

Part 1

Question 1 (1)

No, AUTHORS attribute cannot be a primary key as there are some values that are not unique. For example, "Joe Celko" occurs in multiple rows as he has published many books.

Question 1 (2)

Schema book 3 can answer this query:

```
SELECT TOP(99) SalesRank, Title, PaperbackPrice, HardcoverPrice,  
EbookPrice, AudiobookPrice  
FROM Book3  
ORDER BY SalesRank ASC;
```

Question 2 (1)

By using horizontal fragmentation, we can split the data into two fragments

Fragment 1: publication_year > 21

Fragment 2: publication_year <=21

Question 2(2)

Since the predicate publication_day is not a primary key, we need to check all fragments and make sure the primary key of the record to be inserted does not already exist. If the primary key does not already exist in the database, then proceed to insert the record into the correct fragment that matches the record's publication_day. Otherwise, a primary key violation will mean the tuple cannot be inserted.

Part 2

Question 3

<i>Day</i>	<i>Publisher</i>	<i>Language</i>	<i>Sales</i>
Dimension column	Dimension column	Dimension column	Fact column

Question 4 (1)

One of the main advantages of bitmap indexing is the reduction of space and I/O cost. By using bits, there are less values stored compared to the entire varchar value of a given attribute, for example the size in bytes of the Publisher attribute as it contains many characters. Another advantage of bitmap indexing is that query processing time is reduced when using bit operations.

The type of columns that are suitable for bitmap indexing are the ones that are infrequently updated. It is suitable for low cardinality domains where the attributes don't have a high number of distinct values.

Question 4 (2)

Publisher

Day	AAAI Press	Springer International Publishing	Springer London	IEEE Computer Society Press
07/15/1984	1	0	0	0
05/05/1990	0	1	0	0
06/04/1995	0	0	1	0
12/11/2000	0	0	0	1
04/03/2004	1	0	0	0
05/01/2008	0	1	0	0
11/19/2012	0	0	1	0
08/06/2014	0	0	0	1

Language

Day	English	Spanish
07/15/1984	1	0
05/05/1990	1	0
06/04/1995	1	0
12/11/2000	1	0
04/03/2004	0	1
05/01/2008	0	1
11/19/2012	0	1
08/06/2014	0	1

Question 4 (2)

Find total sales of Spanish books published by “AAAI Press”:

- 1.) Scan the “AAAI press” vector and select all records where “AAAI press” = 1.
- 2.) Scan the Spanish vector and select all records where “Spanish” = 1.
- 3.) Summarise and select the records that appear in both of the bitmap indexes above.
i.e. by combining both bit values for each record, find the records with 11.
- 4.) Return the sales value of those records from the original table.

Part 3

Question 5 (1)

Global conceptual schema:

Books(id, book_title, authors, publication_date, publisher, ISBN13, pages)

Question 5 (2)

Structural heterogeneity- key conflicts where book1.id != book2.id. The same book record may appear in 2 different schemas but have different id on different schemas. Book1 and Book2 refer to id while Book3 and Book4 refer to ID.

Solution: We need to maintain a mapping in the tables where id can be different but refer to the same book. This will be Book1 <-> Book3, Book1 <-> Book4, Book2 <-> Book3, Book2 <-> Book4 where Book1 and Book2 stores the ID from Book3 and Book4.

Semantic heterogeneity- imprecise wording with publication_date. This can refer to different formats for day, month and year. For example, in Book1 pubyear, pubmonth and pubday are specified which means the date can be derived from those fields. However, in Book3 and Book4 only the exact formatting of the specific publication date is available.

Solution: Need to derive each of the publication day, month and year from Book3 and Book4 so that when all data is integrated from all 4 schemas, the publication_date is consistent with one format dd/mm/yy where dd, mm, yy are integers from 0-9.

Part 4

Question 6

```
public class Q6 {

    public static void main(String[] args) {
        //QUESTION 6 (1)
        DBConnection.OpenConnection();
        String query = "SELECT * FROM BOOK3 WHERE MOD(id,100) = 0";
        book3[] books = DBConnection.ExecuteQueryBook3(query);
        DBConnection.CloseConnection();
        System.out.println("SampleSet = " + books.length);

        //QUESTION 6(2)
        int nullCells = nullFields(books);
        System.out.println("NULL fields = " + nullCells);

        //QUESTION 6 (3)
        int COLUMNS = 17;
        int totalCells = books.length * COLUMNS;
        double EPMO = ((double) nullCells / totalCells) * 1000000;
        System.out.println("EPMO = " + EPMO);

    }

    private static int nullFields(book3[] books) {
        int nullCounter = 0;
        for (int i = 0; i < books.length; i++) {

            if (books[i].getAuthor1() == null) {
                nullCounter++;
            }
            if (books[i].getAuthor2() == null) {
                nullCounter++;
            }
            if (books[i].getAuthor3() == null) {
                nullCounter++;
            }
            if (books[i].getDate() == null) {
                nullCounter++;
            }
            if (books[i].getProductDimensions() == null) {
                nullCounter++;
            }
            if (books[i].getSalesRank() == null) {
                nullCounter++;
            }
            if (books[i].getRatingsCount() == null) {
                nullCounter++;
            }
            if (books[i].getRatingValue() == null) {
                nullCounter++;
            }
            if (books[i].getPaperbackPrice() == null) {
                nullCounter++;
            }
            if (books[i].getHardcoverPrice() == null) {
                nullCounter++;
            }
        }
    }
}
```

```
        if (books[i].getEbookPrice() == null) {
            nullCounter++;
        }
        if (books[i].getAudiobookPrice() == null) {
            nullCounter++;
        }
        if (books[i].getPublisher() == null) {
            nullCounter++;
        }
        if (books[i].getISBN13() == null) {
            nullCounter++;
        }
        if (books[i].getPages() == null) {
            nullCounter++;
        }
    }
    return nullCounter;
}
```

```
C:\Users\juloe\.jdk\corretto-11.0.11\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
SampleSet = 37
NULL fields = 286
EPMO = 454689.98410174885

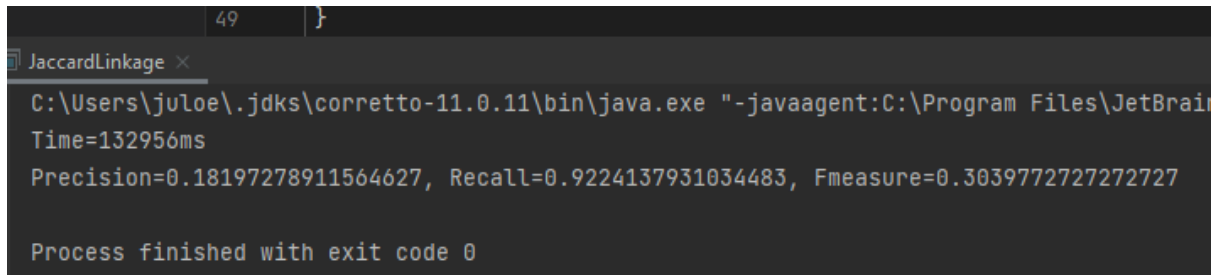
Process finished with exit code 0
```

- (1) SampleSet = 37
- (2) NULL fields = 286
- (3) EPMO = 454689.984

Question 7 (1)

The Jaccard distance will regard them as more similar compared to edit distance function. Both strings differ the most with the ordering of the first author and last author. All 3 authors are actually identical with some differences with characters “,” and “;”. So, since the word order is the biggest difference, the Jaccard function will find them more similar as the Jaccard function is less sensitive to word ordering than edit distance function.

Question 7 (2)

A screenshot of a Java command prompt window titled "JaccardLinkage". The window shows the execution of a Java command using the Java agent. The output displays the execution time, precision, recall, and f-measure values, followed by a message indicating the process finished successfully.

```
49 }  
C:\Users\juloe\.jdk\corretto-11.0.11\bin\java.exe "-javaagent:C:\Program Files\JetBrain  
Time=132956ms  
Precision=0.18197278911564627, Recall=0.9224137931034483, Fmeasure=0.30397727272727  
Process finished with exit code 0
```

When compared to the gold standard results, the data linkage produced a very low precision and a very high recall. There were many titles in the linkage that were considered similar but were actually not similar; yielding many false positive results.