Midterm Review Sheet and Sample Questions

Information

- The midterm exam will be held on Monday May 4 from 11:00 AM until 12:15 PM in the Gates Computer Science Building Room B01 (the Hewlett-Packard Auditorium) on the Stanford campus. All students, including SITN students, are expected to attend the exam on-campus. There will be no early or makeup exams.
- The exam will be closed book. However, each student may bring up to three pages of prepared notes. That's six total sides of writing on 8-1/2"x11" paper.
- A sample solution for Written Assignment #3 will be available via the course Web page by 7:00 PM on Friday May 1.

Review Session

A question-and-answer review session will be conducted by the TA's on Sunday May 3 from 7:00–8:30 PM in Gates Computer Science Building Room 100. Use the main corner entrance to Gates; there's a door to room 100 on the right near the top of the stairs to the basement. Please bring questions or specific problems for the TA's. The review session will not be televised.

Note that there will be *no* Monday afternoon review session the day of the exam.

Material Covered

The exam will cover:

- All lectures through Wednesday April 29
- Textbook readings: Chapters 1, 2.1–2.6, 3.1–3.8, 4.1, 5.1–5.7
- Written Assignments #1–3
- Programming Assignment #1

Since the material on SQL covered in class and in Chapter 5 has not yet been exercised in an assignment, any questions regarding this material on the exam will be fairly straightforward.

What follows is an outline of the material we've covered so far. All of this material is fair game for the midterm exam unless specified otherwise.

1. Basic motivation and database terminology
2. Object-oriented database design using ODL
   - Classes, attributes, relationships, keys
   - Set and inverse relationships
   - Subclasses, inheritance

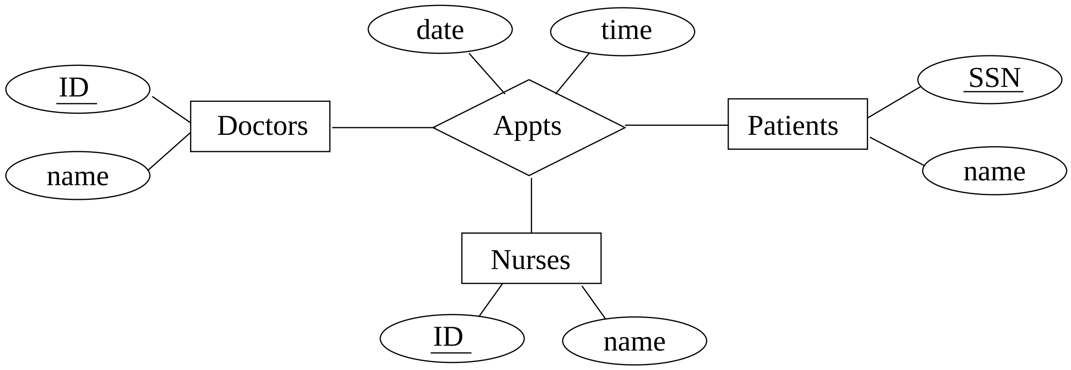
3. Entity-relationship model and E/R diagrams
   - Entity sets, relationship sets, attributes, keys
   - Multiplicity of relationships (one-to-one, one-to-many, etc.)
   - Weak entity sets
   - Subclasses
4. Relational model
   - Relations (tables), attributes (columns), tuples (rows)
   - Schema versus instance
   - Keys, null values
   - Translating E/R and ODL designs to relations
5. Relational database design
   - Functional dependencies (FD's): motivation, definitions, rules
   - Closure of attribute set with respect to FD's
   - Design flaws: redundancy, update & deletion anomalies
   - Boyce-Codd Normal Form (BCNF): motivation, definition, decomposition algorithm
   - Third Normal Form (3NF): motivation, definition, decomposition algorithm
   - Multivalued dependencies (MVDs): motivation, definitions, rules
   - Fourth Normal Form (4NF): motivation, definition, decomposition algorithm
6. Relational algebra
   - Basic operators: select (σ), project (π), Cartesian product ($\times$), union ($\cup$), difference ($-$), rename (ρ)
   - Abbreviations: natural join ($\bowtie$), theta join ($\bowtie_\theta$) intersection ($\cap$)
7. SQL – coverage of SQL on the exam will be cursory; material not covered in lecture is not fair game for the exam
   - Data definition: **create table**, **drop table**
   - Data manipulation: **select** command
   - Subqueries, aggregates, duplicates, null values
   - Data modification: **insert**, **delete**, **update**

Sample Questions

What follows are the questions from Prof. Widom's 1996 midterm or final exams covering the material that will be covered in the midterm exam this year.

1. **Entity-Relationship model and ODL**

   Consider the following entity-relationship diagram representing appointments that include one doctor, one nurse, and one patient:



   (a) Translate the E/R diagram to a relational schema using the method described in class and in the course notes. Underline key attributes for the relations derived from entity sets only.

   (b) Suppose that each doctor, nurse, and patient can be scheduled for at most one appointment at a given date and time. For the relation you derived in part (a) corresponding to the relationship set in the E/R diagram, list all of the possible minimal keys.

   (c) Specify an ODL schema that captures the database design in the above E/R diagram as closely as possible. Assume that names, dates, and times are represented as strings, while IDs and SSNs are represented as integers. It is not necessary to specify keys. You will be graded on clarity and simplicity as well as on correctness.

   (d) Translate your ODL schema from part (c) to a relational schema. Use the method described in class where ODL relationship pairs are translated to a separate relation, rather than the technique described in the course notes where relationship pairs are represented within the relations for the classes. For the translation you may need to make assumptions about keys for your ODL classes. Please state any assumptions you make about keys if (and only if) the assumptions differ from or are in addition to keys specified in the original E/R diagram. If you need to add any new attributes to your ODL schema for the translation to "work," please do so. It is not necessary to underline key attributes in the relational schema.

   (e) Ignoring the issue of keys, is there any database instance using your relational schema from part (a) that cannot be represented in your schema from part (d)? Assume that null values are not permitted in relations. If so, show the simplest such instance you can find.

(f) Ignoring the issue of keys, is there any database instance using your relational schema from part (d) that cannot be represented in your schema from part (a)? Assume that null values are not permitted in relations. If so, show the simplest such instance you can find.

2. **Queries in relational algebra**

Consider the following relational schema:

```
course(course#, dept-name)     // course# is the key
enroll(studentID, course#)     // <studentID,course#> is the key
```

(a) Write a relational algebra expression to find the IDs of all students who are not enrolled in course# 145.

(b) Write a relational algebra expression to find the IDs of all students who are enrolled in courses in at least two different departments.

3. **More relational algebra and SQL**

There is a relational operator called "outerjoin" that we have not (yet) studied in class. The outerjoin of two relations $R(A, B)$ and $S(B, C)$ is defined as follows: We take the natural join $R \bowtie S$. We add to the result a tuple $(a, b, null)$ for every tuple (a, b) in R such that (a, b) does not participate in the result of $R \bowtie S$ (i.e., there is no joining value for b in S), and we add to the result a tuple $(null, b, c)$ for every tuple (b, c) in R such that (b, c) does not participate in the result of $R \bowtie S$ (i.e., there is no joining value for b in R). Write an SQL query that computes the outerjoin of relations $R(A, B)$ and $S(B, C)$, without using the SQL `outerjoin` operator of course. You may `select` the constant value `NULL` if you find it useful; for example, the following query returns all tuples in $Q(A, B, C)$ with all B values replaced by `NULL`:

```
select A,NULL,C from Q
```

Write the SQL query for outerjoin.

4. **Functional Dependencies**

Consider the following "rule" for functional dependencies:

If $A \rightarrow B$ and $B, C \rightarrow D$ then $A, C \rightarrow D$

(a) Is this "rule" correct or incorrect? Answer with one word, please.

(b) If you said in part (a) that the rule is correct, sketch a simple proof. If you said in part (a) that the rule is incorrect, give a simple counterexample that includes a relation schema and a set of tuples that satisfy the "if" part of the rule but not the "then" part.

5. **Relational Database Design**

Consider the following relational schema:

```
Sale(clerk, store, city, date, item#, size, color)
  // records that a clerk sold an item on a particular day
Item(item#, size, color, price)
  // records prices and available sizes and colors for items
```

Make the following assumptions, *and only these assumptions*, about the real world being modeled:

- Each clerk works in one store.
- Each store is in one city.
- A given item# always has the same price, regardless of size or color.
- Each item is available in one or more sizes and one or more colors, and each item is available in all combinations of sizes and colors for that item.

Here are the problems:

(a) Based on the assumptions above (and no others), specify all keys for relations `Sale` and `Item`.

(b) Based on the assumptions above (and no others), specify an appropriate set of completely nontrivial functional dependencies for relations `Sale` and `Item`.

(c) Are relations `Sale` and `Item` in Boyce-Codd Normal Form (BCNF)? If not, decompose the relations so they are in BCNF.

(d) Now consider your decomposed relations from part (c), or the original relations if you did not need to decompose them for part (c). Based on the assumptions above (and no others), specify all nontrivial multivalued dependencies for the relations. Do not include multivalued dependencies that also are functional dependencies.

(e) Are the relations you used in part (d) in Fourth Normal Form (4NF)? If not, decompose the relations so they are in 4NF.