BCS THE CHARTERED INSTITUTE FOR IT

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

ADVANCED DATABASE MANAGEMENT SYSTEMS

September 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

a) The following 'data models' are alternatives to the Relational data model :-

Object oriented
Document oriented
Geographical or spatial

i) Briefly explain the meaning of the term 'data model' given the above context.

3 marks

ii) With the aid of examples, explain how each of the above data models differ from the relational model.

12 marks

b) Recently there has been a lot of interest in alternatives to relational database systems known colloquially as 'NoSQL' databases and often referred to as 'Not Only SQL'. These alternatives have emerged to handle the massive growth of data - so called Big Data. Examples of Big Data applications include Facebook (and social media in general) and Google Analytics (Intelligent Product marketing).

With reference to the above statement, discuss the main requirements and challenges of developing and running Big Data applications.

10 marks

ANSWER POINTER

i) A data model could be defined as an abstraction of reality - a model of some underlying data structure with real world meaning. For example, a relation could be a model of a table of employee data. Alternative non-relational database systems have been designed to incorporate specific representations of modelling constructs more aligned to either the programming paradigm (for example, OO) or to support operations that specifically deal with non-relational data structures such as documents and geospatial data.

ii) Object oriented
Examples should be provided of an OO model plus some explanation.
The Object Oriented data model may exist as memory classes that map to relations (using an OR mapper) or as pure object stores that represent persistent data as classes. Classes are different from relations in that they encapsulate methods and properties that are used to instantiate objects. One way of mapping OO to a relational model is to extend ER models to incorporate Generalisation/Specialisation, whole or part, however, inheritance associations are not implemented in the same way.

Document oriented
Document oriented data models require a more flexible semi-structured data model. This is why XML /JSON are the principal approaches. A simple XML or JSON example, with xml doc and XSLT stylesheet would gain credit.
Comparison is as follows:
A document-oriented database contains records that describe the data in the document, as well as the actual data. Documents can be quite complex data structures and can contain nested data to provide additional sub-categories of information about the document. One or more documents can also be used to represent a real-world object. There is no concept of a schema as in a relational database.

Geographical/spatial
Spatial data models differ from relational models in the way they add a spatial dimension to ordinary data. A person could be a relational object (a row in a table) but with added functionality to keep track of movements in multi-dimensional space. Some of these systems store maps of a 2-D dimensional description of their objects. They require a data model that is optimised to represent objects and with a 2-D geospatial relationship. They have to model geographic objects or surfaces using representations either as vectors (points, lines or polygons) or raster (bit maps that store collections of bit patterns that form the geographic object).

b)

Massive volume
Not suited to conventional OLTP more suited to OLAP Solution flexible structures non-relations stored and viewed in text format throughout.

Performance
The overheads of logging to enable transaction recovery and integrity (for example row locking) may be relaxed under appropriate circumstances. Data is de-normalised/flattened without adherence to a conventional data model. Instead, programs build structure in memory as classes.

Availability
Most datasets are partitioned across many servers. This allows managed replication across multiple servers along with parallel processing.

Query optimisation :
Queries are mainly OLAP in nature set against Cubes. Therefore, the query traffic is more unpredictable. Declarative style SQL, pattern matching and use of external statistical/analytical routines that perform 'what if' queries, are also likely to be executed.

(The language called R has this capability)

Richer data models
Models that encompass some if not all three alternatives need to be handled. This would tend to require more work to be done at the program level through memory based data structures that map to the simple persistent data structures. Processing normally involves columns of aggregated data rather than rows. This is similar to processing star schemas in which the rules of normalisation are often relaxed.

High level of user interaction is required to support streaming of data

Legacy data solutions need to have algorithms that extract, transform and load data into formats that support merging different data sets across different platforms. (This capability can be provided by ETL tools) The different forms of data are not necessarily structured as in relational database systems.

Examiners' Comments
This was not a popular question with few candidates attempting it. The average mark was higher than averages for other questions with the vast majority of candidates achieving pass level marks. The first part covered data modelling and in particular alternatives to the relational data model. The evidence shows that this part was generally well answered. The second part was more topical and covered *new database applications and architectures*, referenced as 3.1 in the syllabus. It is important that candidates keep up to date with these ideas and in particular the subject matter of the question - the emergence of the so called NoSQL databases that deal with Big Data. Again, given this topic is fairly new, performance was generally good from the small number of candidates. The best answers were from candidates who structured their answers into distinct paragraphs. Each paragraph citing the challenges and then describing the requirements needed to address a particular challenge.

A2

a) Explain the distinction between 'Semi-structured data' as exhibited by data description languages such as XML and JSON and 'Structured data' as exhibited by the relational model.

4 marks

ANSWER POINTER

Semi-structured data is data that may be irregular or incomplete and has a structure that may change rapidly or unpredictably. It generally has some structure, but does not conform to a fixed schema
"Schema-less" and self-describing, in that the data carries information about its own schema. This information could be in the form of XML element tags).
Structured data is data that has a regular structure and is tightly constrained by the rigour of an underlying data model – for example the relational model. Data is strongly typed and associated with a schema or data dictionary that imposes further control on the data input.

b) Outline the advantages of using semi-structured data over structured data typified by the relational model of data.

5 marks

ANSWER POINTER

Programmers who want to store semi-structured objects such as documents for their database applications do not need to worry about object-relational impedance mismatch but can often serialise objects via a light-weight library that easily stores/translates objects as raw text.
Support for nested or hierarchical data often simplifies data models representing complex relationships between entities.
Support for lists of objects simplifies data models by avoiding complex translations of lists into a relational data model.

c) Consider the XML document and XSL stylesheet code listings in Figures A2.1 and A2.2 below.

i) Show the result of running mystylesheet.xsl on borrowers.xml.

5 marks

ANSWER POINTER

```
<html>
<body>
Michael, Hanson: 1000<br/>
Michael, Hanson: 500<br/>
</body>
</html>
```

Because only Michael Hanson has loans

ii) Write an XPath expression that returns all the names of borrowers, who have at least one loan.

3 marks

ANSWER POINTER

```
        //loaner[loan]/name
```

the expression should also work on other similar XML documents

d) Refer to the XQuery expression in Figure A2.3 and borrowers.xml in Figure A2.1 below.
i) Describe the function of the code.

4 Marks

ii) Compare and contrast XQuery and XPath

4 marks

Figure A2.1: XML document borrowers.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="mystylesheet.xsl" type="text/xsl"?>
<loans>
<borrower>
   <name>
   <first>Michael</first>
   <last>Hansen</last>
   </name>
   <address>Freisha Way, 23456 Copenhagen, Denmark</address>
   <loan>
```

```
          <amount>1000</amount>
          <payout-date>2016-01-01</payout-date>
          <repayment amount="100" date="2017-01-01"/>
          <repayment amount="100" date="2018-01-01"/>
     </loan>
     <loan>
          <amount>500</amount>
          <payout-date>2017-01-01</payout-date>
          <repayment amount="100" date="2015-01-01"/>
     </loan>
</borrower>
<borrower>
     <name>
          <first>Mansha</first>
          <last>Daleep</last>
     </name>
     <address>Costoria Street 8232, 98764 Mumbai, India</address>
</borrower>
</loans>
```

Figure A2.2 XSL stylesheet, mystylesheet.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="loans">
     <html>
     <body>
          <xsl:apply-templates select="//loan"/>
     </body>
     </html>
</xsl:template>
<xsl:template match="loan">
<xsl:apply-templates select="../name/last"/>,
<xsl:apply-templates select="../name/first"/>:
<xsl:apply-templates select="amount"/><br/>
</xsl:template>
</xsl:stylesheet>
```

Figure A2.3 Query expression

```
<borrowers>
{
for $x in doc("borrowers.xml")//borrower
return <borrower>
{$x/name}
<debt> {fn:sum($x/loan/amount) - fn:sum($x/loan/repayment/@amount)}
</debt>
</borrower>
}
</borrowers>
```

ANSWER POINTER part d)

i) For each borrower in `borrowers.xml` compute the total amount by summing the numbers in the amount elements and subtracting the sum of the numbers in the repayment attribute of the repayment elements. The output must be valid XML that for each borrower states their name and the outstanding amount (in a debt element).

ii) XPath allows traversal of the structure, such as the code in part c) :
XQuery is an XML query language that makes use of XPath to query XML structures.
However, it also allows for functions to be defined and called, as well as complex querying of data structures using FLWOR expressions. FLWOR allows for join functionality between data sets defined in XML.

Examiners' Comments
This was an unpopular question with very few candidates attempting it. Performance on this question was similar to that on previous years' questions that addressed this subject matter.
The evidence shows that parts a) and b) were generally well attempted with most candidates producing rather stock answers. The main problem was that explanations needed illustrative examples of semi-structured data compared to the equivalent structured data. This could be JSON code or XML with the former now becoming more popular and accessible.

A3

Examiner's General Comment:

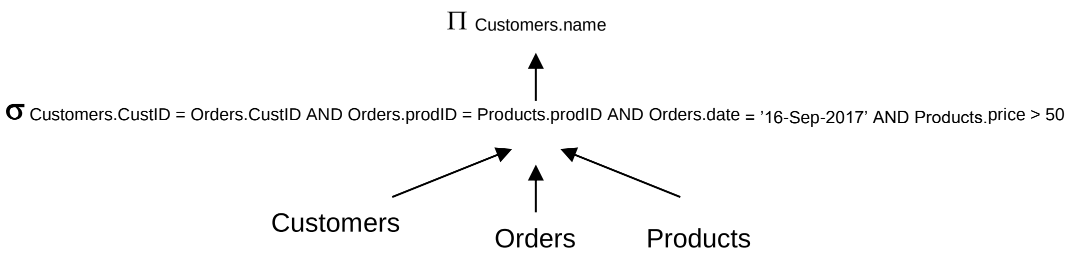
This was a very popular question with candidates. The pass rate was 84.7%, with an average mark of 16.8 and a standard deviation of 5.7.

(a) Given the following three linked tables:

```
Customers (custID, name, country)
Products (prodID, price)
Orders (orderID, custID*, prodID*, date)
```

Suppose we have the following query and its corresponding initial parse tree:

```
SELECT Customers.name
FROM Customers, Orders, Products
WHERE Customers.custID = Orders.custID
AND    Orders.prodID = Products.prodID
AND      Orders.date = '16-Sep-2017'
AND      Products.price > 50;
```


Π Customers.name
σ Customers.CustID = Orders.CustID AND Orders.prodID = Products.prodID AND Orders.date = '16-Sep-2017' AND Products.price > 50
Customers
Orders
Products

(i) What problems arise if the query is executed based on the above parse tree?

2 marks

(ii) Transform the above parse tree into one that corresponds to the most efficient way of processing the query.

11 marks

(iii) Which indexes could be created to potentially further enhance the execution of this query?

2 marks

(b) A company employee has been using the password “APPLE” for the past six months to access a database. Discuss why this poses a security risk and suggest ways in which the company could improve password management.

10 marks

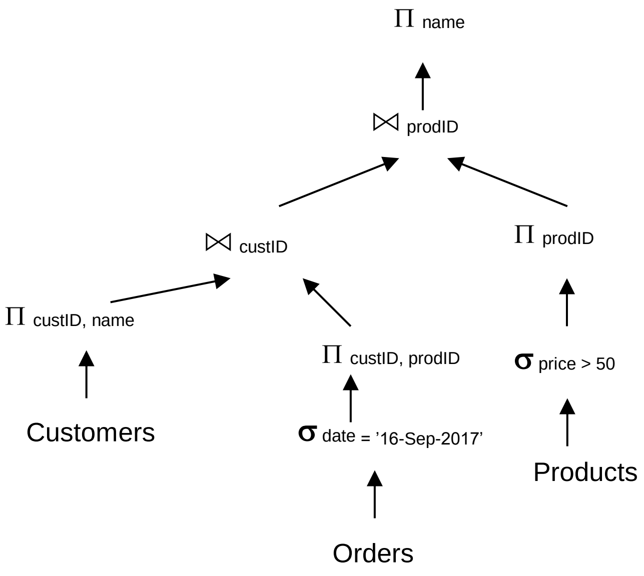
ANSWER POINTER

(a)

(i) By joining the tables first, large intermediate tables may be created which may be too large to fit into the main memory and would require expensive reading and writing to secondary memory. This would increase processing time.

2 marks

(ii) A more efficient parse tree could be:



(iii) Attributes that could theoretically be considered are: the foreign keys `custID` and `prodID` in the `Orders` table (to speed up the joins) (1 mark) and the `date` and `price` attributes (1 mark).

Examiner’s Comment:

Almost all candidates managed to produce an optimised query tree. In part (iii) however, there is evidence that candidates were not able to demonstrate the use of indexes in relation to query execution**.**

ANSWER POINTER

(b) The password is not secure because it is a simple dictionary word and has been used for so long that a malicious user could have cracked it

The company needs to create a robust password policy that includes features such as;

- **Complexity**— imposing a required length and character type combination (e.g., number, letter, uppercase, lowercase, symbols) on a password.
- **Expired passwords**—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** — specifying the number of password changes that a user must make before being allowed to reuse a password. Again, this is to minimise the damage and risk that can be done if a password is breached.
– **Failed attempts**— imposing an upper limit on the number of failed password attempts. A password that has been attempted too many times without avail can be an indication of an intruder. Therefore, it is best to lock an account that has had too many failed password attempts.

Examiner's Comment

All candidates recognised the features that characterise a good password policy.

Section B

Answer Section B questions in Answer Book B

B4

Using your own simple examples and suitable diagrams, explain the following data warehousing and big data concepts. **Five marks each.**

(a) Data Cleansing

5 marks

(b) Indexing & Optimization

5 marks

(c) Aggregation & Summarized Data

5 marks

(d) Materialized Views

5 marks

(e) Roll-up, drill-down and slicing & dicing

5 marks

ANSWER POINTER

(a) Data cleansing involves the transformation of data into a consistent format. This action could require the selection/rejection, re-formatting, ordering and (dis)aggregation of data into a consistent structure and regular format. Decisions may need to be made on what to do with missing (NULL) entries, the standardisation to a particular format (for example 'M/F', 'Male/Female' for Gender), how many fields to use for the Address data or application of business rules.

Examiner's Comment

Generally answered very well with most candidates grasping the key ideas and then going into the ETL process in some depth.

ANSWER POINTER

(b) Indexing: The primary (but not only) method for query optimization. It can include: Bitmap indexes to support queries based on low cardinality data and more 'open' type of queries rather than the usual B-tree indexes used to search for individual identifiers. Advanced join methods - such as in memory hash joins and partition-wise joins – very useful for the large data volumes involved in a Data Warehouse. Advanced optimizers - that can use statistics and histograms - especially on skewed data distributions. (An example of which is Oracle's cost-based optimizer and its ability to optimise star schema queries). Function-based indexes - that can pre-index complex calculations and other expressions often used in Data Warehouse applications. Sampling functions that can gather mathematical data like averages without having to read every row.

Examiner's Comment

Less well done than part (a). Many covered indexing well enough but the evidence shows that the optimization aspect was generally poorly explained with vague, unfocused responses. Very few mentioned the points here in the marking scheme.

ANSWER POINTER

(c) Aggregation: data warehouse queries are often seeking aggregated data, not the fine detail of individual rows, but aggregated data aligned to specific dimensions (month, product, region etc.) so the DBMS must support pre-computed summaries and aggregates to avoid run-time computation. Often used with *analytical functions* – many Business Intelligence and DW applications need to use SQL ranking and aggregate functions, cumulative aggregates or maybe the CUBE and ROLLUP operators for OLAP analysis.

Examiner's Comment

This was a strong section for most candidates with just about all grasping the key ideas and many providing good examples. Most mentioned AVG, SUM etc. but fewer went on to discuss CUBE or ROLLUP.

ANSWER POINTER

(d) A materialized view is an object that contains the results of a query (like a normal view) but this view is 'materialized' (made concrete) and is a form of 'pre-computation'. This is done for performance reasons in data warehouses where the volumes of data are so large that computing/building a normal view 'on the fly' is not feasible in terms of the time needed. Hence, the desire to 'pre-compute' the view and store it. Materialized views which store data based on remote tables are also known as snapshots. The down side is that once materialized, the view now occupies its own space and over time, if not re-built, will become increasingly out of date as the main base tables from which it is constructed change. Hence a refresh strategy must be developed to prevent the view from going stale.

Examiner's Comment

The evidence shows that the majority of candidates understood what a 'view' meant but they were less successful in explaining what a 'materialised' view was. Some of the better responses did venture onto discussing snapshots.

ANSWER POINTER

(e) Roll-up involves the consolidation of data that can be accumulated and computed in one or more dimensions. For example, all sales offices are rolled up to the sales department or sales division to anticipate sales trends. By contrast, the drill-down is a technique that allows users to navigate through the details. For instance, users can view the sales by individual products that make up a region's sales. Slicing and dicing is a feature whereby users can take out (slicing) a specific set of data of the OLAP cube and view (dicing) the slices from different viewpoints. These viewpoints are sometimes called dimensions (such as looking at the same set of sales data by salesperson or by date or by customer or by product or by region).

Examiner's Comment

Most candidates provided a well-annotated diagram of a data cube and clearly understood the key concepts of rolling-up, drilling-down and ‘slicing & dicing’ that cube –a good sub-question for most.

B5

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

(a) Two-phase locking
5 marks

(b) Two-phase commit
5 marks

(c) Cascaded rollback
5 marks

(d) Wait-for graphs
5 marks

(e) Checkpoints
5 marks

(a) Within the Two-Phase Locking (2PL) protocol, locks are applied and removed in two distinct phases during the transaction's execution:

1. Expanding phase: locks are acquired and no locks are released.
2. Shrinking phase: locks are released and no locks are acquired.

Two types of locks are: *Shared* and *Exclusive* locks. Using locks that block processes, 2PL may give rise to deadlocks that result from the mutual blocking of two or more transactions.

Examiner's Comment

Probably the best section in Question 5 with most candidates producing good-quality diagrams and strong supporting explanations of the two phases mentioned in the marking scheme.

(b) The 2PC protocol consists of two distinct phases:

1. The *commit-request phase* (or *voting phase*), in which a *coordinator* process attempts to prepare all the transaction's participating processes to take the necessary steps for either committing or aborting the transaction and to *vote*, either "Yes": commit (if the transaction participant's local execution has ended properly), or "No": abort (if a problem has been detected with the local portion), and
2. The *commit phase*, in which, based on *voting* of the processes, the coordinator decides whether to commit (only if *all* have voted "Yes") or abort the transaction (otherwise), and notifies the result to all the processes. The local processes must then follow with the required actions (commit or abort).

Examiner's Comment

Another strong question – most candidates clearly explained these concepts.

(c) Rollback is an operation which returns the database to some previous state. The ability to rollback is important for database integrity because it allows the database to be restored to a clean copy even after erroneous operations are performed. It is also crucial for recovering from database server crashes. Transactions which were active at the time of the crash can be rolled back in order to recover the database to a consistent state. A *cascading rollback* occurs in database systems when a transaction (T1) causes a failure and hence requires a rollback to be performed. If other transactions have been allowed to see or use the effects of the changes brought about by T1 before its failure, they must also be rolled back, thus causing a cascading effect. That is, the failure of one transaction can cause many to fail. This is why transactions should be able to see only committed data that cannot be rolled back.

Examiner's Comment

Generally, well done but some responses did not provide enough detail.

(d) A wait-for graph is used for deadlock detection & recovery from such a situation by competing transactions. The wait-for graph is used to track which other processes a particular process is currently blocking. In a wait-for graph, processes are represented as nodes, and an edge from process T1 to T2 implies T1 is holding a resource that T2 needs and thus is waiting for it release its lock on that resource.

Examiner's Comment

Again, mostly good responses with clear diagrams.

(e) The purpose of checkpointing is to provide a snapshot of the data within the database. A checkpoint is any identifier or other reference that identifies the state of the database at a point in time. Modifications to database pages are performed in memory and are not necessarily written to disk after every update. Therefore, periodically, the database system must perform a checkpoint to write these updates which are held in memory to the storage disk. Writing these updates to storage disk creates a point in time in which the database system can apply changes contained in a transaction log in order to recover after an unexpected shut down or crash of the database system. If a checkpoint is interrupted and a recovery is required, then the database system must start recovery from a previous successful checkpoint, processing all transaction logs created since that point in time.

Examiner's Comment

Yet another good sub-question with most candidates proving clear answers, supported by well-annotated diagrams.