**THE BCS PROFESSIONAL EXAMINATION**
**Diploma**

**April 2004**

**EXAMINERS' REPORT**

**Database Systems**

## Question 1

**1.** *a)* Using your own simple examples, describe and explain the main features and limitations of the file-based approach to data storage in software development. **(10 marks)**

*b)* Using examples to illustrate your answer, describe and explain the main features and benefits of the database approach to data storage in software development. **(10 marks)**

*c)* Draw a single diagram that illustrates how the following concepts are related:

- Data
- Database
- Database Management System (DBMS)
- Application Logic/Query
- Results
- User Interface **(5 marks)**

## Answer Pointers

(a) Marks spread equally over the following key issues:

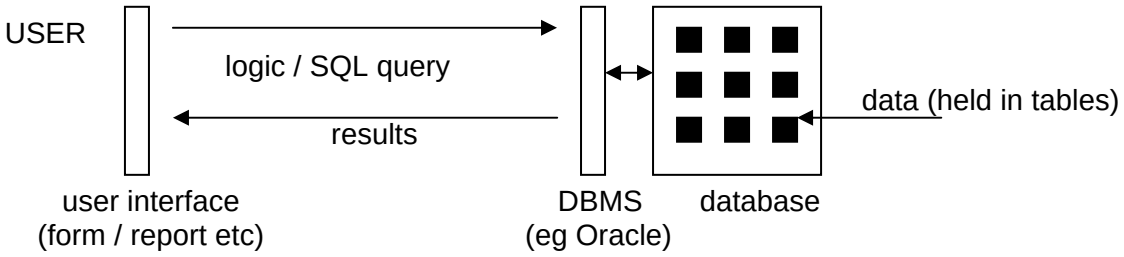
- File-program dependency (data structure in file determined by program logic)
- Redundant & duplicated data spread across multiple discrete files
- Possible inconsistencies between two or more different data items
- Changes must now be rippled across multiple copies of same data
- Any changes to a file format will affect all programs using this file
- This all leads to extra maintenance and programming effort
- Distributed / replicated data means less security (no centralised control)
- Duplicated data may lead to unnecessary and wasted storage
- Manipulating the file data requires technical (imperative) programming skills
- No support for a higher-level query language like SQL
- Similar sensible comments

(b) Marks spread equally over the following key issues:

- Centralised control via DBMS
- A single, unified pool of data
- Greatly enhanced security
- Minimal duplication and redundancy of data
- Data independence from application logic
- Supports a higher-level (declarative) query language like SQL
- More intuitive and less 'low-level' than writing 3GL code
- Lack of data duplication and distribution makes maintenance easier
- Data independence minimizes maintenance ripple across application programs

- Similar sensible comments

(c) Something similar to:




(Marked holistically out of 5 marks)

**Examiners' Comments**

(a) Students were asked to discuss (with examples) the main limitations of the file-based approach to data storage. Some students gave very full and detailed answers whilst others rambled, interpreted 'file-based' to mean manual systems or simply missed many key points such as the program-file dependency, redundancy and possible inconsistencies. Reasonable attempts on the whole.

(b) Students were asked to discuss (with examples) the main benefits and features provided by the database approach to data storage This was, in the main, done quite well. Like part (a), the answers ranged from full and detailed to vague, rambling and minimalist. Most students got some of the key points.

(c) Students were asked to produce a single diagram showing how a typical database system fits together (interface, logic, data, DBMS, database etc). The diagrams supplied ranged from the excellent and succinct, through the vague and rambling to the outright wrong. Most students understood the concepts but relayed that information poorly whilst a small minority simply had a profound misconception of these key issues. A few students did not even draw a diagram – but simply described the component parts. Some even drew Entity-Relationship Diagrams. Needless to say, they lost marks accordingly!

## Question 2

**2.** Using your own SQL code examples, discuss and explain all the relevant concepts that must be addressed by a database developer when implementing the following Data Definition Language (DDL) constructs:

*a)* Table creation **(10 marks)**
*b)* Index creation **(10 marks)**
*c)* View creation **(5 marks)**

**Answer Pointers**

(a) Marks spread equally over the following issues:

- Table name
- Column names

- Data types and sizes of columns eg VARCHAR(25) etc
- Need to create own domains?
- Role of NULL and NOT NULL on different columns
- Primary Key constraint (atomic and composite)
- Foreign Key constraints (if any)
- CHECK constraints (range checks)
- Default values
- Security and privilege sets (who can access table and what can they do?)
- Size of table (row count) and rate of growth (insert heavy or not) and capacity planning – storage issues
- Similar sensible comments

(b) Marks spread equally over the following issues:

- SQL query analysis (which columns are candidates for indexing)
- Selectivity metric of different columns
- Size of table (row count) – small tables not worthwhile
- Indexed columns – simple or concatenated columns?
- Unique or non-unique index (does column already have duplicate values?)
- Index suppression issues in WHERE clause (use of leading wildcards, functions, use of NULL etc)
- Different index types (B-Tree, function-based, reverse-key, bit-map)
- Downside to indexing – index maintenance for tables with heavy DML activity
- Index size and storage concerns
- Index refresh rates and performance impact
- Similar sensible comments

(c) Marks spread equally over the following issues:

- Name of view
- Definition of logic that defines the view (CREATE VIEW statement)
- Will view rename original column names?
- Will view contain computed / derived data from base table(s)?
- Will the view definition contain aggregate functions?
- Will the view be based on a join of two or more base tables?
- Will the view be capable of being updated?
- The use of views as security screens
- Security and privilege sets for the view
- Similar sensible comments

**Examiners' Comments**

(a) Students were asked to discuss 'table creation' issues in SQL. It was generally poorly done (although still better than parts b and c – see below). Most students got the basics such as primary keys, entity integrity rules, the use of null/not null, data types and uniqueness in naming tables and columns but many students lost many marks by omitting other key concepts such as foreign keys, referential integrity rules (including the cascade options) whilst very few mentioned CHECK constraints or default values. Also, the supplied code was often not supported by any annotation or discussion – despite the question specifically asking for a *discussion of the concepts*.

(b) Students were asked to discuss 'index creation' issues in SQL. This was by far the worst of the three sub-questions. Almost without exception it was badly done. Many students did manage to mention the function of access performance on tables and gave a simple code example (often syntactically incorrect). However, the wider issues of when/when not to index, what an index actually *is* (many students thought it was itself a table, rather than a tree structure) storage issues and index suppression etc where completely overlooked. The occasional better student did highlight the difference between uniqueness and non-uniqueness in indexes and the role of implicitly created indexes on primary keys – but these deeper answers were few and far between. Overall, very poor discussions further let down by inaccurate syntax in the SQL examples.

(c) Students were asked to discuss 'view creation' issues in SQL. Most students highlighted the security aspects of views and the fact that they are virtual tables. Most students supplied a decent code example (often slightly inaccurate but acceptable in concept). The better students brought out the difference between levels of data visibility in base tables and views (and related access control / privilege issues). Better done than the indexing answers.

It is obvious that these students have a strong conceptual grasp of tables and views but are much weaker on indexing.

## Question 3

**3.** *a)* List all the Relational Algebra (RA) operators. (Simply itemize the operators - do not describe them or give examples). **(4 marks)**

*b)* For each of the above Relational Algebra (RA) operations, using your own simple examples, describe and explain how the operation works with the relations found in relational theory (include appropriate diagrams to enhance your answer). **(21 marks)**

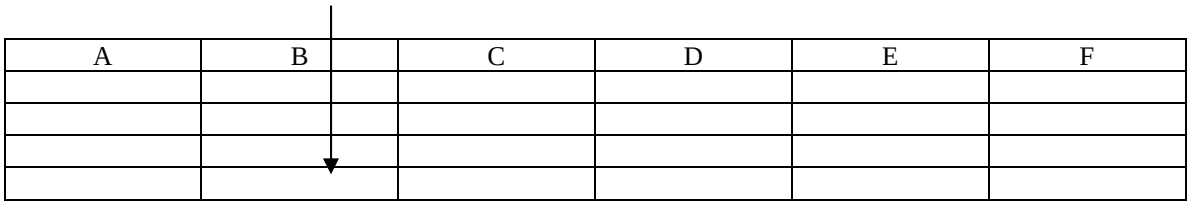
## Answer Pointers

(a) ½ mark each for:

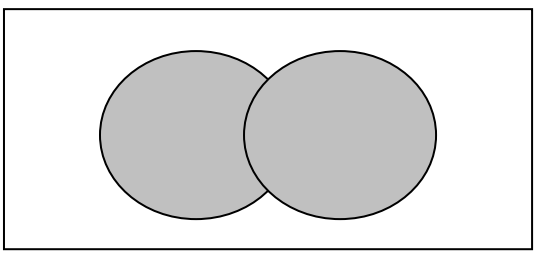
SELECT, PROJECT, JOIN, DIVIDE, UNION, INTERSECT, MINUS (DIFFERENCE), CARTESIAN PRODUCT (TIMES)

(b) 3 marks per operation (up to max total of 21). Marks spread evenly across all operators but full marks will only be given if an actual example is included. Good diagrams will gain bonus marks. Typical diagrams could include:

PROJECT (vertical slice)




UNION (using Venn diagram)



**Examiners' Comments**

(a) Students were asked to list all the Relational Algebra (RA) operators. Most of the students listed most of the relational algebra operation such as Select, Project etc and nearly all of them got the full mark for this question.

(b) Students were asked to describe and explain how each of the RA operation works using their own example. Most of the students answered this question but few of them were finding it difficult to differentiate between some operations such as Select and Project or Join and Cartesian product. The examiner suggests that candidates should know how RA operations are translated into SQL statement and vice versa as most of the students are confused between for instance SQL Select and RA Project or Select operations.

## Question 4

**4. Please refer to Appendix A for Part *a)* and *b)* in this question.**

*a)* Explain how a DBMS transaction manager maintains database integrity in a multi-user environment, for example where update requests are received by a centralised DBMS at the same time. **(12 marks)**

*b)* **Refer to Table A1.** Within the context of the application, describe the range of decisions that need to be made if a situation arose whereby a particular action in a Leg failed to commit or was cancelled.

*c)* Produce example transaction schedules that show how you would re-organise the transaction schedule in order to recover from a failed or cancelled transaction. **(13 marks)**

Part a)

**Answer Pointers**

Transaction integrity is handled by a DBMS transaction manager (TM) in situations where the database is being updated concurrently. The execution of user programs needs to have good DBMS performance thus different types of concurrency control are necessary to balance performance and integrity. These types are not expected in the answer only a broad appreciation that the DBMS (not the host operating system) must provide concurrency control of db transactions using techniques such as locking and isolation levels.
Candidates should explain (perhaps using a scenario) how concurrency control works in a DBMS, thus :
Concurrent transactions may be interleaved actions (reads/writes of DB objects) from different users. A collection of transactions are serialised so that the database can apply integrity rules as if each transaction was atomic (ie follows the ACID properties of a transaction.

**[6 marks]**

Each transaction must leave the database in a consistent state if the DB is consistent when the transaction begins.
Issues follow certain principles that are affected by the interleaving of transactions and the recovery of failed transactions. These are as follows:

**[couple of marks each point total 6 marks]**

**Atomicity of Transactions**

A very important property guaranteed by the DBMS for all transactions is that they are atomic. That is, a user can think of a transaction as always executing all its actions in one step, or not executing any actions at all.

**Commit or Rollback**

A transaction might commit after completing all its actions, or it could abort (or be aborted by the DBMS) after executing some actions. Therefore other transactions that appear to have succeeded will then need to be 'rolled back' to maintain integrity.

**Synchronous vs Asynchronous integrity control**

Tight control has significant performance problems that can be relaxed by providing asynchronous transaction protocol particularly where databases are disconnected for any length of time. TM logs all actions so that it can undo the actions of aborted transactions.

Part b)
This part applies knowledge from 1b) to a practical example where transaction integrity effects the application integrity.
Candidates may interpret different scenarios but the examiner is checking that the scenario meets the test for serialisability thus it should include the following steps:

**1. Serialise a schedule of actions**

Produce a schedule that does not interfere with the actions of different transactions.
This certainly does not occur in the transport scenario as different resources are used at different times during Legs (even though 2 jobs share the same resources).
(Note: If each transaction preserves consistency, every serializable schedule preserves consistency.)

**2. Recovery- needs a cascade abort**

Show that recovery from failure may prevent pre-conditions and dependencies carrying on for other dependent transactions. For example a cascade abort will be needed to de-couple interleaved transactions.
*3. Evaluate equivalent schedules*: For any database state, the effect (on the set of objects in the database) of executing the first schedule is identical to the effect of executing the second schedule. This should be reflected in the schedule produced if one Job fails by allocating resources to another Job that can progress without the side effect of a aborted schedule.

**Examiners' Comments**

The first part of the question was answered quite well. However the depth to candidates' answers varied quite a lot. Candidates should try and cover enough depth and be related to the number of marks on offer rather than simply recall and regurgitate a lot of irrelevant text book material. The second part applied the principles of concurrency and serialisability to a real world example. In the main answers to this part was very poor with many candidates producing contrived answers irrelevant to the scenario. The examiner suggests that candidates should try and 'read around' topics such as this much more and practice analysing similar problems of transaction integrity on a real world problem.

## Question 5

**5. Refer to Appendix A in this question.**

The haulage company is investigating the possibility of tracking the location of their vehicles whilst Jobs are being undertaken. The company has heard that technologies such as satellite navigation can generate accurate location information (such as latitude and longitude) if tracking equipment is installed on tractors/vehicles. The identity of each tracked vehicle and the location information could be collected and relayed to a central server located at HQ and stored on a traffic database. Transport Operators (TOs) can view this information on client computers allowing them to monitor and interact with the traffic database.

Produce a design for a user interface supporting the information system outlined above. Your design is restricted to the client-side user interface supporting a TO view of the information system.

The following should accompany your answer:

The design decisions and any assumptions you make. Draft screen shots showing examples of the way that a TO would interact with the information system. The data sources such as tables and views that you would need to support the presentation of information to a TO. **(25 marks)**

### Answer Pointers

Candidates could produce a range of user interface designs but these must meet defined design parameters of the information system described in the scenario. An appreciation of processing spatial information is not expected but the presentation of such information is required. Also an appreciation of client-server processing would be helpful in formulating the presentation of views of data held in tables. The marking was flexible but a typical break down is as follows :

1. Design decisions – justification for html/GUI browser style versus for example a Windows/GUI application. Limitations dictated by client server connections and state handling. Structure of the screen – eg action areas, passive display areas and supporting drill down menus assisting users progress of interaction. Practical issues – image data linking maps to data of locations. Display of route information is it real time or a static display? Candidates need to suggest some way of distinguishing planning from active transport plans. **[8 marks]**

2. Screen shots and visibility of interaction patterns and navigation. Presentation quality and ease of use. This is the realisation of the design in part 1 answer above. **[10 marks]**

3. Visibility of table structure must be apparent in order to produce a framework of interaction with a TO view/user model. Marks are awarded to those candidates that show the visibility of SQL processing and/or stored procedures triggers when a TO requests or updates information on the user interface designed above.

**[6 marks]**

### Examiners' Comments

This question was generally poorly answered. As expected a range of answers was produced depending on the quality of candidate's experience of designing database user interfaces. There were very few high scoring answers. The main problem the examiner had was discerning the candidates reasoning and analysis of the application. This was exacerbated by poor structure to answers and a lack of commentary to reinforce design decisions that candidates made. The worst case of this was evident with answers that seemed to be just a jumble of undocumented screen shots (ie not supported by comments and design decisions).

## Question 6

**6. Refer to Appendix A for Part *b)* in this question.**

*a)* Explain the differences between:
- *i)* an object id in an Object data model and a Primary Key in a Relational data model.
- *ii)* a Class in an Object data model and an Entity Type in an Entity Relationship data model. **(8 marks)**

*b)* *i)* Produce an ER model that models the relationships that exist between the following Entity Types:

Jobs; Transport Operators; Containers; Products; Legs; Tractors; Trailers; Vehicle Units and Locations.

*ii)* Include the relationship degrees and participation constraints in your model. Assign attributes to the Entity and Relationship Types and identify which of these attributes are primary keys.

*iii)* State any assumptions that you make and the notation you have used to model the relationship degrees and participation constraints. **(17 marks)**

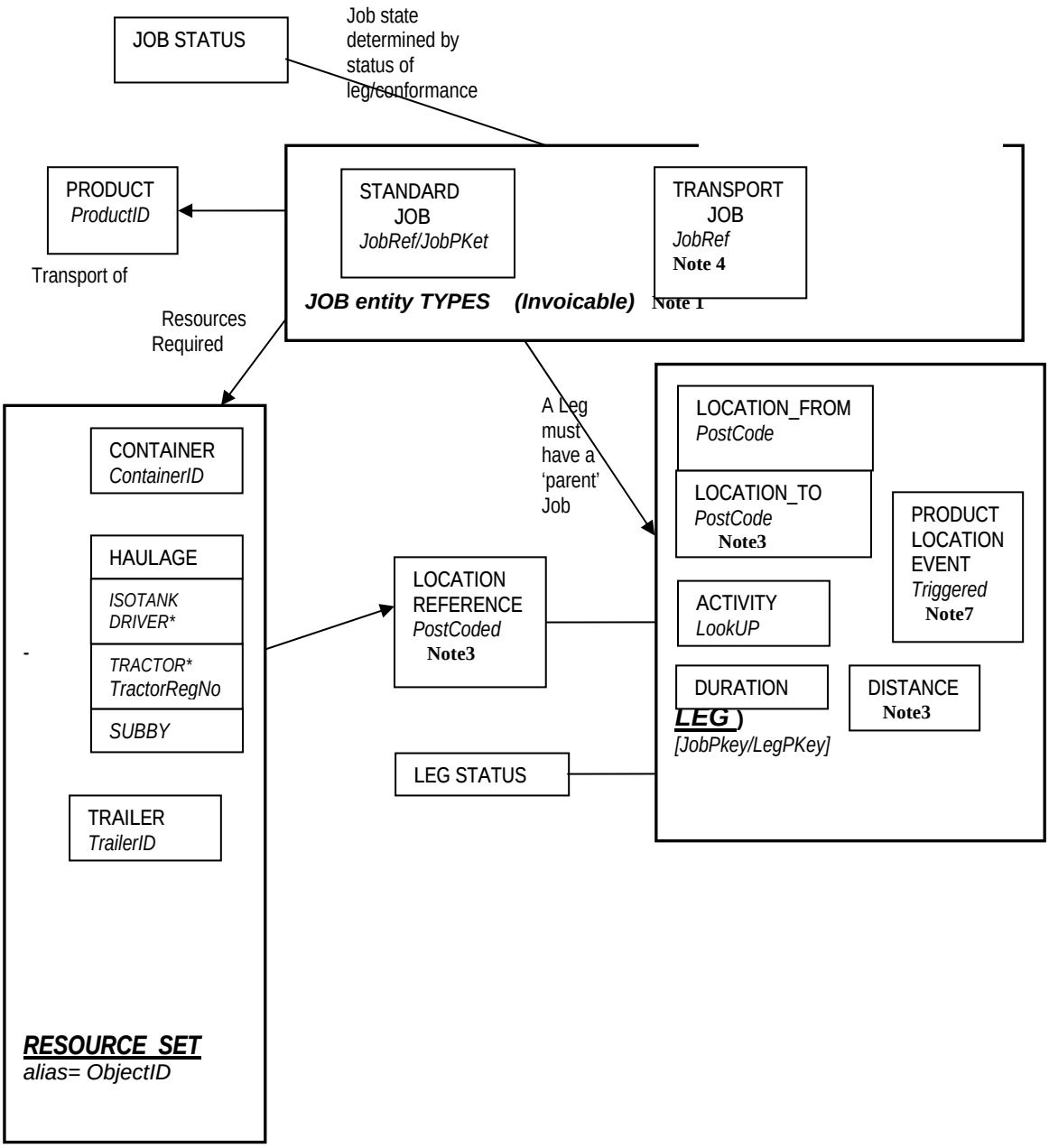
## Answer Pointers

a) (4 marks each pair)
objectid identifies using a surrogate that persists outside the implementation model (i.e. only expires when explicitly deleted). Primary key optional in a RDB module and depends for its existence on the persistence of the DBMS.

Class is a OO programming concept that defines the characteristic behaviours and structure of a collection of objects. Entity type only defines the data structure and maps to tables at the implementation level (usually). Class could be non-persistent whereas Entity type is an abstract model of some persistent artifact.

b) Marks for each identifying correct relationship and participation class approximately 4 marks per relationship.
Diagram similar to the following – very little scope for a range of interpretation unless assumptions state otherwise, but objective is to demonstrate accuracy and modelling skills in interpreting a fairly complex scenario.
Part Data Model reflecting the key association between Employee (role = TO) customer products/orders and Jobs and Legs.

**BCS Database April 2004 - QUESTION 6 diagram**




[the above model would attract full marks only if assumptions and legend supplied to ER model – i.e. different standards exist but those supplied must not be ambiguous]

**Examiners' Comments**

(a) Students were asked to explain the differences between an object id in an Object data model and a Primary Key in a Relational data model and also the difference between Class in an Object data model and an Entity Type in an Entity Relationship data model. Most of the students answered correctly the similarity between ObjectID and Primary in identifying the uniquely objects and relations respectively, but few presented the difference between and ObjectID and primary key. Also, only half of students have clearly answered correctly the difference between a Class in an Object data model and an Entity Type in an Entity Relationship data model.

(b) Students were asked to produce an ER model that models the relationships that exist between a set of Entity Types. They were also asked to include the relationship degrees and participation constraints in their model and also assigning attributes to the Entity and Relationship Types and identify which of the attributes are primary keys. They were also asked to state any assumptions that they make and the notation they used to model the relationship degrees and participation constraints. Nearly all students have come up with an ERD that models the relationships that exist between the Entity Types: Jobs; Transport Operators; Containers; Products; Legs; Tractors; Trailers; Vehicle Units and Locations. They also assigned attributes to the Entity and Relationship Types and identified which of these attributes are primary key.

However, few have presented a correct relationship degrees and participation constraints in the model. The examiner suggests that candidates should read about and practice how to design relationship degrees and also represent constraints in a data model as most of the students have lost marks when it comes to relationship degrees and representing constraints.