



Examination Answer Books

12 Pages

Instructions to Candidates

- Please give the information requested in the boxes below.
- Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
- 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.
- Write your answers in ink.
- Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
- No part of this book is to be torn out; it must be handed in complete.
- If more books than one are used, fasten them together using the tags available from the invigilator(s).
- Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.

1	Student Number	139857
2	Title of Examination Paper	Database Management Systems
3	Module Code	ALCA5014
4	Date and Time of Examination	09/05/2019
5	Examination Venue	Pembroke College
6	Seat Number	15
7	Number of Answer Books handed in	1

Office Use	List the numbers of the questions you have attempted	1
Office Use		18
Office Use		2
Office Use		4
Office Use		9
Office Use		34

Complete, fold and seal
(see note 2 below)

SURNAME: Long

FORENAME(S): James

Alexander

SECTION A

1

CREATE TABLE TutorBooking (

BookingRef VARCHAR(5) NOT NULL,

BookingDate DATE NOT NULL,

TutorNo VARCHAR(5) NOT NULL,

StudentNo VARCHAR(5) NOT NULL,

SubjectId VARCHAR(5) NOT NULL,

SubjectLevel VARCHAR(20) CHECK (SubjectLevel = 'KSLevel', 'ALevel', 'HELevel', 'HourlyRate DECIMAL(4,2) NOT NULL);

7

!!

ALTER TABLE TutorBooking (

CONSTRAINT [PK-TutorBooking] PRIMARY KEY (BookingRef);

1

!!!

ALTER TABLE TutorBooking (

CONSTRAINT [FK-TutorBooking-Tutor] FOREIGN KEY REFERENCES Tutor (TutorNo),

CONSTRAINT [FK-TutorBooking-Student] FOREIGN KEY REFERENCES Student (StudentNo),

CONSTRAINT [FK-TutorBooking-Subject] FOREIGN KEY REFERENCES Subject (SubjectNo);

3

~~CREATE TABLE TutorBooking (~~
~~BookingRef VARCHAR(15)~~

IV

CREATE TABLE TutorSubject (

TutorNo VARCHAR(5) CONSTRAINT FOREIGN KEY
REFERENCES Tutor(TutorNo),
Subject VARCHAR(5) PRIMARY KEY, NOT NULL,

8 foreign

MinHourlyRate DECIMAL(4,2) NOT NULL, *1
MaxHourlyRate DECIMAL(4,2) NOT NULL *1

*1 CHECK MinHourlyRate > 5.60 ✓

6

FROM
SELECT Tutor, TutorNo, FirstName, Surname,
Email, City, Postcode, PhoneNo, DBS ExpiryDate WHERE
DBS ExpiryDate <= June 2019
ORDER BY DBS ExpiryDate ASC

UPDATE Tutor DBS ExpiryDate *2
WHERE DBS ExpiryDate < GETDATE()

*2 'Expiry'

!!!

SELECT Tutor
Min(MinHourlyRate), Max(MaxHourlyRate), AVG(AverageHourlyRate)

!!!

It is important to analyse transactions before ~~they enter~~ ^{implementing} the database indexes into the database because there is a chance that implementing the index to the wrong table/fields can actually slow down the process. Also, this can be incorrect data in transactions which could be potentially harmful to the database and cause it to crash or give errors. To decide on what ~~data~~ criteria to add to an index relies on an easily identifiable field or a heavily queried field or set of fields. ~~Interventions~~ An index will then speed up the querying process for these areas of the database. To summarise, any area which would benefit from ~~your~~ speed up querying process should be analysed for a potential index.

!!

The purpose of an index in a database is to speed up searching / querying data from the database.

! b

a

SECTION B

2

Question No.

Examiner Only

9

a

4

*⁵ 10:00:00

INSERT INTO TutorBooking (BookingRef, BookingDate, SubjectId, TutorNo) VALUES ('500', '31-May-2019', '10:00:00', '100', '66');

1 ✓ INSERT INTO TutorBooking (BookingRef, BookingDate, SubjectId, TutorNo) VALUES ('500', '30-May-2019', '10:00:00', '100', '66');

✓ V ~~INSERT INTO Booking~~ ~~BookingRef~~ (BookingRef)

IV One column which would benefit from a secondary index is Status. This column will be in constant use as it will be changing frequently depending on whether the tutor is available or not. AM Retrieval this data would be made quicker if a secondary index is added.

2 ✓

Examiner Only

• 1000 400

Time	T1	T2
t1	BEGIN Trans	balx / 1500
t2	Begin Trans	Read balx
t3	Read balx	balx = balx + 100
t4	balx = balx - 50	Write balx
t5	...	Commit
t6	Write balx	500 ✓
t7	COMMIT	500 ×

The last update problem here happened at 65 in the T1 column. As if I lost an update here, the previous update at 64 ran twice and left the end user with the same amount of money as before the transaction started.

This needs to be redone again. This kind of exiting and going back and forth is because it's not a good idea to be running when the crash occurred and there may be some incomplete data added to the database

T1 is unaffected by the crash as it was completed or aborted before the checkpoint and the crash. T2 is also unaffected by the crash as it was completed and committed before the checkpoint and crash.

! T3 needs to be redone as when the checkpoint occurred it was half way through. If the database was called back to the next checkpoint then only half of the data in T3 would be added

9

✓ 1