DATABASE SYSTEMS

CS – 355/ CE – 373

Instructor: Maria N. Samad

September 25th, 2024

ADDITIONAL OPERATIONS

- The additional relational algebra operations that can be derived from the fundamental operations are:
  - Intersection(∩)
  - Join (⋈)
  - Assignment (←)

INTERSECTION (∩)

- A relational operation that results in tuples that are both in *R* and *S* in common
- Notation: $R \cap S$
- This takes the mutually common tuples from both relations to form a new one
- Just like the Union operation, the relations must be compatible according to the following two conditions:
  - *r*, *s* have the *same arity* (i.e. same number of attributes)
  - Attributes of *r* and *s* are compatible

INTERSECTION (∩)

- Example:
  - Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

The *section* relation.

- Query:
  - $\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2017}(section)) \cap \Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2018}(section))$

- Result →

course_id
CS-101

INTERSECTION (∩) – EXERCISES

- Activity Sheet:
  - Attempt **Part H**
- These are intersection examples, so even though these can be achieved in other ways, you are supposed to use Intersection operations
- Assumption for Q3: Instructors who have taught in any of the two semesters, which means Instructor A could have taught in Spring only; and Instructor B could have taught in Fall only; and Instructor C could have taught in both Spring and Fall. The query should consider and access all three of them.

INTERSECTION (∩) – EXERCISES

- Activity Sheet **Part H** Solution:

1	$\prod_{ID,\ grade}(\sigma_{(course\_id = \text{“CS-355”} \wedge semester = \text{“Fall”} \wedge year = 2024)}(takes)) \cap \prod_{ID,\ grade}(\sigma_{(course\_id = \text{“CS-224”} \wedge semester = \text{“Fall”} \wedge year = 2024)}(takes))$
2	$\prod_{building}(classroom) \cap \prod_{building}(department)$
3	$\prod_{ID}(\sigma_{(semester = \text{“Spring”} \wedge year = 2020)}(teaches) \cup \sigma_{(semester = \text{“Fall”} \wedge year = 2020)}(teaches)) \cap \prod_{ID}(advisors)$

JOIN (⋈)

- The ***join*** operation is used to combine *related tuples* from two relations into single, longer tuples
- An important operation for any RDBMS, because it allows us to process relationships among relations
- Basically, it allows us to combine a ***select*** operation and a ***cartesian product*** operation into a single operation
- Notation: $\boldsymbol{r} \bowtie_{\boldsymbol{\theta}} \boldsymbol{s}$

JOIN (⋈)

- In simpler terms, a ***join*** operation is a ***cartesian product*** between two relations followed by a ***select*** operation on some predicate, $\boldsymbol{\theta}$
  - $\boldsymbol{r} \bowtie_{\boldsymbol{\theta}} \boldsymbol{s} = \boldsymbol{\sigma}_{\boldsymbol{\theta}} (\boldsymbol{r} \times \boldsymbol{s})$
- $\boldsymbol{\theta}$ can be any comparison operators: {=, ≠, ≥, >, ≤, <} and can be combined with any of the connectives: {∧, ∨, ¬}
- This is also known as ***theta join*** or ***inner join***

JOIN (⋈)

- The result of the ***join*** operation between relations *R* and *S*, is a new relation with arity = $n_R + n_S$, where n_R is the number of attributes in *R*, and n_S is the number of attributes in *S*
- The resultant relation will have one tuple of each combination of tuples – one from *R* and one from *S*, but only when the join condition satisfies

JOIN (⋈)

- ***Difference between Cartesian Product and Join operations:***
  - In ***Cartesian Product***, all combinations of tuples are included in the result
  - In ***Join*** operation however, only those combination of tuples that satisfy the join condition will appear in the result
- The tuples whose join attributes are NULL or for which the condition is FALSE, will not appear in the resultant relation
- This means it does not always preserve the original information of the participating relations, hence it is crucial to define joins only when necessary

JOIN (⋈)

- Variations of ***join*** operation:
  - ***Equi Join***
  - ***Natural Join***
  - ***Outer Join*** – will be discussed in SQL later

JOIN (⋈)

- ***Equi Join***:
  - An extension of the ***join*** operation where θ is simply an equality predicate
  - That is, only equal condition is allowed
  - For example, find names of all instructors together with course IDs of all the courses they taught:
    - $\Pi_{name,\ course\_id}\ (instructor\ \bowtie_{instructor.ID\ =\ teaches.ID}\ teaches)$

$$\sigma_{instructor.id\ =\ teaches.id}\ (instructor\ \times\ teaches)$$

JOIN (⋈)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

The *instructor* relation

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

The *teaches* relation

JOIN (⋈)

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

Result of $\sigma_{instructor.ID = teaches.ID}(instructor \times teaches)$ **OR** $instructor \bowtie_{instructor.ID = teaches.ID} teaches$

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

Result of $\prod_{name,\ course\_id}$ (*instructor* $\bowtie_{instructor.ID = teaches.ID}$ *teaches*)

JOIN (⋈)

- ***Natural Join***:
  - From Equi Join, we can see that even though the number of tuples were reduced to only relevant ones as opposed to Cartesian Product, but there were attributes being repeated each time
  - This problem can be resolved by using Natural Join
  - The new relation that is generated as a result of Natural Join joins the attributes with the same name in both relations, but do not appear repeatedly in the resultant relation
  - Notation: $R * S$ or $R \bowtie S$ (without any predicate)

JOIN (⋈)

- ***Natural Join***:
  - As there is no joining condition specified in the query, it automatically selects the common attributes between the relations, i.e. identical names of the attributes as the joining attribute, and combines the relations based on those attributes, while removing the superfluous information
  - For example, in Department and Dept_Locations tables (see next slide for the relations), we have a common attribute called Dnumber, the natural join can combine the tuples based on that attribute. Write down the query for the natural join
    - *Department * Dept_Locations* **OR** *Department ⋈ Dept_Locations*

JOIN (⋈)

Given DEPARTMENT and DEPT_LOCATIONS, below

DEPARTMENT

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

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

A natural join of these relations is shown as follows:

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

JOIN (⋈)

- ***Natural Join***:
  - If the common attributes do not have the same name, they must be renamed first and only then natural join can be performed on them
  - Example, a department number is called Dnum in Project relation, and Dnumber in Department relation, even though they mean the same thing (see next slides for the relations). Before a natural join can be applied to it, they should be renamed first, and then used in the query, while removing the repetition
    - *Project* * ($\rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}$ (*Department*))

JOIN (⋈)

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

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

Result of natural join operation proj_dept ← project * dept.

JOIN (⋈)

- ***Natural Join***:
  - Multiple join operations can be combined together
  - Both join and union are associative operations, i.e.
    - **(*instructor* ⋈ *teaches*) ⋈ *course* = *instructor* ⋈ (*teaches* ⋈ *course*)**
  - For Example, find names of all instructors in CS department and their course titles that they teach
    - $\Pi_{name,\ title}\ (\sigma_{dept\_name\ =\ "CS"}\ (instructor\ \bowtie\ teaches\ \bowtie\ course)\ )$

JOIN (⋈) – EXERCISES

- Activity Sheet:
  - Attempt **Part I**

JOIN (⋈) – EXERCISES

- Activity Sheet **Part I** Solution:

1	Π_{title} ((*course)* ⋈ $\sigma_{\neg(\text{semester = “Summer” ^ year = 2024})}$ (*section*))
2	$\Pi_{\text{course\_id, start\_time, end\_time}}$ (($\sigma_{(\text{semester = “Fall” ^ year = 2024})}$ (*section*)) ⋈ ($\sigma_{\text{day = “Monday”}}$ (*time_slot*)))
3	$\Pi_{\text{name, grade}}$ ($\sigma_{\text{dept\_name = “CE” ^ semester = “Fall” ^ year = 2024 ^ (ID ≥ 301 ^ ID ≤ 399)}}$ (*students* ⋈ *takes*))
4	*course* ⋈ *prereq*
5	$\Pi_{\text{room\_number, capacity}}$(*classroom* ⋈$_{\text{(classroom.building = department.building)}}$ ($\sigma_{\text{dept\_name = “ECE”}}$ (*department*)))
6	$\Pi_{\text{s\_name, i\_name}}$ (($\rho_{\text{(s\_ID, s\_name, s\_dept\_name, tot\_cred)}}$(*student*) ⋈ *advisors*) ⋈ $\rho_{\text{(i\_ID, i\_name, i\_dept\_name, salary)}}$(*instructor*))

ASSIGNMENT (←)

- It is convenient at times to write a relational-algebra expression by assigning parts of it to ***temporary relation variables***.
- The assignment operation is denoted by ← and works like assignment in a programming language.
- Notation: $temp1 \leftarrow \boldsymbol{\sigma_{\theta}}(R)$

  $\prod_{A1,\, A2}(temp1)$

ASSIGNMENT (←)

- Example: Find all instructor in the “Physics” and “Music” department.

$$Physics \leftarrow \sigma_{dept\_name=\text{“Physics”}}(instructor)$$
$$Music \leftarrow \sigma_{dept\_name=\text{“Music”}}(instructor)$$
$$Physics \cup Music$$

- With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query.

ASSIGNMENT (←) – EXERCISES

- Activity Sheet:
  - Attempt **Part J**

ASSIGNMENT (←) – EXERCISES

- Activity Sheet **Part J** Solution:

1	***fall*** $\leftarrow \sigma_{(\text{semester = "Fall" ^ year = 2023})}$ (*teaches*) ***spring*** $\leftarrow \sigma_{(\text{semester = "Spring" ^ year = 2024})}$ (*teaches*) $\prod_{\text{course\_id}}$ (*course*) – $\prod_{\text{course\_id}}$ (***fall*** ∪ ***spring***)
2	***class*** $\leftarrow \prod_{\text{building}}$ (*classroom*) ***dept*** $\leftarrow \prod_{\text{building}}$ (*department*) ***class*** ∩ ***dept***
3	***all*** $\leftarrow \prod_{\text{course\_id}}$ (*course*) ***summer*** $\leftarrow \prod_{\text{course\_id}}$ ($\sigma_{(\text{semester = "Summer" ^ year = 2024})}$ (*section*)) ***notSummer*** $\leftarrow$ ***all – summer***
4	***fall*** $\leftarrow \sigma_{(\text{semester = "Fall" ^ year = 2024})}$ (*section*) ***monday*** $\leftarrow \sigma_{\text{day = "Monday"}}$ (*time_slot*) ***joined*** $\leftarrow$ ***fall*** ⋈ ***monday*** ***result*** $\leftarrow \prod_{\text{course\_id, start\_time, end\_time}}$ (***joined***)

EQUIVALENT QUERIES

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000
- Query 1: $\sigma_{dept\_name=\text{“Physics”} \wedge salary > 90{,}000}$ (*instructor*)
- Query 2: $\sigma_{dept\_name=\text{“Physics”}}(\sigma_{salary > 90.000}$ (*instructor*))
- The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

SELF PRACTICE

8.18. Consider the LIBRARY relational database schema shown in Figure 8.14, which is used to keep track of books, borrowers, and book loans. Referential integrity constraints are shown as directed arcs in Figure 8.14, as in the notation of Figure 5.7. Write down relational expressions for the following queries:

a. How many copies of the book titled *The Lost Tribe* are owned by the library branch whose name is 'Sharpstown'?

b. How many copies of the book titled *The Lost Tribe* are owned by each library branch?

c. Retrieve the names of all borrowers who do not have any books checked out.

d. For each book that is loaned out from the Sharpstown branch and whose Due_date is today, retrieve the book title, the borrower's name, and the borrower's address.

e. For each library branch, retrieve the branch name and the total number of books loaned out from that branch.

f. Retrieve the names, addresses, and number of books checked out for all borrowers who have more than five books checked out.

g. For each book authored (or coauthored) by Stephen King, retrieve the title and the number of copies owned by the library branch whose name is Central.

** Content from Dr. Faisal Alvi's slides **

SELF PRACTICE

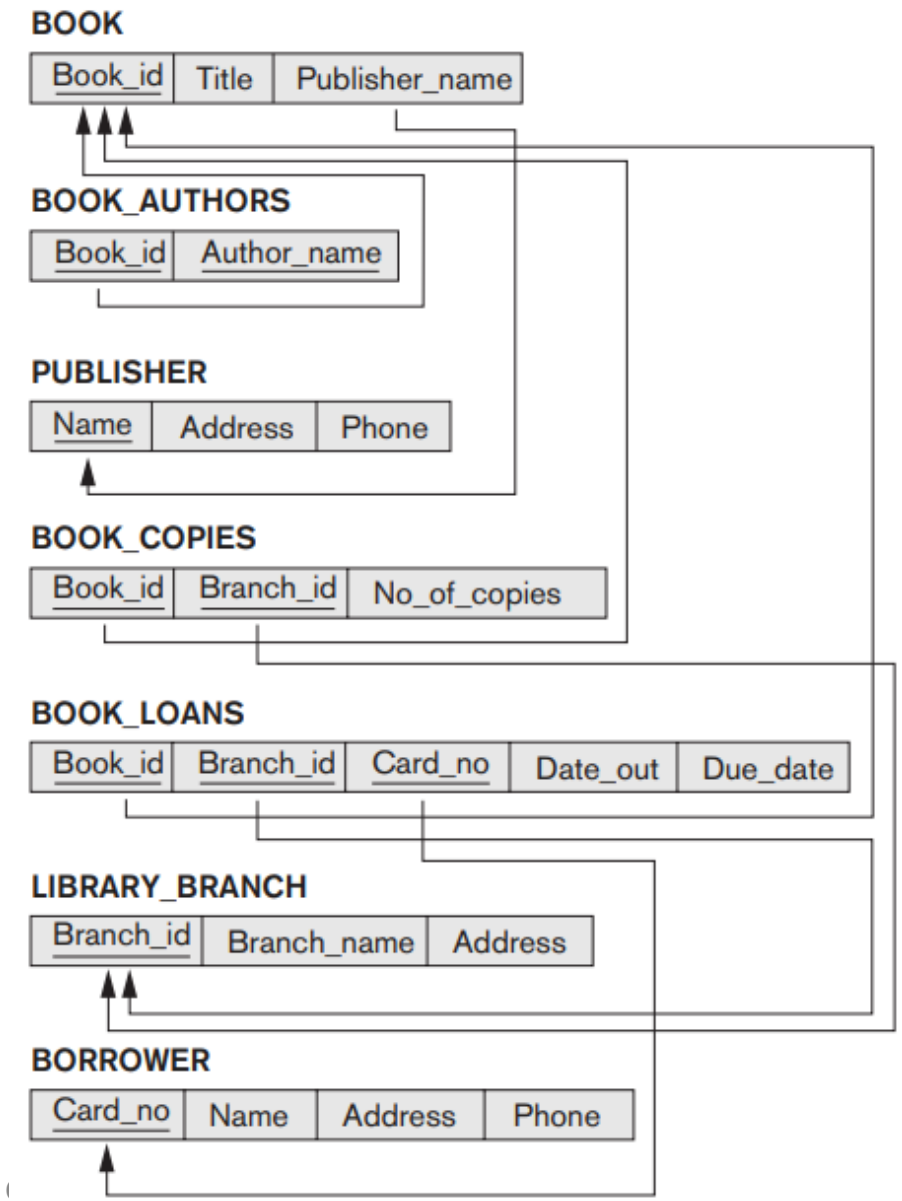

Figure 8.14
A relational database schema for a LIBRARY database.

SELF PRACTICE

8.17. Consider the AIRLINE relational database schema shown in Figure 5.8, which was described in Exercise 5.12. Specify the following queries in relational algebra:

a. For each flight, list the flight number, the departure airport for the first leg of the flight, and the arrival airport for the last leg of the flight.

b. List the flight numbers and weekdays of all flights or flight legs that depart from Houston Intercontinental Airport (airport code 'iah') and arrive in Los Angeles International Airport (airport code 'lax').

c. List the flight number, departure airport code, scheduled departure time, arrival airport code, scheduled arrival time, and weekdays of all flights or flight legs that depart from some airport in the city of Houston and arrive at some airport in the city of Los Angeles.

d. List all fare information for flight number 'co197'.

e. Retrieve the number of available seats for flight number 'co197' on '2009-10-09'.

** Content from Dr. Faisal Alvi's slides **

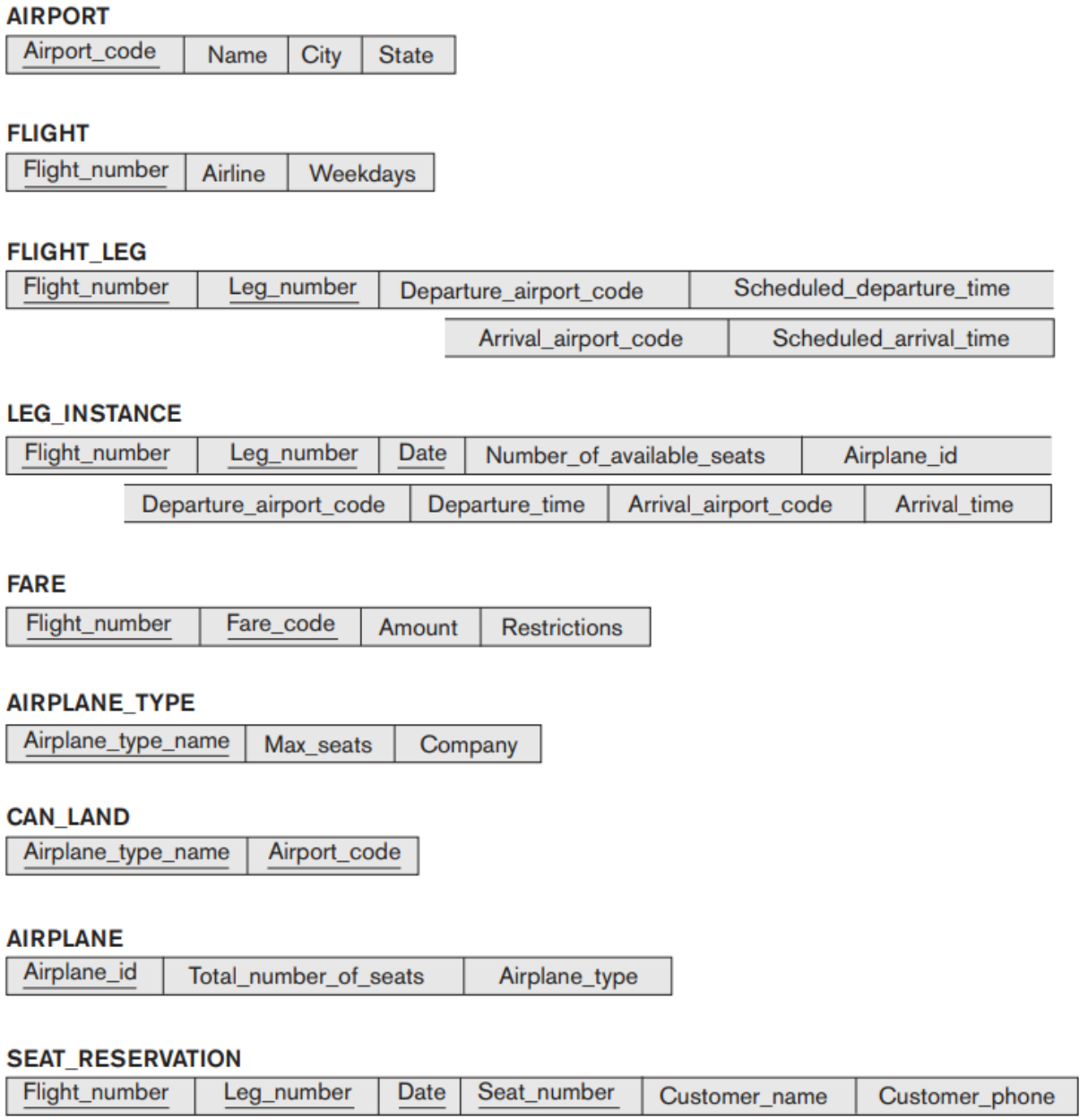

Figure 5.8
The AIRLINE relational database schema.

SELF PRACTICE

14.29. Consider the following relations for an order-processing application database at ABC, Inc.

ORDER (O#, Odate, Cust#, Total_amount)
ORDER_ITEM(O#, I#, Qty_ordered, Total_price, Discount%)

Assume that each item has a different discount. The Total_price refers to one item, Odate is the date on which the order was placed, and the Total_amount is the amount of the order. If we apply a natural join on the relations ORDER_ITEM and ORDER in this database, what does the resulting relation schema RES look like? What will be its key? Show the FDs in this resulting relation. Is RES in 2NF? Is it in 3NF? Why or why not? (State assumptions, if you make any.)