matching tuples are in the result

R ⟕ S

- left outer join - all tuples of R are in the result regardless ...

R ⟖ S

- right outer join - all tuples of S are in the result regardless ...

R ⟗ S

- full outer join - all tuples of R and S are in the result regardless ...

Left Outer Joins

r1

A	B1
a1	b1
a2	b2
a3	b3

r2

B2	C
b1	c1
b3	c3
b4	c4

$r1 \;⟕_{B1=B2}\; r2$

A	B1	C
a1	b1	c1
a2	b2	null
a3	b3	c3

Right Outer Joins

r1

A	B1
a1	b1
a2	b2
a3	b3

r2

B2	C
b1	c1
b3	c3
b4	c4

$$r1 \underset{B1=B2}{⟖} r2$$

A	B1	C
a1	b1	c1
a3	b3	c3
null	b4	c4

Full Outer Joins

r1

A	B1
a1	b1
a2	b2
a3	b3

r2

B2	C
b1	c1
b3	c3
b4	c4

$r1 ⟗_{B1=B2}\ r2$

A	B1	C
a1	b1	c1
a2	b2	null
a3	b3	c3
null	b4	c4

Outer Joins

Employee ⟕ Department
ssn=mgrssn

Result: a list of all employees and also the department they manage if they happen to manage a department.

Project ⟕ Works_on
pno=pnumber

Works_on ⟖ Project
pno=pnumber

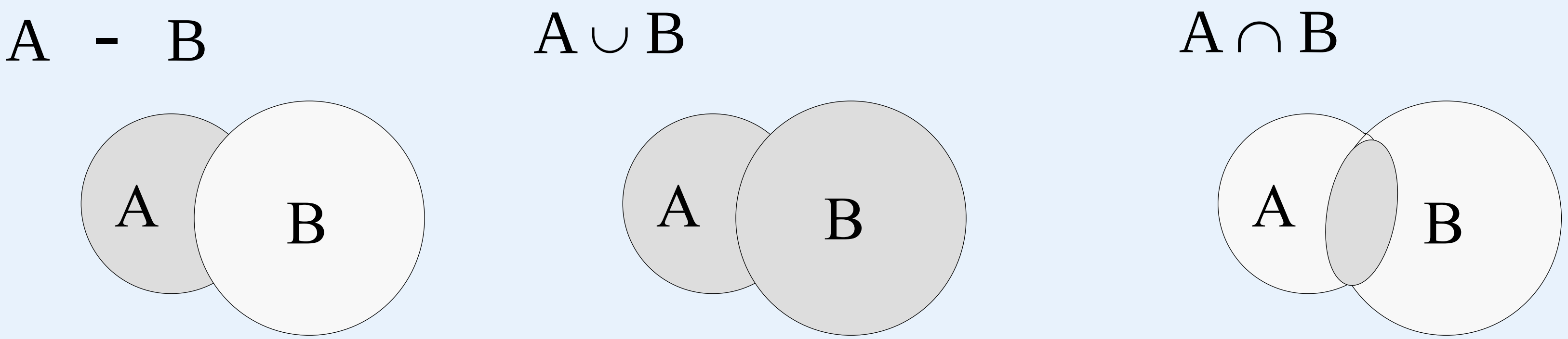
Dependent ⟖ Employee
ssn=essn

Set difference, union, intersection

$A - B$

$A \cup B$

$A \cap B$



Division

$T \leftarrow R \div S$

R

A	B
a1	b1
a1	b2
a2	b1
a3	b1
a3	b2

$\div$

S

B
b1
b2

=

T

A
a1
a3

Division

Query: Retrieve the name of employees who work on all the projects that ‘John Smith’ works on.

$\text{SMITH} \leftarrow \sigma_{\text{FNAME = 'John' and LNAME = 'Smith'}}(\text{EMPLOYEE})$

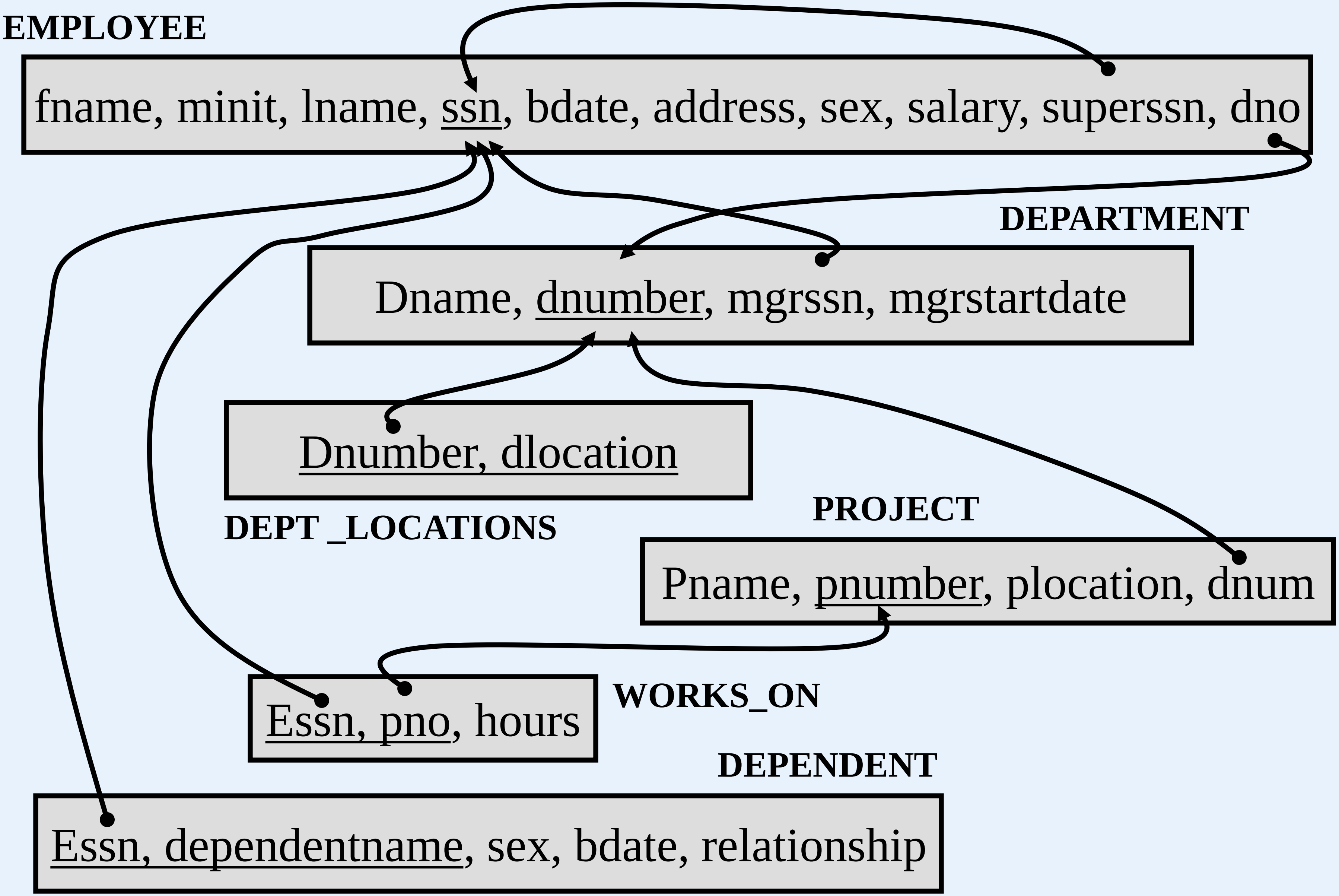
$\text{SMITH\_PNOs} \leftarrow \pi_{\text{PNO}}(\text{WORK\_ON} \bowtie_{\text{ESSN = SSN}} \text{SMITH})$

$\text{SSN\_PNO} \leftarrow \pi_{\text{ESSN,PNO}}(\text{WORK\_ON})$

$\text{SSNS(SSN)} \leftarrow \text{SSN\_PNO} \div \text{SMITH\_PNOs}$

$\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME}}(\text{SSNS} * \text{EMPLOYEE})$

Relational Data Model



EMPLOYEE

ssn	fname	lname
1	John	Smith
2	John	Smith
3	Marry	Black

WORK_ON

essn	PNo	hours
1	1	...
1	2	...
2	3	...
3	1	...
3	2	...
3	3	...
3	4	...

$$SMITH \leftarrow \sigma_{\text{FNAME = 'John' and LNAME = 'Smith'}}(\text{EMPLOYEE})$$

SMITH

ssn	fname	lname
1	John	Smith
2	John	Smith

$$\text{SMITH\_PNOs} \leftarrow \pi_{\text{PNO}}(\text{WORK\_ON} \bowtie_{\text{ESSN = SSN}} \text{SMITH})$$

$\text{WORK\_ON} \bowtie_{\text{ESSN = SSN}} \text{SMITH}$

ssn	fname	lname	essn	PNo	hours
1	John	Smith	1	1	...
1	John	Smith	1	2	...
2	John	Smith	2	3	...

SMITH_PNOs

Pno
1
2
3

$SSN\_PNO \leftarrow \pi_{ESSN,PNO}(WORK\_ON)$

SSN_PNO

essn	PNo
1	1
1	2
2	3
3	1
3	2
3	3
3	4

SSNS(SSN) ← SSN_PNO ÷ SMITH_PNOs

SSN_PNO

essn	PNo
1	1
1	2
2	3
3	1
3	2
3	3
3	4

÷

SMITH_PNOs

Pno
1
2
3

=

SSNS(SSN)

ssn
3

$\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME}}(\text{SSNS} * \text{EMPLOYEE})$

RESULT

ssn	fname	lname
3	Marry	Black

Division

The DIVISION operator can be expressed as a sequence of π, $\times$, and - operations as follows:

$Z = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$, $X = \{B_1, \ldots, B_m\}$,
$Y = Z - X = \{A_1, \ldots, A_n\}$,

$$R(Z) \div S(X) \Rightarrow \left\{ \begin{array}{l} T_1 \leftarrow \pi_Y(R) \\ T_2 \leftarrow \pi_Y((S \times T_1) - R) \\ T \leftarrow T_1 - T_2 \end{array} \right.$$

result