CS319, Hugh Darwen's Lectures: Exam Hints

How to Handle Missing Information Without Using NULL

No hints. See past papers.

Temporal Data and The Relational Model

Note: From 2012 you can expect the exam question to address the SQL features that were added to the lecture that year.

The purpose of these lectures was to describe the special problems arising in connection with data representing not only recorded facts but also the intervals in time *throughout* which those facts are believed to hold (or to have held) true. The problems were described in the context of relational databases in particular. A general approach to their solution was presented, using interval types, relations with attributes of such types, and a general extension (the USING construct) to each of the operators of the database language (**Tutorial D** being used to illustrate these proposals).

The book *Temporal Data and The Relational Model* (by C.J. Date, Nikos A. Lorentzos, and myself) has exercises at the end of each chapter. In the selection from these exercises that follows:

- The exercise number includes the chapter number, which in turn gives you a reference to the relevant slides in my lectures.
- The ones shown in bold are ones that you might be able to answer, having attended my lectures and studied the handouts.
- The ones not shown in bold are either **Tutorial D** exercises that I do not expect you to be able to take on in exam conditions (you might like to have a go at some of them!) or are on points that were probably not covered adequately in my lectures but nevertheless you might like to think about.

3.1 What is a timestamped proposition? Give some examples.

Think of the propositions represented by tuples in the various relvars (nontemporal, semitemporal and fully temporal) used as examples in my lectures.

3.2 Distinguish between "valid time" and "transaction time."

Recall that we proposed "stated time" and "logged time" to replace these terms.

3.3 What is a time quantum? What is a time point?

3.4 What do you understand by the term granularity?

A term you might encounter in the literature, but I preferred "scale" as being more self-explanatory. What scale was being used in nearly all my examples?

3.5 What do you understand by the terms "beginning of time" and "end of time"?

4.1 State as precisely as you can the predicates for:

a. Relvars S and SP as illustrated in [slide 9 in the handout];

b. Relvars S_SINCE and SP_SINCE as illustrated in [slide 10 in the handout];

c. Relvars S_FROM_TO and SP_FROM_TO as illustrated in [slide 11 in the handout].

4.3 Write **Tutorial D** expressions for the following queries on the database illustrated in [slide 10]:

a. Get supplier numbers for suppliers who are currently able to supply at least two different parts, showing in each case the date since when they have been able to do so.

b. Get supplier numbers for suppliers who are currently unable to supply at least two different parts, showing in each case the date since when they have been unable to do so.

What about analogs of these two queries on the database illustrated in [slide 11]? At least try to state such analogs in natural language, even if you decide you would rather not attempt to come up with any corresponding **Tutorial D** formulations.

5.1 State as precisely as you can the predicates for relvars S_DURING and SP_DURING as illustrated in [slide 12].

5.2 List as many advantages as you can think of in favour of replacing FROM-TO attribute pairs by individual DURING attributes.

5.4 Define the terms point type and interval type. Complete the following sentence in your own words: "A type T is usable as a point type if ... ."

5.5 Is a singleton scalar type—i.e., one containing just a single scalar value—a valid point type?

5.6 Let *i* be an interval. Define the operators BEGIN(*i*), END(*i*), and $p \in i$.

5.7 What is a unit interval?

5.9 Consider the following database definition (for a hypothetical university database about students and courses):

```
VAR COURSE RELATION
{ COURSE# COURSE#,
  CNAME NAME,
  AVAILABLE DATE }
KEY { COURSE# } ;

VAR CANCELLED_COURSE RELATION
{ COURSE# COURSE#,
  CANCELLED DATE }
KEY { COURSE# }
FOREIGN KEY { COURSE# } REFERENCES COURSE ;
```

```
VAR STUDENT RELATION
{ STUDENT# STUDENT#,
  SNAME NAME,
  REGISTERED DATE }
KEY { STUDENT#, REGISTERED } ;

VAR UNREG_STUDENT RELATION
{ STUDENT# STUDENT#,
  UNREGISTERED DATE }
KEY { STUDENT#, UNREGISTERED } ;

VAR ENROLMENT RELATION
{ COURSE# COURSE#,
  STUDENT# STUDENT#,
  ENROLLED DATE }
KEY { COURSE#, STUDENT# }
FOREIGN KEY { COURSE# } REFERENCES COURSE
FOREIGN KEY { STUDENT# } REFERENCES STUDENT ;

VAR COMPLETED_COURSE RELATION
{ COURSE# COURSE#,
  STUDENT# STUDENT#,
  COMPLETED DATE,
  GRADE GRADE }
KEY { COURSE#, STUDENT# }
FOREIGN KEY { COURSE# } REFERENCES COURSE ;
```

The predicates are as follows:

- COURSE: *Course COURSE#, named CNAME, became available on date AVAILABLE.*
- CANCELLED_COURSE: *Course COURSE# ceased to be available on date CANCELLED.*
- STUDENT: *Student STUDENT#, named SNAME, registered with the university on date REGISTERED.*
- UNREG_STUDENT: *Student STUDENT# left the university on date UNREGISTERED.*
- ENROLMENT: *Student STUDENT# enrolled in course COURSE# on date ENROLLED.*
- COMPLETED_COURSE: *Student STUDENT# completed course COURSE# on date COMPLETED, achieving grade GRADE.*

Selector operator GRADE, which takes a single argument of type INTEGER (with value in the range 1 through 5), is available for type GRADE.

a. Assuming this database constitutes a record of the relevant part of a typical university's business, what additional constraints (expressed in natural language) might be required?

b. Suppose the following relvar is added, with the intent (eventually) of using it to replace relvar COMPLETED_COURSE:

```
VAR STUDIED RELATION
{ COURSE# COURSE#,
  STUDENT# STUDENT#,
  DURING INTERVAL_DATE,
  GRADE GRADE }
KEY { COURSE#, STUDENT# }
FOREIGN KEY { COURSE# } REFERENCES COURSE ;
```

The predicate is: *Student STUDENT# studied course COURSE# during interval DURING, achieving grade GRADE.*

Write a **Tutorial D** query, making use of relvars ENROLLMENT and COMPLETED_COURSE, whose result corresponds to exactly this predicate (and can therefore usefully be assigned to relvar STUDIED).

c. Write a **Tutorial D** definition for a relvar called COURSE_AVAILABILITY that combines relvars COURSE and CANCELED_COURSE analogously to the way STUDIED combines relvars ENROLLMENT and COMPLETED_COURSE. Include at least one KEY specification and all appropriate FOREIGN KEY specifications.

6.1 Let *i* be an interval. Define the operators PRE(*i*), POST(*i*), and *p* FROM *i*.

6.2 If *a* and *b* are relations (or sets), then it is a fact that *a* INTERSECT *b* ≡ *a* MINUS (*a* MINUS *b*)

Is the same true if *a* and *b* are intervals?

6.3 Let *i* be a value of type INTERVAL_INTEGER. Write an expression to obtain the interval that results from extending *i* by its own length in both directions (e.g., [5:7] becomes [2:10]). In what circumstances will evaluation of your expression fail at run time?

6.4 Again, let *i* be a value of type INTERVAL_INTEGER. Write an expression to obtain the interval that is the middle third of *i*. You can assume that COUNT(*i*) is a multiple of three.

6.5 Let *i1, i2,* and *i3* be intervals such that there is a single interval *i4* consisting of every point *p* such that *p* ∈ *i1* or *p* ∈ *i2* or *p* ∈ *i3*. Write an expression, using operators defined in this chapter, that when evaluated yields *i4*. (Beware of the trap!)

6.6 Given the relvar STUDIED from Exercise 5.9, write a **Tutorial D** query whose result shows for every grade the average length of study for all students who achieved that grade. (We are assuming for the sake of this exercise that the university's courses are "self-study" ones and are completed at the student's own pace.)

6.7 Given the relvars STUDENT and ENROLLMENT from Exercise 5.9, write a query whose result is a relation pairing the student number of each student who has enrolled in at least two courses with the interval from that student's earliest registration date to the date of that student's second enrollment. Note

that there might be several enrollments for the same student on the same date.

7.1 Consider the two sets of intervals shown as values of the DURING attribute in relvars S_DURING and SP_DURING in [slide 13]. Are those two sets equivalent? What are the corresponding expanded and collapsed forms?

7.2 Let MOD3 be the type whose values are the integers 0, 1, and 2. Consider the type RELATION { DURING INTERVAL_MOD3 }. How many relations *xr* of this type satisfy the condition EXPAND(*xr*) = *xr?* List every relation *cr* of this type that satisfies the condition COLLAPSE(*cr*) = *cr*.

8.4 a. Write a **Tutorial D** expression making use of relvar SP_DURING (as illustrated in Endpaper Panel 4) for the query "For each part that has ever been capable of being supplied by supplier S3, get the part number and the applicable intervals of time."

b. Give formulations of the query "For each day on which some part has been capable of being supplied by supplier S3, get that day and the applicable ranges of parts," using (1) relvar S_PARTS_DURING from Section 5.4 and (2) relvar SP_DURING.

9.3 Find a pair of relations *r1* and *r2* of the same type such that the unpacked form of *r1* is a proper subset of the unpacked form of *r2* but the packed form of *r1* is not a proper subset of the packed form of *r2* (where the packings and unpackings are done on the basis of the same attributes in every case, of course).

9.4 Does the following identity hold?

USING (*A*) ◂ *r1* INTERSECT *r2* ▸
≡ USING (*A*) ◂ *r1* MINUS (USING (*A*) ◂ *r1* MINUS *r2* ▸) ▸

10.1 Explain in your own words the horizontal and vertical decompositions described in [slides 28-31].

10.3 Define sixth normal form (6NF). When is 6NF recommended?

10.4 "The moving point *now*" is not a value but a variable. Discuss.

10.6 Consider the following revised version of the database of Exercise 5.9.

```
VAR CURRENT_COURSE RELATION
{ COURSE# COURSE#,
  CNAME NAME,
  AVAILABLE DATE }
KEY { COURSE# } ;

VAR OLD_COURSE RELATION
{ COURSE# COURSE#,
  CNAME NAME,
  AVAILABLE_DURING INTERVAL_DATE }
KEY { COURSE# } ;

VAR CURRENT_STUDENT RELATION
{ STUDENT# STUDENT#,
```

```
  SNAME NAME,
  REGISTERED DATE }
KEY { STUDENT#, REGISTERED } ;

VAR STUDENT_HISTORY RELATION
{ STUDENT# STUDENT#,
  SNAME NAME,
  REG_DURING INTERVAL_DATE }
KEY { STUDENT#, REG_DURING } ;

VAR ENROLMENT RELATION
{ COURSE# COURSE#,
  STUDENT# STUDENT#,
  ENROLLED DATE }
KEY { COURSE#, STUDENT# }
FOREIGN KEY { COURSE# } REFERENCES CURRENT_COURSE
FOREIGN KEY { STUDENT# } REFERENCES CURRENT_STUDENT
;

VAR COMPLETED_COURSE RELATION
{ COURSE# COURSE#,
  STUDENT# STUDENT#,
  STUDIED_DURING INTERVAL_DATE,
  GRADE GRADE }
KEY { COURSE#, STUDENT# } ;
```

The predicates are as follows:

- **CURRENT_COURSE:** ***Course COURSE#, named CNAME, has been available since date AVAILABLE.***
- **OLD_COURSE:** ***Course COURSE#, named CNAME, was available throughout interval AVAILABLE_DURING.***
- **CURRENT_STUDENT:** ***Student STUDENT#, named SNAME, registered with the university on date REGISTERED.***
- **STUDENT_HISTORY:** ***Student STUDENT#, named SNAME, was registered with the university throughout interval REG_DURING.***
- **ENROLMENT:** ***Student STUDENT# enrolled in course COURSE# on date ENROLLED.***
- **COMPLETED_COURSE:** ***Student STUDENT# studied course COURSE# throughout interval STUDIED_DURING, achieving grade GRADE.***

No course number appears in both CURRENT_COURSE and OLD_COURSE.

a. For each relvar, state whether it is in 6NF. If it is not, identify any problems that might be solved by decomposing it appropriately.

b. Write a query to obtain a relation showing, for each student, the number of courses completed during each registration interval for that student.

c. Assume that for each course there are zero or more offerings, each taking place over a given interval of time (possibly contiguous or overlapping). Some offerings have already taken place; others are currently under way (but have a scheduled completion date); others are scheduled to start at some future time (but, again, have a scheduled completion date). When students enroll in courses, they must specify which offering they are enrolling for. Each offering has a quota, and the number of students enrolled in that offering must not exceed that quota.

Write the predicate and an appropriate Tutorial D definition for a relvar COURSE_OFFERING to reflect these requirements.

11.1 Explain the redundancy, circumlocution, and contradiction problems in your own words.

11.2 Explain in your own words:

a. KEY constraints;

b. PACKED ON constraints;

c. WHEN / THEN constraints;

d. U_key constraints.

11.3 Give examples of relvars that involve at least one interval attribute and require

a. both a PACKED ON and a WHEN/THEN constraint

b. a PACKED ON but no WHEN/THEN constraint

c. a WHEN/THEN but no PACKED ON constraint

d. neither a PACKED ON nor a WHEN/THEN constraint

11.4 Explain how classical keys can be regarded as a special case of U_keys.

11.6 Consider the revised version of courses-and-students from Exercise 10.6, with the following as an appropriate definition for relvar COURSE_OFFERING:

```
VAR COURSE_OFFERING RELATION
{ COURSE# COURSE#,
  OFFERING# POSINT,
  QUOTA POSINT,
  OFFERED_DURING INTERVAL_DATE }
KEY { COURSE#, OFFERING# } ;
```

(where POSINT is a type whose values are the integers greater than zero). The predicate is: *Offering OFFERING# of course COURSE# took place or is scheduled to take place during interval OFFERED_DURIN*G.

Revise the database definition again to include all such PACKED ON, WHEN/THEN, and/or U_key constraints as you think necessary.

12.1 What is a "denseness constraint"?

12.2 **Requirements R2, R5, and R8 [see slides 42-44] cannot possibly be satisfied, in general, if full vertical decomposition into 6NF has not been performed. Why not?**

12.3 **State the nine requirements from [slides 42-44] in a form (natural language only) that applies to courses and students instead of suppliers and shipments.**

You should end up with 21 distinct requirements this time!

12.4 **Given your answer to Exercise 12.3, show the corresponding formal constraints for the version of the courses-and-students database discussed in Exercise 11.6.**

12.5 **Given your answer to Exercise 4, what SINCE_FOR and HISTORY_IN specifications, if any, would you add to the database definition?**

12.6 **The database definition resulting from Exercise 6 in Chapter 11 really needs to be extended still further to allow us to record, in connection with an enrolment, the particular offering to which that enrolment is assigned. State whatever additional natural language requirements you can think of that might arise in connection with such an extension.** Also make the corresponding changes to the **Tutorial D** definition of the database.

14.1 **Using the version of the courses-and-students database discussed in Exercise 11.6, give realistic examples (in natural language) of updates that might be required on that database. Be sure to include examples that will require (a) multiple assignment, (b) U_INSERT or U_DELETE or U_UPDATE operations,** (c) "PORTION" specifications.

14.2 **Given your answer to Exercise 14.1, write suitable Tutorial D statements to perform the specified updates.**

14.3 **How might virtual relvars help with Exercise 14.2?**

16.8 **Summarize in your own words the arguments for and against continuous point types.**

In other words, what happens to the operators defined for intervals and relations with interval-typed attributes if no scale is given for each interval and thus quantisation does not apply? Which of them survive intact and which do not?

SQL and The Relational Model

The purpose of these lectures was to reinforce your understanding of the relational model of data and its motivation, by studying the consequences of deviating from it—in the various noted ways that SQL does in particular.

Databases, Types, and The Relational Model: The Third Manifesto by C.J. Date and myself (published by Addison-Wesley in February 2006) has exercises at the end of each chapter. I do not refer to the exercises when devising exam questions on this subject, but here are some that are relevant and I would expect you to be able to attempt as a result of my CS253 contribution.

1. Identify some of the major differences between SQL and the relational model.

"Some" is too vague for an exam question, of course. I would be more likely to list some salient features of the relational model and ask you to assess SQL's degree of conformance to each of them and the consequences of any contraventions. Where SQL contravenes, I would expect you to be able to explain exactly *how* it does so, giving examples. (Minor syntax errors in SQL examples would not be penalised.)

2. The Third Manifesto *requires "=" to apply to every type; more specifically, it requires* v1 = v2 *to evaluate to TRUE if and only if* v1 *and* v2 *are the very same value. SQL, by contrast, does not require "=" to apply to every type, nor does it prescribe the required semantics in all cases where it does apply. What are the implications of this state of affairs?*

   I did not spend a lot of time on this point, but see the slide "'=' Is Not 'equals'". Also consider how SQL's application of a three-valued logic affects equality comparisons. I'm sure I didn't cover all the implications in my lecture, but I hoped to stimulate your own thinking on these issues and it would not be unreasonable to give you a chance to express your own conclusions in exam conditions.

3. *The following SQL query is not valid but should be:*

   ```
   SELECT DISTINCT
   FROM   S, SP
   WHERE  S.S# = SP.S#
   AND    S.CITY = 'Athens' ;
   ```

   What do you think it should return? Give a natural language interpretation of the query. What do you think the interpretation should be if DISTINCT were omitted?

   S and SP are Date's familiar "suppliers" and "shipments" relations whose temporal counterparts formed the running example of my second series of lectures.

4. *Which of the RM Proscriptions does SQL violate? Which of those violations constitute violations of* The Information Principle*?*

   The "RM Proscriptions" are things that a DBMS is not allowed to do if it is to be relational DBMS. Each of them is a consequence of the relational model of data. Here is a copy of the relevant section of *The Third Manifesto:*

   RM PROSCRIPTIONS

   1. **D** shall include no concept of a "relation" whose attributes are distinguishable by ordinal position. Instead, for every relation *r* expressible in **D,** the attributes of *r* shall be distinguishable by *name*.
   2. **D** shall include no concept of a "relation" whose tuples are distinguishable by ordinal position. Instead, for every relation *r* expressible in **D,** the tuples of *r* shall be distinguishable by *value*.
   3. **D** shall include no concept of a "relation" containing two distinct tuples *t1* and *t2* such that the comparison "*t1 = t2*" evaluates to TRUE. It follows that (as already stated in RM Proscription 2), for

every relation *r* expressible in **D,** the tuples of *r* shall be distinguishable by *value*.

4. **D** shall include no concept of a "relation" in which some "tuple" includes some "attribute" that does not have a value.
5. **D** shall not forget that relations with no attributes are respectable and interesting, nor that candidate keys with no components are likewise respectable and interesting.
6. **D** shall include no constructs that relate to, or are logically affected by, the "physical" or "storage" or "internal" levels of the system.
7. **D** shall support no tuple-at-a-time operations on relvars or relations.
8. **D** shall not include any specific support for "composite" or "compound" attributes, since such functionality can more cleanly be achieved, if desired, through the type support already prescribed.
9. **D** shall include no "domain check override" operators, since such operators are both *ad hoc* and unnecessary.
10. **D** shall not be called SQL.

I think this could be a useful checklist for you, but please ignore nos. 8 and 9, which refer to matters I did not discuss at all and in any case are not relevant to SQL. No. 10 is an easy one, of course (it was included as a joke, and to make the numbers up to a round 10).

And here is *The Information Principle* (as E.F. Codd called it):

> *The entire information content of the database is represented (at any given time) in one and only one way: namely, as explicit values in attribute positions in tuples in relations.*
>
> Equivalently and more concisely: *Every variable in the database shall be a relation variable.*

Finally, here is an exercise from my first lecture on CS252. *Consider the following table:*

A	**B**	**A**
1	2	3
4		5
6	7	8
9	9	?
1	2	3

Give three reasons why it cannot be representing a relation.

For CS253 students I could reasonably add: *Assuming that the blank space in the B column of the second row represents NULL, show how an SQL query could result in this table.*