**ARINDA HILLARY**

**DBMS**

**ASSIGNMENT 2**

**2/27/2024**

**1.**[5]**Use the sample of a database shown below to work Problems 1.1 through 1.5.**Each question carries 1 point.  
**This database has 3 relations.**  
  
                        **STUDENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SDT\_ID** | **SF\_NAME** | **SLNAME** | **PHONE** | **MAJOR** |
| 935499 | Jill | Meek | (345)345-5216 | CSC |
| 355869 | Cathy | Liu | (356)467-7488 | PHY |
| 577885 | James | Smith | (456)435-4658 | CHM |
| 127345 | Joy | Brown | (108)365-8976 | CSC |
| 456328 | Lisa | Williams | (239)879-3723 | ENG |
| 732489 | Cathy | Cheng | (213)895-4367 | MUS |

**COURSE**

|  |  |  |  |
| --- | --- | --- | --- |
| **C\_CODE** | **CRS\_NAME** | **CRDT** | **QUALIFIED\_FACULTY** |
| PHY304 | Relativity | 3 | Wolfe, Lathrope, Macy |
| CHM208 | Organic Chemistry | 4 | Walker, Bosch |
| CHM209 | Physical Chemistry | 4 | Walker, Shara |
| MUS338 | Jazz Ensemble | 3 | Gross |
| CSC121 | Programming | 3 | LeMack, Kurup, Naik |
| ENG345 | Creative Writing | 3 | Hanna, Cooley |
| CSC122 | Organization | 3 | LeMack, Kurup |
| CSC124 | Architecture | 3 | Kurup, Naik, Ray |
| PHY207 | Dynamics | 4 | Wolfe, Lathrope, Levy |

**GRADE**

|  |  |  |  |
| --- | --- | --- | --- |
| **CRS\_NAME** | **SL\_NAME** | **GRADE** | **FACULTY** |
| Creative Writing | Cheng | B | Cooley |
| Dynamics | Brown | C | Lathrope |
| Dynamics | Smith | D | Wolfe |
| Programming | Brown | C | Kurup |
| Relativity | Cheng | A | Wolfe |
| Relativity | Meek | B | Wolfe |
| Jazz Ensemble | Williams | A | Gross |
| Organic Chemistry | Williams | C | Walker |

* 1. For each of the tables in the database, identify super keys, candidate keys, primary key and the foreign keys. If a table does not have a foreign key, write NONE.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Super keys** | **Candidate keys** | **Primary keys** | **Foreign keys** |
| STUDENT | 1. STD\_ID 2. PHONE 3. STD\_ID, PHONE 4. STD\_ID, SF\_NAME, SLNAME, MAJOR 5. STD\_ID, SF\_NAME, SLNAME 6. STD\_ID, SF\_NAME 7. STD\_ID, SLNAME 8. STD\_ID, MAJOR 9. STD\_ID, SF\_NAME, MAJOR 10. STD\_ID, SLNAME, MAJOR 11. PHONE, SF\_NAME, SLNAME, MAJOR 12. PHONE, MAJOR 13. PHONE, SF\_NAME, SLNAME 14. PHONE, SF\_NAME 15. PHONE, SLNAME 16. PHONE, SF\_NAME, MAJOR 17. PHONE, SLNAME, MAJOR | 1. STD\_ID 2. PHONE | STD\_ID | NONE |
| COURSE | 1. C\_CODE 2. C\_CODE, CRS\_NAME 3. C\_CODE, CRDT 4. C\_CODE, QUALIFIED\_FACULTY 5. C\_CODE, CRS\_NAME, CRDT 6. C\_CODE, CRS\_NAME, QUALIFIED FACULTY 7. C\_CODE, CRDT, QUALIFIED\_FACULTY 8. C\_CODE, CRS\_NAME, CRDT, QUALIFIED\_FACULTY | 1. C\_CODE | C\_CODE | NONE |
| GRADE | NONE | NONE | NONE | NONE |

* 1. Do the tables satisfy entity integrity? Answer Yes or No, then justify your answer.

Tables STUDENT and COURSE satisfy entity integrity as they each have a primary key and non of them is null. However, table GRADE does not satisfy entity integrity because it has no potential primary key.

* 1. Do the tables satisfy referential integrity? Answer Yes or No, then justify your answer.

NO.

Grade table does not satisfy referential integrity because none of the attributes in this table matches candidate keys of other tables. Student and Course tables don’t have references to any other tables.

* 1. Comment on each table. Can you propose a better organization of data? Justify your answer.

|  |  |  |
| --- | --- | --- |
| **Table** | **Proposed organization** | **Justification** |
| STUDENT | The STUDENT table seems well organized for the most part. However, the MAJOR column could be further normalized. Instead of storing the major as a string, a separate MAJOR table could be created with its own unique identifier (MAJOR\_ID) and a descriptive name for the major. The STUDENT table would then contain a MAJOR\_ID column that references the MAJOR table. | This would reduce redundancy and allow for easier updates or expansions of the major's information. |
| COURSE | The COURSE table's main concern is the QUALIFIED\_FACULTY column, which violates the principle of atomicity in database normalization. This column contains a list of faculty names, which should instead be handled by a separate associative table that could be named COURSE\_FACULTY. | This new table would have a foreign key referencing the C\_CODE from the COURSE table and another foreign key referencing an identifier from a FACULTY table (which we're assuming would need to be created). Each row in the COURSE\_FACULTY table would represent a link between a course and a faculty member qualified to teach it. |
| GRADE | GRADE table should not use CRS\_NAME and SL\_NAME as identifiers. It would be more appropriate to reference the candidate keys from the COURSE and STUDENT tables using C\_CODE and STD\_ID, respectively. Also, introducing a FACULTY table and using a FACULTY\_ID in the GRADE table would be necessary to uniquely identify the faculty member. Moreover, if grades are given for specific semesters or terms, adding a SEMESTER\_ID or similar column would be crucial for a more accurate representation. | The GRADE table would benefit from including more robust referencing mechanisms to ensure that each grade is properly linked to the actual course taken and the student who took it, as well as the faculty member who taught it. |

**1.5)**For each of the tables in the database, create two new rows such that the first one violates entity integrity, and the second one violates referential integrity. If such a row does not exist, write NONE.

STUDENT

                        **STUDENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SDT\_ID** | **SF\_NAME** | **SLNAME** | **PHONE** | **MAJOR** |
|  | James | Balimu | (345)345-5216 | CSC |
| NONE | | | | |

**COURSE**

|  |  |  |  |
| --- | --- | --- | --- |
| **C\_CODE** | **CRS\_NAME** | **CRDT** | **QUALIFIED\_FACULTY** |
|  | Relativity | 4 | Wolfe, Lathrope |
| NONE | | | |

**GRADE**

|  |  |  |  |
| --- | --- | --- | --- |
| **CRS\_NAME** | **SL\_NAME** | **GRADE** | **FACULTY** |
| NONE | | | |
| NONE | | | |

**2. [**2]**Find union, intersection, and difference of the following two relations R and S.  
     Further, find** **∏A,C (σ(A=a2) ˄ (B=b2)(R)).**

A table with blue squares and white text

Description automatically generated

**R U S**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| a1 | b1 | c1 |
| a1 | b1 | c2 |
| a2 | b2 | c1 |
| a2 | b1 | c2 |
| a1 | b2 | c2 |

**R n S**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| a1 | b1 | c2 |
| a2 | b2 | c1 |

**R-S**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| a1 | b1 | c1 |
| a2 | b1 | c2 |

**∏A,C (σ(A=a2) ˄ (B=b2)(R)).**

|  |  |
| --- | --- |
| **A** | **C** |
| a2 | c1 |

A screenshot of a white sheet with black text

Description automatically generated

1. Equi-join

R⋈ R.C=S.C S

​

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **R.C** | **D** | **S.C** | **E** |
| a1 | b1 | c1 | d1 | c1 | e2 |
| a2 | b1 | c2 | d3 | c2 | e2 |
| a2 | b1 | c2 | d4 | c2 | e1 |
| a3 | b2 | c1 | d1 | c1 | e2 |

1. Natural join

R⋈S

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** |
| a1 | b1 | c1 | d1 | e2 |
| a2 | b1 | c2 | d3 | e2 |
| a2 | b1 | c2 | d4 | e1 |
| a3 | b2 | c1 | d1 | e2 |

1. Left outer join

R ⟕ S

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** |
| a1 | b1 | c1 | d1 | e2 |
| a2 | b1 | c2 | d3 | e2 |
| a2 | b1 | c2 | d4 | e1 |
| a3 | b2 | c1 | d1 | e2 |
| a4 | b1 | c3 | null | null |

1. Right outer join

R **⟖ S**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** |
| a1 | b1 | c1 | d1 | e2 |
| a2 | b1 | c2 | d3 | e2 |
| a2 | b1 | c2 | d4 | e1 |
| a3 | b2 | c1 | d1 | e2 |
| null | null |  | d2 | e3 |

**4.**[3]**A relational database contains details about journeys from Chicago to a variety of destinations and contains the following relations:**

**Operator (opCode, opName)**

**Journey (opCode,  destCode, price)**

**Destination (destCode, destName, distance)**

**Each operator is assigned a unique code (opCode) and the relation Operator records the association between this code and the Operator’s name (opName).  
  
Each destination has a unique code (destCode) and the relation Destination records the association between this code and the destination name (destName), and the distance of the destination from Chicago.**

**The relation Journey records the price of an adult fare from Chicago to the given destination by a specified operator; several operators may operate over the same route.**

**Formulate the following queries using relational algebra.**

1. Find the names of all destinations within 20 miles.

*Π* destName​(*σ*distance≤20​(Destination))

1. List the names of all operators with at least one journey priced at under $5.

*Π* opName​(Operator⋈Operator.opCode=Journey.opCode ​*σ*price<5​(Journey))

1. List the names of all operators and prices of journeys to ‘Boston’.

*Π* opName,price​(Operator⋈Operator.opCode=Journey.opCode​(Journey ⋈Journey.destCode=Destination.destCode ​*σ* destName=′*Boston*′​(Destination)))

**5.**[3]**Describe in English the relations that would be produced by the following relational algebra operations.**

1. σHotel.hotelNo = Room.hotelNo(Hotel × Room)

This operation represents a selection after a Cartesian product. The Cartesian product ×× combines every row of the **Hotel** relation with every row of the **Room** relation. The selection *σ* then filters these rows to only include those where the **hotelNo** attribute in the **Hotel** relation matches the **hotelNo** in the **Room** relation. Essentially, this operation gives us all possible combinations of hotels and their rooms but only where the hotel numbers are the same, meaning it gives us the rooms associated with their respective hotels.

1. ΠhotelName(Hotel ⋈ Hotel.hotelNo = Room.hotelNo (σprice > 50 (Room)))

This operation represents a join with a condition followed by a projection. First, the selection *σ* is applied to the **Room** relation to filter only those rooms where the **price** is greater than 50. Next, a natural join ⋈⋈ with a condition is performed between the **Hotel** relation and the result of the selection on **Room**, using **hotelNo** to match hotels to the filtered rooms. Finally, the projection ΠΠ extracts only the **hotelName** attribute. The result is a list of names of hotels that have at least one room with a price over 50.

1. Guest ⋊ (σdateTo ≥ ‘1-Jan-2024’(Booking))

This expression is performing a left outer join between the **Guest** relation and a selection from the **Booking** relation.

This operation will produce a relation that includes:

* All records from the **Guest** relation.
* Matched records from the **Booking** relation where the **dateTo** is on or after '1-Jan-2024'. If a guest does not have a booking that meets this condition, the attributes from the **Booking** relation in the resulting tuple will be filled with null values (or the equivalent in the specific database system being used).