THE BCS PROFESSIONAL EXAMINATION
Professional Graduate Diploma

April 2001

EXAMINERS' REPORT

Advanced Database Management Systems

QUESTION ONE.

One of the problems in distributed database system management is deadlock detection.

Consider the following strategy for building a wait-for graph in a centralised system.

If transaction Ti is waiting for a data item held by Tj , make Ti and Tj nodes in the wait-for graph and draw an edge from Ti to Tj

a) Explain why deadlock detection is a problem in distributed systems.
(5 marks)

b) Explain how the strategy stated above can be used to detect deadlock in centralised systems.
(4 marks)

c) State why the given strategy is not suitable for distributed systems and discuss how it might be amended and used to detect deadlock in such systems. Provide outline algorithms for deadlock detection for two distributed database system topologies.
(16 marks)

Answer Pointers

a) Problem because of no centralised system to oversee which transactions are waiting for which data. A local site would know only that a global transaction is slow in completing or waiting for data. Would not know that deadlock had occurred because would be unaware of what is happening at other sites.
(5 marks)

b) The system draws a graph to include all waiting transactions. If there is a cycle in the graph it means that two transactions are (either directly or indirectly) waiting for each other. Hence deadlock has occurred.
(4 marks)

c) It is not suitable for distributed systems because a local site which initiates the transaction would know only that a transaction is waiting for data held at another site. It would not know that what relationships exist between transactions at other sites and how these might effect this transaction.

Would need more information to be held on the local wait-for graphs and a mechanism for combining local graphs to get a global picture.

The local graphs might be extended to show what local data is held by each transaction active at the site and also what data has been requested for other sites.

Need a global deadlock detection strategy. The strategy will involve combining local graphs.

(6 marks)

For instance, in a ring network:

If a local site suspects deadlock it send its graph to next site on the ring.

repeat until return to local initiating site or deadlock detected

The receiving site combines the graph received with its own

If there is a cycle therein, deadlock has been detected, the graph is sent to sites involved and transactions involved are rolled back.

If there is no cycle, the receiving site sends the combined graph to the next site on the ring.

If no deadlock has been detected the original site continues as normal

In a tree network:

If a local site suspects deadlock

```
it requests the wait-for graphs for each of its descendents
it combines these
if deadlock is not detected
    repeat until deadlock detected or global graph constructed
        send graph to parent
        parent requests graphs from each of its descendent
        parent combines these
if deadlock detected roll back transactions
```

(10 marks - 5 marks for each topology)

Examiners Guidance Notes

The first two parts of this question were answered reasonably well. Some candidates strayed off track in the third part. A common mistake was a discussion of methods of deadlock prevention rather than algorithms for deadlock detection.

QUESTION TWO.

It is usually worth a database management system expending considerable effort in determining the best method of executing a query before actually executing the query. Discuss the role and process of query optimisation in database management systems.

(25 marks)

Answer Pointers

This question is about query optimisation. As there are usually many ways to execute a given user query and some methods are far more expensive than others in processing terms, it is usually worth any DBMS having a query optimisation subsystem.

Candidates should describe the process of query optimisation and execution and should give examples of how alternative strategies can greatly affect the number of disk accesses (usually the main bottleneck) in executing a particular query.

Points around which discussion/ illustration could be based:

- Converting a user query (e.g. SQL) into an internal representation (e.g. relational algebra). Equivalence of expressions. Implied processing patterns.
- Keeping statistics in order to determine size of various results obtained during the processing of a query.
- Use of indexes to speed up query processing. Making use of blocks and physical ordering. Sorting
- Joins are the most expensive operation. Making use of different join strategies.
- Parallel processing. Disk stripping.

Marks Breakdown:

Basic knowledge of role and stages of optimisation	5
Equivalence of expressions in SQL and internal query representation	4
Use of statistics	4
File organisation and indexes	4
Algorithms	4
Parallel Processing Approaches	4
	(25 marks)

Examiners Guidance Notes

A common mistake in this question was for candidates to assume the context of a distributed system. Many answers given assumed only distributed systems. This was not the intention of the question. Optimisation applies to both centralised and distributed systems.

QUESTION THREE.

Explain how one might judge whether a database management system was based on the relational database model.

(25 marks)

Answer Pointers

There are a number of possible approaches to answering this question. Candidates could choose to quote Codd's twelve rules with adequate explanation. A better answer might be to define what is meant by a relation, define the three main integrity constraints: domain, entity and referential integrity, and outline the necessary relational algebra operations.

Examiners Guidance Notes

A popular question, which in general was answered well. Many candidates were able to quote Codd's twelve rules but few were able to explain them. In general the clearer answers were those that followed the second approach outlined in the answer pointers.

QUESTION FOUR.

Discuss the features that should be supported by a temporal database management system.

(25 marks)

Answer Pointers

The answer to this question depends on which temporal database paradigm you follow. One answer might be:

Support for intervals and points
Scalar operators including BEFORE, MEETS, OVERLAPS, DURING, STARTS, FINISHES
Possibly MERGES, CONTAINS
Aggregate operators: UNFOLD and COALESCE

Examiners Guidance Notes:

Not a popular question, suggesting that this is not a topic that appears on many courses. Most of the answers submitted did not receive any credit because candidates had read *temporary* instead of *temporal*. Most candidates who did read the question correctly and chose to answer it, submitted answers similar to the outline given above.

QUESTION FIVE.

Compare and contrast features and origins of object-oriented versus object-relational database systems.

(25 marks)

Answer Pointers

From a user perspective the technologies are converging. Object-oriented database systems came mainly from the software engineering field, although database researchers were also working towards database models more capable of representing complex data items and processes. The software engineering community developed abstract data types. It was recognised that persistence was needed within application areas that used complex types and hence object-oriented databases came about. A sector of the database community, which did not want to lose the investment made in relational systems, developed the object -relational approach as an extension of the relational approach. Thus a relational model can be used with object-oriented extensions where needed.

The concept of object-relational captures relational as well as relational plus objects. Object-Relational databases are an evolution of relational databases. They allow the expression of most OODBMS concepts. The data model used by ORDBMSs is also called the SQL-3 Data Model as it is defined by the Data Definition Language (DDL) of SQL-3.It is compatible with the relational model as defined by SQL-2. The query language SQL-3 is compatible with SQL-2. Thus standard relational queries can be expressed. New features include: *complex types*, *dereferencing* ; *double dot notation; nesting and unnesting.*

Third generation DBMSs should have the following characteristics according to the object-relational 'manifesto':

- Rich Type System
- Generalisation Hierarchies
- Functions (including procedures and methods) are useful
- System should allocate OIDs if a primary key is not available
- Active rules and passive rules will become an essential component of third generation DBMSs
- SQL is the reference language for DBMSs

Object-oriented databases should have the following properties according to the object-oriented manifesto

- Overriding, overloading and Late Binding
- Computational Completeness
- Extensibility
- Durability
- Efficiency
- Concurrency
- Structural Complexity
- Object Identity
- Encapsulation
- Types and/or Classes
- Class and/or Type Hierarchies
- Reliability

- Declarativeness – i.e. high level languages

Some further optional characteristics are

- Multiple inheritance
- Type-checking at compilation time
- Data distribution
- Management of long or embedded transactions

The Object Database Management Group is a committee in which the main constructors of OODBMSs are represented. The committee was brought together in the late 80s. It has proposed a data model with a definition language (ODL), a query language (OQL) and mechanisms for the definitions of methods in languages such as C++ and Smalltalk. OQL looks like SQL.

From a user's perspective common features of both approaches are the support for complex object types i.e. object classes can be stored within object classes or relations can be stored within relations, operations or procedures can be associated with objects or relations, type and class hierarchies can be defined. Main difference between the approaches is the underlying relational technology in object-relational systems, lack of compulsion for object identifier in object-relational systems and possibility to merge relational with object-relational in object-relational systems. Downside of object-relational systems is that they might be less efficient because of the differences just mentioned. Upside is that ad hoc querying and links to predicate logic based systems are supported better. Object-oriented systems might be seen more as programmers tools for specialised systems whereas object-relational might be seen as a more generalised approach.

Marks Breakdown :

Origins	5 marks
Features	12 marks
Comparison	8 marks
	(25 marks)

Examiners Guidance Notes

A common problem in this question was that a number of candidates did not seem to know what is meant by an object-relational system. The object-oriented system seemed to be understood quite well but not object-relational. In fact some candidates compared object-oriented with relational rather than object-relational and lost a lot of marks in the process.