Assignment #2

1.13 - Give examples of systems in which it may make sense to use traditional file processing instead of a database approach.

**Answer:**
1. Small internal utility to locate files
2. Small single user application that does not require security (such as a customized calculator or a personal address and phone book)
3. Real-time navigation system (with heavy computation and very little data)
4. The students may think of others.

2.14 - if you were designing a Web-based system to make airline reservations and to sell airline tickets, which DBMS Architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?

**Answer:**
2.5.4 Three-Tier Client/Server Architecture for Web Application is the best choice. The Client consists of Web User Interface. The Web Server contains the application logic which includes all the rules and regulations related to the reservation process and the issue of tickets; the Database Server contains the DBMS.

2.5.1 Centralized DBMS Architecture would not work since the user interface and database server are on different machines for a web-based system.

2.5.2 Basic Client/Server Architecture and 2.5.3 Two-Tier Client/Server Architecture would work if the Business Logic can reside on server other than the DBMS Server. In general, if the business logic was on the DBMS Server, it will put an excessive burden on the server. If the business logic were to reside on the web client, it will burden the communication network as well as possibly thin client.
3.14 - Consider the following six relations for an order-processing database application in a company:

CUSTOMER (Cust#, Cname, City)
ORDER (Order#, Odate, Cust#, Ord_Amt)
ORDER_ITEM (Order#, Item#, Qty)
ITEM (Item#, Unit_price)
SHIPMENT (Order#, Warehouse#, Ship_date)
WAREHOUSE (Warehouse#, City)

Here, Ord_Amt refers to total dollar amount of an order; Odate is the date the order was placed; Ship_date is the date an order (or part of an order) is shipped from the warehouse. Assume that an order can be shipped from several warehouses. Specify the foreign keys for this schema, stating any assumptions you make. What other constraints can you think of for this database?

**Answer:**

Strictly speaking, a foreign key is a set of attributes, but when that set contains only one attribute, then that attribute itself is often informally called a foreign key. The schema of this question has the following five foreign keys:

1. the attribute Cust# of relation ORDER that references relation CUSTOMER,
2. the attribute Order# of relation ORDER_ITEM that references relation ORDER,
3. the attribute Item# of relation ORDER_ITEM that references relation ITEM,
4. the attribute Order# of relation SHIPMENT that references relation ORDER, and
5. the attribute Warehouse# of relation SHIPMENT that references relation WAREHOUSE.

We now give the queries in relational algebra:

a. \( \text{SHIP}_{-}\text{W2} \leftarrow \text{ORDER} \leftarrow \text{CUST}\)  
   \(\text{RESULT} \leftarrow \pi_{\text{Order#}, \text{Ship}\_\text{date}} \left(\text{SHIP}_{-}\text{W2}\right)\)

b. \( \text{ORDERS}_{-}\text{JL} \leftarrow \text{ORDER} \leftarrow \text{CUST}\)  
   \(\text{RESULT} \leftarrow \pi_{\text{Order#}, \text{Warehouse#}} \left(\text{ORDERS}_{-}\text{JL} \leftarrow \text{SHIPMENT}\right)\)

c. \( \text{BY\_CUSTNUM} \leftarrow \text{CUST} \leftarrow \text{ORDER}\)  
   \(\text{RESULT} \leftarrow \pi_{\text{Cust#}, \text{COUNT}\_\text{Ord\#}, \text{AVERAGE}\_\text{Ord\_Amt}} \left(\text{ORDER}\right)\)
   \(\pi_{\text{Custname}, \text{COUNT}\_\text{Ord\#}, \text{AVERAGE}\_\text{Ord\_Amt}} \left(\text{BY\_CUSTNUM} \leftarrow \text{CUSTOMER}\right)\)

d. \( \text{TIMELY\_SHIPPED} \leftarrow \text{ORDER} \leftarrow \text{SHIPMENT}\)  
   \(\text{RESULT} \leftarrow \pi_{\text{Order#}} \left(\text{ORDER} \leftarrow \text{SHIPMENT}\right) \leftarrow \pi_{\text{Order#}} \left(\text{TIMELY\_SHIPPED}\right)\)

The above query lists all orders for which no “timely” shipment was made, including orders for which no shipment was ever made. It is instructive to compare the above query with the one below that lists those orders for which at least one “late” shipment was made.
Consider the following relations for a database that keeps track of business trips of salespersons in a sales office:

SALESPERSON (SSN, Name, Start_Year, Dept_No)
TRIP (SSN, From_City, To_City, Departure_Date, Return_Date, Trip_ID)
EXPENSE (Trip_ID, Account#, Amount)

Specify the foreign keys for this schema, stating any assumptions you make.

**Answer:**
The schema of this question has the following two foreign keys:
1. the attribute SSN of relation TRIP that references relation SALESPERSON, and
2. the attribute Trip_ID of relation EXPENSE that references relation TRIP.

In addition, the attributes Dept_No of relation SALESPERSON and Account# of relation EXPENSE are probably also foreign keys referencing other relations of the database not mentioned in the question. We now give the queries in relational algebra:

a. COSTLY_TRIPS = SUM_Amount > 2000 (Trip_Id \ F \ SUM_Amount (EXPENSE))
   RESULT _ TRIP * \ \ Trip_Id (COSTLY_TRIPS)

b. RESULT _ \ \ SSN (_ To_City = 'Honolulu' (TRIP))

c. RESULT _ \ F \ SUM_Amount (_ SSN = '234-56-7890' (TRIP) * EXPENSE)

3.18 - Database design often involves decisions about the storage of attributes. For example a Social Security Number can be stored as a one attribute or split into three attributes (one for each of the three hyphen-delimited groups of numbers in a Social Security Number—XXX-XX-XXXX).

However, Social Security Number is usually stored in one attribute. The decision is usually based on how the database will be used. This exercise asks you to think about specific situations where dividing the SSN is useful.

**Answer:**
a. We need the area code (also know as city code in some countries) and perhaps the country code (for dialing international phone numbers).

b. I would recommend storing the numbers in a separate attribute as they have their own independent existence. For example, if an area code region were split into two regions, it would
change the area code associated with certain numbers, and having area code in a separate attribute will make it easier to update the area code attribute by itself.

c. I would recommend splitting first name, middle name, and last name into different attributes as it is likely that the names may be sorted and/or retrieved by the last name, etc.

d. In general, if each attribute has an independent logical existence based on the application, it would make sense to store it in a separate column otherwise there is no clear advantage. For example, SSN need not be split into its component unless we are using the subsequences to make deductions about validity, geography, etc. In the two cases above, it made logical and business sense to split the attributes.

3.20 - Recent changes in privacy laws have disallowed organizations from using SSN to identify individuals unless certain restrictions are satisfied. As a result, most US universities cannot use SSNs as primary keys (except for financial data). In practice, StudentID, a unique ID, a unique identifier, assigned to every student, is likely to be used as the primary key rather than SSN since StudentID is usable across all aspects of the system.

a. Some database designers are reluctant to use generated keys (also known as surrogate keys) for primary keys (such as StudentID) because they are artificial. Can you propose any natural choices of keys that can be used to store the student record in a UNIVERSITY database?

b. Suppose that you were able to guarantee uniqueness of a natural key that included last name. Are you guaranteed that the last name will not change during the lifetime of the database? If the last name last name can change, what solutions can you propose for creating a primary key that still includes last name but remains unique?

b. What are the advantages and disadvantages of using generated (surrogate) keys?

Answer:

a. By keeping the name attributes separated, we allow the possibility of looking these pieces of their name. In a practical use, it is not likely that the user will know the correct primary key for a given student and so we must consider how a user will locate the correct row without this information. If we were to collapse the name into a single attribute, then we have complicated any sort of “lookup by name” query; such a query would then require partial string matching and any results might not disambiguate between FirstName and LastName. Therefore, a practical system should allow name searches by FirstName and LastName; we must leave MiddleInitial separated still to avoid ambiguities from combining these pieces together.

b. A single attribute Phone# would no longer suffice if a student were able to have multiple phone numbers. We could possibly have multiple rows for a single student to allow this to happen, but then we have violated key principles of database design (e.g. having redundant data). A better solution would be to include the additional attributes HomePhone, CellPhone, and OfficePhone and allow the possibility of these attributes to have no value. Again, this is not most desirable because most students will not have all three of these attributes, and we will have many valueless key/attribute pairs. An excellent solution would be add an additional relation Phone# (SSN, Type, Number) while removing PhoneNumber from the Student relationship. This new relationship would allow the one-to-many relationship from students to phone numbers without creating redundant data or wasting space on sparse, valueless attributes.

6.16 - Specify the following queries on the COMPANY relational database schema shown in Figure 3.5, using the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state of Figure 3.6.
(a) Retrieve the names of employees in department 5 who work more than 10 hours per week on the 'ProductX' project.

(b) List the names of employees who have a dependent with the same first name as themselves.

(c) Find the names of employees that are directly supervised by 'Franklin Wong'.

(d) For each project, list the project name and the total hours per week (by all employees) spent on that project.

(e) Retrieve the names of employees who work on every project.

(f) Retrieve the names of employees who do not work on any project.

(g) For each department, retrieve the department name, and the average salary of employees working in that department.

(h) Retrieve the average salary of all female employees.

(i) Find the names and addresses of employees who work on at least one project located in Houston but whose department has no location in Houston.

(j) List the last names of department managers who have no dependents.

**Answers:**
In the relational algebra, as in other languages, it is possible to specify the same query in multiple ways. We give one possible solution for each query. We use the symbol \( s \) for SELECT, \( P \) for PROJECT, \( J \) for EQUIJOIN, * for NATURAL JOIN, and \( f \) for FUNCTION.

(a) \( \text{EMP\_W\_X} \leftarrow ( s \ PNAME='ProductX' (PROJECT)) \ J (\text{PNUMBER},(\text{PNO}) (\text{WORKS\_ON}) \text{EMP\_WORK\_10} \leftarrow (\text{EMPLOYEE}) \ J (\text{SSN},(\text{ESSN}) \ ( s \ \text{HOURS}>10 (\text{EMP\_W\_X})) \text{RESULT} \leftarrow ( s \ DNO=5 (\text{EMP\_WORK\_10})) \)

Result:
<table>
<thead>
<tr>
<th>LNAME</th>
<th>FNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>John</td>
</tr>
<tr>
<td>English</td>
<td>Joyce</td>
</tr>
</tbody>
</table>

(b) \( E \leftarrow (\text{EMPLOYEE}) \ J (\text{SSN},FNAME), (\text{ESSN},DEPENDENT\_NAME) \ (DEPENDENT) \ R \leftarrow \ P \ LNAME,FNAME \ (E) \)

Result (empty):
| LNAME  | FNAME  |

(c) \( \text{WONG\_SSN} \leftarrow \ P \ \text{SSN} \ ( s \ \text{FNAME}='Franklin' \ \text{AND} \ \text{LNAME}='Wong' (\text{EMPLOYEE})) \text{WONG\_EMPS} \leftarrow (\text{EMPLOYEE}) \ J (\text{SUPERSSN},(\text{SSN}) \ (\text{WONG\_SSN}) \text{RESULT} \leftarrow ( s \ \text{LNAME},\text{FNAME} \ (\text{WONG\_EMPS}) \)

Result (empty):
| LNAME  | FNAME  |
Result:

LNAME FNAME
Smith John
Narayan Ramesh
English Joyce

(d) $\text{PROJ\_HOURS(PNO,TOT\_HRS) \leftarrow PNO \sum \text{HOURS(WORKS\_ON)}}$
RESULT $\leftarrow P \ PNAME,TOT\_HRS \ (\text{PROJ\_HOURS(J(PNO),(PNUMBER)(PROJECT)}))$

Result:

PNAME TOT\_HRS
ProductX 52.5
ProductY 37.5
ProductZ 50.0
Computerization 55.0
Reorganization 25.0
Newbenefits 55.0

(e) $\text{PROJ\_EMPS(PNO,SSN) \leftarrow P PNO,ESSN(WORKS\_ON)}$
$\text{ALL\_PROJS(PNO) \leftarrow P PNUMBER(PROJECT)}$
$\text{EMPS\_ALL\_PROJS \leftarrow PROJ\_EMPS \div \text{ALLPROJS} (* \text{DIVISION operation} *)}$
RESULT $\leftarrow P \ LNAME,FNAME \ (\text{EMPLOYEE \* EMP\_ALL\_PROJS)}$

Result (empty):

LNAME FNAME

(f) $\text{ALL\_EMPS \leftarrow P SSN(EMPLOYEE)}$
$\text{WORKING\_EMPS(SSN) \leftarrow P ESSN(WORKS\_ON)}$
$\text{NON\_WORKING\_EMPS \leftarrow ALL\_EMPS \div WORKING\_EMPS (* \text{DIFFERENCE} *)}$
RESULT $\leftarrow P \ LNAME,FNAME \ (\text{EMPLOYEE \* NON\_WORKING\_EMPS)}$
Result (empty):

LNAME FNAME

(g) $\text{DEPT\_AVG\_SALS(DNUMBER,AVG\_SAL) \leftarrow DNO \ f \ AVG\_SALARY(EMPLOYEE)}$
RESULT $\leftarrow P \ DNUMBER,AVG\_SAL \ (\text{DEPT\_AVG\_SALS \* DEPARTMENT})$

Result:

DNUMBER AVG\_SAL
Research 33250
Administration 31000
Headquarters 55000
(h) \[\text{RESULT}(\text{AVG}_F \_ \text{SAL}) \leftarrow f \text{ AVG SALARY (s SEX='F' (EMPLOYEE))}\]

Result:
AVG _F_SAL
31000

(i) \[\text{E}_P \_ \text{HOU}(\text{SSN}) \leftarrow \]

\[\text{P ESSN (WORKS_ON J(PNO),(PNUMBER) (s PLOCATION='Houston' (PROJECT)))} \]

\[\text{D}_\text{NO_HOU} \leftarrow \]

\[\text{P DNUMBER (DEPARTMENT) - P DNUMBER (s DLOCATION='Houston' (DEPARTMENT))} \]

\[\text{E}_\text{D}_\text{NO_HOU} \leftarrow \text{P SSN (EMPLOYEE J(PNO),(DNUMBER) (D_NO_HOU))} \]

\[\text{RESULT}_\text{EMPS} \leftarrow \text{E}_P \_ \text{HOU} - \text{E}_D \_ \text{NO_HOU} (* \text{this is set DIFFERENCE } *) \]

\[\text{RESULT} \leftarrow \text{P LNAME,FNAME,ADDRESS (EMPLOYEE \_ \text{RESULT}_\text{EMPS})} \]

Result:
LNAME FNAME ADDRESS
Wallace Jennifer 291 Berry, Bellaire, TX

(j) \[\text{DEPT}_\text{MANAGERS}(\text{SSN}) \leftarrow \text{P MGRSSN (DEPARTMENT)} \]

\[\text{EMPS}_\text{WITH}_\text{DEPENDENTS}(\text{SSN}) \leftarrow \text{P ESSN (DEPENDENT)} \]

\[\text{RESULT}_\text{EMPS} \leftarrow \text{DEPT}_\text{MANAGERS} - \text{EMPS}_\text{WITH}_\text{DEPENDENTS} \]

\[\text{RESULT} \leftarrow \text{P LNAME,FNAME (EMPLOYEE \_ \text{RESULT}_\text{EMPS})} \]

Result:
LNAME FNAME
Borg James

6.24 - Specify queries (a), (b), (c), (e), (f), (i), and (j) of Exercise 6.16 in both tuple and domain relational calculus.

Answer:
(a) Retrieve the names of employees in department 5 who work more than 10 hours per week on the 'ProductX' project.

Tuple relational Calculus:

\{ e.LNAME, e.FNAME | \text{EMPLOYEE(e) AND e.DNO=5 AND (EXISTS p) (EXISTS w) (WORKS_ON(w) AND PROJECT(p) AND e.SSN=w.ESSN AND w.PNO=p.PNUMBER AND p.PNAME='ProductX' AND w.HOURS>10 )} \}


Domain relational Calculus:

\{ qs \mid EMPLOYEE(qrstuvwxyz) \text{ AND } z=5 \text{ AND } (\exists a) (\exists b) (\exists e) (\exists f) (\exists g) \ (\text{WORKS_ON}(efg) \text{ AND } PROJECT(abcd) \text{ AND } t=e \text{ AND } f=b \text{ AND } a=\text{\textquoteleft}ProductX\textquoteright\text{ AND } g>10 ) \}

(b) List the names of employees who have a dependent with the same first name as themselves.

Tuple relational Calculus:

\{ e.LNAME, e.FNAME \mid EMPLOYEE(e) \text{ AND } (\exists d) \ (\text{DEPENDENT}(d) \text{ AND } e.SSN=d.ESSN \text{ AND } e.FNAME=d.DEPENDENT\_NAME) \}

Domain relational Calculus:

\{ qs \mid (\exists t) (\exists a) (\exists b) \ (\text{EMPLOYEE}(qrstuvwxyz) \text{ AND } DEPENDENT(abcde) \text{ AND } a=t \text{ AND } b=q ) \}

(c) Find the names of employees that are directly supervised by 'Franklin Wong'.

Tuple relational Calculus:

\{ e.LNAME, e.FNAME \mid EMPLOYEE(e) \text{ AND } (\exists s) \ (\text{EMPLOYEE}(s) \text{ AND } s.FNAME='Franklin' \text{ AND } s.LNAME='Wong' \text{ AND } e.SUPERSSN=s.SSN) \}

Domain relational Calculus:

\{ qs \mid (\exists y) (\exists a) (\exists c) (\exists d) \ (\text{EMPLOYEE}(qrstuvwxyz) \text{ AND } EMPLOYEE(abcdefg) \text{ AND } a='Franklin' \text{ AND } c='Wong' \text{ AND } y=d) \}

(e) Retrieve the names of employees who work on every project.

Tuple relational Calculus:

\{ e.LNAME, e.FNAME \mid EMPLOYEE(e) \text{ AND } (\forall p) \ (\text{NOT}(PROJECT(p)) \text{ OR } \text{WORKS_ON}(w) \text{ AND } p.PNUMBER=w.PNO \text{ AND } w.ESSN=e.SSN) \}

Domain relational Calculus:

\{ qs \mid (\exists t) \ (\text{EMPLOYEE}(qrstuvwxyz) \text{ AND } (\forall b) \ (\text{NOT}(PROJECT(abcd)) \text{ OR } \text{\textquoteleft}WORKS\_ON(efg) \text{ AND } e=t \text{ AND } f=b)) \}
(f) Retrieve the names of employees who do not work on any project.

Tuple relational Calculus:

\{ e.LNAME, e.FNAME | EMPLOYEE(e) AND NOT(EXISTS w) ( WORKS_ON(w) AND w.ESSN=e.SSN ) \}

Domain relational Calculus:

\{ qs | (EXISTS t) ( EMPLOYEE(qrstuvwxyz) AND NOT(EXISTS a) ( WORKS_ON(abc) AND a=t ) ) \}

(i) Find the names and addresses of employees who work on at least one project located in Houston but whose department has no location in Houston.

Tuple relational Calculus:

\{ e.LNAME, e.FNAME, e.ADDRESS | EMPLOYEE(e) AND (EXISTS p) (EXISTS w) ( WORKS_ON(w) AND PROJECT(p) AND e.SSN=w.ESSN AND w.PNO=p.PNUMBER AND p.PLOCATION='Houston' AND NOT(EXISTS l) ( DEPT_LOCATIONS(l) AND e.DNO=l.DNUMBER AND l.DLOCATION='Houston' ) ) \}

Domain relational Calculus:

\{ qsv | (EXISTS t) (EXISTS z) ( EMPLOYEE(qrstuvwxyz) AND (EXISTS b) (EXISTS c) (EXISTS e) (EXISTS f) ( WORKS_ON(efg) AND PROJECT(abcd) AND t=e AND f=b AND c='Houston' AND NOT(EXISTS h) NOT(EXISTS i) ( DEPT_LOCATIONS(hi) AND z=h AND i='Houston' ) ) ) \}

(j) List the last names of department managers who have no dependents.

Tuple relational Calculus:

\{ e.LNAME | EMPLOYEE(e) AND (EXISTS d) ( DEPARTMENT(d) AND e.SSN=d.MGRSSN AND NOT(EXISTS x) (DEPENDENT(x) AND e.SSN=x.ESSN) ) \}

Domain relational Calculus:

\{ s | (EXISTS t) ( EMPLOYEE(qrstuvwxyz) AND (EXISTS c) ( DEPARTMENT(abcd) AND t=c AND NOT(EXISTS e) (DEPENDENT(efghi) AND e=t) ) \}
6.32 - A nested query is query within a query. More specifically, a nested query is a parenthesized query that can be used as a value in a number of places, such as instead of a relation or a selection condition. Specify the following queries on the database specified in Figure 3.5 using the concept of nested queries and the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state of Figure 3.6.

a. List the names of all employees who work in the department that has the employee with the highest salary among all employees.
b. List the names of all employees whose supervisor’s supervisor has ‘888665555’ for SSN.
c. List the names of employees who make at least $10,000 more than the employee who is paid the least in the company.

Answers:
a) \[
\text{RESULT} \leftarrow \text{P LNAME ( S DNO = ( P DNO ( S SALARY = \text{MAX(SALARY)} \text{ EMPLOYEE ) )} )}
\]

b) \[
\text{RESULT} \leftarrow \text{P LNAME ( S SUPERSSN = (P SSN ( S SUPERSSN = ‘88866555’ \text{ EMPLOYEE ) )} )}
\]

c) \[
\text{RESULT} \leftarrow \text{P LNAME ( S SALARY >= 10000 + P SALARY ( S SALARY = \text{MIN(SALARY)} \text{ EMPLOYEE ) })}
\]

7.17 - Composite and multi-valued attributes can be nested to any number of levels. Suppose we want to design an attribute for a STUDENT entity type to keep track of previous college education. Such an attribute will have one entry for each college previously attended, and this entry is composed of: college name, start and end dates, degree entries (degrees awarded at that college, if any), and transcript entries (courses completed at that college, if any). Each degree entry is formed of degree name and the month and year it was awarded, and each transcript entry is formed of a course name, semester, year, and grade. Design an attribute to hold this information. Use the conventions of Figure 7.5.

Answer:
\[
\text{PreviousEducation ( CollegeName, StartDate, EndDate,}
\text{ Degree (DegreeName, Month, Year) ),}
\text{ Transcript (CourseName, Semester, Year, Grade) ) ) )}
\]

7.19 - Consider the ER diagram of Figure 7.20, which shows a simplified schema for an airline reservations system. Extract from the ER diagram the requirements and constraints that resulted in this schema. Try to be as precise as possible in your requirements and constraints specification.

Answer:
(1) The database represents each AIRPORT, keeping its unique AirportCode, the AIRPORT Name, and the City and State in which the AIRPORT is located.

(2) Each airline FLIGHT has a unique number, the Airline for the FLIGHT, and the Weekdays on which the FLIGHT is scheduled (for example, every day of the week except Sunday can be coded as X7).

(3) A FLIGHT is composed of one or more FLIGHT LEGs (for example, flight number CO1223 from New York to Los Angeles may have two FLIGHT LEGs: leg 1 from New York to Houston and leg 2 from Houston to Los Angeles). Each FLIGHT LEG has a DEPARTURE AIRPORT and Scheduled Departure Time, and an ARRIVAL AIRPORT and Scheduled Arrival Time.
(4) A LEG INSTANCE is an instance of a FLIGHT LEG on a specific Date (for example, CO1223 leg 1 on July 30, 1989). The actual Departure and Arrival AIRPORTs and Times are recorded for each flight leg after the flight leg has been concluded. The Number of available seats and the AIRPLANE used in the LEG INSTANCE are also kept.

(5) The customer RESERVATIONs on each LEG INSTANCE include the Customer Name, Phone, and Seat Number(s) for each reservation.

(6) Information on AIRPLANEs and AIRPLANE TYPEs are also kept. For each AIRPLANE TYPE (for example, DC-10), the TypeName, manufacturing Company, and Maximum Number of Seats are kept. The AIRPORTs in which planes of this type CAN LAND are kept in the database. For each AIRPLANE, the Airplaneld, Total number of seats, and TYPE are kept.

7.27 - Cardinality ratios often dictate the detailed design of a database. The cardinality ratio depends on the real-world meaning of the entity types involved and is defined by the specific application. For the binary relationships below, suggest cardinality ratios based on common-sense meaning of the entity types. Clearly state any assumptions you make.

<table>
<thead>
<tr>
<th>Entity 1</th>
<th>Cardinality Ratio</th>
<th>Entity 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Student</td>
<td></td>
<td>SocialSecurityCard</td>
</tr>
<tr>
<td>2. Student</td>
<td></td>
<td>Teacher</td>
</tr>
<tr>
<td>3. ClassRoom</td>
<td></td>
<td>Wall</td>
</tr>
<tr>
<td>4. Country</td>
<td></td>
<td>CurrentPresident</td>
</tr>
<tr>
<td>5. Course</td>
<td></td>
<td>TextBook</td>
</tr>
<tr>
<td>6. Item (that can be found in an order)</td>
<td></td>
<td>Order</td>
</tr>
<tr>
<td>7. Student</td>
<td></td>
<td>Class</td>
</tr>
<tr>
<td>8. Class</td>
<td></td>
<td>Instructor</td>
</tr>
<tr>
<td>9. Instructor</td>
<td></td>
<td>Office</td>
</tr>
<tr>
<td>10. E-bay Auction item</td>
<td></td>
<td>E-bay bid</td>
</tr>
</tbody>
</table>
Answer:

<table>
<thead>
<tr>
<th></th>
<th>Entity 1</th>
<th>Cardinality Ratio</th>
<th>Entity 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Student</td>
<td>1-many</td>
<td>SocialSecurityCard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A student may have more than one social security card (legally with the same unique social security number), and every social security number belongs to a unique student.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Student</td>
<td>Many-many</td>
<td>Teacher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generally students are taught by many teachers and a teacher teaches many students.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ClassRoom</td>
<td>Many-many</td>
<td>Wall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Don’t forget that the wall is usually shared by adjacent rooms.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Country</td>
<td>1-1</td>
<td>CurrentPresident</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assuming a normal country under normal circumstances having one president at a time.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Course</td>
<td>Many-many</td>
<td>TextBook</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A course may have many textbooks and a text book may be prescribed for different courses.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Item (that can be found in an order)</td>
<td>Many-many</td>
<td>Order</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assuming the same item can appear in different orders.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Student</td>
<td>Many-many</td>
<td>Class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>One student may take several classes. Every class usually has several students.</td>
<td></td>
</tr>
</tbody>
</table>
### 8. Class

- **Many-to-1**
- Assuming that every class has a unique instructor. In case instructors were allowed to team teach, this will be many-many.

#### Instructor

### 9. Instructor

- **1-1**
- Assuming every instructor has only one office and it is not shared. In case of offices shared by 2 instructors, the relationship will be 2-1. Conversely, if an instructor has a joint appointment (in two departments) and offices in both departments, then the relationship will be 1-2. In a very general case, it may be many-many.

#### Office

### 10. E-bay Auction item

- **1-many**
- 1 item has many bids and a bid is unique to an item (assuming a regular auction format).

#### E-bay bid

---

**7.30** - Illustrate the UML Diagram for exercise 7.16. Your UML design should observe the following requirements:

- a. The student should have the ability to compute his/her GPA and add or drop majors and minors.
- b. Each department should be able to add or delete courses and hire or terminate faculty.
- c. Each instructor should be able to assign or change a grade to a student for a course.

*Note: Some of these functions may be spread over multiple classes.*
This one isn't finished yet. The relationships need to be added and some of the functions should be moved to span classes.
8.19 - Identify all the important concepts represented in the library database case study described here. In particular, identify the abstraction of classification (entity types and relationship types), aggregation, identification, and specialization/generalization. Specify (min, max) cardinality constraints whenever possible. List details that will affect the eventual design but which have no bearing on the conceptual design. List the semantic separately. Draw an EER diagram of the library database.

**Answer:**

![EER Diagram](image)

8.21 - Figure 8.12 shows an example of an EER diagram for a small private airport database that is used to keep track of airplanes, their owners, airport employees, and pilots. From the requirements for this database, the following information was collected: Each airplane has a registration number [Reg#], is of a particular plane type [of_type], and is stored in a particular hangar [stored_in]. Each plane_type has a model number [Model], a capacity [Capacity], and a weight [Weight]. Each hangar has a number [Number], a capacity [Capacity], and a location [Location]. The database also keeps track of the owners of each plane [owns] and the employees who have maintained the plane [maintain]. Each relationship instance in owns relates an airplane to an owner and includes the purchase date [Pdate]. Each relationship instance in maintain relates to an employee to a service record [service]. Each plane undergoes service many times; hence, it is related by [plane_service] o
a number of service records. A service record includes as attributes the date of maintenance [Date], the number of hours spent on the work [Hours], and the type of work done [Workcode]. We use a weak entity type [service] to represent airplane service, because the airplane registration number is used to identify a service record. An owner is either a person or a corporation. Hence, we use a union type (category) [owner] that is a subset of the union of corporation [Corporation] and person [Person] entity types. Both pilots [Pilot] and employees [Employee] are subclasses of person. Each pilot has specific attributes license number [Lic_Num] and restrictions [Restr]; each employee has specific attributes salary [Salary] and shift [Shift]. All person entities in the database have data kept on their social security number [Ssn], name [Name], address [Address], and telephone number [Phone]. For corporation entities, the data kept includes name [Name], address [Address], and telephone number [Phone]. The database also keeps track of the types of planes each pilot is authorized to fly [Flies] and the types of planes each employee can do maintenance work on [Works_on]. Show how the small airport EER schema of Figure 8.12 may be represented in UML notation. (Note: We have not discussed how to represent categories (union types) in UML, so you do not have to map the categories in this and the following question.)

Answer:
8.27 - Consider the following EER diagram that describes computer systems at a company. Provide your own attributes and key for each entity type. Supply max cardinality constraints justifying your choice. Write a complete narrative description of what this EER diagram represents.
**Answer:**

Note: We represented BOOK_AUTHORS as a multi-valued attribute of BOOK in the above ER diagram. Alternatively, it can be represented as a weak entity type.

**9.5** Map the BANK ER schema of Exercise 7.23 (shown in Figure 7.21) into a relational schema. Specify all primary keys and foreign keys. Repeat for the AIRLINE schema (Figure 7.20) of Exercise 7.19 and for the other schemas for Exercises 7.16 through 7.24.

Partial Answer:

BANK
CODE NAME ADDR
ACCOUNT
ACCTNO BALANCE TYPE BCODE BNO
CUSTOMER
SSN NAME PHONE ADDR
LOAN
LOANNO AMOUNT TYPE BCODE BNO
LIBRARY_BRANCH
BCODE BRANCHNO ADDR
A_C
SSN ACCTNO
L_C
SSN LOANNO
f.k.
f.k. f.k.
f.k.
f.k. f.k.
f.k. f.k.

10.23 - A possible DDL corresponding to Figure 3.1 is shown below:
CREATE TABLE STUDENT(
    NAME VARCHAR(30) NOT NULL,
    SSN CHAR(9) PRIMARY KEY,
    HOMEPHONE VARCHAR(14),
    ADDRESS VARCHAR(40),
    OFFICEPHONE VARCHAR(14),
    AGE INT,
    GPA DECIMAL(4,3)
);

Discuss the following detailed design decisions:

a. The choice of requiring NAME to be NON NULL.
b. Selection of SSN as the PRIMARY KEY.
c. Choice of field sizes and precision.
d. Any modification of the fields defined in this database.
e. Any constraints on individual fields.

Answer:

a) This means that no Students can exist without a Name.

b) This has a possible problem because there are some restrictions on using SSN as an identifier for people, hence why most schools assign a student ID. It also does not allow (easily) for a change to a student’s SSN if this key is used elsewhere in the DB.

c) Giving Name a size of 30 does not allow for long names (think foreign students), especially since it contains a student’s entire name (first, middle, last). Homephone and Officephone, with size 14, might not allow for overseas or foreign formatted phone numbers. Age is expected to be (roughly) between 1 and 100, so using an INT is much more space than needed. A TinyInt would be better, since it allows 0 to 255. The GPA field does not allow for GPA systems that go above 9 (such as the 10 point scale and the 12 point scale), though these are rare.

d) I would suggest breaking the Name field into three fields: FirstName, MiddleName, and LastName. It is more complicated and less specific to search for a student by name if the name data is in one field. See question 3.19.

The same reasoning can be applied to the Homephone and Officephone fields, breaking them into multiple fields such as AreaCode, PhoneNumber, Extension, etc. Another possible modification is to increase the size of the Homephone, Officephone, and SSN fields so that formatting characters can be entered, such as SSN becoming a VARCHAR(11) to allow for the two dashes in XXX-YY-ZZZZ.
The table design allows for two phone numbers, Homephone and Officephone, either of which may be NULL. This means that some rows will have empty fields for those attributes. A solution is then to add a Phone# (SSN, Type, Number) table and remove the two phone number fields from the Students table. This allows for any student to have zero or more phone numbers of various types, which we can specify (Home, Office, Cell, etc.).

Consider not storing the AGE data and instead storing DateOfBirth. Typically only the year component of Age is stored which is less accurate than if DateOfBirth were stored and the age computed from it. That is, we can generate Age from DateOfBirth, but we cannot generate DateOfBirth from Age.

e) A suggested constraint is on the GPA field, to restrict its values from 0.000 to 4.000. This would, however, prevent the (rarely seen) 10 or 12 point grade scale. Another possible constraint is to limit Age to be from 0 to 100 (or even a lower maximum value).