COMSW4111_003_2024_3: Selected Final Exam Answers

Introduction

It is easy to find the correct answers to many of the questions to the W4111 - Introduction to Databases, Section 002, spring 2025 midterm exam. For many questions, slides from the lectures or from the slides associated with the recommended textbook directly provide answers. ChatGPT is extremely good and generating correct answers, although it tends to be to verbose.

The correct answers for some questions are not as easily determined. Moreover, the rubric or what we were specifically looking for might be unclear. This notebook contains answers and explanations for questions from the midterm exam.

Initialize

In [1]:
```
%load_ext sql
```

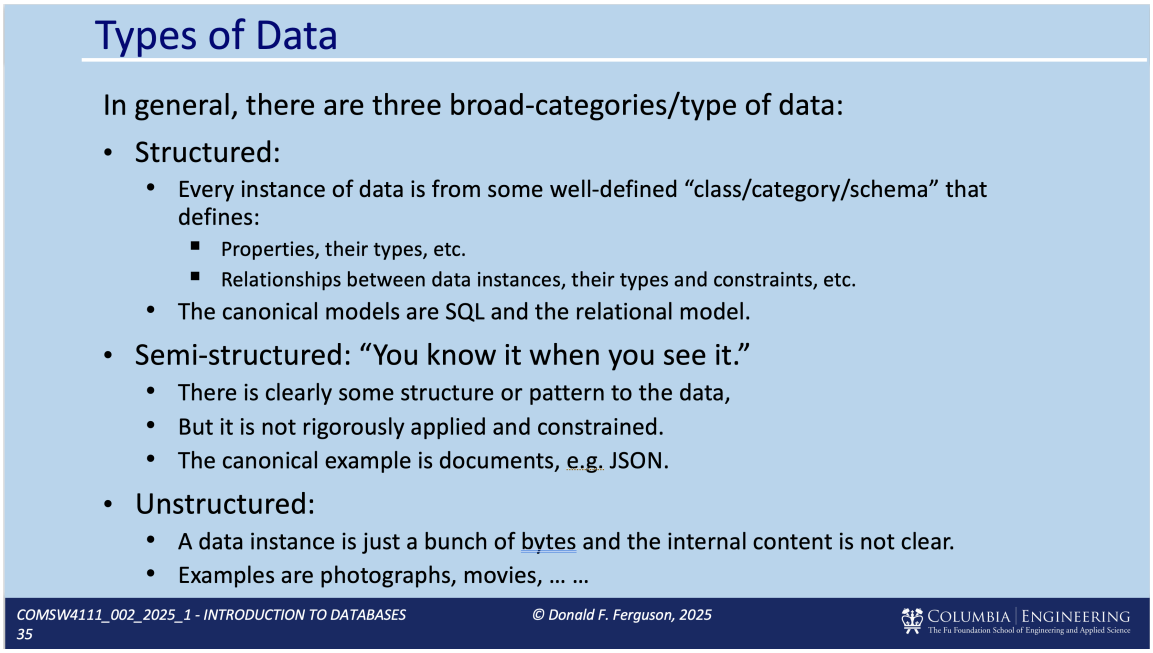
In [2]:
```
%sql mysql+pymysql://root:dbuserdbuser@localhost
```

Question 1

Question

Briefly explain the concepts of structured, semi-structured and unstructured data. Which type best describes spreadsheets and why?

Answer



Spreadsheets are *semi-structured* data. Spreadsheet data is tabular with columns and rows, and has some coherent structure. A spreadsheet lacks a well-formed schema that formally defines and constraints the structure, types and value.

Comments

Question 2

Question

A relational database table and a spreadsheet both have columns and rows and have a table-like structure. List 3 differences between a spreadsheet and a relational database table.

Answer

Complications:

1. The rows in a spreadsheet do not have to have the same number of columns.
2. The column values in a spreadsheet do not have to come from the same domain or be of the same type.
3. A spreadsheet does not have integrity constraints like primary or unique keys, check constraints, not null, etc.

Comments

There are other differences we would accept.

Question 3

Question

Consider the information below taken from a lecture slide. Give two examples of the complications in operators that NULL causes. Give an example of a query for which NULL is necessary to get a logically correct result.

Attributes

- The set of allowed values for each attribute is called the domain of the attribute
- Attribute values are (normally) required to be atomic; that is, indivisible
- The special value null is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

Answer

Complications:

1. `WHERE` class testing if a value if `NULL` must use `IS` and not `=` because `NULL = NULL` does not evaluate to `TRUE.`
2. If any value in an expression is `NULL,` the expression evaluates to `NULL.` For example `SELECT concat(nameFirst, ' ', nameMiddle, ' ', nameLast) AS nameFull FROM people` evaluates to `NULL` and not the string `nameFirst nameLast` if `nameMiddle` is `NULL.` This requires a test for `NULL` in string expressions. So, the correct expression is `SELECT concat(nameFirst, ' ', IFNULL(nameMiddle, ' '), ' ', nameLast) AS nameFull FROM people`.

In aggregate operations in `GROUP BY` queries, `NULL` does not form part of the computation of the result. If a null indicator like `-1` were used instead of `NULL,` sums, averages, etc. would produce an incorrect answer.

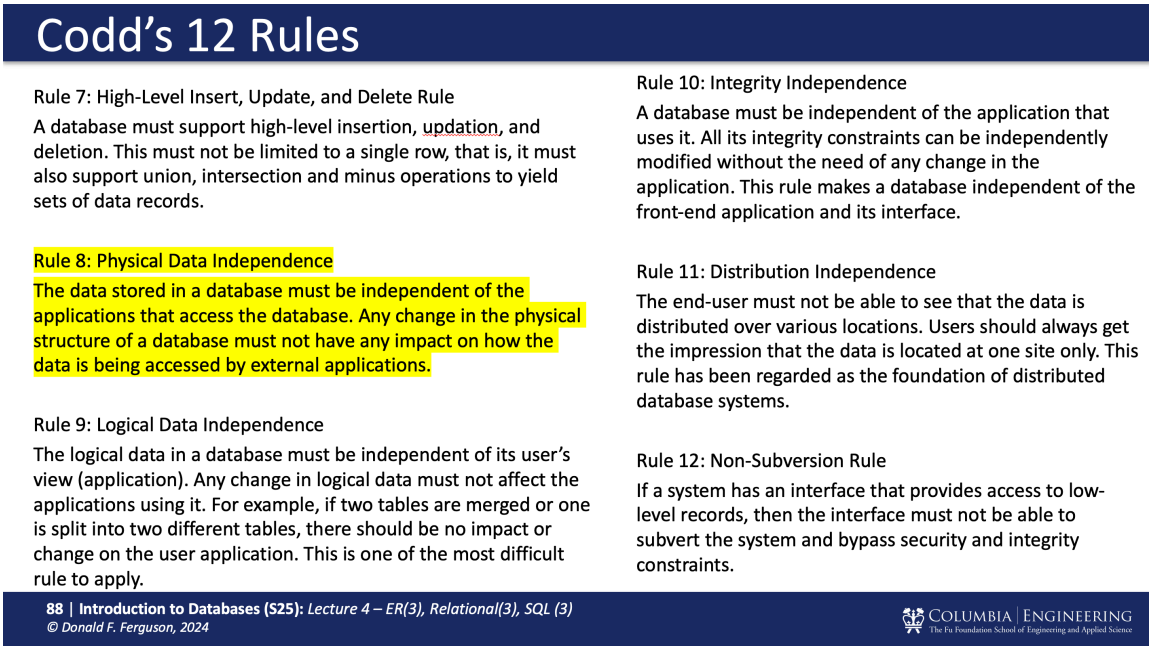
Comments

Question 4

Question

Briefly explain the concept of Physical Data Independence.

Answer



Comments

Question 5

Question

Briefly explain what a database administrator is and what tasks the administrator performs. Give 3 examples of SQL statements that an administrator is likely to use that a naïve user would not.

Answer

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

Comments

Question 6

Question

Briefly explain why all associative entities are also weak entities.

Answer

"A weak entity set is one whose existence is dependent on another entity, called its identifying entity."

All associative entities must be weak entities because they depend on the existence of the entities they associate — they cannot exist independently.

More specifically, the attributes in the primary key of the associative entity are typically `NOT NULL` foreign keys referencing the entities they associate.

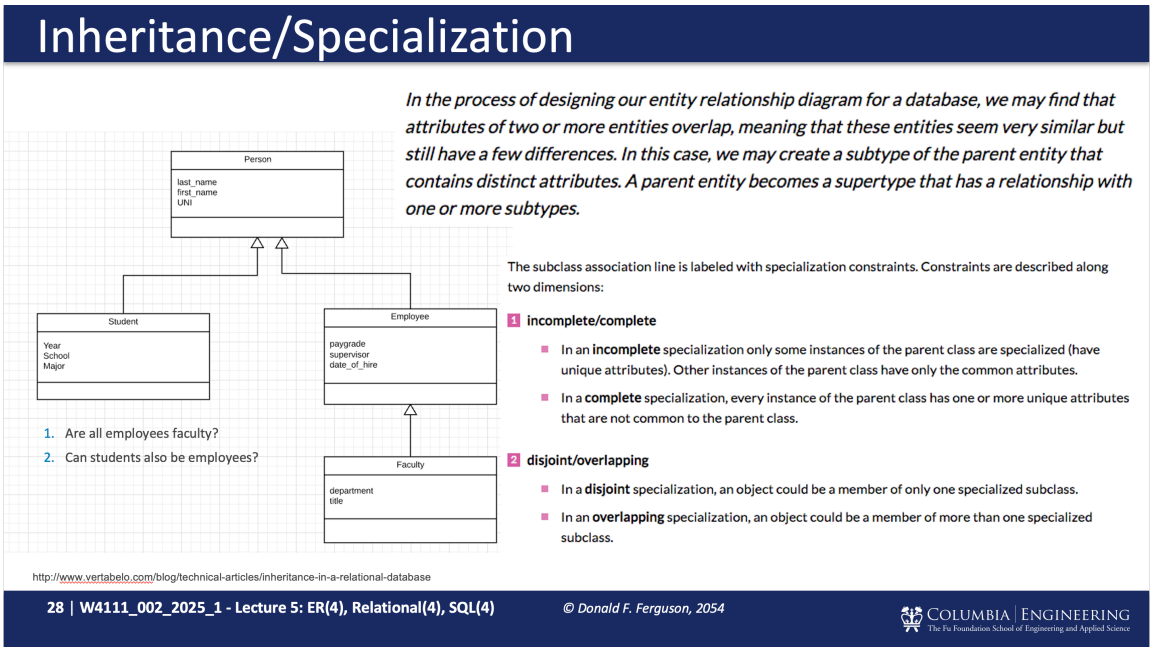
Comments

Question 7

Question

With respect to inheritance (specialization/generalization) in ER modeling, briefly explain the concepts of incomplete/complete and disjoint/overlapping.

Answer



Comments

Question 8

Question

What are 3 reasons/motivations for using/defining views in a SQL database.

Answer

1. If a table has sensitive information, e.g. salary, SSNO, the database administrator can define a view that allows users to query the table without being able to access the sensitive information.
2. Complex queries are often required to facilitate simple queries. Simple users can produce the easy queries but cannot are not skilled enough to write the complex queries. The the *Classic Models* database, a complex query is necessary to support

a simple query like `SELECT * FROM revenue_country WHERE revenue >= 1000000`. Producing the table `revenue_country` requires several joins an an aggregation. The DBA can define a view from the complex query, making simple queries on `revenue_country` possible.
3. Views enable changing and evolving the underlying detailed, physical data model without breaking existing applications and queries.

Comments

Question 9

Question

Briefly explain 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. What is the benefit of making the database independent of front-end applications?

Answer

Without integrity constraints defined in the schema and enforced by the database, maintaining data integrity requires:

1. Users to know the constraints and write correct queries.
2. All programmers writing application to understand the constraints and implement correct code.

One benefit is that relying on users and programmers to correct manipulate data is error prone. Second, if the constraints change, it is necessary to modify possibly many applications that use the data.

Comments

Question 10

Question

Briefly explain the concepts of a database connection and database session.

Answer

Some Terms

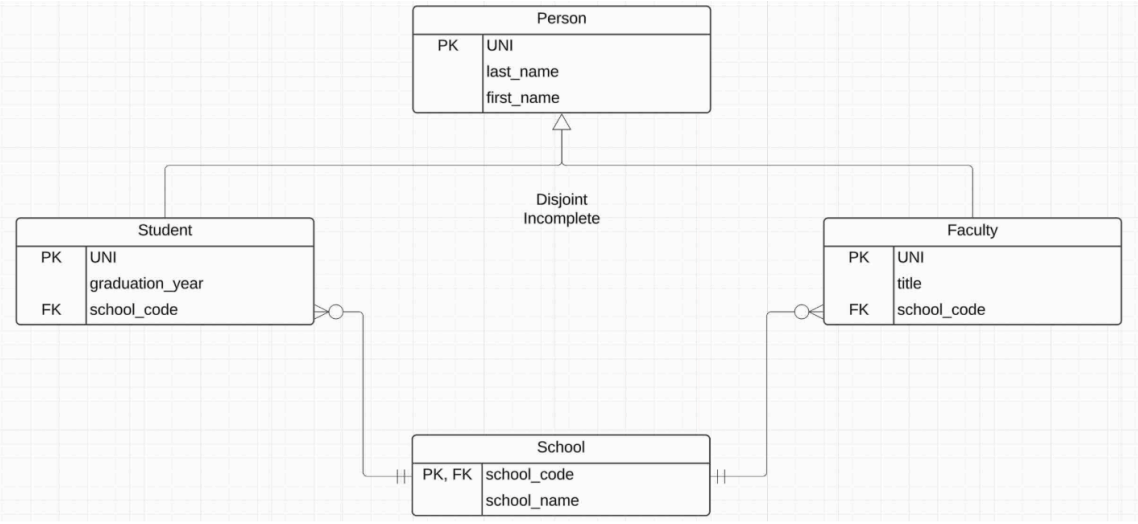
- "A database connection is a facility in computer science that allows client software to talk to database server software, whether on the same machine or not. A connection is required to send commands and receive answers, usually in the form of a result set." (https://en.wikipedia.org/wiki/Database_connection)
- Session:
  - "Connection is the relationship between a client and a MySQL database. Session is the period of time between a client logging in (connecting to) a MySQL database and the client logging out (exiting) the MySQL database." (https://stackoverflow.com/questions/8797724/mysql-concepts-session-vs-connection)
  - "A session is just a result of a successful connection."
- Network protocols are layered. You can think of:
  - Connection as the low-level network connection.
  - Session is the next layer up and has additional information associated with it, e.g. the user.
- Connection libraries like pymysql sometimes blur the distinction.

Comments

Question 11

Question

Translate the following ER diagram into SQL DDL statements. You must use the one-table solution/design. You may have to add constraints, additional columns, etc. You can assume that the data type for all columns is VARCHAR(64). Add text notes to document any design decisions or assumptions. You do not need to write procedures, triggers or functions.



Answer

```
/*
 The one table solution, not surprisingly, requires all of
the data to be in one table.
 */
create table midterm_person
(
    UNI varchar(64) primary key not null,

    /*
     NOT NULL is not required for a correct answer. This is a
design choice I made.
     */
    last_name varchar(64) not null,
    first_name varchar(64) not null,

    /*
     Since this is a one table solution, this column will be
null for faculty
     */
    graduation_year varchar(64) null,

    /*
     Since this is a one table solution, this columns will be
null for student
     */
    title varchar(64) null,

    /*
     The || ending indicates that this value must be not
null.
     */
    school_code varchar(64) NOT NULL
);

create table midterm_school
(
    school_code varchar(64) primary key not null,

    /*
     Making this not null is a design choice.
     */
    school_name varchar(64)             not null
);

/*
 Foreign key from person to school.
 */
alter table midterm_person
```

```
    add constraint
midterm_person_midterm_school_school_code_fk
        foreign key (school_code) references midterm_school
(school_code);

/*
 This foreign key does not make any sense. If you observed
and noted that it did not make sense,
 you could have omitted it and explained why.

 This definition makes it impossible to insert any data. I
cannot create the
 first school without a student or faculty and vice versa.
 */
alter table midterm_school
    add constraint
midterm_school_midterm_person_school_code_fk
        foreign key (school_code) references midterm_person
(school_code);

/*
 Since the specialization is incomplete, there are "persons"
that are neither a student nor a faculty.
 */
create or replace view person_view as
    select uni, last_name, first_name from midterm_person
        where graduation_year is NULL and title is NULL;

 /*
 The existence of graduation_year value indicates student.
 */
create or replace view student_view as
    select uni, last_name, first_name, graduation_year from
midterm_person
        where graduation_year is not NULL;

 /*
 The existence of title value indicates faculty.
 */
create or replace view faculty_view as
    select uni, last_name, first_name, title from
midterm_person
        where title is not NULL;

/*
 Since the specialization is disjoint, a person cannot be
both.
 */
alter table midterm_person
    add constraint check_disjoint
        check (((graduation_year is NULL) or (title is
```

```
NULL)));
```

Comments

Question 12

Question

Consider the two subsets of the IMDB data for the entity sets name_basics and title_basics. Write DDL statements to define the schema that you would use to hold the data. knownForTitles contains comma delimited tconst values from title_basics.

You may have to add constraints, additional columns, tables, etc. You can assume that the data type for all columns is VARCHAR(64), except for primaryName. primaryName is VARCHAR(129). Add text notes to document any design decisions or assumptions.

name_basics

nconst	primaryName	primaryProfession	knownForTitles
nm0389698	B.J. Hogg	actor,music_department	tt0986233,tt1240982,tt0970411,tt
nm0269923	Michael Feast	actor,composer,soundtrack	tt0120879,tt0472160,tt0362192,tt
nm0727778	David Rintoul	actor,archive_footage	tt1139328,tt4786824,tt6079772,tt
nm6729880	Chuku Modu	actor,writer,producer	tt4154664,tt2674426,tt0944947,tt
nm0853583	Owen Teale	actor,writer,archive_footage	tt0102797,tt0944947,tt0485301,tt
nm0203801	Karl Davies	actor,producer	tt3428912,tt7366338,tt0944947,tt
nm8257864	Megan Parkinson	actress,director,writer	tt0944947,tt26934073,tt4276618,
nm0571654	Fintan McKeown	actor	tt0112178,tt0110116,tt0166396,tt0
nm1528121	Philip McGinley	actor,archive_footage	tt0944947,tt1446714,tt0053494,tt
nm0000980	Jim Broadbent	actor,writer,producer	tt0203009,tt1431181,tt1007029,tt0
nm0649046	Deobia Oparei	actor,archive_footage	tt1343727,tt0419706,tt0118929,tt0
nm1783582	Sahara Knite	actress,archive_footage	tt0944947,tt9814116,tt15249564,t
nm8127149	Nathanael Saleh	actor,soundtrack	tt5028340,tt0944947,tt1635327,tt
nm1074361	Luke Roberts	actor,producer,composer	tt5809150,tt2375692,tt0944947,tt

nconst	primaryName	primaryProfession	knownForTitles
nm3586035	Maisie Williams	actress,producer,soundtrack	tt4682266,tt0944947,tt3294200,t
nm0538869	Patrick Malahide	actor,writer,producer	tt0116908,tt0143145,tt0090521,tt(
nm4207240	Phil Barnhill	actor	tt1785299
nm0568400	Ian McElhinney	actor,director,soundtrack	tt0116477,tt0944947,tt3748528,tt(

title_basics

tconst	titleType	primaryTitle	runtimeMinutes
tt0054518	tvSeries	The Avengers	50
tt0054571	tvSeries	Three Live Wires	30
tt0055556	movie	Two Living, One Dead	105
tt0056105	movie	The Swingin' Maiden	98
tt0056696	movie	Young and Willing	110
tt0057435	movie	The Punch and Judy Man	96
tt0058142	movie	Girl with Green Eyes	91
tt0058596	movie	Smokescreen	70
tt0059106	movie	Die! Die! My Darling!	97
tt0059191	tvEpisode	For the West	75
tt0059607	movie	The Pleasure Girls	88
tt0059660	movie	Rotten to the Core	90
tt0060039	tvMiniSeries	The Wars of the Roses	
tt0060094	movie	The Alphabet Murders	90
tt0060100	tvEpisode	Amerika	120
tt0060506	tvEpisode	A Hero of Our Time	85
tt0060532	movie	Time Lost and Time Remembered	91
tt0060866	movie	The Psychopath	82
tt0061170	movie	Walk Don't Run	114

Answer

```
create table midterm_name_basics
(
    nconst      varchar(64) not null,
    first_name  varchar(64) not null,
    last_name   varchar(64) not null,

    /* This column is clearly auto-generated. */
    primaryName varchar(129) as (concat(first_name, ' ',
last_name)) stored,

    constraint midterm_name_basics_pk
        primary key (nconst)
);

create table midterm_title_basics
(
    tconst          varchar(64) not null,

    /*
        This column, based on the data, is an enum. Defining
a table of titleTypes and using
        an associative entity would also be OK.
    */
    titleType       ENUM('tvSeries', 'tvEpisode', 'movie')
not null,

    primaryTitle    varchar(256) not null,
    runtimeMinutes  int not null,

    constraint midterm_name_basics_pk
        primary key (tconst),

    /* This seems reasonable. */
    check (runtimeMinutes > 0)
);

create table midterm_professions
(
    profession_id    int primary key auto_increment,
    profession_label varchar(64) not null
);

/*
    An associative entity is the approach to handle the
multi-values attribute.
*/
create table midterm_names_professions
(
    nconst              varchar(64)          not null,
    profession_id       int                  not null,
```

```
    constraint midterm_names_profession_pk
        primary key (nconst, profession_id),

    constraint midterm_names_professions_names_fk
        foreign key (nconst) references midterm_name_basics
(nconst),
    constraint midterm_names_professions_profession_fk
        foreign key (profession_id) references
midterm_professions (profession_id)
);

/*
    This one is really tricky. An associative entity is the
correct way to handle multi-values,
    BUT knownFor is a property of the association.
*/
create table midterm_names_titles
(
    nconst     varchar(64)           not null,
    tconst     varchar(64)           not null,
    isKnownFor boolean default false not null,
    constraint midterm_names_titles_pk
        primary key (nconst, tconst),
    constraint
midterm_names_titles_midterm_name_basics_nconst_fk
        foreign key (nconst) references midterm_name_basics
(nconst),
    constraint
midterm_names_titles_midterm_title_basics_tconst_fk
        foreign key (tconst) references midterm_title_basics
(tconst)
);
```

Comments

Question 13

Question

Translate the following relational schema definition in an SQL statement. You can assume all data types are VARCHAR(32).

$$section(\underline{courseNo},\ \underline{sectionNo},\ \underline{semester},\ \underline{year},\ timeSlotID,\ capacity) \quad (1)$$

Answer

```
create table midterm_section
(
    courseNo   varchar(32) not null,
    sectionNo  varchar(32) not null,
    semester   varchar(32) not null,
    `year`     varchar(32) not null,
    timeSlotID varchar(32) not null,

    /* The question said this can be varchar(32) and we would
accept it. */
    capacity   varchar(32) not null,

    constraint midterm_section_pk
        primary key (courseNo, sectionNo, semester, `year`)
);
```

Comments

Question 14

Question

The left semijoin (⋉) is a join similar to the natural join and written as

R ⋉ T

where R and T are relations. The result is the set of all tuples in R for which there is a tuple in T that is equal on their common attribute names. The difference from a natural join is that other columns of T do not appear.

The result of R ⋉ T is

R.a	R.b	R.c
1	'a'	'd'
4	'd'	'f'
5	'd'	'b'

Write a relational algebra expression that produces the same result.

Answer

$$\pi_{R.a,\ R.b,\ R.c}\ (\ R \bowtie T\ )$$

1 ms

R.a	R.b	R.c
1	'a'	'd'
4	'd'	'f'
5	'd'	'b'

Comments

Question 15

Question

The RelaX calculator relation for the section information from the database associated with the recommended textbook is:

section.course_id	section.sec_id	section.semester	section.year	section.building	se
'BIO-101'	1	'Summer'	2009	'Painter'	51
'BIO-301'	1	'Summer'	2010	'Painter'	51
'CS-101'	1	'Fall'	2009	'Packard'	10
'CS-101'	1	'Spring'	2010	'Packard'	10
'CS-190'	1	'Spring'	2009	'Taylor'	31
'CS-190'	2	'Spring'	2009	'Taylor'	31
'CS-315'	1	'Spring'	2010	'Watson'	12
'CS-319'	1	'Spring'	2010	'Watson'	10
'CS-319'	2	'Spring'	2010	'Taylor'	31
'CS-347'	1	'Fall'	2009	'Taylor'	31

Two sections conflict they occur in the same year, semester and time_slot_id. A section does not conflict with itself. Write a relational algebra expression that computes sections in the year 2009 that conflict. Your query should show each pair only once. My answer is:

one.course_id	one.sec_id	one.semester	one.year	one.building	one.room_number
'PHY-101'	1	'Fall'	2009	'Watson'	100

Note that the following result is incorrect

one.course_id	one.sec_id	one.semester	one.year	one.building	one.room_number
'CS-347'	1	'Fall'	2009	'Taylor'	3128
'PHY-101'	1	'Fall'	2009	'Watson'	100

Answer

```
σ one.course_id > two.course_id ∧ one.year=2009
(
        (ρ one (section))
                ⋈ one.time_slot_id=two.time_slot_id ∧
                        one.year = two.year ∧
                        one.semester = two.semester
        (ρ two (section))
)
```

Comments

There are a few subtle "tricks" to this query. I gave examples of the tricks in lecture.

1. Since this is a self-join, it is necessary to alias/rename the tables.
2. Courses cannot conflict with themselves. This would suggest using something like `σ one.course_id ≠ two.course_id` but this will produce a table of the form AB and BA because it is a self-join. Using `one.course_id > two.course_id` results in choosing only one of the conflicting pair.

Question 16

Question

The following is the initial definition of an SQL table.

```
create table if not exists s2025_examples.person_midterm
(
    UNI             varchar(16)  null,
    last_name       varchar(64)  null,
    first_name      varchar(64)  null,
    preferred_email varchar(128) null
);
```

Modify the definition so that:

1. The UNI is automatically computed on insert from the first character of the first name and the first character of the last name plus 1 more than the UNI of existing persons with the same initials.
2. It is not possible to change a UNI.
3. There is an additional field default_email which is of the form UNI@columbia.edu.

Modifying the definition may require additional DDL statements.

For example, if the table data is currently:

UNI	last_name	first_name	preferred_email	default_email
df1	Ferguson	Donald	dff@shire.gov	df1@columbia.edu
df2	Franklin	Douglas	doug@mordor.org	df2@columbia.edu

Executing the statement

```
insert into person_midterm(first_name, last_name,
preferred_email)
        values('Dilbert', 'Frankfurt', 'dilbie@orthanc.com')
```

would produce

UNI	last_name	first_name	preferred_email	default_email
df1	Ferguson	Donald	dff@shire.gov	df1@columbia.edu
df2	Franklin	Douglas	doug@mordor.org	df2@columbia.edu
df3	Frankfurt	Dilbert	dilbie@orthanc.org	df3@columbia.edu

Answer

```
/*
    The first part of the question requires a function.
    We did an example in class.

    We are not concerned about define or deterministic.
*/
create
    definer = root@localhost function compute_uni(first_name
varchar(64), last_name varchar(64)) returns varchar(12)
    deterministic
begin
    declare new_uni varchar(12);
    declare first_initial varchar(1);
    declare last_initial varchar(1);
    declare prefix varchar(3);
    declare prefix_count int;

    set first_initial = lower((select substr(first_name, 1,
1)));
    set last_initial = lower((select substr(last_name, 1,
1)));
    set prefix = concat(first_initial, last_initial, "%");

    select count(*) into prefix_count from person_midterm
where uni like prefix;
    set new_uni = concat(first_initial, last_initial,
prefix_count);
    return new_uni;
end;


/*
    Condition 1 also requires automatically setting the UNI
on insert.
*/
create definer = root@localhost trigger set_uni
    before insert
    on person_midterm
    for each row
```

```
begin
    set new.UNI = compute_uni(new.first_name, new.last_name);
end;


/*
    Condition 2 requires a trigger on update to prevent a
change.

    We are not concerned with the details of how you signal
an error.
*/
create definer = root@localhost trigger protect_uni
    before update
    on person_midterm
    for each row
begin
    if (new.uni != old.uni) then
        SIGNAL SQLSTATE '50001'
            SET MESSAGE_TEXT = 'UNI is invariant.',
MYSQL_ERRNO = 1234;
    end if;
end;


/*
    Condition 3 requires an auto-generated value for
default_email.
*/
alter table person_midterm
    add default_email varchar(64) as (concat(UNI, '@',
'columbia.edu')) stored;
```

Comments

Like the other questions, we are focusing on your understanding of how to apply the concepts, not the details of the correct syntax.

Question 17

Question

The data for the course and prereq tables are below.

course_id	prereq_id
BIO-301	BIO-101
BIO-399	BIO-101

course_id	prereq_id
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

course_id	title	dept_name	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

Write an SQL statement the produces a table of the form

`course_prereq(course_id, course_title, prereq_id, prereq_title)`

The table contains the `course_id` and `title` of a course, and the `course_id` of the `course` and the `prereq's`` course title.```

You cannot use a JOIN. The result of my query is

course_id	course_title	prereq_id	course_title
BIO-301	Genetics	BIO-101	Intro. to Biology
BIO-399	Computational Biology	BIO-101	Intro. to Biology
CS-190	Game Design	CS-101	Intro. to Computer Science
CS-315	Robotics	CS-101	Intro. to Computer Science
CS-319	Image Processing	CS-101	Intro. to Computer Science

course_id	course_title	prereq_id	course_title
CS-347	Database System Concepts	CS-101	Intro. to Computer Science
EE-181	Intro. to Digital Systems	PHY-101	Physical Principles

Answer

```
/*
    Since course_id is a column in both course and prereq, it
is necessary to be explicit about the tables.
*/
select
    course_id,
    (select title from course where
course.course_id=prereq.course_id) as course_title,
    prereq_id,
    (select title from course where
course.course_id=prereq.prereq_id) as prereq_title
from prereq;
```

Comments

Question 18

Question

The data for instructor and teaches is below.

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000

ID	name	dept_name	salary
98345	Kim	Elec. Eng.	80000

ID	course_id	sec_id	semester	year
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
10101	CS-101	1	Fall	2017
45565	CS-101	1	Spring	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
10101	CS-315	1	Spring	2018
45565	CS-319	1	Spring	2018
83821	CS-319	2	Spring	2018
10101	CS-347	1	Fall	2017
98345	EE-181	1	Spring	2017
12121	FIN-201	1	Spring	2018
32343	HIS-351	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017

Write a SQL statement that produces the following table where `ID` and `name` come from `instructor.` The column `sections` is a semi-colon delimited string where each entry is of the form `'course_no-section-semester-year'` for a section the instructor teaches.

ID	name	sections
10101	Srinivasan	CS-101-1-Fall-2017;CS-315-1-Spring-2018;CS-347-1-Fall-2017
12121	Wu	FIN-201-1-Spring-2018
15151	Mozart	MU-199-1-Spring-2018
22222	Einstein	PHY-101-1-Fall-2017
32343	El Said	HIS-351-1-Spring-2018
33456	Gold	NULL
45565	Katz	CS-101-1-Spring-2018;CS-319-1-Spring-2018
58583	Califieri	NULL
76543	Singh	NULL

ID	name	sections
76766	Crick	BIO-101-1-Summer-2017;BIO-301-1-Summer-2018
83821	Brandt	CS-190-1-Spring-2017;CS-190-2-Spring-2017;CS-319-2-Spring-2018
98345	Kim	EE-181-1-Spring-2017

Answer

```
with one as (
    select
        ID,
        concat(course_id, '-', sec_id, '-', semester, '-',
year) as sections
    from teaches),
two as (
    select instructor.ID, name, one.sections from instructor
left join one using(ID)
    )
select
    two.ID, name, group_concat(sections, ';') as sections
from
    two
group by ID, name
order by ID;
```

Comments

There were a few specific things we were looking for on this question:

1. A basic understanding of `concat`.
2. A basic understanding of `group by`.
3. The critical realization that the `NULL` values require a left join on with `instructor.`