1.8 Databases, SQL and Normalisation: Test Pack

OCR H446 · Calculators not allowed · about 20 minutes

Total Marks

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 This question is about transaction processing in the Ferngate Leisure Centre database.

(a) State what each letter of the acronym ACID stands for, in the context of database transaction processing. **(4)**

(b) When a member books a class, the database must both insert a new row into Booking and update the count of places remaining for that class in FitnessClass, as a single transaction. Discuss why atomicity and isolation are both essential properties for this transaction, using this booking scenario to illustrate your answer. **(4)**

(Total for Question 1 is 8 marks)

© Revision Library | revisionlibrary.co.uk

2 This question is about how the Ferngate Leisure Centre database manages multiple users accessing it at the same time.

(a) State what is meant by record locking in a multi-user database system. **(2)**

(b) State one difference between pessimistic locking and optimistic (validation-based) concurrency control. **(2)**

(c) Two receptionists at different desks try, at almost the same moment, to book the last remaining place on the same fitness class for two different members. Discuss how the DBMS's concurrency control prevents this resulting in the class becoming overbooked, and explain what is meant by a deadlock and one method the DBMS could use to resolve one. **(4)**

(Total for Question 2 is 8 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 1.8 Databases, SQL and Normalisation: Test Pack

Question 1

(a) Atomicity **B1**

(a) Consistency **B1**

(a) Isolation **B1**

(a) Durability **B1**

(a) Answer: Atomicity, Consistency, Isolation, Durability.

(b) **L1** Basic, undeveloped reference to atomicity and/or isolation, with little or no link to the booking scenario. **1-2**

(b) **L2** Clear explanation of both atomicity and isolation, each illustrated using the booking scenario, showing what could go wrong for the leisure centre if either property were missing. **3-4**

(b) Indicative content:

- Atomicity means the transaction is treated as a single indivisible unit: either both the Booking insert and the FitnessClass places-remaining update happen, or neither does.
- Without atomicity, a failure partway through (e.g. a power cut after the Booking row is inserted but before places remaining is decremented) could leave the database inconsistent: a booking exists but the class still appears to have a free place.
- Isolation means concurrent transactions (e.g. two receptionists booking different members onto classes at the same time) do not interfere with each other's intermediate/uncommitted results.
- Without isolation, two simultaneous bookings for the last remaining place could both read the same 'one place left' value before either commits, and both insert a Booking row, resulting in the class being overbooked (a lost update).

(b) Answer: N/A - levels-marked extended response.

Question 2

(a) Locking prevents a second transaction from accessing (reading and/or writing) a record while another transaction is currently using/updating it (oe) **B1**

(a) Second point, e.g. the lock is released once the first transaction commits or rolls back, at which point another transaction may then acquire it (oe) **B1**

(a) Answer: Record locking stops a second transaction from reading and/or writing to a record while another transaction is in the middle of using it; the lock is released once that transaction commits or rolls back.

(b) Pessimistic locking locks a record for the duration of a transaction before any changes are made, assuming conflicts are likely (oe) **B1**

(b) Optimistic concurrency control allows transactions to proceed without locking, and only checks for conflicts (validates) at the point of commit, assuming conflicts are rare (oe) **B1**

(b) Answer: Pessimistic locking locks a record up front, for the whole transaction, on the assumption that conflicts are likely. Optimistic concurrency control does not lock in advance; it lets transactions proceed and only checks for a conflict when the transaction tries to commit, on the assumption that conflicts are rare.

(c) **L1** Basic, undeveloped reference to locking preventing overbooking and/or a partial description of a deadlock, with little application to the scenario. **1-2**

(c) **L2** Clear, well-applied explanation of how locking prevents the lost-update/overbooking problem in this scenario, together with an accurate explanation of deadlock and a valid resolution method. **3-4**

(c) Indicative content:

- When the first receptionist's transaction reads/updates the FitnessClass row (places remaining), the DBMS places a lock on that record, so the second receptionist's transaction attempting to update the same row must wait.
- The first transaction completes (commits), releasing the lock and reducing places remaining to zero; the second transaction can then proceed, but now sees zero places remaining and its booking is correctly rejected, preventing overbooking.
- A deadlock occurs when two transactions each hold a lock that the other transaction needs, and each is waiting for the other to release its lock, so neither can ever proceed.
- The DBMS can resolve a deadlock by using a timeout (aborting/rolling back a transaction that has waited longer than a set time) or by running a deadlock-detection algorithm that identifies the circular wait and forces one transaction to roll back, releasing its locks so the other can continue.

(c) Answer: N/A - levels-marked extended response.

© Revision Library | revisionlibrary.co.uk