COMSW4111_003_2024_03: Midterm

Version 1.0, 2024-10-16 5:00 AM

Instructions

- We will be managing time **very strictly.**
    - Do NOT open this cover page until the exam moderator/proctor states that you can begin.
    - You will be given a 10 minute warning and 5 minute warning before the exam time is complete.
    - You must stop writing immediately when the proctor announces that the exam is over. Failing to stop will result in substantial point deductions on the exam.
- You cannot use, look at, touch, … any electronic devices. All bags, books, coats, etc. should be in one of the collection areas at the front of the classroom.
- Your desk/writing area must be clear of everything other than this exam document, the cheat sheet we provided, and your one page "cheat sheet."
- Write your answers in the spaces provided after questions. If you need additional space, you may use the backside of a page. Please indicate that your answer continues on the backside of the page.

"A man who uses a great many words to express his meaning is like a bad marksman who instead of aiming a single stone at an object takes up a handful and throws at it in hopes he may hit." — Samuel Johnson

We will deduct points if your answers are not concise and to the point.

Written Questions

This section requires short, written answers. No question requires more than 5 sentences or bullet points.

W1 What are the three basic concepts in the ER data model?

ER model -- Database Modeling

- The ER data mode was developed to facilitate database design by allowing specification of an **enterprise schema** that represents the overall logical structure of a database.
- The ER data model employs three basic concepts:
  - entity sets,
  - relationship sets,
  - attributes.
- The ER model also has an associated diagrammatic representation, the **ER diagram**, which can express the overall logical structure of a database graphically.

W2 Give three benefits of using ER Diagrams for database projects.

ER Modeling – Reasonably Good Summary

Advantages of ER Model

Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.

Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.

Effective communication tool: It is an effective communication tool for database designer.

Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.

Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

Disadvantages of ER Model

Limited constraints and specification

Loss of information content: Some information be lost or hidden in ER model

Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.

No representation of data manipulation: It is difficult to show data manipulation in ER model.

Popular for high level design: ER model is very popular for designing high level design

No industry standard for notation

https://pctechnicalpro.blogspot.com/2017/04/advantages-disadvantages-er-model-dbms.html

Note:

- If you get to use Google to help with take home exams, HW, etc.
- I get to use Google to help with slides.

W3 SQL has *types,* e.g. *varchar(5).* ER modeling recommends that a columns values come from an *atomic domain*. Briefly explain the difference between a domain and a SQL data type. Briefly explain the concept of atomic domain. Provide a simple example to illustrate your answer.

A *domain* is the set of valid, allowed values for a column. This is usually a restriction on the values from all of the types possible values that are allowed. For example, US zip codes are of the form “10027” and “02108”. The column type is *varchar(5)* or *char(5),* but not all 5 character values are from the domain. “02109” is a valid zip code. The 5 character string “mouse” is not.

Atomic means that the values of the domain are not divisible into two domains that make sense. We discussed several examples in class. Telephone numbers are of the form “+1 212-555-1212” or “+44 020 7219 3000.” These are two domains: *country code* and the *telephone number.* A second example is the simple domain of a person’s “name.” Representing the attribute *name* as a *varchar(128)* with values of the form “Ferguson, Donald, F” is not atomic. This is 3 domains: *(last_name, first_name, middle_initial)*.

W4 Explain the concept of *Physical Data Independence.*

Physical Data Independence

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

For example, the logical representation of an entity type and the interaction with it is independent of *how the data is physically stored in the database or on disks/storage.*

W5 A fundamental type of database user is a *Database Administrator.* List three functions or Tasks that a database administrator performs.

Database Administrator

A person who has central control over the system is called a **database administrator (DBA)**, whose functions are:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

W6 Briefly explain the concepts of *natural key* and *surrogate key.*

A *natural key* is one or more columns/attributes of an entity that uniquely identify the entity. Assuming that in the entity domain, *email* is a unique, identifying value for *student(last_name, first_name, middle_name, email).* The attribute *email* is a *natural key.* Names are typically not guaranteed to be unique. So, if the form of the relation/table is *student(last_name, first_name, middle_name)*, we would need to add an arbitrary *surrogate key* to provide a primary key. This could be an auto-incremented column, a generated UUID, etc.

W7 Consider the SQL statement,

```
delete from instructor
where salary < (select avg (salary)
                from instructor);
```

Briefly explain why executing this statement may produce incorrect results.

The SQL engine processes the delete one row at a time. That is, the engine examines a row and if its salary is less than the computed average, the row is deleted. This deletion changes the average, however. Which means for the next row, the test/condition is different.

W8 Is it always possible to insert, delete or update that base tables that a SQL CREATE VIEW statement references? If not, give two reasons why it is not always possible. You can use the schema that accompanies the recommended textbook to provide an example.

No.

1. The underlying table(s) have a column that cannot be NULL and is not present in the view.
2. The view uses GROUP BY and aggregation operations.

Example: Consider the table *instructor(ID, name, dept_name, salary)* and assume that all columns are NOT NULL. We might create a view that hides the value of *salary* for privacy reasons. For example,

create view instructor_private as select ID, name, dept_name from instructor.

We cannot insert into *instructor_private* because the value for salary in the underlying table would be NULL, which is not allowed.

Example: Consider a view that replaces an instructor's salary with the percentage of the salary relative to the total salary paid by the department to instructors.

```
create view this_is_weird as

with one as (
   select dept_name, sum(salary) as salary_total from instructor
                                    group by dept_name
)
select
   id, name, dept_name,
   salary / (select salary_total from one
      where one.dept_name=instructor.dept_name) as percent_of_salary
from
   Instructor;
```

There would be no way to change the percentage and have the correct result on the underlying salaries.

W9 Consider Codd's Rule 10: Integrity Independence:
A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

List 3 concepts from the SQL Data Definition Language that implement this rule. What is an example of a problem caused when all integrity rules are only implemented in applications using the database.

1. Primary keys, unique keys.
2. Foreign keys.
3. check constraints

If the intrinsic constraints of the data model change, it is necessary to "find" all applications that update the data and correctly reimplement the constraints.

W10 Briefly explain the concepts of a *composite primary key.*

A *composite primary key* is a primary key whose definition contains more than one column. For example, consider sections of Columbia University courses

section(course_number, section_number, semester, year, enrollment, capacity).

The primary key would be the composite *(course_number, section_number, semester, year).*

W11 Why is examining the data currently in a table not always a valid approach to identifying possible keys?

Querying the existing data may lead to the incorrect conclusion/definition of keys. Consider a table containing the names of people. If the amount of data in the table is small, it may be the case that no two people in the table have the same name. But, in the space of all people, the name is not unique.

Relational Algebra

The questions in this section reference the following relations.

Note: The terms like **R.c** or **T.d** simply indicate the difference between a column **c** and an attribute in a row with the value “c.” We will not deduct points based on column names as long as your answer is clear and correct.

R

R.a	R.b	R.c
1	a	d
3	c	c
4	d	f
5	d	b
6	e	f

S

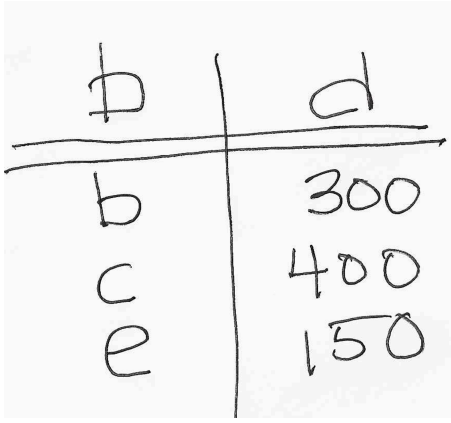
S.b	S.d
a	100
b	300
c	400
d	200
e	150

T

T.b	T.d
a	100
d	200
f	400
g	120

Note: You can hand draw your results. An example might be:

The result of S - T is



Hopefully, your handwriting and drawing skills are better than your professor’s

R1 What is the result of executing (π a, b (R)) ⋈ (T ∪ S)

$$(\pi_{a, b} (R)) \bowtie (T \cup S)$$

Execution time: 0 ms

R.a	R.b	T.d
1	'a'	100
3	'c'	400
4	'd'	200
5	'd'	200
6	'e'	150

R2 What is the result of executing

σ a > 3 (
(π a,b (R) ∪ π a,c (R))
) ⋊ S

$$\sigma_{a>3}(\pi_{a,b}(R) \cup \pi_{a,c}(R)) \bowtie S$$

Execution time: 2 ms

R.a	R.b	S.d
4	'd'	200
5	'd'	200
6	'e'	150
4	'f'	*null*
5	'b'	300
6	'f'	*null*

R3 The relation below is the the result of R anti-join T (R ▷ T)

R.a	R.b	R.c
3	c	c
6	e	f

Write an equivalent relational expression that does not use the anti-join operator: ▷

π a, b, c (σ d=null (R ⟕ T))

R4 In relational algebra, the symmetric difference △ of two relations is the set of tuples that are in either relation but not in both. That is S △ T = (S - T) ∪ (T - S). The result of S △ T is

b	d
'b'	300
'c'	400
'e'	150
'f'	400
'g'	120

Write a relational algebra expression that that produces S △ T. You may only use the following relational operators: π, σ, ←, ⋈, ⟕, ⟖, =, ≠. Hint: You may have to use the θ-condition on your joins.

__Note:__ **Just give full credit if it looks like they tried. I forgot how I did this one and spent 20 minutes trying to remember. It is too hard.**

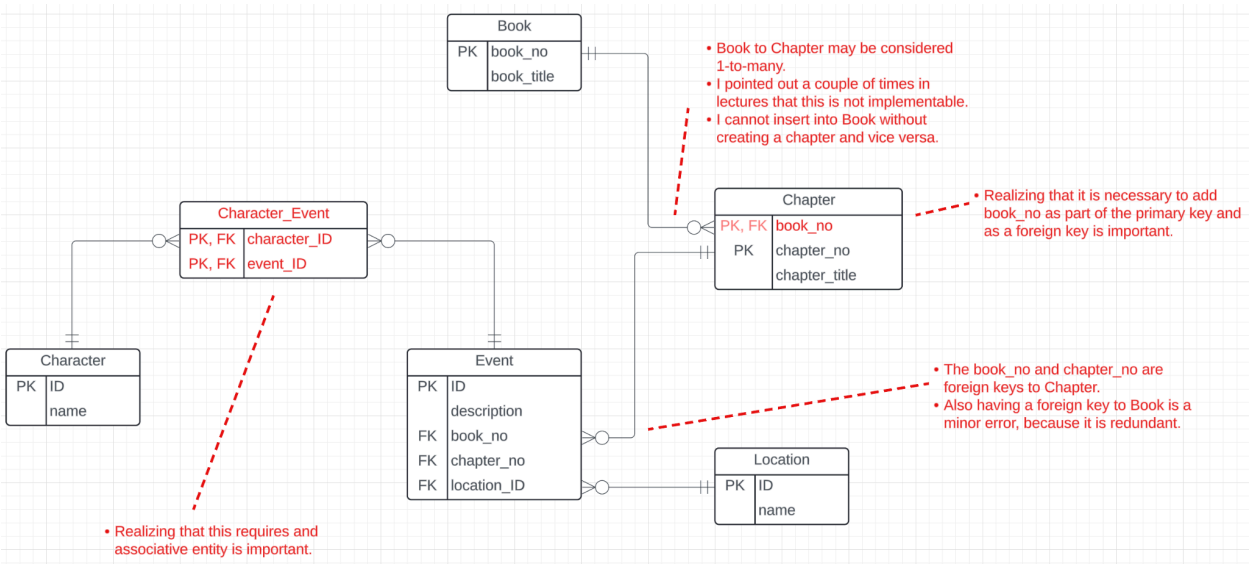
Entity Relationship Modeling

Consider a data model representing concepts from the Harry Potter series of books. The data model has the following entity types:

- **Character**:
    - ID: A unique ID number for the character.
    - Name: The name of the character.
- **Book**:
    - Book_No: The volume numbers, i.e. 1, 2, 3, 4, 5, 6, 7
    - Book_Title: The title of the book, e.g. “Harry Potter and the Philosopher's Stone,” “Harry Potter and the Chamber of Secrets.”
- **Chapter**:
    - Chapter_No: The chapter number within the volume.
    - Chapter_Title: The text of the chapter title. For example,
        - “The Boy Who Lived” is the 1st chapter of ““Harry Potter and the Philosopher's Stone.”
        - “At Flourish and Blotts” is the 4th chapter of “Harry Potter and the Chamber of Secrets.”
- **Location**:
    - ID: A unique ID for a location.
    - Name: The name of the location, e.g. “Hogwarts,” “The Forbidden Forest.”
- **Event**:
    - ID: A unique ID for the event.
    - Description: A short description of the event.

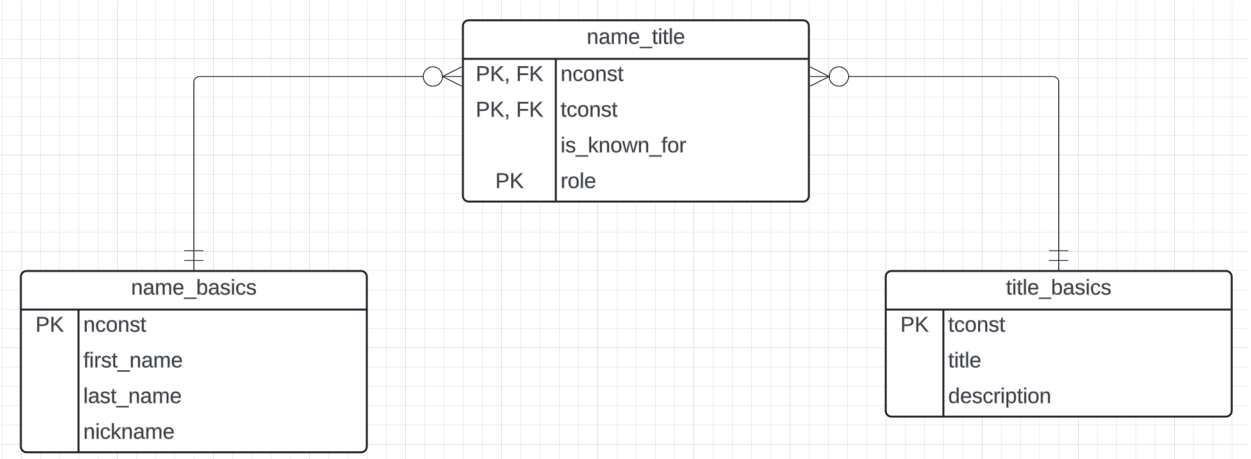
An specific **Event** occurs at *exactly one* location, in *exactly one* **Chapter** of *exactly one* **Book.** There *0, 1 or many* **Characters** participating in the event.

Draw a Crow’s Foot Logical Model Diagram that realizes the description above. You may have to add attributes to entity types. You must show relationships and foreign keys.



Realizing an ER Model in SQL DDL

Write SQL CREATE TABLE statements that create the schema for the following data model.



Note:

- *is_know_for* is a boolean.
- *role* is one of *Actor, Director, Writer, Producer.*
- All columns except *nickname* are NOT NULL.

You do not need to worry about defining indexes.

```
/*
name_basics is straightforward. If there is a mistake, it is
```

```
most likely forgetting that nickname can be NULL and all others
are NOT NULL.
*/
create table F24_examples.name_basics_midterm
(
   nconst     varchar(16) not null
       primary key,
   first_name varchar(64) not null,
   last_name  varchar(64) not null,
   nickname   varchar(64) null
);

/*
name_basics is straightforward. There are not traps
*/
create table F24_examples.title_basics_midterm
(
   tconst      varchar(16)   not null
       primary key,
   title       varchar(128)  not null,
   description varchar(1024) not null
);

/*
Some things to look for:
-   Foreign key constraints have names.
-   is_know_for can be BOOLEAN or TINYINT(1). BOOLEAN is just a shortcut
   for TINYINT(1).
-   The PRIMARY KEY has to contain (nconst,tconst, role)
-   role must either be an ENUM with the allowed values, or a foreign key
   into a table of the form (role_id, role_name) with role_id a PK and both
   columns NOT NULL. A table of the form (role_name) with role_name a PK is
   OK but less good.
*/
create table F24_examples.name_title_midterm
(
   nconst      varchar(16)                                          not null,
   tconst      varchar(16)                                          not null,
   is_know_for tinyint(1)                                           not null,
   role        enum ('Actor', 'Director', 'Writer', 'Producer') not null,
   primary key (nconst, tconst, role),
   constraint name_title_midterm___name_basics_fk
       foreign key (nconst) references f24_examples.name_basics_midterm
(nconst),
   constraint name_title_midterm___name_title_fk
       foreign key (tconst) references f24_examples.title_basics_midterm
(tconst)
);
```

SQL

The following are tables from the database associated with the textbook.

Takes

ID	course_i	sec_id	semeste	year	grad
128	CS-101	1	Fall	2017	A
128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	NULL

Department

dept_nam	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Course

course_ic	title	Dept_Nam	Credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Student

ID	name	dept_nam	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

SQL1 Produce a query result of the form:

(student_ID, no_of_courses, computed_credits, recorded_credits)

Where:

- student_ID is the student's *ID* from *student.*
- *name* is the student's *name* from *student.*
- *no_of_courses* is the count of the courses the student took from *takes.*
- *compute_credits* is the sum of the credits for courses the student took.
- *recorded_credits* is *tot_cred* from the table *student.*

Please put your SQL below.

```
/*
-  The JOIN path is Student --> Takes --> Course.
-  The table two below is optional. It is OK to directly JOIN course.
-  The GROUP BY and renaming of the columns is important.
-  If they do not use WITH and Common Table Expressions, we should deduct some
points.
   I was quite clear all semester that using WITH is a good practice and make
the queries
   easy to understand.
*/
with one as (select ID as student_ID,
                    name,
                    course_id
             from student
                      left join takes using (ID)),
    two as (select course_id, credits
            from course),
    three as (select *
              from one
                       join two using (course_id)),
    four as (select student_id,
                    name,
                    count(*)     as no_of_courses,
                    sum(credits) as computed_credits
             from three
             group by student_id, name)
select *,
      (select student.tot_cred from student where student.ID = student_id) as
recorded_credits
from four;
```

For the remaining questions, use the schema for *department* from the sample database associated with the textbook. The definition is:

```
create table if not exists db_book.department
(
   dept_name varchar(20)    not null
       primary key,
   building  varchar(15)    null,
   budget    decimal(12, 2) null,
   check (`budget` > 0)
);
```

SQL2 Write a statement to add a row *('Chem. Eng.', 'Taylor', 150,000)* to the table.

```
/*
-  It is OK to not have the columns after department, but if that
   is the case, the value must be in the order dept_name, building, budget.
-  Having the decimal .00 is best practice but not critical. Most databases
   will convert the type.
-  The exam has a "," in the number, which CANNOT be in the values.
*/
insert  into    department_midterm(dept_name, building, budget)
   value ('Chem. Eng.', 'Taylor', 150000.00);
```

SQL3 Write a statement that changes the building name for a department in *Taylor* to *Northwest* except for the department 'Comp. Sci.'

```
/*
-  This is pretty simple.
-  The WHERE clause with the AND is important.
*/
update department_midterm
   set building = 'Northwest'
   where building = 'Taylor' AND dept_name != 'Comp. Sci.';
```