Chapter 5 Questions and Exercises: 5.1, (also define the term Super key, Key, Candidate Key, Primary Key).5.3, 5.4, 5.6, 5.9, 5.11(b, c, e,f, h, i, k), 5.12, 5.14, 5.17. Chapter 7 Exercise: (1) Map the ER diagram in Figure 3.21 on PP 95 into relational schemas. Justify your choice of mapping options. (2) Map the EER diagram in Figure 4.9 on page 120 into relational schemas. Justify your choice of mapping options.

5.1

* Domain – a set of atomic values. Usually the values a data type can take.
* Attribute – the name of a role played by some domain in the relation schema.
* N-tuple – an ordered list of n values where each element belongs to an element of its corresponding attribute domain or is NULL.
* Relation schema – a relation name R and a list of attributes and a set of integrity constraints.
* Relation state – only the valid tuples that represent a particular state of the real world. A set of n-tuples.
* Degree of a relation – aka arity, is the number of attributes in the relation schema.
* Relational database schema – the union of all relation schemas for all relation schemas in the database and a set of integrity constraints.
* Relational database state – the union of all relation states for all relations in the database.
* Super key – specifies a uniqueness constraint that no two distinct tuples in any state r of R can have the same value for SK.
* Candidate key – each of the possible keys. A relation may have more than one key, and each one is known as a candidate key.
* Primary key – the candidate key chosen to uniquely identify tuples in the relation.

5.3

* Duplicate tuples aren’t allowed in a relation because a relation is defined as a set of tuples. By definition all elements of a set are distinct, and thus, all tuples in a relation must also be distinct.

5.4

* The difference between key and super key is that a key is a super key in which you cannot remove anymore attributes lest it no longer be a key. A key is a minimal super key.

5.6

* Characteristics of relations that make them different from ordinary tables and files
  + A relation is subject to more constraints than a table.
  + Ordering of tuples in a relation doesn’t matter. You can reorder the tuples and the two relations are equivalent. The same isn’t true for files.
  + A relation must have unique tuples, whereas a table or file can have duplicate entries.
  + A table is n-dimensional, where a relation is 2-dimensional.
  + Attributes are independent in a table, they aren’t in a relation.
  + A table is a physical instance of a relation, but a relation is a logical construct.

5.9

* Foreign key – a set of attributes is a foreign key in R1 that references relation R2 if
  + The attributes in FK have the same domain as the primary key attributes in R2
  + A value of an attribute in FK in R1 exists in the corresponding attribute in R2 or is NULL
  + This is used to reference one relation’s key from another relation to enforce referential integrity.

5.11

* B
  + Referential integrity constraint of Dnum. No tuples with DEPARTMENT.Dnumber =2 .
  + Fix by not inserting the tuple and asking the user to correct the error.
* C
  + Uniqueness constraint of Dnumber. Tuple with DEPARTMENT.Dnumber=4 already exists.
  + Referential integrity constraint of Mgr\_ssn. No tuples with EMPLOYEE.Ssn=’943775543’.
  + Fix by not inserting the tuple and asking the user to correct the error.
* E
  + Violates the domain of attribute DEPENDENT.Relationship. ‘spouse’ is not equivalent to ‘Spouse’ (with a capital S).
  + Fix by capitalizing the S.
* F
  + No violations.
* H
  + Referential integrity constraint of Pnumber. WORKS\_ON.Pno references this and two tuples will have invalid values for WORKS\_ON.Pno.
  + Fix by either rejecting, setting WORKS\_ON.Pno=1 to WORKS\_ON.Pno=NULL, or by cascading the delete to delete the two tuples in WORKS\_ON.Pno=1.
* I
  + No violations.
* K
  + No violations.

5.12

* A – The operations for this are insert operation(s)
  + INSERT <Flight\_number, Leg\_number, Date, Seat\_number, Customer\_name, Customer\_phone> into SEAT\_RESERVATION with the given information. The customer will supply their name and phone number. Flight\_number, Leg\_number, Date, and Seat\_number will all be shown to the user as a list of choices and they will choose a particular one.
* B – I would expect to check that Flight\_number exists in FLIGHT, Leg\_number exists in FLIGHT\_LEG and LEG\_INSTANCE, Date exists in LEG\_INSTANCE, Seat\_number doesn’t exceed Max\_seats in AIRPLANE\_TYPE, that Number\_of\_available\_seats in LEG\_INSTANCE isn’t 0, and that the Customer\_name and Customer\_phone aren’t NULL. Make sure that the combination of the 4 attributes for the key type aren’t already in SEAT\_RESERVATION and that none of these attributes are NULL.
* C
  + Key constraints – Ensuring that {Flight\_number, Leg\_number, Date, Seat\_number} is unique.
  + Entity integrity constraints – Ensuring that none of {Flight\_number, Leg\_number, Date, Seat\_number} are NULL.
  + Referential integrity constraints – Ensuring that Flight\_number exists in FLIGHT, Leg\_number exists in FLIGHT\_LEG and LEG\_INSTANCE.
  + Other constraints – Ensuring that Customer\_name and Customer\_phone are not NULL. Checking to make sure the Number\_of\_available\_seats > 0 and that Seat\_number < Max\_seats.
* D
  + FLIGHT\_LEG.Flight\_number references FLIGHT.Flight\_number.
  + FLIGHT\_LEG.Departure\_airport\_code references AIRPORT.Airport\_code.
  + FLIGHT\_LEG.Arrival\_airport\_code references AIRPORT.Airport\_code.
  + LEG\_INSTANCE.Flight\_number references FLIGHT\_LEG.Flight\_number.
  + LEG\_INSTANCE.Airplane\_id references AIRPLANE.Airplane\_id.
  + LEG\_INSTANCE.Departure\_airport\_code references FLIGHT\_LEG.Airport\_code.
  + LEG\_INSTANCE.Arrival\_airport\_code references FLIGHT\_LEG.Airport\_code.
  + FARE.Flight\_number references FLIGHT.Flight\_number.
  + CAN\_LAND.Airplane\_type\_name references AIRPLANE\_TYPE.Airplane\_type\_name.
  + CAN\_LAND.Airport\_code references AIRPORT.Airport\_code.
  + AIRPLANE.Airplane\_type references AIRPLANE\_TYPE.Airplane\_type\_name.
  + AIRPLANE.Total\_number\_of\_seats references AIRPLANE\_TYPE.Max\_seats (maybe).
  + SEAT\_RESERVATION.Flight\_number references FLIGHT.Flight\_number.
  + SEAT\_RESERVATION.Leg\_number references LEG\_INSTANCE.Leg\_number.
  + SEAT\_RESERVATION.Date references FLIGHT.Weekdays and FLIGHT\_LEG.Scheduled\_departure\_time.

5.14

* Referential integrity constraints
  + ORDER.Cust# references CUSTOMER.Cust#.
  + ORDER\_ITEM.Order# references ORDER.Order#.
  + ORDER\_ITEM.item# references ITEM.Item#.
  + SHIPMENT.Order# references ORDER.Order#.
  + SHIPMENT.Warehouse# references WAREHOUSE.Warehouse#.
* Other constraints
  + In addition to the referential integrity constraints listed, since the foreign keys are used as part of the primary key in some relations, we have some additional NOT-NULL constraints.
  + ORDER\_ITEM.Order# may not be NULL.
  + ORDER\_ITEM.Item# may not be NULL.
  + SHIPMENT.Order# may not be NULL.
  + SHIPMENT.Warehouse# may not be NULL.

5.17

* Referential integrity constraints
  + OPTION.Serial\_no references CAR.Serial\_no.
  + SALE.Salesperson\_id references SALESPERSON.Salesperson\_id.
  + SALE.Serial\_no references SALESPERSON.Salesperson\_id.
  + Since bartering usually happens, a SALE.Sale\_price does not necessarily equal a CAR.Price. Usually the sale price is less than the advertised price.
* Some example populations

CAR

|  |  |  |  |
| --- | --- | --- | --- |
| 123456789 | Sonata | Hyundai | 12500.00 |
| 987654321 | Cobalt | Chevrolet | 7500.00 |
| 101010101 | Windstar | Ford | 19500.00 |

OPTION

|  |  |  |
| --- | --- | --- |
| 123456789 | power\_locks | 250.00 |
| 987654321 | spoiler | 200.00 |

SALE

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 123456789 | 08-15-2008 | 11500.00 |
| 1 | 987654321 | 07-25-2010 | 6750.00 |

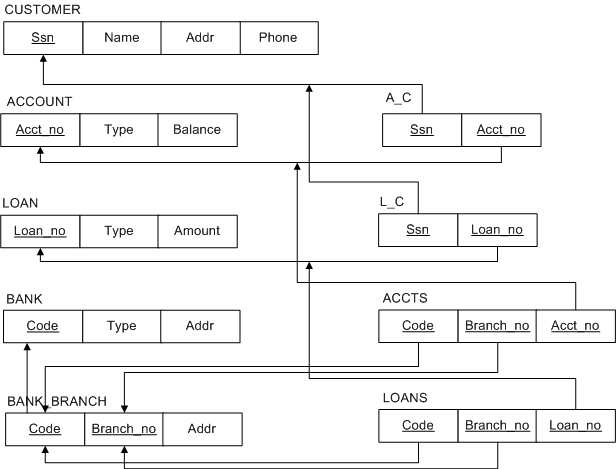
SALESPERSON

|  |  |  |
| --- | --- | --- |
| 1 | Gary | 888-555-9999 |
| 2 | Rebecca | 888-666-9999 |

* SALE violation: INSERT <3, 111111111, NULL, NULL>.
* SALESPERSON violation: INSERT<1, Jeremy, NULL>.
* SALE no violation: INSERT<1, 101010101, 10-15-2012, 18750.00>.
* SALESPERSON no violation: INSERT<3, Katy, 888-777-9999>.

7 (1) The following is my derived database schema for the diagram. Some of the arrow heads don’t go all the way to the referenced attribute, and instead merge with another arrow that was already pointing to that attribute. The referenced attribute should still be clear.

* Justifications: I followed the procedures set forth in class to make BANK\_BRANCH its own entity, whose partial key combined with the identifying entity’s key creates a key for it. ACCTS and LOANS are at a specific BANK\_BRANCH rather than with a specific BANK.



7(2) The following is my derived database schema for the diagram. I typed it instead of drawing it because it is remarkably cleaner looking since Visio isn’t remarkably flexible with its arrowheads.

* I separate PERSON, FACULTY, and STUDENT into three entities for minimal data duplication.
* INSTRUCTOR\_RESEARCHER must be its own entity type because it has an M:N relationship.
* SECTION and CURRENT\_SECTION are rolled into one because CURRENT\_SECTION has only two specific attributes.
* MAJOR and MINOR get rolled into STUDENT because they are 1:N.
* ADVISOR gets rolled into GRAD\_STUDENT because it is 1:N.
* TEACH gets rolled into SECTION because it is 1:N.
* CS gets rolled into SECTION because it is 1:N.
* DC gets rolled into COURSE because it is 1:N.
* CD gets rolled into DEPARTMENT because it is 1:N.
* CHAIRS gets rolled into FACULTY because it is 1:1 and I chose that side.
* PI gets rolled into GRANT because it is 1:N.

These are the entities in my schema:

PERSON(Ssn, Fname, Minit, Lname, Bdate, Sex, No, Street, Apt\_no, City, State, Zip)  
FACULTY(Ssn, Rank, Foffice, Fphone, Salary, Chairs)  
STUDENT(Ssn, Class, Minor, Major)  
GRAD\_STUDENT(Ssn, College, Degree, Year, Advisor)  
GRANT(No, Title, Agency, St\_date, Faculty)  
INSTRUCTOR\_RESEARCHER(Ssn)  
DEPARTMENT(Dname, Dphone, Office, College)  
COLLEGE(Cname, Dean, Coffice)  
COURSE(C#, Cname, Cdesc, Department)  
SECTION(Sec#, Year, Qtr, Instructor, Course)  
REGISTERED(Student, Section)  
TRANSCRIPT(Student, Section, Grade)  
COMMITTEE(Faculty, Grad\_student)  
SUPPORT(Instructor, Faculty, Start, End, Time)  
BELONGS(Faculty, Department)

These are the referential integrity constraints:

FACULTY.Ssn references PERSON.Ssn  
FACULTY.Chairs references DEPARTMENT.Dname  
STUDENT.Ssn references PERSON.Ssn  
STUDENT.Minor references DEPARTMENT.Dname  
STUDENT.Major references DEPARTMENT.Dname  
GRAD\_STUDENT.Ssn references PERSON.Ssn  
GRAD\_STUDENT.College references COLLEGE.Cname  
GRAD\_STUDENT.Advisor references FACULTY.Ssn  
GRANT.Faculty references FACULTY.Ssn  
INSTRUCTOR\_RESEARCHER references FACULTY.Ssn or GRAD\_STUDENT.Ssn  
DEPARTMENT.College references COLLEGE.Cname  
COURSE.Department references DEPARTMENT.Dname  
SECTION.Instructor references INSTRUCTOR\_RESEARCHER.Ssn  
SECTION.Course references COURSE.C#  
REGISTERED.Student references STUDENT.Ssn  
REGISTERED.Section references SECTION.Sec#  
TRANSCRIPT.Student references STUDENT.Ssn  
TRANSCRIPT.Section references SECTION.Sec#  
COMMITTEE.Faculty references FACULTY.Ssn  
COMMITTEE.Grad\_student references GRAD\_STUDENT.Ssn  
SUPPORT.Faculty references FACULTY.Ssn  
SUPPORT.Instructor references INSTRUCTOR\_RESEARCHER.Ssn  
BELONGS.Faculty references FACULTY.Ssn  
BELONGS.Department references DEPARTMENT.Dname