1. Define the following terms and then create an example. (3 points each)
   1. Superkey:

A superkey is a set of one or more columns (attributes) that is able to identify a unique tuple in a table. There are many superkeys in a table (assuming there are more than one attribute).

* 1. Candidate key:

A candidate key is a type of superkey that cannot be reduced and still be able to identify a unique tuple in a table. That is, if a candidate key is made up of the attributes {A1, A2, A3}, none of the attributes can be removed and still have the key identify all of the unique tuples.

* 1. Primary key:

A primary key is simply the candidate key that is selected to key the table. That is, there may be multiple candidate keys in a table, but the only one is selected as the primary key. This key can later be used as a foreign key in other tables.

* 1. Foreign key:

A foreign key is an attribute in a table that contains the values of the primary key of another table. This foreign key represents the relation to that other table.

1. Look at the time\_slot relation. You can find this relation on page 1271 of the textbook. In this relation the primary key is made up of 3 attributes. The only attribute not in the key is end\_time. Try to tell me why this is the case. (2 points)

End time does not help identify a tuple in the table. That is, all four attributes make up a superkey, but not a candidate key. Each individual class section has only one time\_slot\_id, but the class may have multiple different days and starts times. However, for a given day and start time for a given class, that class is only going to have a single end time (typically, in the real world). Thus, having all four attributes is not a candidate key, and would not be selected as a primary key. End time could also have been selected as the third attribute instead of start time, but as most people likely care more about start time than end time, the designer chose the candidate key that included start time to be the PK.

1. What is the result of the following compound relational algebra statement? Make sure you base it on the textbook data (pages 1276 – 1281) and show your work (using the textbook data) not just the answer. In otherwords, show each table of the original data, then the intermediate table(s) in the order they occur, and finally the resulting table. (5 points)
   1. σs\_id=ID(student x advisor)

First, we take the cartesian product of student and advisor. We note that this table is going to be massive. The student relation has 13 tuples, and the advisor relation has 9 tuples, so the cartesian product of these relations has 117 tuples. The original tables (from the textbook), and the massive resulting intermediate result are shown below.

A screenshot of a cell phone

Description automatically generatedA close up of text on a white background

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *ID* | *name* | *dept\_name* | *tot\_cred* | *s\_id* | *I\_id* |
| 00128 | Zhang | Comp. Sci. | 102 | 00128 | 45565 |
| 00128 | Zhang | Comp. Sci. | 102 | 12345 | 10101 |
| 00128 | Zhang | Comp. Sci. | 102 | 23121 | 76543 |
| 00128 | Zhang | Comp. Sci. | 102 | 44553 | 22222 |
| 00128 | Zhang | Comp. Sci. | 102 | 45678 | 22222 |
| 00128 | Zhang | Comp. Sci. | 102 | 76543 | 45565 |
| 00128 | Zhang | Comp. Sci. | 102 | 76653 | 98345 |
| 00128 | Zhang | Comp. Sci. | 102 | 98765 | 98345 |
| 00128 | Zhang | Comp. Sci. | 102 | 98988 | 76766 |
| 12345 | Shankar | Comp. Sci. | 32 | 00128 | 45565 |
| 12345 | Shankar | Comp. Sci. | 32 | 12345 | 10101 |
| 12345 | Shankar | Comp. Sci. | 32 | 23121 | 76543 |
| 12345 | Shankar | Comp. Sci. | 32 | 44553 | 22222 |
| 12345 | Shankar | Comp. Sci. | 32 | 45678 | 22222 |
| 12345 | Shankar | Comp. Sci. | 32 | 76543 | 45565 |
| 12345 | Shankar | Comp. Sci. | 32 | 76653 | 98345 |
| 12345 | Shankar | Comp. Sci. | 32 | 98765 | 98345 |
| 12345 | Shankar | Comp. Sci. | 32 | 98988 | 76766 |
| 19991 | Brandt | History | 80 | 00128 | 45565 |
| 19991 | Brandt | History | 80 | 12345 | 10101 |
| 19991 | Brandt | History | 80 | 23121 | 76543 |
| 19991 | Brandt | History | 80 | 44553 | 22222 |
| 19991 | Brandt | History | 80 | 45678 | 22222 |
| 19991 | Brandt | History | 80 | 76543 | 45565 |
| 19991 | Brandt | History | 80 | 76653 | 98345 |
| 19991 | Brandt | History | 80 | 98765 | 98345 |
| 19991 | Brandt | History | 80 | 98988 | 76766 |
| 23121 | Chavez | Finance | 110 | 00128 | 45565 |
| 23121 | Chavez | Finance | 110 | 12345 | 10101 |
| 23121 | Chavez | Finance | 110 | 23121 | 76543 |
| 23121 | Chavez | Finance | 110 | 44553 | 22222 |
| 23121 | Chavez | Finance | 110 | 45678 | 22222 |
| 23121 | Chavez | Finance | 110 | 76543 | 45565 |
| 23121 | Chavez | Finance | 110 | 76653 | 98345 |
| 23121 | Chavez | Finance | 110 | 98765 | 98345 |
| 23121 | Chavez | Finance | 110 | 98988 | 76766 |
| 44553 | Peltier | Physics | 56 | 00128 | 45565 |
| 44553 | Peltier | Physics | 56 | 12345 | 10101 |
| 44553 | Peltier | Physics | 56 | 23121 | 76543 |
| 44553 | Peltier | Physics | 56 | 44553 | 22222 |
| 44553 | Peltier | Physics | 56 | 45678 | 22222 |
| 44553 | Peltier | Physics | 56 | 76543 | 45565 |
| 44553 | Peltier | Physics | 56 | 76653 | 98345 |
| 44553 | Peltier | Physics | 56 | 98765 | 98345 |
| 44553 | Peltier | Physics | 56 | 98988 | 76766 |
| 45678 | Levy | Physics | 46 | 00128 | 45565 |
| 45678 | Levy | Physics | 46 | 12345 | 10101 |
| 45678 | Levy | Physics | 46 | 23121 | 76543 |
| 45678 | Levy | Physics | 46 | 44553 | 22222 |
| 45678 | Levy | Physics | 46 | 45678 | 22222 |
| 45678 | Levy | Physics | 46 | 76543 | 45565 |
| 45678 | Levy | Physics | 46 | 76653 | 98345 |
| 45678 | Levy | Physics | 46 | 98765 | 98345 |
| 45678 | Levy | Physics | 46 | 98988 | 76766 |
| 54321 | Williams | Comp. Sci. | 54 | 00128 | 45565 |
| 54321 | Williams | Comp. Sci. | 54 | 12345 | 10101 |
| 54321 | Williams | Comp. Sci. | 54 | 23121 | 76543 |
| 54321 | Williams | Comp. Sci. | 54 | 44553 | 22222 |
| 54321 | Williams | Comp. Sci. | 54 | 45678 | 22222 |
| 54321 | Williams | Comp. Sci. | 54 | 76543 | 45565 |
| 54321 | Williams | Comp. Sci. | 54 | 76653 | 98345 |
| 54321 | Williams | Comp. Sci. | 54 | 98765 | 98345 |
| 54321 | Williams | Comp. Sci. | 54 | 98988 | 76766 |
| 55739 | Sanchez | Music | 38 | 00128 | 45565 |
| 55739 | Sanchez | Music | 38 | 12345 | 10101 |
| 55739 | Sanchez | Music | 38 | 23121 | 76543 |
| 55739 | Sanchez | Music | 38 | 44553 | 22222 |
| 55739 | Sanchez | Music | 38 | 45678 | 22222 |
| 55739 | Sanchez | Music | 38 | 76543 | 45565 |
| 55739 | Sanchez | Music | 38 | 76653 | 98345 |
| 55739 | Sanchez | Music | 38 | 98765 | 98345 |
| 55739 | Sanchez | Music | 38 | 98988 | 76766 |
| 70557 | Snow | Physics | 0 | 00128 | 45565 |
| 70557 | Snow | Physics | 0 | 12345 | 10101 |
| 70557 | Snow | Physics | 0 | 23121 | 76543 |
| 70557 | Snow | Physics | 0 | 44553 | 22222 |
| 70557 | Snow | Physics | 0 | 45678 | 22222 |
| 70557 | Snow | Physics | 0 | 76543 | 45565 |
| 70557 | Snow | Physics | 0 | 76653 | 98345 |
| 70557 | Snow | Physics | 0 | 98765 | 98345 |
| 70557 | Snow | Physics | 0 | 98988 | 76766 |
| 76543 | Brown | Comp. Sci. | 58 | 00128 | 45565 |
| 76543 | Brown | Comp. Sci. | 58 | 12345 | 10101 |
| 76543 | Brown | Comp. Sci. | 58 | 23121 | 76543 |
| 76543 | Brown | Comp. Sci. | 58 | 44553 | 22222 |
| 76543 | Brown | Comp. Sci. | 58 | 45678 | 22222 |
| 76543 | Brown | Comp. Sci. | 58 | 76543 | 45565 |
| 76543 | Brown | Comp. Sci. | 58 | 76653 | 98345 |
| 76543 | Brown | Comp. Sci. | 58 | 98765 | 98345 |
| 76543 | Brown | Comp. Sci. | 58 | 98988 | 76766 |
| 76653 | Aoi | Elec. Eng. | 60 | 00128 | 45565 |
| 76653 | Aoi | Elec. Eng. | 60 | 12345 | 10101 |
| 76653 | Aoi | Elec. Eng. | 60 | 23121 | 76543 |
| 76653 | Aoi | Elec. Eng. | 60 | 44553 | 22222 |
| 76653 | Aoi | Elec. Eng. | 60 | 45678 | 22222 |
| 76653 | Aoi | Elec. Eng. | 60 | 76543 | 45565 |
| 76653 | Aoi | Elec. Eng. | 60 | 76653 | 98345 |
| 76653 | Aoi | Elec. Eng. | 60 | 98765 | 98345 |
| 76653 | Aoi | Elec. Eng. | 60 | 98988 | 76766 |
| 98765 | Bourikas | Elec. Eng. | 98 | 00128 | 45565 |
| 98765 | Bourikas | Elec. Eng. | 98 | 12345 | 10101 |
| 98765 | Bourikas | Elec. Eng. | 98 | 23121 | 76543 |
| 98765 | Bourikas | Elec. Eng. | 98 | 44553 | 22222 |
| 98765 | Bourikas | Elec. Eng. | 98 | 45678 | 22222 |
| 98765 | Bourikas | Elec. Eng. | 98 | 76543 | 45565 |
| 98765 | Bourikas | Elec. Eng. | 98 | 76653 | 98345 |
| 98765 | Bourikas | Elec. Eng. | 98 | 98765 | 98345 |
| 98765 | Bourikas | Elec. Eng. | 98 | 98988 | 76766 |
| 98988 | Tanaka | Biology | 120 | 00128 | 45565 |
| 98988 | Tanaka | Biology | 120 | 12345 | 10101 |
| 98988 | Tanaka | Biology | 120 | 23121 | 76543 |
| 98988 | Tanaka | Biology | 120 | 44553 | 22222 |
| 98988 | Tanaka | Biology | 120 | 45678 | 22222 |
| 98988 | Tanaka | Biology | 120 | 76543 | 45565 |
| 98988 | Tanaka | Biology | 120 | 76653 | 98345 |
| 98988 | Tanaka | Biology | 120 | 98765 | 98345 |
| 98988 | Tanaka | Biology | 120 | 98988 | 76766 |

Now that we have the cartesian product, we can evaluate our select algebra on the intermediate result. We are selecting all of the rows where the ID value is equal to the s\_id value.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *ID* | *name* | *dept\_name* | *tot\_cred* | *s\_id* | *I\_id* |
| 00128 | Zhang | Comp. Sci. | 102 | 00128 | 45565 |
| 12345 | Shankar | Comp. Sci. | 32 | 12345 | 10101 |
| 23121 | Chavez | Finance | 110 | 23121 | 76543 |
| 44553 | Peltier | Physics | 56 | 44553 | 22222 |
| 45678 | Levy | Physics | 46 | 45678 | 22222 |
| 76543 | Brown | Comp. Sci. | 58 | 76543 | 45565 |
| 76653 | Aoi | Elec. Eng. | 60 | 76653 | 98345 |
| 98765 | Bourikas | Elec. Eng. | 98 | 98765 | 98345 |
| 98988 | Tanaka | Biology | 120 | 98988 | 76766 |

From the textbook:

1. (2.6) Consider the following expressions, which use the result of a relational algebra operation as the input to another operation. For each expression, explain in words what the expression does.
   1. σyear≥2009(takes) ⋈ student

This operation first selects all of the tuples from the relation takes, which has a year whose a value is greater than or equal to 2009. The resulting new relation is naturally joined to the student relation. Essentially this is finding all of the classes that every student has taken or is taking since 2009.

* 1. σyear≥2009(takes ⋈ student)

This operation first does a full natural join on the entire takes and student relations. This builds a large intermediary relation that contains all of the students and every class those students have taken. Then, the operation selects only those tuples whose rows have a year value greater than or equal to 2009. The end result should be the same as the first operation, but the processing/query time is not necessarily the same.

* 1. ∏ID,name,course\_id (student ⋈ takes)

This operation also does a natural join on the student and takes relations, building a new relation containing all of the students and the classes they are taking. We then take the projection of this intermediary result, selecting only the ID, name, and course\_id attributes. This results in a final relation that contains only student names, student IDs, and all of the courses that this student has taken (note, duplicates are dropped).

1. (2.7) Consider the relational database of Figure 2.14. Give an expression in the relational algebra to express each of the following queries:
   1. Find the names of all employees who live in city “Miami”.

∏person\_name(σcity=”Miami” (employee))

* 1. Find the names of all employees whose salary is greater than $100,000.

∏person\_name(σsalary≥100000 (employee⋈works))

* 1. Find the names of all employees who live in “Miami” and whose salary is greater than $100,000.

∏person\_name(σsalary≥100000 and city=”Miami”(employee⋈works))

1. (2.8) Consider the bank database of Figure 2.15. Give an expression in the relational algebra for each of the following queries.
   1. Find the names of all branches located in “Chicago”.

∏branch\_name(σbranch\_city=”Chicago” (branch))

* 1. Find the names of all borrowers who have a loan in branch “Downtown”.

∏customer\_name(σbranch\_name=”Downtown” ((borrower⋈loan)⋈branch))

1. (2.10) Consider the advisor relation shown in Figure 2.8, with *s\_id* as the primary key of advisor. Suppose a student can have more than one advisor. Then, would *s\_id* still be a primary key of the advisor relation? If not, what should the primary key of advisor be?

If a student could have multiple advisors, *s\_id* would no longer be the primary key in the advisor relation. This is because you would have multiple different tuples in the relation identified by a single value of *s\_id*. Rather, we would make a new primary key that consisted of both the *s\_id* and the *i\_id* fields.

1. (2.12) Consider the relational database of Figure 2.14. Give an expression in the relational algebra to express each of the following queries:
   1. Find the names of all employees who work for “First Bank Corporation”.

∏person\_name(σcompany\_name=”First Bank Corporation”(employee⋈works))

* 1. Find the names and cities of residence of all employees who work for “First Bank Corporation”.

∏person\_name, city(σcompany\_name=”First Bank Corporation”(employee⋈works))

* 1. Find the names, street address, and cities of residence of all employees who work for “First Bank Corporation” and earn more than $10,000.

∏person\_name, street, city(σcompany\_name=”First Bank Corporation” and salary≥10000(employee⋈works))

1. (2.13) Consider the bank database of Figure 2.15. Give an expression in the relational algebra for each of the following queries:
   1. Find all loan numbers with a loan value greater than $10,000.

∏loan\_number(σamount>10000 (loan))

* 1. Find the names of all depositors who have an account with a value greater than $6,000.

∏customer\_name(σbalance≥6000(depositor⋈account))

* 1. Find the names of all depositors who have an account with a value greater than $6,000 at the “Uptown” branch.

∏customer\_name(σbalance≥6000 and branch\_name=”Uptown”(depositor⋈account))