BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 5 Diploma in IT

DATABASE SYSTEMS

March 2017
Answer **any** THREE questions out of FIVE. All questions carry equal marks.
Time: THREE hours

Answer any Section A questions you attempt in Answer Book A
Answer any Section B questions you attempt in Answer Book B

The marks given in brackets are **indicative** of the weight given to each part of the question.

Calculators are **NOT** allowed in this examination.

Section A

Answer Section A questions in Answer Book A

A1

Examiners' Guidance Notes

This was a fairly popular question, attempted by nearly two-thirds of the candidates, with nearly half of those attempts achieving a pass mark. There was a wide range in the performance of candidates with some candidates producing near fully correct solutions.

Study the following scenario then attempt the question parts that follow:-

Scenario AFC United

AFC United is a football club that host matches (also known as **Fixtures**) against an opposing team. A database is required by the club to hold information to support the booking of seats to watch a match at the club's stadium over the course of a season.

Prior to the start of a season a set of fixtures are arranged between AFC United (the home team) and an opposing team (the away team). Matches are watched by spectators who have registered with the club. These are called **PassHolders.** PassHolders must book seats in advance for any of the 20 home fixtures that season.

Once a **Seat** is booked (and payment made), the Pass Holder is issued with one or more **Tickets**.

A particular seat can have restricted occupancy. This is recorded as the seating type.

A PassHolder can purchases one or more tickets for seats for each fixture.

Important attributes (highlighted in bold font)

TicketNo This uniquely identifies which seat has been purchased on a particular date (the **Purchase Date**) by a particular Passholder for attending a particular fixture.

FixtureID This attribute uniquely identifies a particular football match that is played on a particular date (the **Fixture date**) between the home team and an opponent (called the away team). All games are played at the same stadium - the home ground of ABC United.

AwayTeam a unique 4 character code identifies the Team. Examples are CHEL,MANC,SUND,BORO

SeatID uniquely identifies a seat. Examples MM59,H105

SeatingArea is a seating area within the stadium. There are five distinct seating areas called North, South, East, West and Upper West Stands.

SeatType: Over the course of a season a fixed number of seats have restricted and exclusive occupancy. A restricted seat is recorded as one of four seat types Family; Reserved; Disabled; Away Supporters.

PassHolderID identifies a particular Pass Holder of AFC United, a person who has registered prior to purchasing seats. Passholder details such as **name** and contact details (**PHAddress**, **PhoneNumber**) are recorded.

< End of Scenario>

a) Derive an Entity Relationship data model for the scenario above showing :-

The entity Types **PassHolders**, **Fixtures**, **Seats** and **Tickets** and the Relationships between them.

Relationship constraints:
For each relationship show the degree (1:Many;many :many or 1:1) and participation (mandatory or optional).

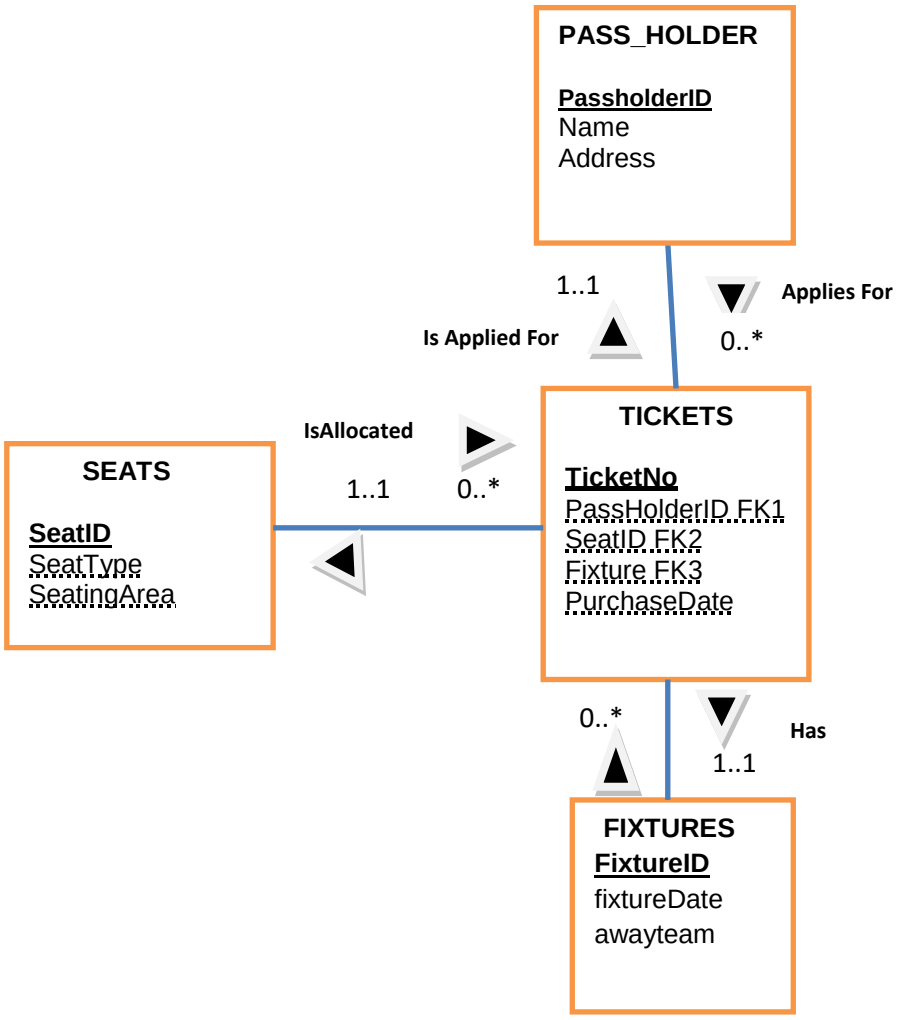
Attributes assigned to entity types using the listed attributes shown above in bold font. Underline the primary key attributes

You may choose any modelling notation but ***you must state*** the notation you have used. State any necessary assumptions you make, on the understanding they do not contradict the scenario.

11 marks

ANSWER POINTER

Figure A1 Solution data model for use in question A1 UML



The ERD should include only 4 entity types.
Marks are gained for a correct (ternary) association between entity types and reasonably correct relationship constraints. Primary key attributes should be indicated and attributes reasonably defined and correct The ERD should depict a ternary relationship.

b) Design a set of Tables, derived from your ERD and illustrate them with a **small amount** of sample data. That is, data that represents all of the degrees of the relationships.

Note: You are not required to include data types or check constraints, but you must specify all columns and state which are primary/foreign keys.

6 marks

ANSWER POINTER
It is important to present tables that are consistent with the ERD and to highlight aspects of data integrity.
Primary key column names should be identified; Foreign key column names specified.

Table `tickets`

TICKETNO	PURCHASEDATE	PASSID	FIXTUREID	SEATID
107823	13-Dec-2016	10032	8320	MM59
959235	13-Dec-2016	10032	8321	H105

736228	*15-Oct-2016*	*3420*	*9770*	*K4*
107922	*15-Dec-2016*	*10035*	*8320*	*H105*
107923	*13-Dec-2016*	*10035*	*8320*	*H106*

Table `fixtures`

FIXTUREID	**FIXTUREDATE**	**AWAYTEAM**
8320	*02-Jan-2017*	*CHEL*
8321	*23-Jan-2017*	*MANC*
8322	*18-Dec-2016*	*SUND*
9767	*15-Mar-2017*	*BORO*
9770	*16-Oct-2016*	*WATF*

Table `passholder`

PASSID	**NAME**	**ADDRESS**
10032	*R.Sayers*	*'Tess' Ilkley Moor*
7243	*P.Smith*	*'Homeblest', Preston Capes*
10035	*V.Singh*	*23 Belle Vue St, Odiham*
3420	*P.Smith*	*Dove Cottage, Stratford*

Table: `seats`

SEATID	**SEATINGAREA**	**SEATTYPE**
C11	*West Stand*	*Reserved*
MM59	*West Upper*	
H105	*South Stand*	*Family*
H106	*South Stand*	*Family*
G3	*East Stand*	*Away*
K4	*North Stand*	*Disabled*

c) Show with the aid of a diagram how the data in your tables is related to support the data access requirements of the following query:-

List the names of the passholders who have purchased tickets with Seat type = "Family" for a particular fixture?

3 marks

ANSWER POINTER

Full credit can be gained by simply drawing the connections between the tables and highlighting the related data that are involved. For example the selected seat type for Family is linked to the seatIDs in the seat table. The seatIDs are then linked to the tickets table from which the PassIDs are found and then the corresponding names of passholders are returned, for example

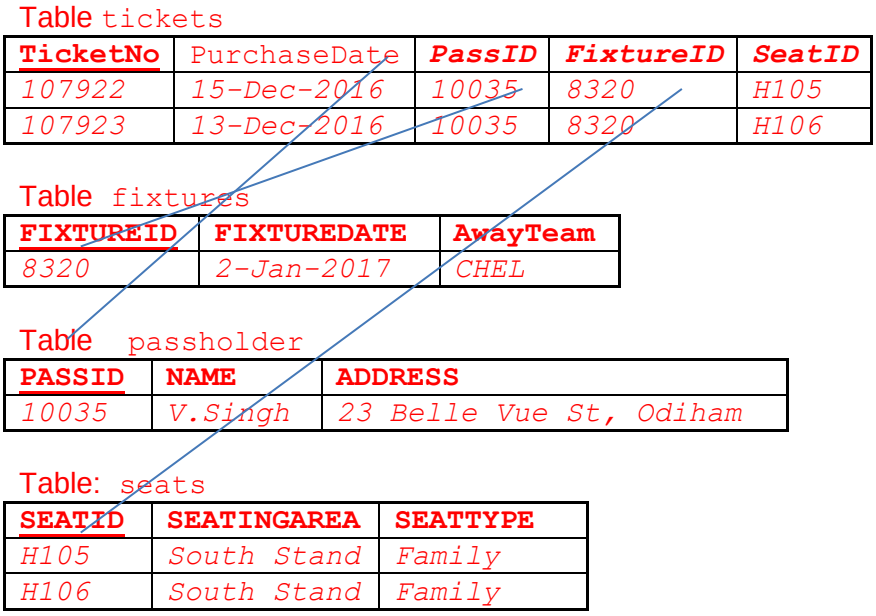


Table tickets

TicketNo	PurchaseDate	PassID	FixtureID	SeatID
107922	15-Dec-2016	10035	8320	H105
107923	13-Dec-2016	10035	8320	H106

Table fixtures

FIXTUREID	FIXTUREDATE	AwayTeam
8320	2-Jan-2017	CHEL

Table passholder

PASSID	NAME	ADDRESS
10035	V.Singh	23 Belle Vue St, Odiham

Table: seats

SEATID	SEATINGAREA	SEATTYPE
H105	South Stand	Family
H106	South Stand	Family

d) Explain briefly what is meant by a Weak Entity Type. Provide an example of a Weak EntityType either from the scenario or using an example of your own.

5 marks

ANSWER POINTER

1. It is existence-dependent on another entity, that is, it cannot exist without the parent entity with which it has a parent-child relationship. .

2. It inherits at least part of its primary key from the entity to which it is related.

The primary key of a weak entity is a composite key that includes the primary key of the entity on which it is existence-dependent. Therefore if the ticket entity type had a composite key of FixtureID,PassID,SeatID then it would be a Weak Entity Type based on three parents because FixtureID etc are primary keys of the entities on which it depends.

Examiner's Comments

(a)

Many candidates appeared not to read that the number of entity types was restricted to the four that were specified. Candidates seemed to dive in and attempt deriving an ERD without reading the scenario thoroughly. Marks were also lost if a clear and unambiguous statement of assumptions was missing - important if the ERD departed from the expected answer. Other guidance was not taken up, in particular, the supplied entity type and table Tickets illustrated the possibility of forming a ternary relationship (a relationship type involving 3 entity types). It was pleasing to see improvement in the accuracy of expressing cardinality and participation constraints.

(b)

The weaknesses revealed in part (a) affected candidates answers to part (b. Often this resulted in a lot of redundant data and un-normalised tables being produced. Again great care must be

taken in the process of deriving the tables. Working should be shown where necessary but the resultant tables should be in at least 2NF and have primary and foreign keys, otherwise marks are invariably lost or few if any marks can be awarded. Some candidates used the sample tables provided but marks were lost if these tables were significantly inconsistent with the candidate's ERD. Marks were awarded for the correct allocation of foreign keys to these tables but it was sometimes unclear how candidates represented the foreign keys.

A number of candidates did not read the question and created SQL Create table code with data types and constraints and hence lost marks and time as a result.

(c)

This section was reasonably well answered despite the problems revealed in part (b). The process of checking the access path of a query was asked for rather than writing the query in SQL or simply stating the result. Again great care must be taken in reading the question thoroughly to check what is being asked.

(d)

This part was generally answered satisfactorily with most candidates producing stock answers. Full marks for this part were awarded for a good explanation and examples related to the scenario

A2

Examiner's Comments

This was the least popular question attempted by less than half of the candidates with half of those achieving a pass mark. A large number of candidates demonstrated good knowledge of SQL and as a result found it relatively easy to gain high marks either for individual parts or overall. Around a half of candidates found SQL difficult and clearly lacked practical experience of the language and either produced poorly expressed SQL or completely blank answers:

Refer to the tables Figures A2.1 and A2.2 below for this question.

Fig A2.1 transactions Table

TRANSACTIONID	ACCOUNT_ID	TRANSACTION_DATE	AMOUNT
7659897	93008	12/4/2017	3.67
7659898	93008	12/4/2017	12.99
7743433	93008	13/4/2017	-7.99
7935320	331449	13/4/2017	-14.76
8756571	93008	13/4/2017	-5.99

Fig A2.2 accounts Table

ACCOUNT_ID	SORT_CODE	ACCOUNT_TYPE	BALANCE
93008	30-54-87	Direct Debit	362.74
331449	31-12-54	Credit	320.26
57746	30-54-87	On-Line Saver	1295.60
16227	12-32-18	Direct Debit	-550.93

a) List the results of running both the following queries (Query A and Query B) and then describe in a few sentences how these results are produced.

8 marks

Query A:

```
SELECT COUNT(*), account_type
FROM accounts
```

```
WHERE balance < 4000
GROUP BY account_type
HAVING COUNT(*) > 1;
```

Query B:

```
SELECT SUM(AMOUNT), t.account_id, transaction_date
FROM transactions t
WHERE t.account_id IN (SELECT a.account_id
                       FROM accounts a
                       WHERE account_type <> 'On-Line Saver')
GROUP BY t.account_id, transaction_date
ORDER BY SUM(amount) ASC;
```

ANSWER POINTER

Query A Returns this output:

```
COUNT(*)   ACCOUNT_TYPE
--------   ------------------
       2   Direct Debit
```

Query A Uses a `GROUP BY` operator to count the total number of account types that have a balance of less than 4000.00 A `HAVING` clause is used to restrict the types of accounts to those that have more than one account.

Query B produces this output

```
SUM(AMOUNT) ACCOUNT_ID TRANSACTION_DATE
----------- ---------- ----------------
     -14.76     331449 13-APR-17
     -13.98      93008 13-APR-17
      16.66      93008 12-APR-17
```

Query B uses a `GROUP BY` operator to return for each account (in ascending order), the total value of transactions of a particular date and restricted to account types that are not on-line savers.

b) Write an SQL query that produces the same output as query B, but instead uses an INNER JOIN operator.
For guidance, the syntax of an `INNER JOIN` operator is:-

```
INNER JOIN <tablea name> ON <tablea.columna> = <tableb.columna>
```

where `columna` is the matching column in `tablea` and `tableb`

5 marks

ANSWER POINTER

Query B query uses INNER JOIN instead of a subquery producing the same result

```
SELECT SUM(amount), a.account_id, transaction_date
FROM accounts a
INNER JOIN transactions t ON a.account_id = t.account_id
WHERE a.account_type <> 'On_line Saver'
GROUP BY a.account_id, transaction_date
```

```
ORDER BY SUM(amount) ASC;
```

c) Explain the differences between LEFT and RIGHT OUTER JOIN and an INNER JOIN. Illustrate your answer by showing how replacing an INNER JOIN operator with either a LEFT or RIGHT OUTER JOIN operator can affect the output your answer in part b) above.

Hint : You must choose between either a RIGHT or LEFT OUTER JOIN to illustrate the different output produced compared with using an INNER JOIN

6 marks

ANSWER POINTER

A `LEFT OUTER JOIN` returns all the rows from the first table, even if there are no matches in the second table. `RIGHT OUTER JOIN` returns all the rows from the second table, even if there are no matches in the first table. INNER JOIN requires there is at least a match in comparing the two tables. Approx. 3 marks for description 3 marks for applying the relevant example

Choose (in this example) a `LEFT OUTER JOIN` to include details of those accounts that do not have any recorded transactions, thus:

```
SELECT SUM(amount), a.account_id, transaction_date
FROM accounts a
LEFT OUTER JOIN transactions t
ON a.account_id = t.account_id
WHERE a.account_type <> 'On_line Saver'
GROUP BY a.account_id, transaction_date
ORDER BY SUM(amount) ASC;
```

```
 SUM(AMOUNT) ACCOUNT_ID TRANSACTION_DATE
 ----------- ---------- ----------------
      -14.76     331449 13-APR-17
      -13.98      93008 13-APR-17
       16.66      93008 12-APR-17
        null      57746 null
        null      16227 null
```

d) Write an SQL `UPDATE` statement that updates the running total of the balance for account_ID 93008 for transactions made on this account on 13-APR-2017. Following this update the new balance for this account should be 348.76

6 marks

ANSWER POINTER

```
UPDATE accounts a
SET balance = balance + (SELECT SUM(amount)
                         FROM transactions t
                         WHERE transaction_date = '13-APR-2017'
                         AND t.account_id = 93008)
WHERE a.account_id = 93008;
```

The inner `SELECT` calculates the total amount to be added to the balance for the account 93008 (in this case -13.98) and then the Update is applied to the same account using the last `WHERE` clause

BEFORE update

```
ACCOUNT_ID    BALANCE
----------    -------
     93008     362.74
```

AFTER Update

```
ACCOUNT_ID     BALANCE
----------     -------
     93008      348.76
```

Examiner's Comments

Part (a) Generally the best answered part, however some candidates could not work out the result at all and only attempted a very generic explanation of SQL clauses such as GROUP BY.

Part (b). Generally well answered. This was an opportunity to write SQL code. Having the syntax for a JOIN operation helped most candidates in producing a conversion of the subquery.

Part (c) Fairly well answered with good knowledge of the two OUTER JOIN operators, producing mainly stock answers. Example of either RIGHT/LEFT JOIN was another opportunity to write SQL code.
This involved simply replacing the INNER JOIN with an OUTER JOIN The result set reveals the NULLs due to the non-matching values in the column `transaction_date`. However, many candidates failed to apply knowledge of OUTER JOINs into code showing perhaps a lack of practice in using SQL.

Part (d) Most candidates attempted a simpler version of the UPDATE statement that did not **calculate** the balance (using the subquery/SUM) operation.
Most candidates explicitly changed the balance directly using **UPDATE … SET Balance = 348.79** given the condition :

```
transaction_date='13-APR-2017' AND t.account_ID = 93008
```

Clearly the best solution involves calculating the balance dynamically because it changes over time. However on this occasion, credit was given for the alternative solution that most candidates produced.

A3

Examiner's General Comment

This was a very popular question as almost all candidates attempted it. Almost two thirds of the attempts were successful.

(a) A company places orders for items. Each order is placed on a given date, and may include a variety of items in different quantities. The following table shows a sample of orders. The Primary Key is (`OrderNo`, `ItemNo`).

orders

ORDERNO	ITEMNO	ITEMDESCRIPTION	DATE	QUANTITY
1	1	Screw	Jan 6th	100

1	2	Bolt	Jan 6th	50
2	3	Flange	Feb 2nd	10
2	2	Bolt	Feb 2nd	40
2	1	Screw	Feb 2nd	80

(i) Explain why this table is not in 2nd normal form.

2 marks

(ii) Describe two types of anomaly that could be caused by update, insert or delete operations, and give an example of each, with reference to the above table.

4 marks

(iii) Transform the table into 2nd normal form. Show the structures of the resultant tables

4 marks

(b) A company wants to create a database to store records of its employees. For each employee, we record their personal details, the location of the branch where they work and their qualifications. Below is a sample of data for two employees.

EmployeeID: E001	**Name:** Fred Gordon	**JobTitle:** Web Developer	
BranchCode: B04		**BranchAddress:** Delhi	
QualificationID	**Qualification**	**Level**	**Year Obtained**
1	BSc	Undergraduate	2012
3	PhD	Postgraduate	2016

EmployeeID: E002	**Name:** Simon Singh	**JobTitle:** IT Manager	
BranchCode: B01		**BranchAddress:** London	
QualificationID	**Description**	**Level**	**YearObtained**
1	BSc	Undergraduate	2009
2	MSc	Postgraduate	2010

The un-normalised table that corresponds to the above format is as follows:
Employee(**EmployeeID**, Name, JobTitle, BranchCode, BranchName, QualificationID, Description, Level, YearObtained)

(i) Identify the repeating group of attributes and transform the above un-normalised table into tables that are in 1st Normal Form.

5 marks

(ii) Identify any partial dependencies and transform into tables that are in 2nd Normal Form.

5 marks

(iii) Identify any transitive dependencies and transform into tables that are in 3rd Normal Form.

5 marks

ANSWER POINTER

(a)

(i) The table is not in 2NF as it contains 2 partial dependencies
OrderNo → Date
ItemNo → Description

(ii)

Any two from: insertion, deletion and modification anomaly, (1 mark) for name, (1 mark) for example

Insertion Anomaly (type 1)
A new item/description cannot be added to the table until that item has been ordered.

Insertion Anomaly (type 2)
If a new order is added for an existing item and a mistake is made in the description, then the result would be two different descriptions for the same item.

Modification Anomaly
If the description of an item changes, and if it is neglected to change all the occurrences of that description, then the table will contain conflicting values for the description of this item.

Deletion Anomaly
If an order is deleted, then information about the items included in that order will be lost, if these items do not appear in any other order.

(iii)

Order(OrderNo, Date)
Item(ItemNo, Description)
OrderLine(OrderNo, ItemNo, Quantity)

Examiner's Comments

Most candidates managed to correctly identify the partial dependency between ItemNo and Description but failed to identify the dependency between OrderNo and Date. As a result of missing that dependency, many have failed to normalise the table correctly. Also, candidates were, in general, able to give a description of the various types of anomalies, but many struggled to provide a good example relevant to the table used in the question.

(b)

(i)

(QualificationID, Description, Level, Year) is a repeating group.
The following tables are in 1NF:
Employee(EmployeeID, Name, JobTitle, BranchCode, BranchName)
Qualified(EmployeeID*, QualificationID, Description, Level, Year)

(ii)

Partial dependencies: QualificationID → Description, Level
Qualified(EmployeeID*, QualificationID*, Year)
QualLevel(QualificationID, Description, Level)

(iii)

Transitive dependency: EmployeeID→BranchCode→BranchName
The tables given in (ii) are already in 3NF.
Employee(EmployeeID, Name, JobTitle, BranchCode*)

Branch(BranchCode, BranchName)

Examiner's Comments

Most candidates managed to correctly normalise the given scenario. However, in general, candidates find it easier dealing with partial and transitive dependencies than correctly attaining 1st normal form. More generally, it seems that most candidates are more comfortable normalising the initial table up to 3rd normal form in one single step as supposed to showing evidence of a step-by-step approach to normalisation.

Section B

Answer Section B questions in Answer Book B

B4

Examiner's Comments

A popular question that was generally very well answered. Every candidate specified SID as PK and correctly specified the FKs. Most candidates did poorly on candidate keys – despite clearly understanding the concept. The attempts at parts (d) and (e) were more of an average standard – with lower marks than expected. All in all, a strong question for most candidates

Using the following sample relation and your knowledge of declarative constraints :

STUDENT (Title, Name, Gender, Birthday, Address, Email, Mobile, SID, MID, PROJID, PLACID)

- *SID is the student identifier*
- *MID is the identifier of the compulsory assigned staff mentor*
- *PROJID is the identifier of the optional project selected by the student*
- *PLACID is the identifier of the optional industrial placement undertaken by the student*

Address each of the following tasks. **Five marks each**.

(a) Suggest and justify a PRIMARY KEY for this relation.
Write the correct SQL code to implement this.

(b) Suggest and justify any FOREIGN KEYS for this relation.
Write the correct SQL code to implement this.

(c) Suggest and justify all CANDIDATE KEYS for this relation.
Write the correct SQL code to implement this.

(d) Suggest and justify one DOMAIN constraint that enforces the entering of a value in a column.
Write the correct SQL code to implement this.

(e) Suggest and justify one DOMAIN constraint that verifies the entered value in a column.
Write the correct SQL code to implement this.

(25 marks)

ANSWER POINTER

(a) Obviously, `SID`, but could be `Email` or `Mobile`. `CONSTRAINT studpk PRIMARY KEY (SID)`

(b) 3 X Foreign Keys : STAFF, PROJECT & PLACEMENT TABLES.

`CONSTRAINT STUDMENTOR FOREIGN KEY (MID) REFERENCES STAFF (StaffID)`

(c) Candidate keys are SID, Email & Mobile (and other various composites). Make each UNIQUE

(d) Any column that MUST have a value (via `NOT NULL`) – Gender, Name, Birthday etc.) but not optional entries like `PROJID` and `PLACID`.

(e) Examples are Gender (M/F, Male/Female) or Title (Ms/Miss/Mrs/Mr/Dr/Prof/etc.) `CHECK(Gender IN ('M','F'))` etc.

.

B5

Examiner's Comments

This was a popular question that was generally answered to a high standard. Candidates achieved good marks on parts (a), (c) and (e) and achieved a reasonable standard on the other parts. This was a strong question for most candidates.

For each of the following types of database interface or access mechanism, explain their key features, strengths and weaknesses, plus the typical type of user situation where that interface would be best utilized.

(a) A command line terminal.

5 marks

(b) A database utility.

5 marks

(c) A graphical user interface comprising forms and reports.

5 marks

(d) An SQL script.

5 marks

(e) A webpage.

5 marks

ANSWER POINTER

(a) A very basic text-only interface – like a SQL*Plus, Unix or MS-DOS window - that demands high levels of technical knowledge. Used by DBAs and developers to write code and/or administer and monitor the database. End-users could not handle such a bare interface.

(b) Examples could include bulk insertion tools, statistical packages used to harvest query histories for tuning purposes or auditing tools. Demands high levels of technical knowledge. Used by the DBA. and operators

(c) Aimed at the non-technical end-user where knowledge of raw coding is not needed but rather a point-and-click WIMP environment is provided. It shields the user from the technical nitty-gritty. The focus is on the easy manipulation of the data within the database rather than controlling, modifying or managing the database itself.

(d) A file of SQL code (such as a Notepad file with .sql extension) used to perform one or more well-defined (and pre-prepared) tasks. Run by the DBA or developer as it demands strong coding skills.

(e) Like the GUI, the web interface is squarely aimed at the non-technical end-user but it has the distinct advantage of offering database access – albeit hidden behind a webpage – to anyone with an internet connection – not just employees within the company. It is thus

aimed more at external customers, although internal staff – technical or otherwise – can obviously still use it.

B6

Examiner's General Comment

Nearly two out of three candidates attempted this question. Slightly over half of the attempts were successful.

(a) Explain the concepts of **transaction scheduling** and **serialisability**, Describe one type of problem (giving an example) that might appear when a transaction schedule is not serializable.

8 marks

(b) Modern database systems provide a mechanism for enforcing a password policy that has the following features:

- Complexity
- Failed attempts
- Expired passwords
- Password reuse

Describe each of the above features and explain how they help protect the database.

8 marks

(c) Backups of the database should be taken in order to protect data. Describe five features of a good backup strategy.

5 marks

(d) Describe the role, and content, of the data dictionary (metadata) in a DBMS.

4 marks

ANSWER POINTER

(a)

Transaction scheduling (2 marks)

A transaction is a command or a set of commands that access or change the contents of a database. It has a start and finish and takes the database from a consistent state to another. Scheduling of transactions is under the control of a transaction manager and may involve different users submitting transactions that execute concurrently. The transaction manager is responsible for interleaving different transactions which may use the same database items. This interference has to be managed carefully as it may lead to inconsistencies.

Serialisability (2 marks)

If a set of transactions executes concurrently, we say that the schedule is correct if it produces the same results as some equivalent serial execution. The objective of serialisability is to find nonserial schedules that allow transactions to execute concurrently without interfering with one another. The result should be the same as if the database state had been produced by a serial execution. Serialisability identifies those executions of

transactions guaranteed to ensure consistency. It is essential to guarantee serialisability in order to ensure database correctness.

Concurrency problems: Explanation (2 marks) plus example (2 marks) such as:

Lost update problem - Two transactions access the same database items (x) and have their operations interleaved in a way that results in the loss of an update made by one of the interleaved transactions.

Uncommitted dependency problem - Occurs when one transaction can see intermediate results of another transaction before it has committed. This can give rise to cascading rollback.

Inconsistent analysis problem. - Occurs when a transaction reads partial results of other incomplete transactions.

Examiner's Comment

This question was designed to test understanding of some fundamental concepts in databases and, unfortunately, most candidates found it difficult to describe those concepts in their own words. It seems that because the question did not directly refer to the common problems of concurrency by name (e.g., Lost Update), many candidates failed to relate to them.

(b)

- **Complexity**— A policy can be created that identifies the required length and character type combination (number, letter, uppercase, lowercase, symbols) of a password. Ensuring that dictionary words are not used would be an example of a protection method used to protect against password attacks.
- **Failed attempts**—described by the number of failed password attempts. A password that has been attempted too many times without success can be an indication of an intruder. Therefore, it is advised to lock an account that has had too many failed password attempts.
- **Expired passwords**—This component specifies the length of time a user can use a password before being forced to change it. This is to minimize the damage and risk that can be done if a password is breached.
- **Password reuse** —The number of password changes that a user must make before being allowed to reuse a password is specified here. Again this is to minimize the damage and risk that can be done if a password is breached.

Examiner's Comment

Most candidates failed to answer the two aspects requested in the question, namely (i) a description of each feature and (ii) its role in protecting the database. The answers were generally about one or the other and rarely about both.
Also, candidates should avoid providing answers such as: "Password complexity means that we need a complex password", because it does not describe what the term "complex" means in this context.

(c) Features such as (1 mark each)

- Encrypt data in the backup.
- Take multiple copies.
- Create copies on different media (e.g., disks, tapes)
- Store in a site different from the database site.
- Test and validate the backups (trial recoveries).

Examiner's Comments

Almost all candidates made a reasonable attempt at this question and managed to identify good features for the backup strategy.

(d) The data dictionary describes the data in the database (1 mark).
It typically stores: (3 marks)

- names, types, and sizes of data items;
- constraints on the data;
- names of authorized users;
- data items accessible by a user and the type of access;
- usage statistics…

Examiner's Comments

Few candidates were able to give a good description of the content and role of a data dictionary. It is not enough to simply describe the metadata as "data about data" without specifying what this term means in the databases context.