



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Examination Answer Books

12 Pages

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

Complete, fold and seal
(see note 2 below)

SURNAME: STRAHAN

FORENAME(S): RYLE

STEPHEN KYLE

1	Student Number	180976
2	Title of Examination Paper	Database Management Systems
3	Module Code	ACCAS014
4	Date and Time of Examination	09/05/19 ~~9:00~~ 9.05 AM
5	Examination Venue	G42
6	Seat Number	—
7	Number of Answer Books handed in	1

List the numbers of the questions you have attempted	Office Use
1	22 $\frac{1}{2}$
2	20 $\frac{1}{2}$
4	21 $\frac{1}{2}$
Total	64 $\frac{1}{2}$

(Gweler yr ochr arall am y fersiwn Cymraeg)

1

a

```

i CREATE TABLE TutorBooking (
  BookingRef int NOT NULL,
  BookingDate date date NOT NULL,
  TutorNo int NOT NULL,
  StudentNo int NOT NULL,
  SubjectID int NOT NULL,
  SubjectLevel int NOT NULL, -- number levels e.g. 1, 2, 3
  HourlyRate decimal NOT NULL, -- decimal for currency
  CONSTRAINT CK_DATE CHECK (BookingDate <=
    GETDATE()) -- checks if booking was made before
    the current date as cannot be in future
);

```

```

ii ALTER TABLE TutorBooking
  ADD CONSTRAINT PK_BookingRef PRIMARY KEY BookingRef;

```

```

iii ALTER TABLE TutorBooking
  ADD CONSTRAINT FK_TutorNo FOREIGN KEY TutorNo REFERENCES
    Tutor(TutorNo),
  ADD CONSTRAINT FK_StudentNo FOREIGN KEY StudentNo
    REFERENCES Student(StudentNo),
  ADD CONSTRAINT FK_SubjectID FOREIGN KEY SubjectID
    REFERENCES Subject(SubjectID);

```

iv. CREATE TABLE TutorSubject (
 TutorNo int NOT NULL,
 SubjectID int NOT NULL,
 KSLevel varchar(3), -- 'Yes' or 'No', Allows null
 ALevel varchar(3), -- 'Yes' or 'No', Allows null
 HELevel varchar(3), -- 'Yes' or 'No', Allows null
 MinHourlyRate decimal NOT NULL,
 MaxHourlyRate decimal NOT NULL,
 CONSTRAINT FK_TutorNo FOREIGN KEY TutorNo
 REFERENCES Tutor(TutorNo),
 CONSTRAINT FK_SubjectID FOREIGN KEY SubjectID
 REFERENCES Subject(SubjectID)
 CONSTRAINT CK_Level CHECK((KSLevel, ALevel, HELevel)
 = 'Yes' OR 'No')
);

b
 i. SELECT TutorNo, ~~name~~ FirstName, Surname, Email, PhoneNo
 FROM Tutor
 WHERE DBSEpiryDate BETWEEN '2019-06-01' AND '2019-06-31'
 ORDER BY DBSEpiryDate ASC;

ii. UPDATE ~~TABLE~~ TABLE Tutor
 SET Status ~~to 'Expired'~~ AS ~~'Expired'~~ 'Pending DBS check'
 WHERE DBSEpiryDate < GETDATE();

iii. SELECT MIN(HourlyRate), MAX(HourlyRate), AVG(~~HourlyRate~~)
 FROM TutorBooking
 WHERE BookingDate BETWEEN ~~'2019-01-01'~~ '2019-01-01' AND '2019-12-31';

2

- a. External Schemas of a database hold the views of a database and shows how the user will interact with the data.

Internal schema defines how and where the data is stored including the SQL code.

Conceptual schemas are the community view of the database.

b.

- i. Indexes are used in databases to ~~create order~~ improve the performance of a database to improve the ~~result~~ query result speed by ordering the results so they can be searched more efficiently.

- ii. Indexes can be useful to tune a database. They should be used on columns that are being queried often, however they should not be used when the data is being updated regularly as the indexes need to be maintained. If a relation is small with only a few tuples, then there is no need to create an index for these relations.

- iii. A clustering index could potentially be used on the name columns of the tutor table, as they will be accessed often and wouldn't necessarily be updated.

~~CREATE INDEX~~

ALTER TABLE Tutor

ADD CONSTRAINT IDX_Name

CLUSTERED INDEX (FirstName);

iv. As mentioned in the previous question, the surname field of the Tutor table could be indexed as that would also be accessed ~~regularly~~ regularly. This would be a big relation so indexing this column is worthwhile.

v. INSERT INTO TutorBooking
VALUES ('500', '2019-05-01', '86', '25', '100', '1', '9.99');
INSERT INTO BookedLesson
VALUES ('500', '2019-05-30 10:00', '120', 'Llanelli', ~~120~~
NULL, '1234567891011123', '14.99')
('500', '2019-05-31 10:00', '120', 'Llanelli', NULL,
'1234567891011123', '14.99');

4.

a. A - Atomicity - This property defines that the transaction should be executed fully or not at all to ensure consistent data.

C - Consistency - All integrity constraints should be maintained to keep database in a consistent state.

I - Isolation - If multiple transactions are executed, this prevents and ensures the data is not inconsistent.

D - Durability - All transaction stored and data is consistent.

b. Concurrency Control is the process of managing the execution of multiple transactions simultaneously. This can prevent inconsistent data and errors to keep the database in a consistent state.

C.

- i. **Lost Updates** - If two transactions are accessing the same data of a table (same row + column), then ~~the~~ the transaction who commit's first ~~or~~ will over-write the ~~second~~ other transactions data.

Uncommitted Dependency - If a transaction updates data but has yet to be committed, a second transaction could read the data before the first transaction has been committed. Potentially causing errors within the data.

Inconsistent Analysis - Also known as a dirty read, a transaction could read inconsistent data written by another transaction.

ii

Time	~~T1~~ T ₁	T ₂	~~bal_x~~ bal _x
T1		BEGIN TRANS	500
T2	BEGIN TRANS	Read bal _x	500
T3	Read bal _x	bal _x = bal _x + 100	500
T4	bal _x = bal _x - 50	Write bal _x	600
T5	...	COMMIT	600
T6	Write bal _x		600
T7	COMMIT		600

At T3, transaction 2 has added 100 to the balance, written it to bal_x at T4, and committed before the first transaction at T5. This overwrites the first transaction.

This could be tackled by executing the transactions at separate times, by using a timestamp-based protocol which gives priority to the oldest transaction. This is a method of ~~concurrency~~ concurrency control.

- d.
- i. Transaction₁ would need to be redone as it has been aborted. Transaction₃ will also need to be redone as the checkpoint is before the commit. Transaction₄ would have to be redone due to it crashing before the commit.
- ii. No transactions need to be undone.
- iii. Transaction₂ is not affected as it has been committed before the checkpoint.

X

$$\begin{array}{r} 21 \\ \times 21 \\ \hline 21 \\ 42 \\ \hline 441 \end{array}$$

$$\begin{array}{r} 22\frac{1}{2} \\ 20\frac{1}{2} \\ 21\frac{1}{2} \\ \hline 64\frac{1}{2} \end{array}$$

56%