THE UNIVERSITY OF NEW SOUTH WALES
Final Exam

# COMP9311
# Database Systems

**TERM 1, 2024**

---

- Time allowed: **3 hours (2:00PM to 5:00PM, 4 May, Sydney Time.)**
- Total number of questions: **5**
- Total number of marks: **100**
- Answer **all** questions.
- You can answer the questions in any order.
- Start each question on a **new page**.
- We accept any format: directly answering using Word or handwriting and converting to Word or PDF. We only require the file to be clear and in **.docx** or **.pdf**.
- Your file should have a prefix of **zid_name**.
- Submit your answer file via Moodle. You may submit multiple times and we will mark the LAST one.
- **Check your submission before you leave.**

---

## Question 1 (18 marks)

(a) (10 marks) Draw an ER diagram to represent the following application requirements for a *Car Rental Database*. You must use the drawing conventions in the lecture notes. Clearly state any additional reasonable assumptions you make.

- A customer is uniquely identified by her/his ID. We record her/his name, email, and driving license ID. A customer may have more than one email. The name is composed of first name and last name.
- A customer may make zero or more reservations. Each reservation must be associated with one customer and one car. Similarly, a car can be reserved zero or more times. Additionally, we record the method used by the customer to make the reservation, i.e., phone, email, or online booking.
- A reservation is uniquely identified by reservation ID. We record its duration, location, and insurance. The duration is composed of pick-up date and return date, and the location is composed of pick-up location and return location. A reservation may have multiple insurances.
- Each reservation must be managed by a single rental company. A company can manage any number of reservations, from none to many, and must own at least one car. Every car must be owned by one specific rental company.
- A rental company is uniquely identified by company ID. We record the rental company's name, address, and the number of cars it owns. A car is uniquely identified by its VIN number. We record its available status, brand, and model. In this rental system, the sole expense for users is the daily rental fee, so we also document the daily rental price for each car.
- The previous maintenance information of a car is crucial. Each maintenance record is identified by its ID and car's VIN. We also document the date of the maintenance, its cost, and the odometer reading. Each maintenance must be logged on one car, and a car can log zero or more maintenance records.
- Each reservation must be associated with one invoice and each invoice must correspond to one specific reservation. An invoice is uniquely identified by the invoice ID. We record the payment method used and the total cost of the reservation on the invoice.

(b) (8 marks) Translate the following ER diagram into a relational model. You must use the drawing conventions in the lecture notes.

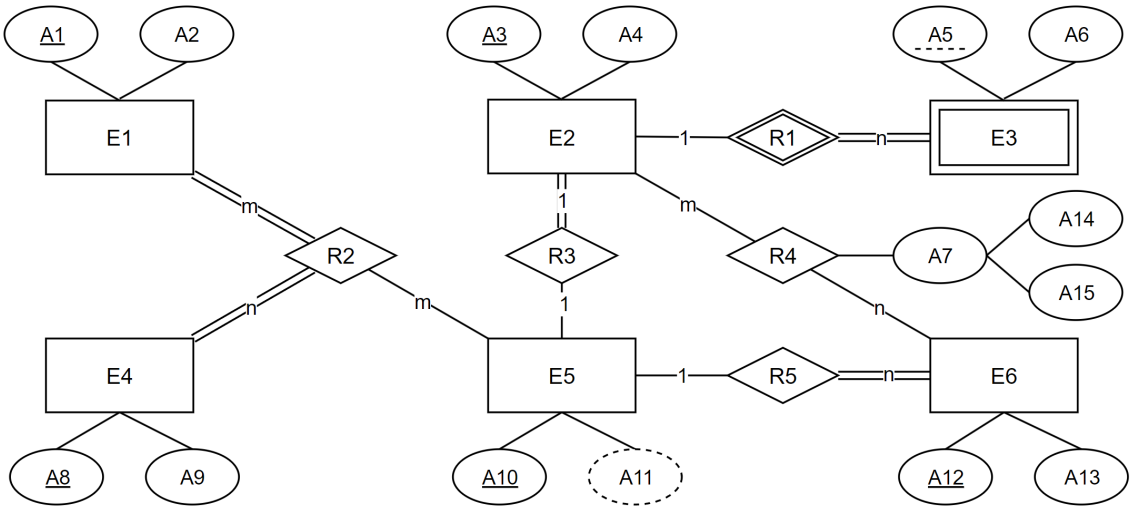


Figure 1: ER Diagram of Q1(b).

## Question 2 (22 marks)

Consider the following relational database schema for chain restaurants. The database schema consists of 5 relations, with the names and their attributes shown below. The underlined attribute names in each relation indicate that the combination of their values uniquely identifies each tuple (i.e., primary key). Note that attributes other than the primary key may not be unique among all tuples.

- Brand (BrandID, BrandName, Headquarters, FoundedYear),
- Branch (BranchID, BrandID, Location, ManagerID, OpenYear)
- Employee (EmployeeID, BranchID, Name, Role, HireYear)
- Menu (MenuID, BranchID, ItemName, Price, Category)
- Feedback (FeedbackID, BranchID, CustomerName, Rating, Comment, Date)

Below are detailed descriptions for the fields in each schema:

- Brand: For each brand, we record the brandID, brandName, headquarters, foundedYear. The brandID serves as the primary key. Headquarters is the location of the brand's headquarters. FoundedYear servers as the year the brand was established.
- Branch: For each Branch Store, we record branchID, brandID, location, managerID, and openYear. The branchID attribute serves as the primary key. Location is the physical location of the branch store. You can check if a location is somewhere using the "in" operation, e.g., "Branch.location" in somewhere. The openYear servers as the year this branch was opened.
- Employee: This relation records employeeID, branchID, name, role, and hireYear, identified by employeeID. The Role represents the job roles like "manager", "chef" or "waitstaff". The hireYear servers as the year the employee was hired.
- Menu: For each menu, we record the menuID, branchID, itemName, price, and category. The "price" attribute is a positive integer. The combination of menuID and branchID forms the primary key. The category is a type of food (e.g., appetizer, main course and dessert).
- Feedback: For each feedback, we record the feedbackID, branchID, customerName, rating, comment, and date in the format YYYY-MM-DD. The "rating" attribute is a positive integer from 0 to 5. The combination of feedbackID and branchID services as the primary key.

Answer the following queries (if not possible, provide a brief explanation). State any reasonable assumptions you made.

(a) (5 marks) Express the following query using **relational algebra**:
List the branchIDs of all the branches that do not have dessert priced lower than $10.

(b) (5 marks) Express the following query using **relational algebra**. You are only allowed to use the 5 operators (i.e., selection $\sigma$, projection $\pi$, cartesian product $\times$, difference $-$, and rename $\rho$):

List the branchIDs of all the branches that have the same opening year as the Brand, and there are no orders rated less than 4.0 in the last 3 years.

(c) (6 marks) Express the following query using **relational algebra**:

List the branchIDs of branches that have maintained an average score of at least 4.5 points over the past ten years and have senior staff who have been hired more than ten years.

(d) (6 marks) Express the following query using **relational algebra**:

List the branchIDs of all branches located in the same area as the headquarters that simultaneously offer both appetizers and main courses, but do not offer desserts, and do not have an assistant manager. The term "assistant manager" refers to the role specified in the "Employee.Role" attribute.

## Question 3 (20 marks)

(a) (6 marks) Consider the relational schema $R(A, B, C, D, E, G, H, I, J)$ and a set of functional dependencies $F = \{A \rightarrow EH, BE \rightarrow G, DI \rightarrow AJ, GH \rightarrow CI, C \rightarrow BDI\}$. Note that $A, B, C, D, E, G, H, I$ and $J$ are attributes. Regarding $F$, given a decomposition $R1 = \{ADIG\}$, $R2 = \{ABEG\}$, $R3 = \{CEGH\}$, and $R4 = \{BECJ\}$ of $R$.

i. Is $R1$ in 3NF? Please justify your answer. *(3 marks)*

ii. Is the decomposition lossless? Please justify your answer. *(3 marks)*

(b) (14 marks) Consider the relational schema $R(A, B, C, D, E, G, H, I, J)$ and the set of functional dependencies $F = \{AE \rightarrow CGH, BE \rightarrow IJG, D \rightarrow AI, GJ \rightarrow DE, AHI \rightarrow B\}$. Note that $A, B, C, D, E, G, H, I$ and $J$ are attributes. Justify your answer to each question.

i. Determine the highest normal form of $R$. *(2 marks)*

ii. Find all the candidate keys for $R$. *(2 marks)*

iii. Find a minimal cover $F_m$ for $F$. *(3 marks)*

iv. Decompose $R$ into BCNF. *(4 marks)*

v. From your answer in iv., is your decomposition dependency preserving? *(3 marks)*

## Question 4 (18 marks)

Consider the following page requests from a transaction in a database system:

$$1, 2, 3, 1, 4, 1, 2, 5, 2, 6, 2, 7, 3, 8, 3, 9, 3, 1, 2$$

(Page 1 is first read from disk, then page 2, page 3, ...)

(a) (8 marks) Assume that the buffer pool is initially empty and there are **6** buffer frames in the buffer pool.

i. Sketch the process of how blocks are replaced in the *Least Recently Used (LRU)* policy. For each page request, indicate whether it's a 'hit' or a 'miss'. *(3 marks)*

ii. Sketch the process of how blocks are replaced in the *First In First Out (FIFO)* policy. For each page request, indicate whether it's a 'hit' or a 'miss'. *(3 marks)*

iii. Calculate the page hits rates of the above two policies and justify which policy performs better. *(2 marks)*

(b) (6 marks) In real applications, the *2Q* caching policy[1] is widely used. Following is a simple version of the *2Q* algorithm. We divide the whole buffer pool into 2 parts, A1 and A2. Initially, when a page is referenced for the first time, it is placed in A1, which is managed on a *First In First Out (FIFO)* basis. If this page is referenced again while it resides in A1, it is likely a "hot" page and is thus transferred to A2, which is managed on a *Least Recently Used (LRU)* basis. This transfer process only occurs in the main memory and does not participate in disk I/O. The pages in A2 will not be transferred back to A1. Conversely, if a page remains unreferenced during its stay in A1, indicating it is likely a "cold" page, it will be managed in A1 according to the *FIFO* policy.

i. Sketch the process of how blocks are replaced by using the *2Q* policy. For each page request, indicate whether it's a 'hit' or a 'miss'. Assume that the buffer pool is initially empty, and there are **3** buffer frames for A1 and **3** buffer frames for A2 in the buffer pool. *(4 marks)*

ii. Between the *LRU* and *FIFO* from question 4(a) and *2Q* policies discussed above, which one performs better in the given query? Why? *(2 marks)*

[1] Johnson, T., & Shasha, D. (1994, September). 2Q: A low overhead high performance buffer management replacement algoritm. In Proc. 20th Int. Conf. Very Large Databases (pp. 439-450).

(c) (4 marks) Consider an employee table with 4 attributes: id, name, birth and salary. The tuples of the relation are stored in a sorted file based on an increasing ordering of the values of birth. Furthermore, suppose id is the only key in the relation. Below is the first 3 and last 3 tuples in the relation.

| id | name | birth | salary |
|---|---|---|---|
| 001 | Jane White | 1975 | 3000 |
| 003 | Mark Lee | 1976 | 2000 |
| 002 | Lynd Keith | 1976 | 8000 |
| ... | ... | ... | ... |
| 004 | John Loy | 2000 | 4000 |
| 011 | Mary Chan | 2001 | 3000 |
| 010 | Peter Karp | 2001 | 5000 |

i. Is it meaningful or reasonable to build a clustered index over the attribute name, if we frequently search people with a query name, e.g., search the tuple with name as "Jane White"? Justify your answer. *(2 marks)*

ii. Is it meaningful or reasonable to build a hash index over the attribute name, if we frequently search people with their first name? Justify your answer. *(2 marks)*

## Question 5 (22 marks)

(a) (8 marks) Consider the following schedule (S1) of the set of transactions $T_1, T_2, T_3, T_4$. ($R_i(X)$ means transaction $T_i$ reads the value of $X$ from the database, and $W_i(X)$ means transaction $T_i$ writes the value of $X$ to the database.)

$$S1 : R_3(A), R_4(C), W_1(A), R_3(C), R_1(B), W_1(D), W_4(A), R_3(B),$$

$$W_4(D), W_2(C), W_2(A), W_2(B), R_2(D)$$

i. Draw the precedence graph of the schedule with all the transactions. *(4 marks)*

ii. State whether the schedule is serializable or not. If it is serializable, write down an equivalent serial schedule. If not, briefly explain why. *(4 marks)*

(b) (8 marks) Given the four transactions $T_1, T_2, T_3, T_4$, and $T_5$ below, please construct schedule according to the requirements. In the constructed schedule, each transaction should keep its operations and the order of operations.

$T_1$: Write(C), Write(A), Write(B), Write(E)
$T_2$: Write(D), Read(F), Write(C)
$T_3$: Write(B), Read(E), Write(D)
$T_4$: Write(B), Read(B), Write(D)
$T_5$: Write(A), Write(D)

i. Construct a schedule of $T_1, T_2, T_3, T_4$, and $T_5$ which **causes** deadlock when using two-phase locking protocol. You should clearly indicate all the **lock** and **unlock operators** in your schedule and draw the **wait-for graph** for the deadlock part only. If no such schedule exists, explain why. *(4 marks)*

ii. Construct a schedule of $T_1, T_2, T_3, T_4$, and $T_5$ which does **NOT cause** deadlock when using two-phase locking protocol. You should clearly indicate all the **lock** and **unlock operators** in your schedule and draw the full **wait-for graph** which includes all the transactions. If no such schedule exists, explain why. *(4 marks)*

(c) (6 marks) Consider the following log file of a database system. The log file contains the log records of the transactions $T_1, T_2, T_3, T_4$. The log records are ordered by their timestamps.

| Time | Log record |
|---|---|
| $t_1$ | [start_transaction, $T_1$] |
| $t_2$ | [read_item, $T_1$, X] |
| $t_3$ | [start_transaction, $T_2$] |
| $t_4$ | [read_item, $T_2$, Z] |
| $t_5$ | [start_transaction, $T_3$] |
| $t_6$ | [write_item, $T_3$, X] |
| $t_7$ | [write_item, $T_2$, Z] |
| $t_8$ | [commit, $T_2$] |
| $t_9$ | [write_item, $T_1$, Y] |
| $t_{10}$ | [start_transaction, $T_4$] |
| $t_{11}$ | [write_item, $T_4$, A] |
| $t_{12}$ | [write_item, $T_1$, Z] |
| $t_{13}$ | [commit, $T_1$] |
| $t_{14}$ | [write_item, $T_3$, Z] |
| $t_{15}$ | [commit, $T_3$] |
| $t_{16}$ | [write_item, $T_4$, B] |
| $t_{17}$ | [commit, $T_4$] |

i. Please determine the time points $t_a$ and $t_b$ in the log record for setting a checkpoint and a crash, respectively, to ensure that transaction $T_2$ remains unaffected, transactions $T_1$ and $T_3$ require redo operations, and transaction $T_4$ needs to be undone. Specify the times for $t_a$ and $t_b$, and briefly explain your reasoning. If it is impossible to create this scenario, please explain why. *(3 marks)*

ii. If the check point made immediately after $t_4$ and system crashes immediately after $t_9$, what should be done to the transactions $T_1, T_2, T_3, T_4$ when the system is restarted? Please briefly explain the reason. *(3 marks)*

**END OF EXAM PAPER**