CMPT 354
Midterm Exam Key

Total marks: 72 2000-1
Date: March 1, 2000 Instructor: G. Louie

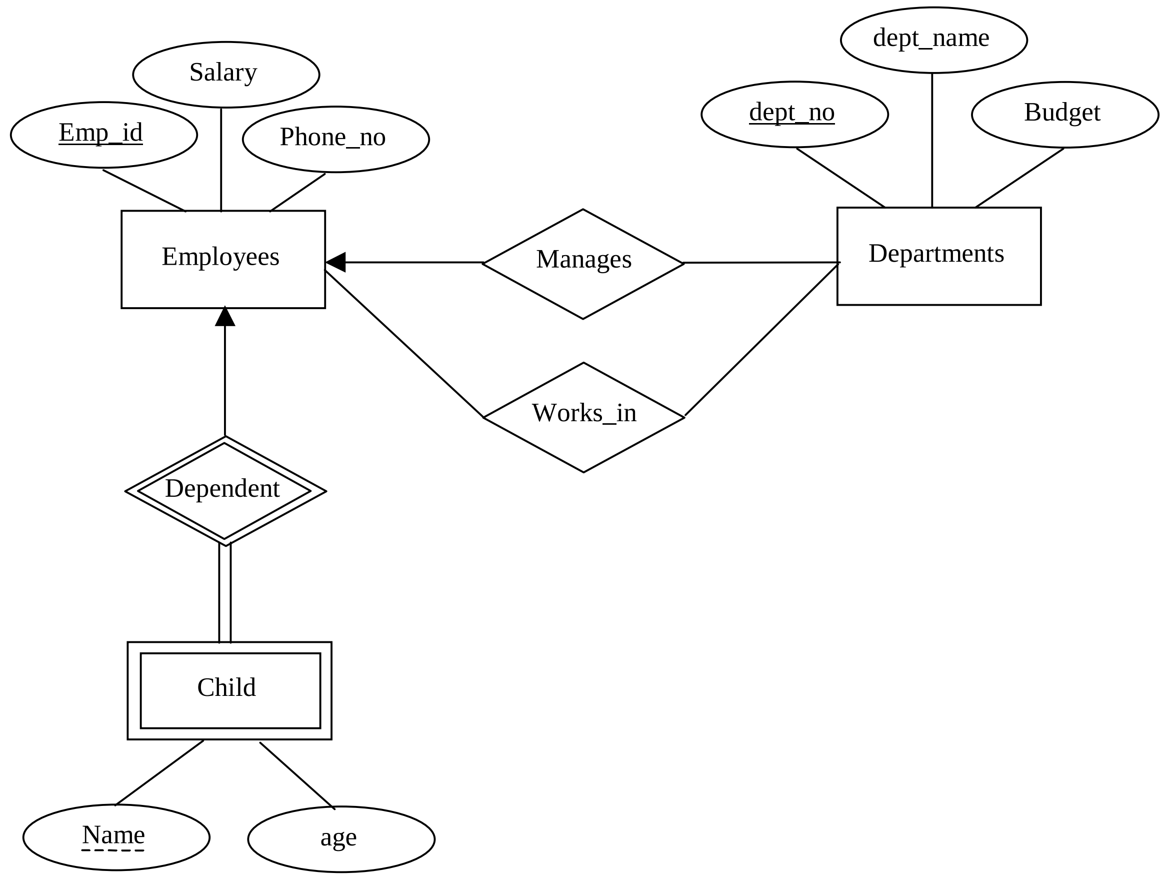
This exam is **closed** book. You have **1** hour and **15** minutes. **No** talking! You may answer the questions in any order that you choose, but clearly indicate the question number in your answer. Write clearly: if we can't read it, you get no marks. Good luck!

1. A company database needs to store information about employees (identified by *emp_id*, with *salary* and *phone_no* as attributes); departments (identified by *dept_no*, with *dept_name* and *budget* as attributes); and dependent children of employees (with *name* and *age* as attributes).

   Employees *work* in departments (*i.e.* an employee can work in more than one department); each department is *managed by* an employee; a child must be identified uniquely by *name* when the parent (who is an *employee*; you can assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.

a) (7 marks) Construct a clean and concise ER diagram using the Chen notation for this database. Clearly indicate the cardinality mappings in the ER diagram.

A possible solution is presented here. Start w. 7 marks and deduct one for each element in the diagram which is not consistent with the requirements as set out above (i.e. missing entities, relationship sets, or attributes, incorrect mapping cardinalities, lines missing arrowheads, incorrect identification of weak entity set, etc.). Note that other solutions are possible and as long as the requirements specified above are met (*i.e.* entities, relationship sets, attributes & mapping cardinalities are correct) and any reasonable assumptions are properly documented, full marks should be given.



b) (15 marks) Give an SQL schema definition for the ER diagram from part a). Clearly indicate any integrity constraints that should hold on this schema.

Deduct one mark for each incorrect/extraneous relation, incorrect or incompatible domain, or incorrect identification of the primary key. Note that the names of the relations for the relationship sets can be different.

```
create table employees
(   emp_id       integer not null,
    salary       numeric(6,2),
    phone_no     char(10),
    primary key (emp_id))              2 marks
```

```
create table departments
(   dept_no       integer not null,
    dept_name     char(25),
    budget        numeric(12,2),
    primary key (dept_no))                    2 marks
```

```
create table works_in
(   emp_id        integer not null,
    dept_no       integer not null,
    primary key (emp_id, dept_no))          /* i.e. many-to-many */
```

3 marks: 2 marks for table + 1 mark for correctly identifying the primary key for the many-to-many relationship set.

```
create table manages
(   emp_id        integer not null,
    dept_no       integer not null,
    primary key (dept_no))                  /* i.e. one-to-many */
```

3 marks: 2 marks for table + 1 mark for correctly identifying the primary key for the one-to-many relationship set.

```
create table dependent_children
(   emp_id        integer not null,
    name          char(20),
    age           integer,
    primary key (emp_id, name),
    foreign key (emp_id) references employees on delete cascade)
```

5 marks: 2 marks for table, 1 mark for correctly identifying the primary key, 1 mark for the foreign key, and 1 mark for the cascading delete.

2. Short Answer

a) (4 marks) What is a foreign key constraint?

A *foreign key* constraint is a statement of the form that one or more fields of a relation, say *R*, together *refer* to a second relation, say *S*.

The values in these fields of a tuple in *R* are either *null*, or uniquely identify some tuple in *S*. Thus, these fields of *R* should be a (candidate or primary) key.

b) (3 marks) Why are such constraints important?

Foreign key constraints are important because they provide safeguards for insuring the integrity of data. Users are alerted/thwarted when they try to do something that does not make sense. This can help minimize errors in application programs or in data-entry.

c) (3 marks) What is referential integrity?

Referential integrity means all foreign key constraints are enforced.

3. Consider the following schema:

Student(<u>*st_num*</u>, *st_name*, *major*, *grade*, *age*)
Class(<u>*cname*</u>, *meet_ at_time*, *room*, *fac_id*)
Enrolled(<u>*st_num*</u>, <u>*cname*</u>)
Faculty(<u>*fac_id*</u>, *fname*, *deptid*)

Note: do not print any duplicate rows in your answers.

a) (3 marks) Find the names of all grade 10's who are enrolled in a class taught by "John Smith".

```
SELECT DISTINCT S.St_name
FROM Student S, Class C, Enrolled E, Faculty F
WHERE S.st_num = E.st_num AND E.cname = C.cname
    AND C.fac_id = F.fac_id AND F.fname = 'John Smith'
    AND S.grade = 10
```

b) (3 marks) Find the names of all classes that either meet in room 5NE or have five or more students enrolled.

```
SELECT C.cname
FROM Class C
WHERE C.room = “5NE” OR
  C.cname IN (   SELECT E.cname
                 FROM Enrolled E
                 GROUP BY E.cname
                 HAVING COUNT (*) >= 5  )
```

c) (3 marks) Find the names of all students who are enrolled in two classes that meet at the same time.

```
SELECT DISTINCT S.st_name
FROM Student S
WHERE S.st_num IN
   ( SELECT E1.snum
     FROM Enrolled E1, Enrolled E2, Class C1, Class C2
     WHERE E1.st_num = E2.st_num
               AND E1.cname <> E2.cname
               AND E1.cname = C1.cname
               AND E2.cname = C2.cname
               AND C1.meet_at_time = C2.meet_at_time)
```

d) (3 marks) Define a view *part_time_students(*`st_num, st_name, major, grade, age`*)* where a student is considered part-time if he or she takes less than 5 courses.

```
CREATE VIEW part_time_students AS
SELECT student.st_num, st_name, major, grade, age
FROM student, enrolled
WHERE student.st_num = enrolled.st_num
GROUP BY student.st_num, st_name, major, grade, age
HAVING COUNT(cname) < 5
```

4. Consider the following relational database:

Suppliers(*supplier_id*, *supplier_name*, *address*)
Parts(*part_id*, *part_name*, *color*)
Catalog(*supplier_id*, *part_id*, *cost*)

For each of the following queries, give an expression in

i) the relational algebra,
ii) the tuple relational calculus,
iii) the domain relational calculus.

Warning: marks will be deducted even for errors that are propagated *i.e.* if you make a mistake in the relational algebra and then write relational calculus statements based on the incorrect expression, marks will be deducted for *each* incorrect expression!

a) (9 marks) Find the supplier_id's of the suppliers who supply some 'red' or 'green' part.

(3 marks for each correct expression)

i) $\Pi_{supplier\_id}\ (catalog \bowtie \Pi_{part\_id}\ (\sigma_{color\ =\ \text{'red'}\ \vee\ color\ =\ \text{'green'}}\ (parts)))$

ii) $\{t \mid \exists\ c \in catalog\ (\exists\ p \in parts\ ((p[color] = \text{'red'} \vee p[color] = \text{'green'})$
$\wedge\ c[part\_id] = p[part\_id]) \wedge t[supplier\_id] = c[supplier\_id])\}$

iii) $\{<s> \mid \exists\ p, c, c1\ (<s, p, c> \in catalog \wedge <p, n, c1> \in parts$
$\wedge\ cl = \text{'red'} \vee cl = \text{'green'})\}$

b) (9 marks) Find the part_id's of all parts that are supplied by at least two suppliers.

(3 marks for each correct expression)

i) $\Pi_{part\_id}\ (\sigma_{catalog.part\_id\ =\ c2.part\_id\ \wedge\ catalog.supplier\_id\ <>\ c2.supplier\_id}\ (catalog \times \rho_{c2}(catalog)))$

ii) $\{t \mid \exists\ c \in catalog\ (\exists\ c1 \in catalog\ (\ c[part\_id] = c1[part\_id]$
$\wedge\ c[supplier\_id] <> c1[supplier\_id]) \wedge t[part\_id] = c[part\_id])\}$

iii) $\{<p> \mid \exists\ s, c, s1, c1\ (<s, p, c> \in catalog \wedge <s1, p, c1> \in catalog \wedge s <> s1)\}$

5. (10 marks) Suppose you are given a relation *r* with its primary key *j* and a relation *s* with its primary key *k*. Both relations *r* and *s* represent entity sets. Explain how functional dependencies can be used to indicate that a one-to-one relationship set exists between *r* and *s*.

A one-to-one relationship set exists between *r* and *s* if the following functional dependencies hold (1 marks):

j → k	(3 marks)
k → j	(3 marks)

(3 marks for explanation) This is because by the definition of functional dependency, any two tuples in the relationship set with the same *j* value must have the same *k* value and vice versa. This is also the definition for a one-to-one relationship