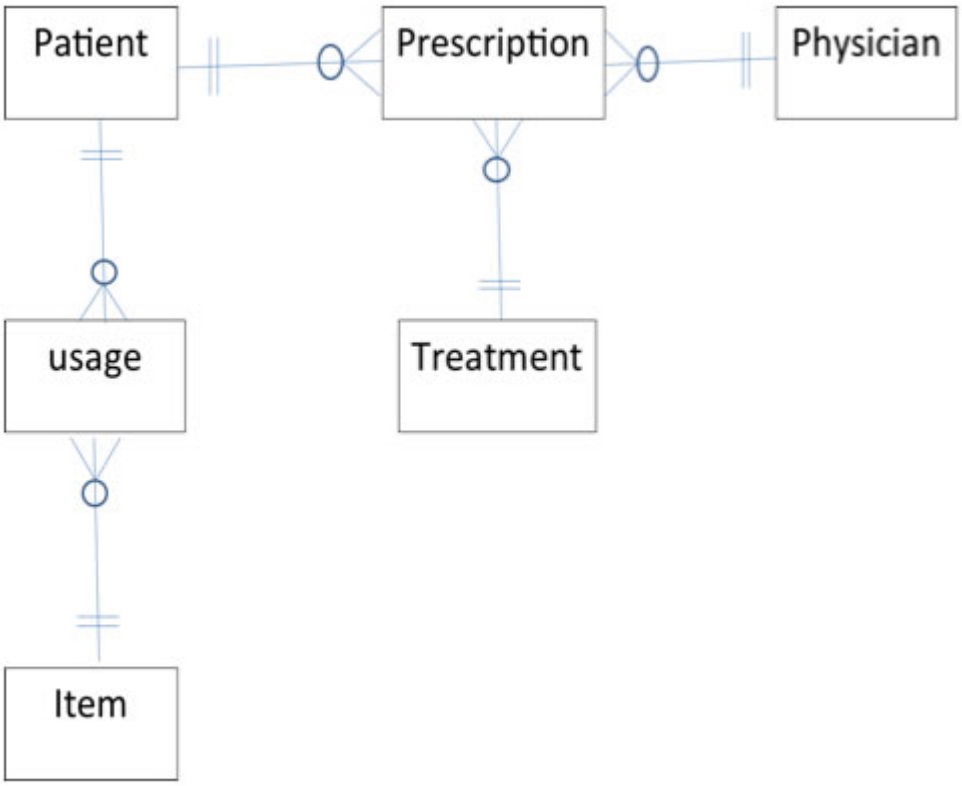
Assignment 2

Course: COMP378
Name: Jason Bishop
Student#: 3042012

Question 1 (12 marks)
Consider the following EER diagram for the Royal Victoria Hospital (RVH) database.

Data

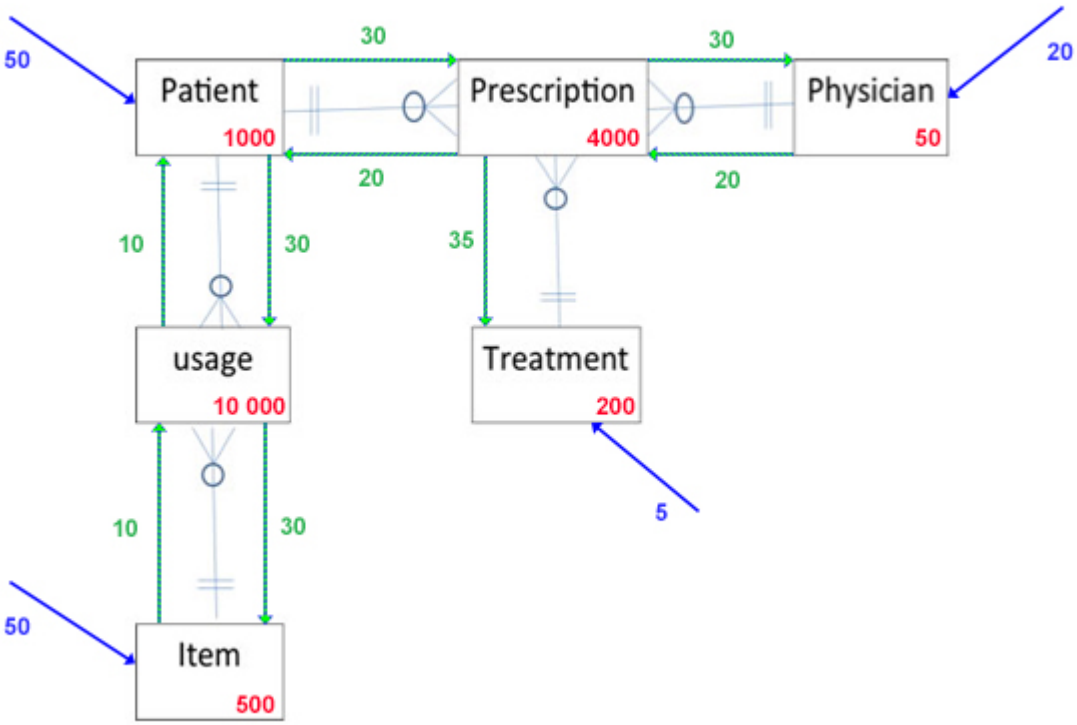


volume and access for this diagram are as follows:

- There are 1 000 patients and 500 items yielding a total of 10 000 usage records in the database.
- There are 50 physicians and a total of 4 000 prescriptions in the database.
- There are 200 treatments in the database.
- There are 50 accesses per day for patient records; of these, 30 request access to both prescription and usage records.
- There are 20 accesses per day for physician records; of these, 20 request access to prescriptions.
- There are 50 accesses per day to item records; of these, 10 request access to usage records.
- There are 5 direct accesses per day for treatment records.
- Of the total accesses to prescription records, 20 request access to patients, 30 request access to physicians, and 35 request access to treatment.

- Of the total accesses to usage records, 10 request access to patients and 30 request access to items.

Draw a composite usage map for the RVH database.



Question 2 (12 marks)
Answer the following questions (250 words max/question).

a. What are the typical integrity controls performed in both data integrity and referential integrity?

There are several types of integrity controls that can be used in both data and referential integrity. Data integrity controls often limit the type and length of data allowed in a table field. This includes using a default value when no data is entered into a particular field. This can speed data entry and reduce errors. Range control puts a limit on the values that field can have. This can also reduce errors. Null value control is used to prohibit or allow the use of null values within a field. It helps enforce business rules and referential integrity.

Referential integrity assumes that a value in a particular table field must exist in another table also. It is a specialized type of data integrity with another set of controls to consider, because there are two different tables to consider when adding, modifying or deleting data. The key considerations come when attempting to delete data because there may be loss of some information. Three typical ways to handle this are using a cascading delete to remove all associated instances of a referential value, not allowing deletion until all references to it have already been deleted, or allowing the

use of a null value to simultaneously keep some of the referential information, while allowing the deletion of the foreign data.

b. Using an example for each situation, illustrate the three common situations that suggest relations should be denormalized.

A situation that suggests denormalization is when two entities share a one-to-one relationship. An example might be an organ donor card with a driver's license. Each person is associated with one card, with one card associated to each driver. These two relations can be combined. This type of denormalization is only recommended when the matching entity exists most of the time and access frequency between the entities is high.

Another situation for denormalization is the existence of an associative entity with non-key attributes. To access full relationship data, two join operations are needed. Denormalizing reduces the number of joins. An example might be at a university with a student and course entity. The associative entity between them would capture the information of when the student enrolled in the course, with the dates and grades. This associative entity could be combined with the student record. However, this type of normalization should only be done when the joining occurs often, and may result in significant duplication of data.

The third candidate for denormalization is when an entity on the one side of a one-to-many relationship participates in no other relationships. An example of this might be if a company required each employee to complete a mandatory sexual harassment course. The course is taken by many employees, but each employee takes the one course. This type of denormalization is recommended when there are few instances of the many side entity for each entity on the one side.

c. What are the advantages and disadvantages of horizontal and vertical partitioning?

The advantages of either type of participation are essentially the same. Partitioning increases the efficiency of the system by storing data together that are frequently accessed together. This also isolates data maintenance. Local optimization is achieved because each partition can be individually tuned for performance. Security is also increased because data can be segregated from users who do not need access to that data. Additionally, recovery of data is easier because the smaller size of partitions means that there is less time and effort to back up and recover sections of data. Also, if the partition is damaged in some way, it is more isolated, which allows the other partitions to still be accessed. This increases uptime of the system. Finally, load balancing within the system can be achieved with greater efficiency. Different partitions can be stored in different areas to reduce access to a particular storage medium and can allow for parallel processing.

Question 3 (12 marks)

Answer the following questions (250 words max/question).

a. What factors should be considered when choosing a file organization?

A file organization involves the physical arrangement of data on a storage medium. There are seven factors to consider when choosing a particular file organization:

1. Fast data retrieval – for example, an indexed organization can allow faster searches than sequential organization
2. High data throughput – processing data input and maintenance transactions can be faster, as when using a join index for instance
3. Efficient use of space – a sequential organization uses a contiguous block with no gaps which is efficient, but an indexed organization requires extra space for the index.
4. Loss and failure protection – if an indexed organization is used, data can be placed across different storage media. If one fails, only part of the data is lost. This can't be done as easily with a sequential organization.
5. Minimization of reorganization – when inserting data into a sequential organization, the all the data after the insertion needs to be copied and moved, this is not the case with other organizations
6. Accommodation for growth – as the amount of data grows, some types of organizations can become inefficient. For example if a hashed index grows to large for the hash key, resulting in a high number of collisions
7. Security – users can be restricted to accessing a certain range of data more easily with sequential and indexed organizations versus a hashed organization because a hashed piece of data is stored based on an algorithm, not the actual data value

b. What is the purpose of clustering data in a file?

Clustering of data means putting rows from different tables in adjacent sections of storage. This is usually used when the data is frequently joined. The purpose of clustering the data together is to reduce the time it takes to perform a join because the related records are located physically close to one another. However, if the data changes frequently, the clustering becomes less efficient and can result in wasted space. Therefore, clustering is typically used on static data.

c. Compare hashed file organization versus indexed file organization. List two advantages of indexed over hashed, and two advantages of hashed over indexed.

A hashed file organization uses a hashing algorithm to compute the storage location of a particular piece of data. This essentially means that data can not be stored sequentially. In an indexed file organization, the data can stored sequentially or non-sequentially. An index is a table used to compute where to store and retrieve information. An index can also point to more than one piece of information by using a secondary key index. This is not possible with hashing, which can only point to one particular piece of data.

One of the advantages of using a hashed organization is that random retrieval of primary key data is faster than using an index. This is because the hashing algorithm is very quick to compute and points to the data quickly, whereas an index table would

have to be searched to find the data. Additionally, adding and deleting records is very easy with a hashed index because space is dynamically allocated.

However, indexing has its own advantages. For one, retrieving data with multiple keys is very fast with multiple indexes but is not possible with a pure hashing organization. Furthermore, data storage is more efficient and allows for growth, because the data can be stored sequentially. With a hashed organization, extra space may be needed to allow for growth and there may be gaps in the storage medium because of the results of the hashing algorithm.

Question 4 (18 marks)

Consider the following database:

Employee(emp-no, name, department, salary), ProjAssigned(emp-no, proj-no, worked-hours)

a. Write one SELECT SQL query to list the numbers and names of all employees with a salary greater than 66 000 who are assigned to projects, the projects they are assigned to, and the corresponding hours worked. Your list should be sorted by employee name.

```
SELECT E.emp-no, E.name, P.proj-no, P.worked-hours
  FROM Employee_t E INNER JOIN ProjAssigned_t PA ON
                E.emp_no = PA.emp_no
        WHERE E.salary > 66000
          ORDER BY E.name;
```

b. Define indexes on selected attributes to speed up your query, and justify your selections.

I would create indexes on both emp-no in Employee and ProjAssigned because it would be the primary key in the Employee table and it would frequently be joined with the value in the ProjAssigned table. This would help speed up the results of any queries. I would also consider creating an index on the emp_no and proj_no columns of the ProjAssigned table because these two values would form a composite primary key for the table.

c. Write SQL queries to create the indexes you defined above.

```
CREATE UNIQUE INDEX Empindex_PK ON Employee_t(emp_no);

CREATE INDEX emp_no_IDX ON ProjAssigned_t(emp_no);

CREATE UNIQUE INDEX PAindex_PK ON ProjAssigned_t(emp_no, proj_no);
```

Question 5 (40 marks)
Consider the following three relations:

TRAVEL_AGENT (name, age, salary)
CUSTOMER (name, departure_city, destination, journey_class)
TRANSACTION (number, cust_name, travel_agent_name, amount_paid)

Write SQL statements to answer the following questions.

a. Compute the number of different customers who have a transaction.

```
SELECT name, COUNT (name) AS transactions
  FROM CUSTOMER_t C, TRANSACTION_t T
        WHERE C.name = T.cust_name
          GROUP BY C.name;
```

b. Display the name of the oldest travel agent.

```
SELECT name
  FROM TRAVEL_AGENT_t
        WHERE age IN (SELECT MAX(age)
                        FROM TRAVEL_AGENT_t);
```

c. List the total number of transactions for each travel agent. Consider only those transactions where the amount paid exceeds 1 000.

```
SELECT travel_agent_name, COUNT(travel_agent_name) AS transactions
  FROM TRAVEL_AGENT_t
        WHERE amount_paid > 1000
          GROUP BY travel_agent_name;
```

d. Display the names and ages of the travel agents who have arranged journeys for customer “John Smith”, in descending order of age (use a subquery).

```
SELECT name, age
  FROM TRAVEL_AGENTS_t
        WHERE name IN (SELECT travel_agent_name
                        FROM TRANSACTION_t
                        WHERE cust_name = 'John Smith')
          ORDER BY age DESC;
```

e. Display the names and ages of travel agents who have arranged journeys for customer “John Smith”, in descending order of age (do not use a subquery).

```
SELECT DISTINCT name, age
  FROM TRAVEL_AGENTS_t INNER JOIN TRANSACTION_t ON
```

```
        name = ta_name
    WHERE cust_name = 'John Smith'
     ORDER BY age DESC;
```

f. Display the age of travel agents who have arranged journeys for customer “John Smith” to “Ottawa” (use a subquery).

```
SELECT age
  FROM TRAVEL_AGENTS _t
        WHERE name IN (SELECT travel_agent_name
                              FROM TRANSACTION_t
                               WHERE cust_name = 'John Smith' AND
                                     cust_name IN (SELECT name
                                                          FROM CUSTOMER_t
                                                              WHERE destination = 'Ottawa'));
```

g. Display the age of travel agents who have arranged journeys for customer “John Smith” to “Ottawa” (do not use a subquery).

```
SELECT DISTINCT age
  FROM TRAVEL_AGENTS_t TA, TRANSACTION_t T, CUSTOMER_t C
        WHERE TA.name = T.travel_agent_name
                AND T.cust_name = 'John Smith'
                AND C.destination = 'Ottawa';
```

h. Display the names and salaries of all travel agents who did not arrange journeys for customer “John Smith”, in ascending order of salary.

```
SELECT name, salary
  FROM TRAVEL_AGENTS_t
        WHERE name NOT IN (SELECT ta_name
                                    FROM TRANSACTION_t
                                    WHERE cust_name = 'John Smith')
         ORDERY BY salary ASC;
```

i. Display the names of travel agents who have five or more transactions.

```
SELECT travel_agent_name
  FROM TRANSACTION_t
        GROUP BY travel_agent_name
         HAVING COUNT(travel_agent_name) >= 5;
```

j. Display the names of all travel agents who have arranged at least ten journeys to “Ottawa”.

```
SELECT DISTINCT name
  FROM TRANSACTION_t
        WHERE cust_name IN (SELECT name
                                    FROM CUSTOMER_t
                                     WHERE destination = 'Ottawa'
                                       GROUP BY name
                                         HAVING COUNT(name) >= 10);
```