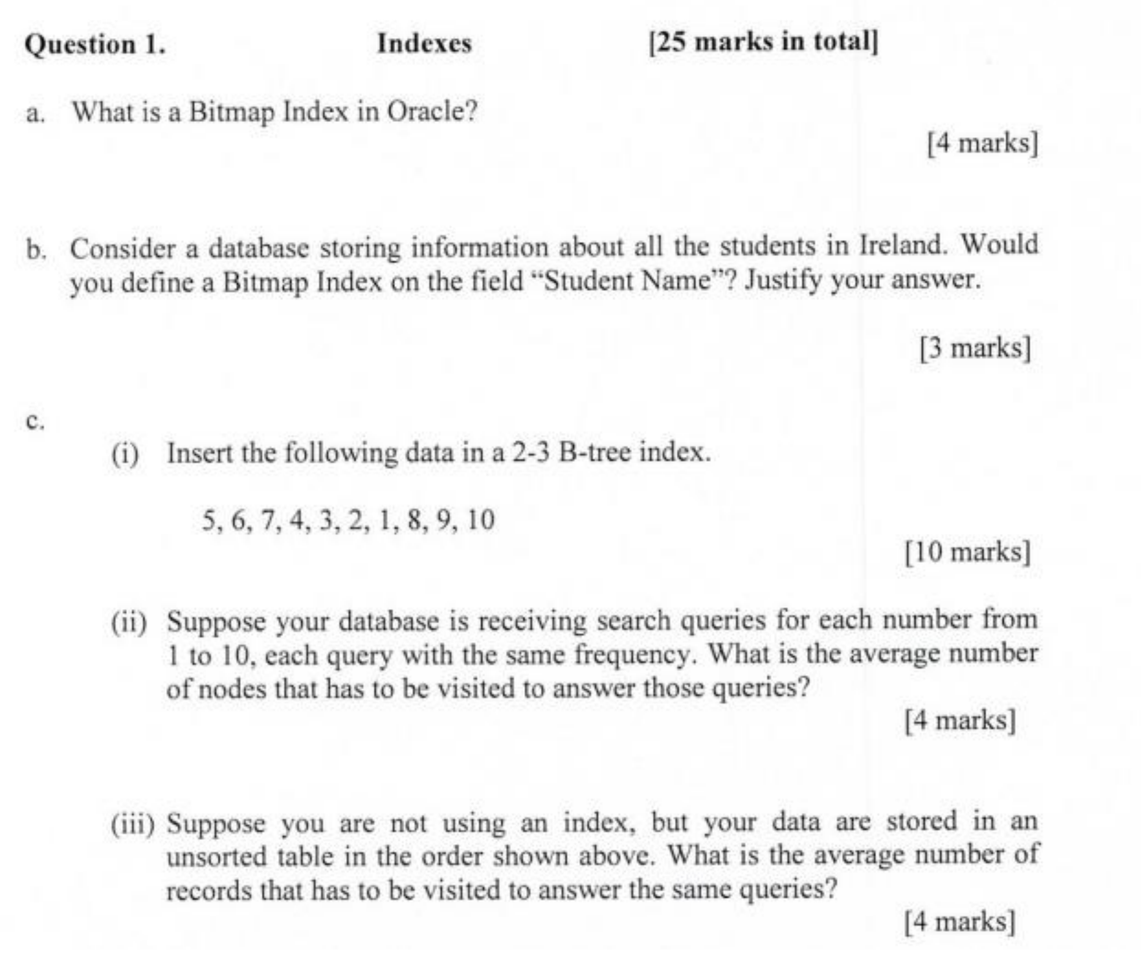


Question 1. **Indexes** **[25 marks in total]**

a. What is a Bitmap Index in Oracle?

[4 marks]

b. Consider a database storing information about all the students in Ireland. Would you define a Bitmap Index on the field "Student Name"? Justify your answer.

[3 marks]

c.

(i) Insert the following data in a 2-3 B-tree index.

5, 6, 7, 4, 3, 2, 1, 8, 9, 10

[10 marks]

(ii) Suppose your database is receiving search queries for each number from 1 to 10, each query with the same frequency. What is the average number of nodes that has to be visited to answer those queries?

[4 marks]

(iii) Suppose you are not using an index, but your data are stored in an unsorted table in the order shown above. What is the average number of records that has to be visited to answer the same queries?

[4 marks]

1.

a. In Oracle, a bitmap index is a type of index that uses a bitmap to represent the index data. Each row in the index corresponds to a bit in the bitmap, and the value of the bit indicates whether the indexed value appears in the corresponding row of the table. Bitmap indexes are most effective when used to index low-cardinality columns, which are columns that have a relatively small number of distinct values.

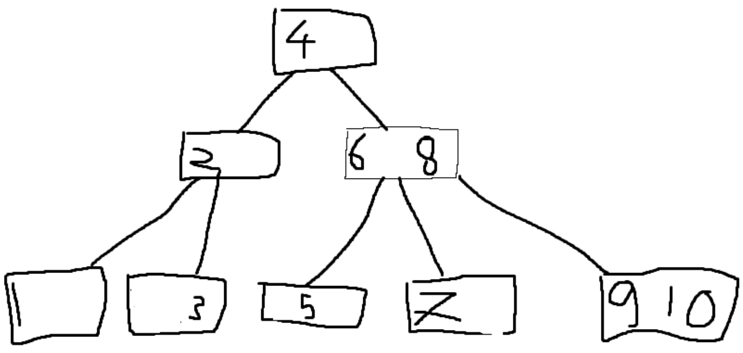
   Bitmap indexes can be used to improve the performance pf queries that filter on low-cardinality columns, because the index can be used to quickly determine which rows in the table match the filter condition.

b. It is generally not recommended to use a bitmap index on a field like “Student name”, because bitmap indexes are most effective when the indexed field has a low cardinality, meaning it has a small number of distinct values. In the case of a field like “student Name “, there are likely to be a very large number of distinct values, since every student will have a unique name.

   In general, bitmap indxes are best suited for fields with a small number of distinct values, such as gender and yes/no field.

c.

   i. https://www.youtube.com/watch?v=bhKixY-cZH



   ii.

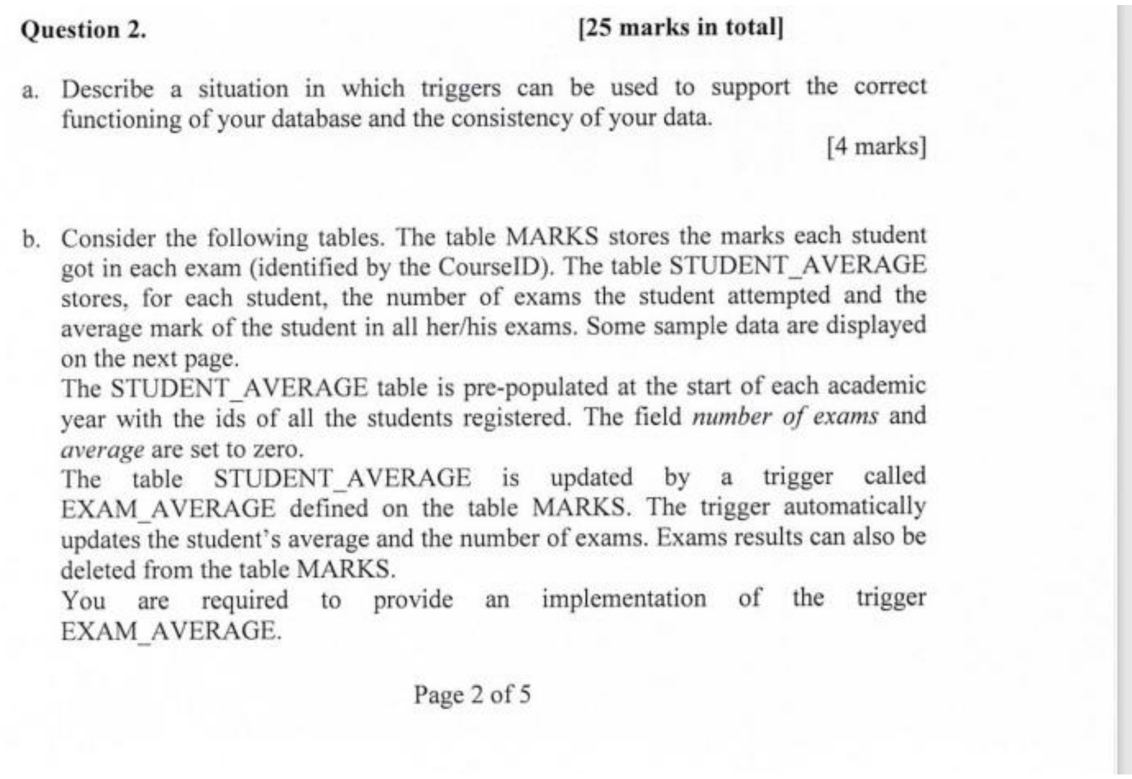
      1. 3
      2. 2
      3. 3
      4. 1
      5. 3
      6. 2
      7. 3

8. 2
9. 3
10. 3

iii.

1. 7
2. 6
3. 5
4. 4
5. 1
6. 2
7. 3
8. 8
9. 9
10. 10

2.



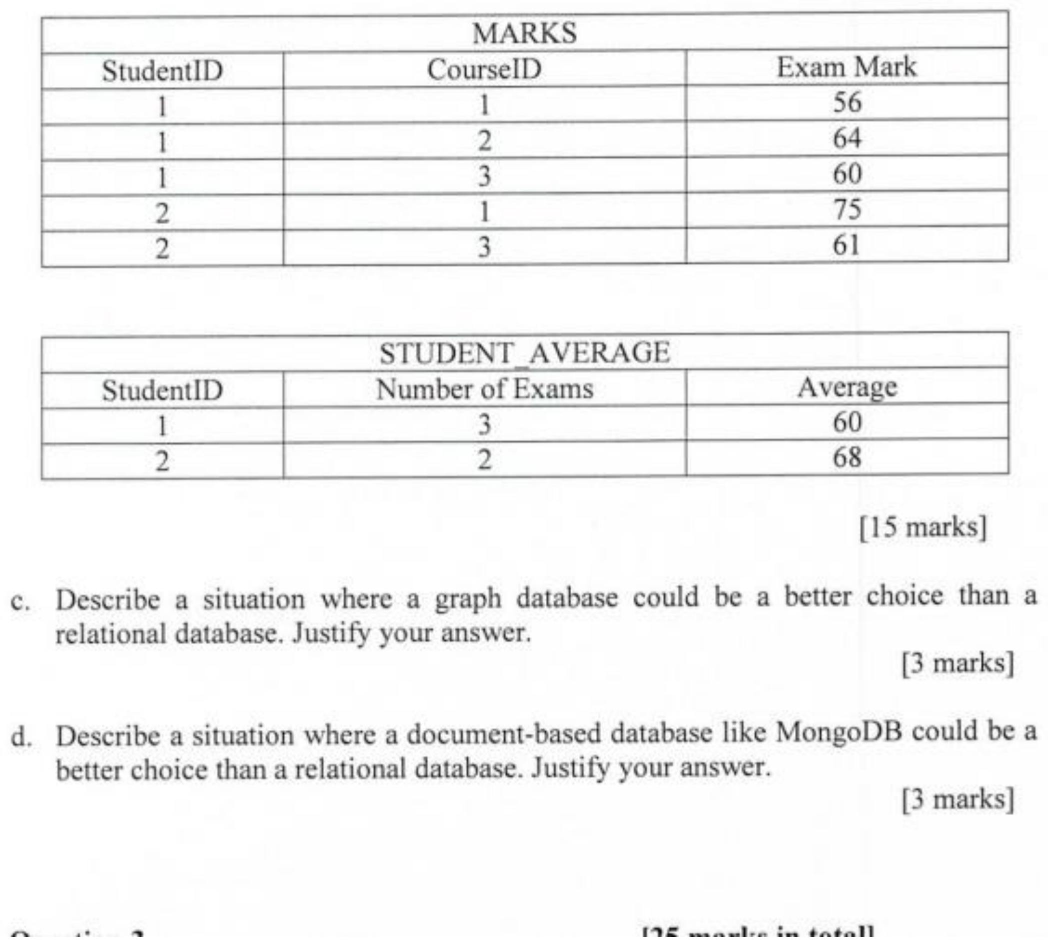
Question 2. **[25 marks in total]**

a. Describe a situation in which triggers can be used to support the correct functioning of your database and the consistency of your data.

[4 marks]

b. Consider the following tables. The table MARKS stores the marks each student got in each exam (identified by the CourseID). The table STUDENT_AVERAGE stores, for each student, the number of exams the student attempted and the average mark of the student in all her/his exams. Some sample data are displayed on the next page.
The STUDENT_AVERAGE table is pre-populated at the start of each academic year with the ids of all the students registered. The field *number of exams* and *average* are set to zero.
The table STUDENT_AVERAGE is updated by a trigger called EXAM_AVERAGE defined on the table MARKS. The trigger automatically updates the student's average and the number of exams. Exams results can also be deleted from the table MARKS.
You are required to provide an implementation of the trigger EXAM_AVERAGE.

Page 2 of 5



MARKS		
StudentID	CourseID	Exam Mark
1	1	56
1	2	64
1	3	60
2	1	75
2	3	61

STUDENT_AVERAGE		
StudentID	Number of Exams	Average
1	3	60
2	2	68

[15 marks]

c. Describe a situation where a graph database could be a better choice than a relational database. Justify your answer.

[3 marks]

d. Describe a situation where a document-based database like MongoDB could be a better choice than a relational database. Justify your answer.

[3 marks]

Question 3 [25 marks in total]

a. A trigger can be used to create a audit trail in order to track updates to order tables. For example, if a customer order table gets updated you can collaborate these changes to an audit table.

Triggers are special type of stored procedure that are automatically executed by the database in response to certain events such as the insertion, update or deletion of a row in a table. Triggers can be used to enforce data integrity and maintain the consistency of the data in a database.

Imagine you have a database that stores information about orders places by customers at an online store. You have table called "Orders" that stores information about each order, including the order status(e.g “pending”, “shipped”, “delivered”). You want to make sure that the order status is always updated correctly and consistently.

To enforce this rule, you can create a trigger that is executed whenever a row in the “Orders” table is updated. The trigger could check the new value of the “order status” field and make sure it is valid. If the new value is invalid. The trigger could roll back the update and return an error message. In this way triggers helps to ensure that the data in the orders table is always consistent and up-to-date and it helps to prevent errors or inconsistencies from creeping into the database

b. Create trigger EXAM_AVERAGE
AFTER
INSERT
ON MARKS
For each row
Update STUDENT_AVARAGE=
Count(MARKS.StudentID) WHERE
STUDENT_AVARAGE.STUDENT_ID
=MARKS.STUDENT_ID , Update
STUDENT_AVARAGE= AVG(MARKS.StudentID)
WHERE STUDENT_AVARAGE.STUDENT_ID
=MARKS.STUDENT_ID
c. Here is an example of a situation where a graph database might be abetter choice than a relational database:
Imagine you are building a social networking website and a you want to store information about users, their connections, and the types of

relationships they have with one another. In this case, a graph database would be a good choice because it allows you to easily store and query the complex relationships between users.

In a graph database, you could represent each user as a node and the relationships between them as edges. You could then use simple queries to find things like the friends of a given user, the friend of their friends, or the shortest path between two users.

By contrast, in a relational database, it would be more difficult to store and query these types of relationships. You might have to create multiple tables and use complex join queries to represent the same information. This can make the database slower and more difficult to work with.

In conclusion, a graph database is a good choice when you need to store and query data that has a complex, interconnected structure and when you need to be able to quickly find relationships and patterns withing the data.

d. A document based database is a type of database that stores data in the form of flexible, semi-structured documents. Document-based databases, such as mongodb ,are designed to be scalable , flexible and easy to use , making them well-suited for a wide range of applications.

Here is an example of a situation where a document based database might be a better choice than a relational database:

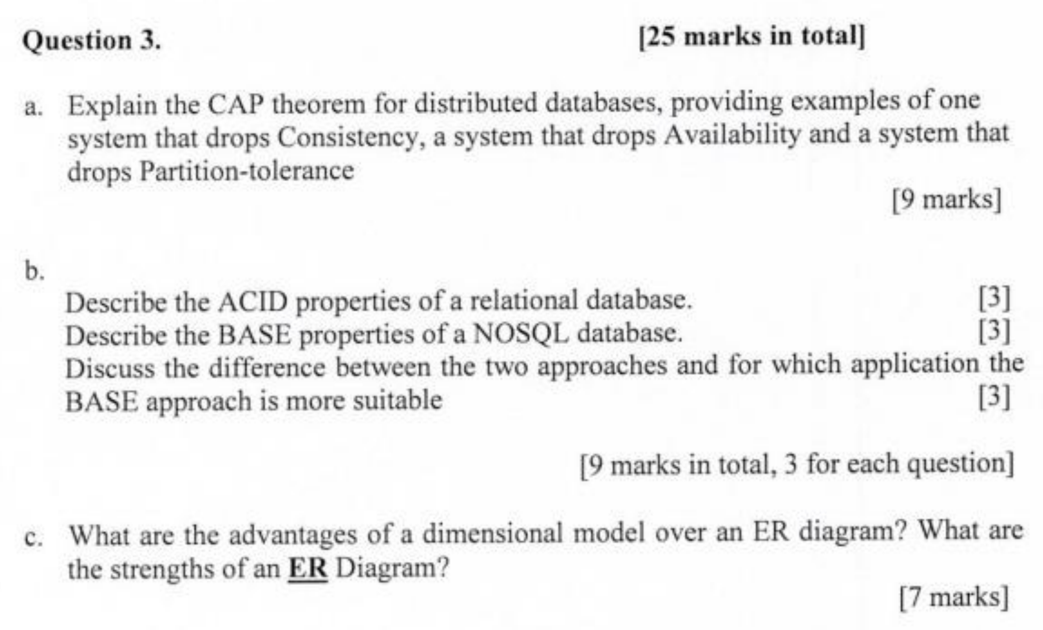
Imagine you are building a content management system for a news website. You need to store a large number of articles, each of which has a title a body , a publication date , and a list of tags , you also need to be able to search for articles based on a various criteria, such as the publication date or the tags,

In this case, a document-based database would be a good choice because it allows you to store the articles as flexible, semi structure documents. You can easily store all the information about an article in a single document, and you can use the database's query language to search for articles based on various criteria.

By contrast, in a relational database, you might have to create multiple tables to store the different types of information about an article and you would have to use complex join queries to search for articles based on multiple criteria. This can make database slower and more difficult to work with.

Overall, a document-based database is a good choice when you need to store and query large amount of semi-structured data, and when you need a flexible and scalable database that is easy to use.

3.


Question 3. **[25 marks in total]**

a. Explain the CAP theorem for distributed databases, providing examples of one system that drops Consistency, a system that drops Availability and a system that drops Partition-tolerance

[9 marks]

b.

Describe the ACID properties of a relational database. [3]

Describe the BASE properties of a NOSQL database. [3]

Discuss the difference between the two approaches and for which application the BASE approach is more suitable [3]

[9 marks in total, 3 for each question]

c. What are the advantages of a dimensional model over an ER diagram? What are the strengths of an **ER** Diagram?

[7 marks]

a. Cap theorem is a fundamental principle that states that it is impossible for a distributed database system to simultaneously provide all three of the following guarantees:
   i. Consistency: All nodes in the system see the same data at the same time.
   ii. Availability: all nodes in the system are able to respond to requests at all times
   iii. Partition-tolerance: the system can continue to operate even if some nodes are unavailable or there is a network partition.

   According to the CAP theorem, a distributed system must choose to either sacrifice consistency, availability, or partition-tolerance in order to function.

   Here are some examples of systems for dropping consistency or partition-tolerance:

- A system that drops consistency: Cassandra is an example of a distributed database system that prioritizes availability and partition-tolerance over consistency. Cassandra is designed to continue operating even if nodes fail or there is a network partition, and it allows users to choose the level of consistency they want for their data. However, this means that it is possible for different nodes to see different versions of the same data at the same time, which can lead to inconsistencies
- A system that drops partition-tolerance: MYSQL is an example of a distributed database system that prioritizes consistency and availability over partition-tolerance. MySQL uses a technique called "master-slave replication " to ensure that all nodes see the same nodes fail. However, if there is a network partition, the system will become unavailable until the partition is resolved.

CAP theorem is a useful way to understand the trade-offs involved in designing a distributed database system, and it helps to inform the design and implementation of systems that need to handle large amount of data across multiple nodes.

b.

i.

ACID is an acronym that stands for four properties of a database transaction: Atomicity, Consistency, Isolation and Durability. These four properties ensure that database transactions are reliable, even in the face of errors, power failures, and other unexpected events.

1. Atomicity: This property ensures that a transaction is either completed in its entirety or not completed at all. In other words, if a transaction involves multiple updates to the database, either all of the updates are applied or none of them are applied. This is known as “all-or-nothing” semantics
2. Consistency: This property ensures that a transaction leaves the database in a consistent state, meaning that the data adheres to all defined rules and constraints. If a transaction violates a constrain or rule, its in not allowed to complete.
3. Isolation: this property ensure that concurrent transactions do not interfere with each other. Each transaction is executed as if it were the only transaction taking places, even if other transaction is occurring at the same times. This prevents data from being corrupted by concurrent updates.
4. Durability: This property ensures that once a transaction has been committed, it will not be lost. The changes made by the transaction are permanently recorded in the database, even if the database crashes or shuts down.

ii. Base is an acronym that stands for four properties of a NOSQL database: Basically

Available, Soft state, Eventual consistency. These properties are a set of design properties for NOSQL databases that prioritize flexibility and scalability over ACID properties of traditional relational databases.

1. Basically Available: This property refers to the ability of the database to continue functioning, even if some parts of the system are unavailable or experiencing errors. This is in contrast to the “all-or-nothing” semantics of the atomicity property in ACID databases, which requires that all parts of the system be available for a transaction to complete.
2. Soft state: This property refers to the fact that the state of a NOSQL database can change over time , even without direct input from users or application. For example, a distributed database may automatically replicate data between nodes, leading to changes in the overall state of the system.
3. Eventual consistency: This property refers to the fact that, over time, all copies of the data in a NOSQL database will eventually converge to the same state. In other words, even if different copies of the data are temporarily inconsistent , they will eventually become consistent as changes are propagated throughout the system. This is in contrast to the consistency property in ACID databases which requires that all

copies of the data be consistent at all times.

Together these properties allow NOSQL databases to be highly available, scalable, and flexible, but at the cost of some of the strong consistency guarantees provided by ACID databases.

iii. One key difference between ACID and BASE is the way they handle consistency. ACID databases prioritize strong consistency, meaning that all copies of the data must be consistent at all times. BASE databases, on the other hand, prioritize availability and may allow for temporary inconsistencies between copies of the data.

In general, ACID databases are more suitable for applications that require strong consistency and transactional guarantees, such as financial system or e-commerce websites. BASE databases are more suitable for applications that require high availability and scalability, such as social media platforms or real-time data processing systems.

It's worth noting that there is a trade-off between consistency and availability, and the appropriate choice will depend on the specific needs of the application.

c.

i. Dimensional model and Entity-Relationship (ER) diagrams are both techniques for designing and organizing data in a database. Both approaches can be used to model the relationships between different entities and their attributes, but they differ in the way they represent and organize data.

One advantage of dimensional modelling over ER diagrams is that it is optimized for querying and analysis. Dimensional models are designed to support fast and efficient querying using a technique called “star schema” which organizes data into a central fact table surrounded by smaller dimension tables. This design allows for fast querying and aggregation of data, making it well suited for data warehousing and business intelligence applications.

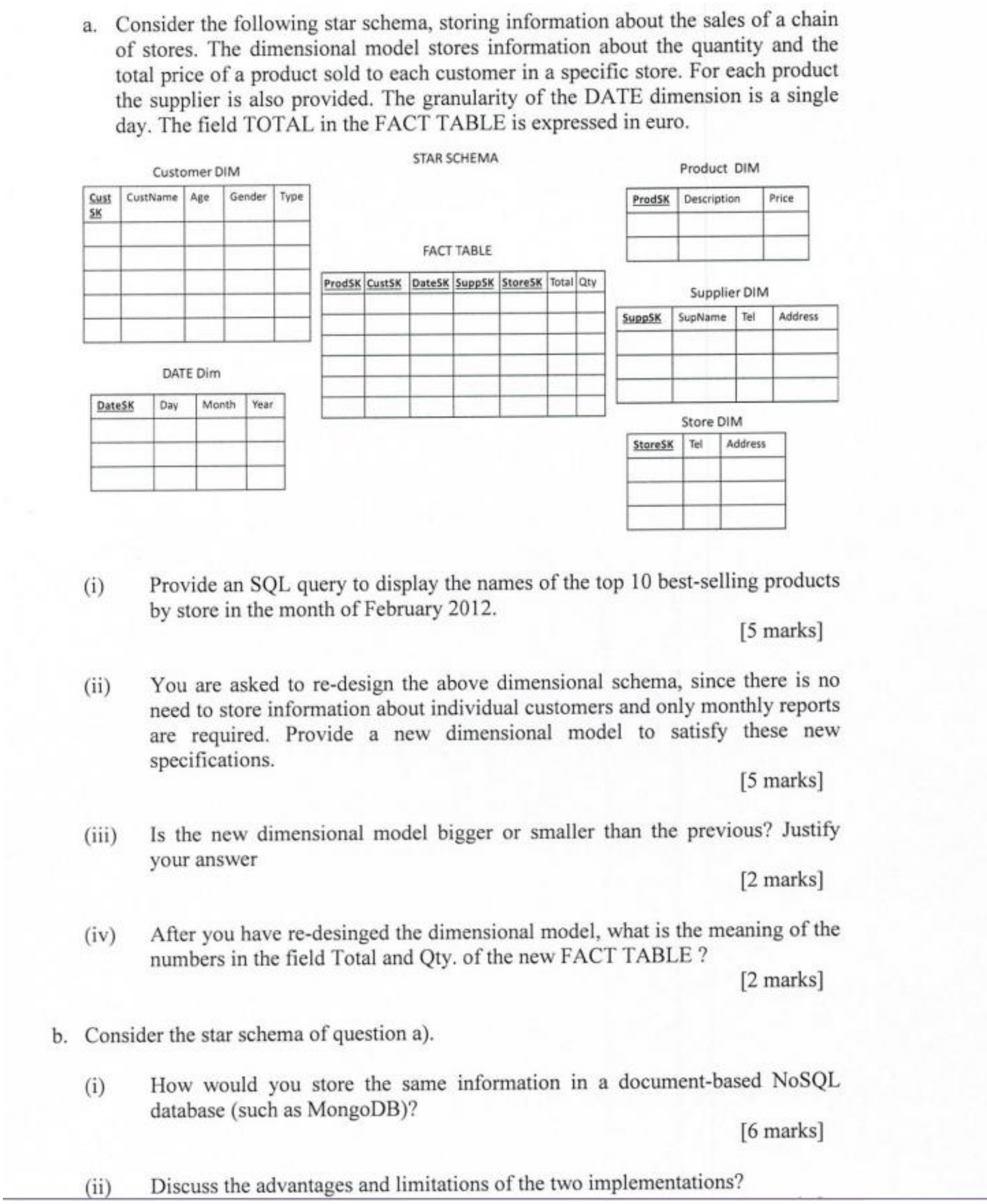
Another advantage of dimensional modelling is that it is more intuitive and easier to understand than ER diagrams. Dimensional models use a more straightforward and visual approach to representing data, making it easier for business users and analysts to understand and work with the data.

Some strengths of ER diagrams include:

1. They provide a visual representation of the database structure: ER diagrams use a graphical notation to represent entities, attributes, and relationships, making it easier to understand the overall structure of the database.

2. They allow for easy identification of relationships between entities: ER diagrams use connectors to show the relationships between different entities making it easy to see how different entities are related to one another.
3. They support database normalization: ER diagrams can be used to design a database that is normalized and dependency. Normalized databases are generally more efficient and easier to maintain.
4. They can be used to generate database schemas: Many database management systems can generate the database schema (the structure of the database) directly from an ER diagram, making it easy to create database on the design.

4.



a. Consider the following star schema, storing information about the sales of a chain of stores. The dimensional model stores information about the quantity and the total price of a product sold to each customer in a specific store. For each product the supplier is also provided. The granularity of the DATE dimension is a single day. The field TOTAL in the FACT TABLE is expressed in euro.

STAR SCHEMA

Customer DIM

Cust SK	CustName	Age	Gender	Type

Product DIM

ProdSK	Description	Price

FACT TABLE

ProdSK	CustSK	DateSK	SuppSK	StoreSK	Total	Qty

Supplier DIM

SuppSK	SupName	Tel	Address

DATE Dim

DateSK	Day	Month	Year

Store DIM

StoreSK	Tel	Address

(i) Provide an SQL query to display the names of the top 10 best-selling products by store in the month of February 2012.

[5 marks]

(ii) You are asked to re-design the above dimensional schema, since there is no need to store information about individual customers and only monthly reports are required. Provide a new dimensional model to satisfy these new specifications.

[5 marks]

(iii) Is the new dimensional model bigger or smaller than the previous? Justify your answer

[2 marks]

(iv) After you have re-desinged the dimensional model, what is the meaning of the numbers in the field Total and Qty. of the new FACT TABLE ?

[2 marks]

b. Consider the star schema of question a).

(i) How would you store the same information in a document-based NoSQL database (such as MongoDB)?

[6 marks]

(ii) Discuss the advantages and limitations of the two implementations?

a.

i. SELECT ProductDIM.Description
 From ProductDIM
 JOIN FACT TABLE on
 PRODUCTDIM.PRODSK=FACTTABLE.PRODSK,

JOIN DATEDIM on
DATEDIM.DATESK=FACTTABLE.DATESK
WHERE MONTH== 2 && YEAR==2012 &&
TOTAL

ii. Cut out the customer table and the day column in datedim
iii. Smaller as there is no need for individuals.
iv. Total money spend and , quantity of of products sold

b.

i. In a document-based database like MongoDB, data is stored in the forms of “documents,” which are structured as collections of key-value pairs. To store the same information that might be represented in a traditional relational database or an ER diagram, you could create a collection of documents, each representing a particular entity or object

   For example, in document-based database you could represent this information using a collection of documents each representing a particular supplier or product.

   For example, a supplier document might contain fields such as SupName, Telephone and Address. To represent the relationships between supplier and product , you could include the ProductId as a field in supplier.

ii.

   1. Dimensional modelling advantages:
      - Simplicity: Dimensional modelling is relatively easy to understand, even

for people who are not familiar with database design. This makes it an attractive option for organizations that want quickly and easily build a data warehouse.

- Performance: Dimensional modelling is optimized for fast query perform performance, which is essential for data warehouses that are used to support business intelligence and analytics.
- Flexibility: Dimensional modelling is highly flexible and can be easily adapted to changing business needs. For example, new dimensions can be added as needed to support new analysis requirements.

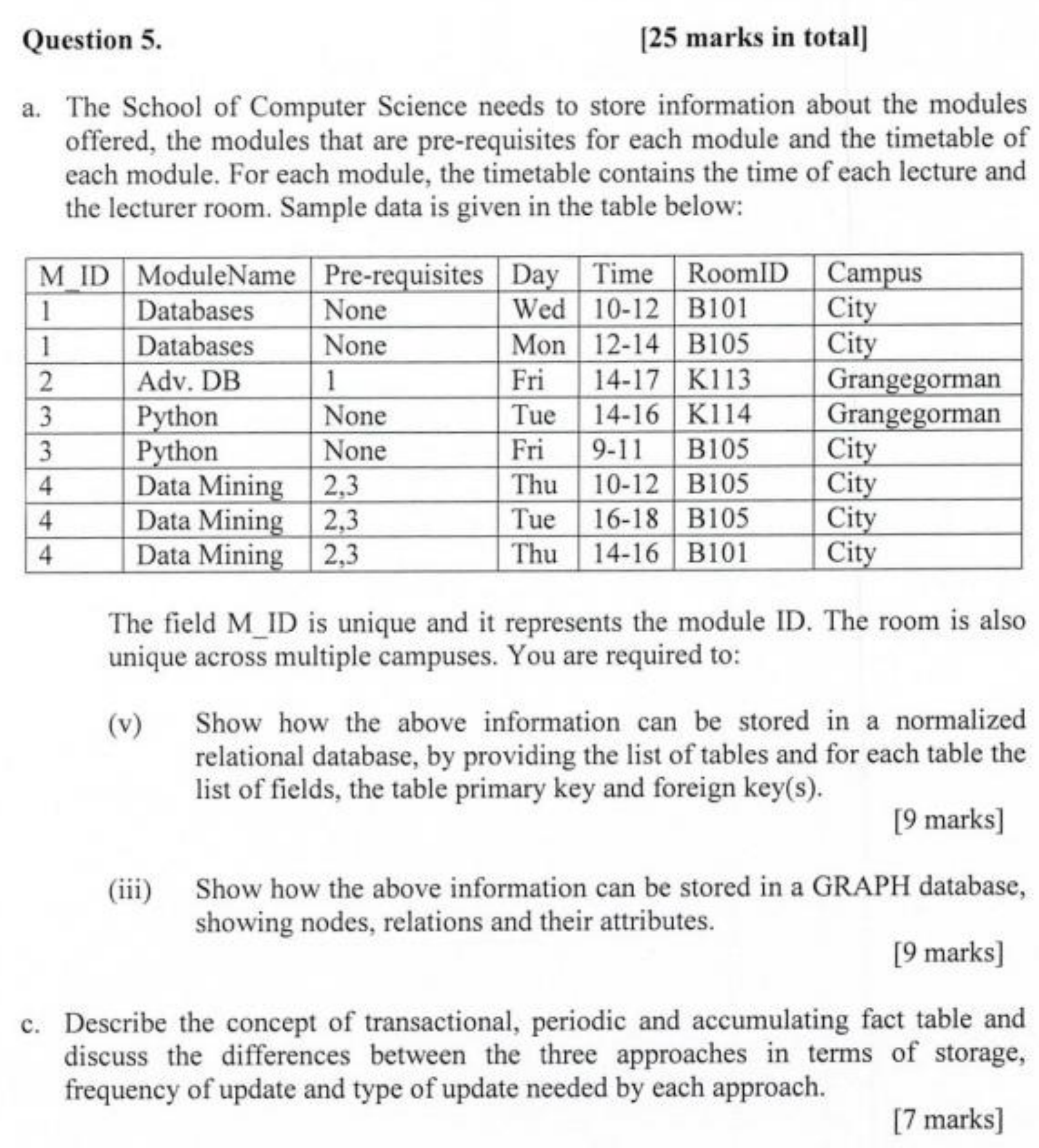
Dimensional modelling limitations:

- Inflexibility: Dimensional modelling can be inflexible when it comes to handling changes to the underlying data. For example, if the structure of a dimension changes, it can be difficult to update the data warehouse to reflect those changes.
- Limited data types: Dimensional modelling is not well-suited for storing complex data types, such as text or images.

2. NOSQL document-based database advantages:
    - Flexibility: Document-based NOSQL databases are highly flexible and can store data in a variety of formats, including text, images, and even binary data. This makes them well-suited for storing complex data structures.
    - Scalability: Document-based NOSQL databases are designed to scale horizontally, which means that they can easily handle large amounts of data and handle high levels of concurrency.
    - Performance: document-based NOSQL databases are generally fast and can handle large numbers of reads and writes.

   NOSQL document-based database Limitations:
    - Complexity: document-based NOSQL databases can be more complex to use than traditional relational databases, which can make it more difficult for developers to work with them.

- Lack of transactions: Some document-based NOSQL databases do not support transactions, which can make it difficult to ensure the integrity of data in certain scenarios.
- Limited Querying: document-based NOSQL databases are not as good at supporting complex queries as traditional relational databases. This can make it difficult to perform certain types of analysis on the data.



Question 5. **[25 marks in total]**

a. The School of Computer Science needs to store information about the modules offered, the modules that are pre-requisites for each module and the timetable of each module. For each module, the timetable contains the time of each lecture and the lecturer room. Sample data is given in the table below:

M_ID	ModuleName	Pre-requisites	Day	Time	RoomID	Campus
1	Databases	None	Wed	10-12	B101	City
1	Databases	None	Mon	12-14	B105	City
2	Adv. DB	1	Fri	14-17	K113	Grangegorman
3	Python	None	Tue	14-16	K114	Grangegorman
3	Python	None	Fri	9-11	B105	City
4	Data Mining	2,3	Thu	10-12	B105	City
4	Data Mining	2,3	Tue	16-18	B105	City
4	Data Mining	2,3	Thu	14-16	B101	City

The field M_ID is unique and it represents the module ID. The room is also unique across multiple campuses. You are required to:

(v) Show how the above information can be stored in a normalized relational database, by providing the list of tables and for each table the list of fields, the table primary key and foreign key(s).

[9 marks]

(iii) Show how the above information can be stored in a GRAPH database, showing nodes, relations and their attributes.

[9 marks]

c. Describe the concept of transactional, periodic and accumulating fact table and discuss the differences between the three approaches in terms of storage, frequency of update and type of update needed by each approach.

[7 marks]

5.

a.

i.

1. Module_table, Timetable, Room_table
2. Module_table: M_ID , ModuleName , Pre_requistes

3. Timetable: T_ID, Day, Time, M_ID_FK, RoomID_FK
4. Room: RoomID, Campus
5. Primary: M_ID, T_ID, RoomID
6. Foreign keys: M_ID_FK, RoomID_FK

ii. ???

b.

c. In dimensional modelling, a fact table is a central table that stores measures or facts about a particular business process. There are three main types of fact tables: transactional , periodic and accumulating.

   i. Transactional fact tables: Transactional fact tables store data at the most granular level and are typically updated in real-time as transactions occur. These tables contain a record of each individual transaction and are typically used to support operational reporting and analysis. transactional fact tables are typically updated very frequently and may require frequent updates to maintain data integrity.

   ii. Periodic fact tables: Periodic fact tables store data at a more aggregated level and are updated on a periodic basis, such as daily or monthly. These tables are used to support trend analysis and are typically used for reporting on longer time periods. Periodic fact tables are updated less frequently than transactional fact tables and may require less frequent updates.

   iii. Accumulating fact tables: Accumulating fact tables store data at a more aggregated level and are updated as events occur. However, unlike transactional fact tables, accumulating fact tables do not store each individual event, but rather store the total accumulated value up to a certain point in time. These tables are used to track the progress of an event over time and are typically used for reporting on long-term trends. Accumulating fact tables are updated less frequently than periodic fact tables and may require fewer updates.

In terms of storage, transactional and periodic fact tables tend to be larger than accumulating fact tables because they store more detailed data. However, accumulating fact tables may require more storage over time as they accumulate more data.

In terms of frequency of update, transactional fact tables are typically updated more frequently than periodic or accumulating fact tables. Periodic fact tables are typically updated less frequently than transactional fact tables but more frequently than accumulating tables. Accumulating fact tables are typically updated the least frequently of the three types.

In terms of the type of update needed, transactional fact tables may require more complex updates to maintain data integrity, as each individual transaction must be accurately recorded. Periodic fact tables may require more simple updates to aggregate data over a set time period. Accumulating fact tables may require the simplest updates, as they only need to track the accumulation of a value over time.