**BCS THE CHARTERED INSTITUTE FOR IT**

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 6 Professional Graduate Diploma in IT

# ADVANCED DATABASE MANAGEMENT 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**

**Examiner's General Comments**

This was a fairly popular question, attempted by around half of the candidates, with two-thirds of the attempts achieving pass level marks. Therefore, the performance was generally quite good with a wide range of marks, some candidates producing near correct solutions.

The definition of a true distributed database system is that it consists of a collection of geographically remote sites of physical databases, connected together via some kind of communication network, in which a user at any site can access data anywhere in the network exactly as if the data were all stored in one single database at the user's own site.

a) Explain the problems associated with achieving a true distributed database in practice.

12 marks

**ANSWER POINTER**

Fragmentation of tables requires maintenance of unique keys across horizontal fragments and consistency of primary keys that are reflected in vertical fragments. Also increased network traffic occurs as many fragments may need to be consulted to perform one query.

Query optimisation would need to be distributed with an optimiser at each site. This could require much network traffic and increased complexity and processing times.

The Data Dictionary Meta data repository must be maintained. If this is distributed, then there would be increased network traffic.

Integrity control for concurrent transactions is required – the necessary lock management results in increased network overhead and difficulties in detecting deadlocks

Recovery of failed transactions at more than one site requires a protocol such as 2-phase commit. Problems can be caused by a crash during the commit phase. Recovery in this situation can be problematic. Rollbacks would need to be co-ordinated across sites.

b) Study the following scenario and then attempt the question parts that follow:

A small Bank has two branches, one located in York and one located in Leeds. Currently the banks centralised database is managed at its HQ in London, where it keeps data about its customer accounts. Other uses include collecting data for the generation of reports and to monitor the services that customers use. Applications have been installed at the two branches to access the centralised database via a fast communication network for accessing the data they need. There is also a communication link between the two branches, which is currently used only when one of the main links to the London headquarters fail.
For simplicity the centralised database system holds a single Customer table (see figure A1 below), where data about customer accounts are kept. The columns of the Customer table are, the account number, the customer's name, the branch where the account is kept and its current balance. The bank has been asked to split its branches so that they can support customers when there is failure of the centralised database or the communication system is slow or breaks down.

Figure 1 Examples of a customer relation held at HQ

| Account_Number | Customer_Name | Branch | Balance |
|---|---|---|---|
| 20012 | Brown | York | 1000.11 |
| 20454 | Patel | Leeds | -250.89 |
| 61344 | Smith | Leeds | 58.99 |
| n75630 | Gray | York | 2956.93 |
| 30435 | Green | Leeds | -33.62 |
| 96567 | Richards | York | 45.76 |
| 01232 | Akhtar | York | 43.92 |

Describe **three** ways to distribute data across the three sites. Comment on the pros and cons of each option.

13 marks

**ANSWER POINTER**

(i) Option 1 The Customer relation is replicated to all three sites. Advantages: The database should always be available to all sites and access to it should be fast even in the case of data communication link failures. Disadvantages: difficult to maintain, update propagation, a lot of storage is required.

(ii) Option 2 There should be no redundancy in the allocation of data
The Customer relation is horizontally partitioned into two fragments, which are stored one at the York branch, and one at the Leeds branch. The London headquarters do not store any data. Fragment York = **s** (Branch="York") and Fragment Leeds =**s** (Branch="Leeds"). Advantages: only local data is stored at a site. No redundancy of data, easier maintenance and security, efficient use of physical storage. Disadvantages: Data becomes unavailable is a site or links to it fail

(iii) Option 3 A reasonable compromise between requirements (i) and (ii). The table is horizontally fragmented and allocated to the York and Leeds branches as above, but a copy of the whole relation is also allocated to the London Headquarters

**Examiner's Comments**

The term distributed database system was overall well understood but database replication (an alternative option) was not well known in general. **Replication** has become the main technology supporting distributed databases because it solves many of the disadvantages of a pure distributed database environment. Replication can improve local database performance and protect the availability of applications because alternate data access options exist. For example, an application may normally access a local database rather than a remote server to minimize network traffic and achieve maximum performance. The application can continue to function if the local server experiences a failure, but other servers with replicated data remain accessible.

**A2**

## Examiner's General Comments

This was not a popular question, attempted by a small set of candidates, with half of the attempts achieving a pass mark. There was a reasonable range in the performance of candidates with some candidates producing near correct solutions.

**This question considers advanced use of SQL in different programming contexts.**

a) The following programming techniques can be thought of as extending the range and functionality of SQL and perform tasks that SQL cannot perform on its own. Describe with the aid of examples how ***each*** programming technique extends the range and functionality of SQL.

i) Stored procedures — 5 marks

ii) Triggers — 5 marks

iii) Embedding SQL in a programming language such as Java/PHP — 5 marks

**ANSWER POINTER**

**Stored procedures** (SPs) can be used to store SQL statements in an executable object stored in the data dictionary. An SP can be written and tested once and can then be accessed by an application. There is no standard of coding SPs across DBMS but all have something like the following common structure:-

```
CREATE or REPLACE PROCEDURE proc_name[parameter list]
AS
[declaration statements
  ...]
BEGIN
  ...  native SQL statements usually DML statements
  ...  block(s) of decision logic such as IF THEN ELSE which extends
  ...  native SQL statements
  ...  ability to use row at a time processing using cursors and loops
[EXCEPTION   (handles error conditions that may be raised in the
executable part of the block.
  ...]
END;
```

An example of a stored procedure is expected following the above general structure, along with the execution call to run it (for example, `EXEC proc_name`). Stored procedures can be compiled and thus might run more efficiently than a set of standalone SQL statements.

**Triggers:** Like a stored procedure, a trigger is a named object that is stored in the data dictionary and can be invoked repeatedly but only via a triggering event. A trigger is created with a `CREATE TRIGGER` statement in which the triggering event is specified in terms of triggering statements and the item (usually a table) on which they act. The timing point is also specified, which determines

whether the trigger fires before or after the triggering statement is executed and whether it fires for each row that the triggering statement affects. A trigger that fires at row level can access the data in the row that is being processed and is able to reference these like local variables so that they can be tested using conditional statements. The old and new values of the row undergoing a change are given different names but generally use keywords such as `OLD`, `NEW`. Triggers can be used to implement business rules and constrain data input. Triggers constrain what transactions can do. A trigger enforces transitional constraints; that is, a trigger only enforces a constraint at the time that the data changes. Triggers are often associated with Business or Enterprise rules such as "the delivery date for an order must be at least seven days from today" and are synonymous with complex constraints within the application logic. Another use of a trigger is to log events made following updates to a table and hence producing an audit trail.
There is no standard coding syntax across DBMSs therefore candidates can use pseudocode or actual code that supports the following general structure. A simple example should reveal the extended SQL elements within the trigger body for example:

```
CREATE OR REPLACE TRIGGER <trigger name>
  BEFORE DELETE OR INSERT OR UPDATE ON <table name>
  (code to Set up cursors to perform row at a time iteration)
  WHEN (NEW.job_id <> 'CEO')  -- condition do not print CEO salary
DECLARE  <variable>
BEGIN
  <Variable> := :NEW.salary  - :OLD.salary;  -- access to OLD/NEW values
  PRINT(:OLD.salary)
  PRINT(:NEW.salary)
 END;
```

**Embedding SQL**: SQL in Java allows the best of both worlds and is the most widely used method of connecting to a database from an application programming interface. The separation of application logic from persistent data is accomplished by creating connections to a database and then exposing the method calls to read or store data. In java, once a connection is opened any database requests are embedded in a method call so that the database can process a so called static prepared SQL statement that can be parsed and restrictions added via parameters. Also, the formulation of the dynamic query can be formed dynamically depending on conditions that are only known at run time. Example of embedded SQL could be

```
Connection db= DriveManager.getconnection(url,loginpassw);
Query = "SELECT * FROM news WHERE source = ?";
PreparedStatement ps = db.preparedStatement(Query);
ps.setString(1, source);
ResultSet rs = ps.executeQuery();
```

b) Another programming technique is to use a native Object to Relational (OR) mapping language (eg LINQ) to access a relational database. Outline the principles behind an OR mapping language and outline the benefits of using this technique compared with a conventional programming technique such as embedding SQL in Java (part b iii)) above. Use examples to assist your answer.

10 marks

**ANSWER POINTER**
This topic has been covered many times when dealing with OO database issues. The current thinking is not to use pure OO databases but to layer existing DBMSs with OO features so that the programmer thinks he is using a virtual persistent object store. This means the use of SQL embedded in the other forms is not required instead the Object Relational mapper automatically translates function calls between the two systems. To maintain the functionality of SQL and to

make queries consistent with classes/methods new query languages have emerged that replicate SQL functionality. One of these is LINQ.

The perceived benefit of LINQ is that the best of both worlds can be obtained in that a persistent but virtual object store is available, along with the rigour and functionality of pure OO programming without side effects (such as having to translate calls and results/recordsets into incompatible types).

A very simple example would be beneficial

```
String sql = "SELECT ... FROM persons WHERE id = 10"
DbCommand cmd = new DbCommand(connection, sql);
Result res = cmd.Execute();
String name = res[0]["FIRST_NAME"];
```

Is equivalent to

```
Person p = repository.GetPerson(10);
String name = p.FirstName;
```

**Examiner's' Comments**

There was a trend to produce descriptive answers with very few examples of code to illustrate the different programming techniques.

There is evidence to suggest that studying Triggers/Stored Procedures/LINQ/JDBC (an important syllabus topic) would provide an opportunity for candidates to gain important key practical skills in database development and advanced SQL techniques in order to answer this question better.

**A3**

**Examiner's General Comment:**

Almost all candidates attempted this question. Well over three quarters of the attempts achieved a pass.

(a) Consider the following tables:

```
Film (filmNbr, title, year)
Director (directID, name)
Directing (directID, filmNbr)
```

And the following query:

```
SELECT Film.title
FROM  Film, Director, Directing
WHERE Film.filmNbr = Directing.filmNbr
AND Director.directID = Directing.directID
AND Director.name = 'Lucas'
AND Film.year = 2015;
```

(i) Draw a query tree that corresponds to the most efficient way of processing this query.

10 Marks

(ii) Assume there is a B-Tree index on the column “title” of the table “Film”. For each of the following queries, explain how this index could be used when

executing each query:

```
SELECT * FROM Film WHERE title = 'Up';
SELECT * FROM Film ORDER BY title;
SELECT COUNT(title) FROM Film;
```

6 marks

b) The security of a database should be built-into the database development lifecycle. For each of the following database development stages, describe how security can be considered:

i) Database Planning Stage

2 Marks

ii) Database Requirements Stage

2 Marks

iii) Database Design Stage

2 Marks

(c) A medical surgery keeps patients' records in a database server located in a dedicated room. These records can be accessed remotely by the staff working at the surgery (doctors and office staff).

(i) Describe an example of physical security control for protecting this database.

1 Mark

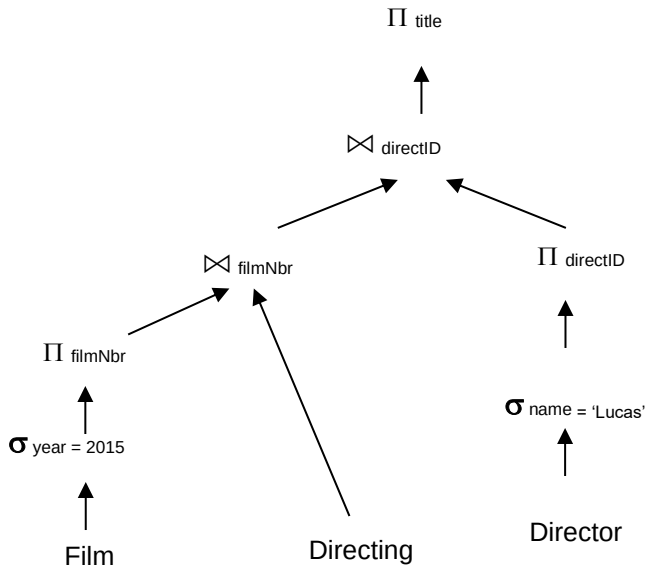
(ii) Describe one way of preventing the office staff from viewing sensitive medical data about patients that should be viewed only by doctors.

2 Marks

**ANSWER POINTER**

(a)

(i)




ii) (using Oracle as an example)

- Query 1: The index will be searched for the title "Up" using an "Index Range Scan". Once this is found, it will be followed by a "Table Access by Index Rowid" in order to fetch the corresponding record.

- Query 2: The index will be used to avoid a separate sorting step (because it is already ordered). The index can be searched using a Full Index Scan, then each entry in the index can point to the corresponding record in the table using a "Table Access by Index Rowid".
- Query 3: All the data needed to execute this query is stored in the index, so there is no need to access the table: A "Index Fast Full Scan" will be performed.

**Examiner's Comment:**

Most students managed to produce an optimised query tree. In part (ii) however, students were not able to show understanding of the use of indexes in relation to query execution.

b)

i) Database Planning Stage: identify which data is sensitive/needs protection; identify the different types of users and their access level

ii) Database Requirements Stage:
functional requirements: which data is to be accessed by which authorised users; which data is to be searched securely…
non-functional requirements: data availability and reliability, security controls; compliance and legal issues.

iii) Database Design Stage: user views, entity and referential integrity rules, business rules, normalisation.

**Examiner's Comments**

Although most candidates can relate to the database development lifecycle and its common steps, not many were able to describe the security considerations involved in each step.

c)

i) Any sensible answer such as
- Securing access to the server room (key, swipe card…)
- Taking regular backups and securing their storage.

ii) Any sensible answer such as
- Creating database views for doctors and office staff and granting access to the views instead of the base tables. These views will hide the columns where sensitive data is stored.
- Implement a VPD (Virtual Private Database) policy to mask columns containing sensitive data. The policy will be applied to all office staff.

**Examiner's Comments**

Almost all candidates provided a sensible answer to part (i) but not to part (ii). It is important for candidates to understand the database mechanisms that can provide segregated access to data. It is not enough to state that different users will have different logins and passwords or different roles and privileges.

## Section B

## Answer Section B questions in Answer Book B

**B4**

Using your own simple examples and suitable diagrams, explain the following transaction-processing concepts.

(a) Guaranteed data consistency via ACID principles.

5 marks

(b) Eventual data consistency via BASE principles.

5 marks

(c) Schedules, serializability and isolation.

5 marks

(d) COMMIT, ROLLBACK, SAVEPOINT and staged or cascaded variants.

5 marks

(e) Locking levels, types and philosophies (optimistic versus pessimistic).

5 marks

(a) ACID: as the fundamental framework for transaction-processing, specifying atomicity, consistency, isolation and durability - each aspect must be named and explained in detail. Bonus marks for a clear worked example or well annotated diagram. The fact that checks are done up-front.

(b) BASE (**B**asically **A**vailable, **S**oft state, **E**ventual consistency) – as used in NoSQL databases - offers a concept of "eventual consistency" in which database changes are propagated to all nodes "eventually". Faster but riskier. Failed transactions must be aborted and re-run thus incurring additional overhead.

(c) A transaction schedule is **serializable** if its outcome (the resulting database state) is equal to the outcome of its transactions executed serially (sequentially without overlapping in time). Transactions are normally executed concurrently (they overlap), since this is the most efficient way. Serializability is the major correctness criterion for concurrent transactions' executions. It is considered the highest level of isolation between transactions.

(d) COMMIT: as a saving operation (of the complete transaction) – as opposed to ROLLBACK – see below. Candidates should then go on to describe the need for the two-phase commit in a distributed database environment and how it constitutes the 'voting' phase (the 'are you ready to commit' message) and the 'decision' phase (the 'commit now' message). Students should also cover the concepts of *global transactions*, *localized sub-transactions*, *transaction coordinator site* (the transaction initiator site), *participant sites*, the need to pass *messages* between sites, the use of *timeouts* to avoid unnecessary blocks or delays, the possibility of a participant site issuing a *local abort* – thus forcing a *global abort* to be issued and the need for unanimous *local commits* before a *global commit* is circulated – all ensuring the *data integrity* and *ACID rules* are satisfied. Bonus marks for a clear worked example or well annotated diagram. ROLLBACK/CASCADED ROLLBACK: ROLLBACK to undo the whole transaction. Cascaded Rollback - when transaction $T_1$ fails and induces a rollback which in turn causes other transactions - which were dependent on the success of $T_1$ – to likewise fail and be rolled back. Bonus marks for a clear worked example or well annotated diagram. SAVEPOINTS as transaction partitioning concepts whereby a large transaction can be sub-divided into smaller units using embedded labels and how the DBMS can roll back to a named savepoint rather than undoing the whole transaction

(e) Locking: Optimistic locking – based on the assumption that inter-transaction conflict is rare so individual transactions can proceed unsynchronized and are only checked for conflicts at the end – just before commit. Useful in low data contention environments because it avoids the overhead of locking/waiting for unlocking but inefficient if data conflicts are common as transactions will have to be repeatedly restarted. Pessimistic locking – assumes a high degree of inter-transaction conflict and locks up all data resources ahead of access immediately – even if other transactions never try and access them. Saves re-running transactions in highly contentious environments but this 'belt and braces' approach can induce overhead in locking/releasing data resources that were never in conflict in the first

place. Locking can be S-locks (shared) or X-locks (exclusive) and be at row-level or table-level.

## Examiners Comments

A very popular question answered extremely well by almost every candidate who attempted it. Good descriptions/explanations with good supporting examples.

**B5**

Using your own simple examples and suitable diagrams, explain the following data modelling concepts.

(a) Entity Relationship Diagrams (ERD), Entity Types, Entity Instantiations and Identifiers
5 marks

(b) Enhanced Entity Relationship Diagram (EERD), Super-Types and Sub-Types
5 marks

(c) Data Cubes & OLAP
5 marks

(d) Star Schemas and the role of normalization/de-normalization
5 marks

(e) Snowflake Schemas and the role of normalization/de-normalization
5 marks

**ANSWER POINTER**

(a) The discussion should include:
ERD is based on entities (logical or physical items of interest within a domain of discourse) and relationships (logical connections between entities). Differentiation of entity (and relationship) types and entity (and relationship) instantiations. Identifiers used to target instantiations.

(b) EERD: The concept of super-types and sub-types – as used on the EERD - should be explained – for example super-type STUDENT having sub-types UNDERGRADUATE AND POSTGRADUATE. Suitable comments on attributes and the use of primary keys/identifiers would gain credit. Good examples/diagrams are expected.

(c) Data cube dimensions, the concept of a multidimensional dataset - given that data can have an arbitrary number of dimensions. The term 'hypercube' is sometimes used, especially for data with more than three dimensions. The definition of OLAP.

(d) Star Schema: a specialized example of an ER Model with the use of a *fact table* (with composite primary key) and a set of *dimension tables* (each with an atomic primary key) related to the fact table via foreign keys – thus producing a *star schema* (star join) model.

Candidates should then go on to discuss issues such as:

the fact table being much larger than the dimension tables

the fact table works best when the 'facts' recorded are numeric (grades, prices, ages etc.) thus allowing aggregated computations to be run leading to summarized data

that dimension tables tend to hold more descriptive data (names, addresses, identifiers)

the use of de-normalized data to replicate attributes across multiple dimension tables (for example, storing address or contact data in several different dimension tables) to

avoid additional joins and enhance query performance.

Accurate examples/diagrams are expected.

(e) Snowflake Schema: an extension of the Star Schema where dimensions can have their own dimensions – caused by normalizing the original dimension table data into two or more child dimension tables, all linked to the 'parent' dimension table via the familiar PK/FK technique. So *star schemas* use de-normalized (repeated) data and *snowflake schemas* use normalized (minimized duplication) data. Good examples/diagrams expected.

## Examiner's Comments

Another very popular question, again answered extremely well by the vast majority of candidates who attempted it. Good descriptions/explanations with good supporting examples and for most, excellent annotated models and diagrams.