Final Examination
"Course 1-1023221-0: Database Systems"
Exam A

Kinneret College School of Engineering

January 28, 2009 9:00am-12:00pm

- Answer the following questions in English or Hebrew.
- You may bring one page of notes to the exam with notes on both sides.
- The number of points for each question is listed next to each one to indicate its weight.
- The number of points for each question is listed next to each one to indicate its weight. You must complete a total of **120** points on the test to receive a 100% grade. Since there are a total of 155 points possible, you may get up to 35 points off and still receive a full grade.
- You will not receive a grade greater than 100%, even if you receive more than 120 points.
- You are encouraged to try all of the problems. I will grade them all.
- Write all of your answers in the test booklet which you received.
- Marks made on the test sheets will not be counted or graded.
- You must return the test questions sheet at the end of the exam.

1 Question 1 (18 points / 3 points each)

Briefly explain the following terms as they relate to databases:

1. Foreign Key
2. Transaction
3. Participation Constraint
4. Recoverable Schedule
5. Forcing Pages (to disk)
6. Serializable Schedule

2 Question 2 (12 points / 6 points each)

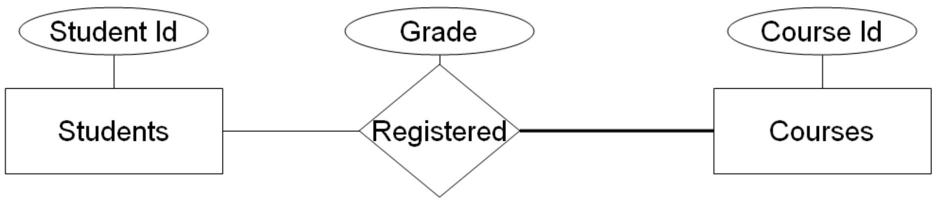
Consider the following scenarios about entities, relationships, and constraints. Draw an ERD for each scenario. Each scenario may be considered *independently* of the others in the list.

1. Each student has a national identification number, a name, an address, and a phone number. Younger students often share the same address, and no address has more than one phone.
2. A local library has borrowers and elderly volunteers. Each volunteer has a national identification number, a day he works at the library and a specialty. Each borrower has a national identification number a unique library card number, and an address. A volunteer can borrow books *only* if she is also registered as a borrower. A book can only be borrowed by one borrower at a time.

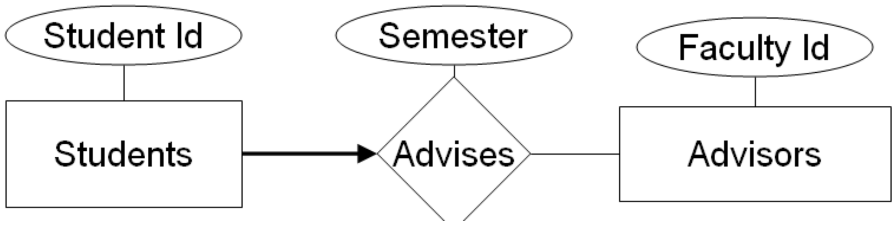
3 Question 3 (20 points / 5 points each)

Consider the following ERD diagrams. Write SQL `CREATE TABLE` commands which create relations which implement them. Include information about primary and foreign keys and `NOT NULL` constraints. If a constraint can not be enforced using primary keys, foreign keys, and `NOT NULL`, write code (`CHECK`, `ASSERTION`, or `STORED PROCEDURE`) which will check it.

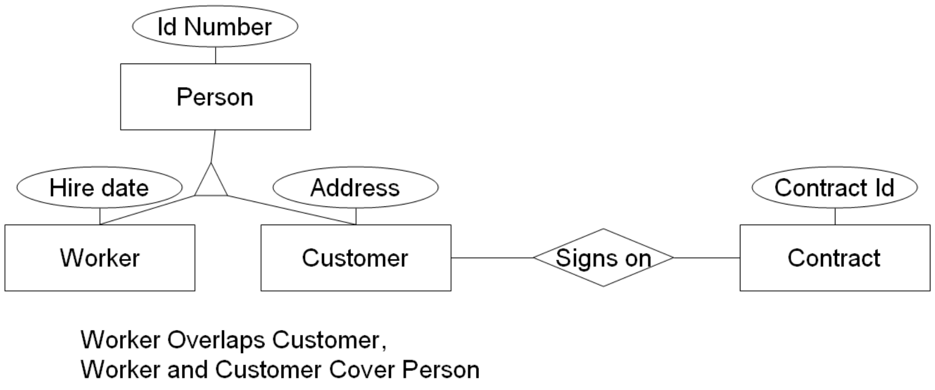
3.1 Diagram 1



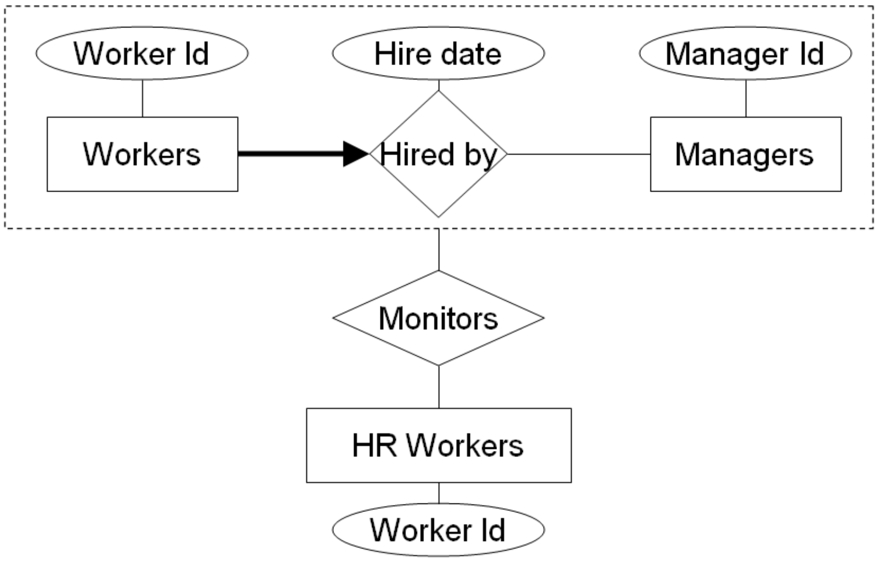
3.2 Diagram 2



3.3 Diagram 3



3.4 Diagram 4



4 Question 4 (66 points)

Consider the following relational schema. An employee can work in more than one department; the pct_time field of the Works relation shows the percentage of time that a given employee works in a given department.

Emp(<u>*eid*:`integer`</u>, *ename*:`string`, *age*:`integer`, *salary*:`real`)
Works(<u>*eid*:`integer`</u>, *did*:`integer`, *pct_time*:`integer`)
Dept(<u>*did*:`integer`</u>, *dname*:`string`, *budget*:`real`, *managerid*:`integer`)

You will first write relational algebra queries and then similar SQL queries.

A table with all of the relational algebra and set manipulation operators and their meanings is shown in Table 1 to help with this question.

Symbol	Meaning
σ	Select (rows)
π	Project (columns)
$\rho(N(\overline{C}), O)$	Renames table O to be called N. Changes column names as per $\overline{C}$
$\bowtie_c$	Conditional Join / Equijoin
$\bowtie$	Natural Join
/	Division
$\times$	Cartesian Cross Product
$\cup$	Set Union
$\cap$	Set Intersect
-	Set Difference

Table 1: Relational Algebra Operators

4.1 Relational Algebra Queries (30 points / 6 points each)

Write relational algebra statements for the following queries. You may use more than one line for the relational algebra statement if necessary.

1. Select the names and ages of each employee who works in both the Hardware department and the Software department.
2. Select the *enames* of managers who manage at least two departments.
3. Select the ages of employees who work in *all* departments in which there is some employee
4. Select the names of employees whose *eid* is greater than at least one department in which they work's *did*.
5. Select the budgets for all departments which employ at least one employee under 18 years old.

4.2 SQL Queries (36 points / 6 points each)

Write SQL statements for the following queries. You may use grouping, nesting, or join operators in your statements.

1. Select the names and ages of each employee who works in both the Hardware department and the Software department.
2. Select the *enames* of managers who manage at least two departments.
3. Select the **MINIMUM age of employees** who work in *all* departments in which there is some employee
4. Select the average salaries of employees under 18 in departments with at least 2 such employees grouped by the number of employees under 18 in the department. For example:

average salary	number under 18
100	2
102	3
4	10
...	...

5. Find the **SUM of the budgets** for all departments which only employ employees under 18 years old.
6. Print the *eid*'s for employees whose total appointments are greater than 100% (this means that the sum of their associated pct_time > 100).

5 Question 5 (24 points)

Consider the following database schema. It is from a factory which has workers and machines. Workers maintain the machines. Information about the machines is stored in the Machines relation, including the last date the machine was maintained. Each maintenance event is recorded in the Maintenance relation, including the date of the maintenance and how many hours it took. Each worker is described in the Workers relation, including the number of hours the worker has worked so far.

Machines (<u>*mid*:`int`</u>, *mname*:`string`, *lastMaintDateTime*:`DateTime`)
Maintenance (<u>*wid*:`int`, *mid*:`int`, *dateTime*:`DateTime`</u>, *numHours*:`int`)
Workers (<u>*wid*:`int`</u>, *wname*:`string`, *hoursWorked*:`int`)

Assume the above tables have been initialized as follows:

Machines:

mid	mname	lastMaintDateTime
1	'AMachine'	2001-1-1 1:01:01
2	'BMachine'	2002-2-2 2:02:02

Workers:

wid	wname	hoursWorked
10	'Chuck'	9
11	'Dan'	10

Maintenance

wid	mid	dateTime	numHours
10	1	2000-1-1 11:11:11	1
10	1	2000-1-1 12:12:12	3
10	1	2001-1-1 1:01:01	5
11	2	2002-2-2 2:02:02	10

5.1 Tables SQL (9 points)

Write SQL code which will create the above tables. Include all relevant primary and foreign key constraints.

5.2 Table Updates (6 points)

- Write SQL code which will change the third row of Maintenance to: (11, 1, 2001-1-1 1:01:01, 6).
- Write SQL code which will insert the following tuples into Maintenance: (11, 1, 2002-2-2 2:02:02, 5), (10, 2, 2003-3-3 3:03:03, -1).

5.3 Triggers (9 points)

Consider the following triggers written in SQL syntax:

```
CREATE TRIGGER trigger1 BEFORE INSERT ON Maintenance
REFERENCING NEW ROW AS nrow
FOR EACH ROW
WHEN nrow.numHours < 0
SET nrow.numHours = 0

CREATE TRIGGER trigger2 AFTER UPDATE ON Maintenance
REFERENCING NEW ROW AS nrow, OLD ROW AS orow
FOR EACH ROW
BEGIN
    UPDATE Workers W
        SET W.hoursWorked = W.hoursWorked - orow.numHours
        WHERE W.wid = orow.wid;
    UPDATE Workers W
        SET W.hoursWorked = W.hoursWorked + nrow.numHours
        WHERE W.wid = nrow.wid;
END

CREATE TRIGGER trigger3 AFTER INSERT ON Maintenance
REFERENCING NEW ROW AS nrow
FOR EACH ROW
WHEN nrow.dateTime > (SELECT MAX(lastMaintDateTime)
                      FROM Machines M
                      WHERE M.mid = nrow.mid)
BEGIN
    UPDATE Machines M
        SET M.lastMaintDateTime = nrow.dateTime
        WHERE M.mid = nrow.mid;
    UPDATE Workers W
        SET W.hoursWorked = W.hoursWorked + nrow.numHours
        WHERE W.wid = nrow.wid;
END
```

Draw the state of the Machines, Workers, and Maintenance tables after you have run the insert and update commands written in 5.2:

6 Question 6 (15 points / 5 points each)

Consider the following two transactions sequences.

T_1: R(A), W(B), R(B), W(A), W(B), Commit.

T_2: R(C), R(A), R(B), W(D), Commit.

Consider the following schedules. For each schedule, write whether it is serializable or not and what serial schedule it is equivalent to. If it is not, explain the problems that prevent it from being serializable.

6.1 Schedule 1

T_1	T_2
R(A)	
W(B)	
	R(C)
R(B)	
	R(A)
W(A)	
W(B)	
Commit	
	R(B)
	W(D)
	Commit

6.2 Schedule 2

T_1	T_2
	R(C)
	R(A)
R(A)	
W(B)	
R(B)	
W(A)	
W(B)	
Commit	
	R(B)
	W(D)
	Commit

6.3 Schedule 3

T_1	T_2
	R(C)
	R(A)
	R(B)
R(A)	
W(B)	
R(B)	
W(A)	
W(B)	
	W(D)
	Commit
Commit	