CSCI-585 - Database Systems
Exam #1
10/3/14, 6-8 pm

Solution key

Name:

Student ID:

Please read each question carefully before answering.

The space provided for the answers should be adequate.

This is a closed-book, closed-notes, closed-devices, closed-peeking (but open mind!) test. If you are caught cheating or discovered to have cheated in any way, your score for the entire test will be 0.

Your score

```
---------------
Q1:       /10
Q2:       /10
Q3:       /10
Q4:       /10
Q5:       /10
Q6:       /10
Q7:       /10
Q8:       /10
Q9:       /10
Q10:      /10
Bonus:    /5
---------------
Total:    /105
---------------
```

Q1 (10 points). A **DBMS** is an intermediary between the end-users and a database - why have one (what are the advantages - list 2 or more)?

A DBMS provides access to MULTIPLE users concurrently, manages the data dictionary, provides backup and recovery, does data transformation and presentation, provides data access languages and APIs, and provides communication interfaces.

Q2 (10 points). In the context of supertype/subtype 'extended' E-R modeling, what is '**partial completeness**'? Explain, using an example.

Partial completeness means that in an entity supertype/subtype relationship, not every supertype occurence is a member of a subtype. Eg. given a EMPLOYEE supertype and ADMINISTRATOR, PROFESSOR subtypes (in a college db schema), not every EMPLOYEE is an ADMINISTRATOR or a PROFESSOR - an employee could be a MAINTENANCE_WORKER, ADMIN_ASST etc.

Q3 (5+5=10 points). What is **referential integrity**, and what is one situation where it could be violated?

Referential integrity is a condition/situation where every reference to an entity instance by another entity instance is valid. It is violated when a foreign key value in a table does not have a corresponding primary key value in the referenced table.

Explaining the above in own words is OK.

Q4 (2.5*4=10 points). What are some (4 or more) heuristics for choosing a **'good' primary key** for a table?

A 'good' primary key would be one that is non-intelligent, unchanging over time, preferably a single (not composite) attribute, preferably numeric, security-compliant.

Explaining the above in own words is OK.

Q5 (2.5*4=10 points). List four '**relational algebra**' operators.

Select, Project, Union, Intersect, Difference, Product, Join, Divide..

Q6 (10 points). When implementing an E-E-R diagram, what is different between implementing '**disjoint types**' and '**overlapping subtypes**'?

For disjoint subtypes, a single 'subtype discriminator' column in the ssupertype will suffice in order to distinguish subtypes. For overlapping subtypes, there needs to be a Boolean (Yes/No) column in the supertype, for each subtype, or a new, separate composite entity created with such subtype Boolean columns.

Q7 (5+5=10 points). These questions have to do with relations between tables. What is a '**weak entity**'? What is '**relationship degree**' (give example(s))?

A weak entity is one that is existence-dependent (on another entity), **and** whose primary key is partially or totally derived from the existence-depending (referred to as a 'parent' or 'owner' or 'identifying') entity. In other words, it (the weak entity) can't be identified by its own attributes alone.

Relationship degree refers to the number of entities involved/participating in a relationship. A unary relationship is where an association is maintained with itself - eg. an EMPLOYEE table contains a 'Manages' relation to itself. A binary relationship is where two entities participate, eg. an EMPLOYEE 'works_for' a COMPANY. In a ternary relation, three tables participate, eg. a DOCTOR prescribes a DRUG to a PATIENT. Note - OK if in the ternary relationship example, a PRESCRIPTION is indicated as a bridge.

Q8 (5+5=10 points) These two questions are about SQL. What is the purpose of a **trigger**? What does the '**GROUP BY**' statement help achieve, in a query?

A database trigger is procedural code that is **automatically** executed in response to certain **events** on a particular table or view in a database. The main thing to note is automatic execution (like a 'callback') of code, in response to an event (eg. a value changes in a column in a table).

The GROUP BY statement is used in conjunction with **aggregate** functions, to group the result-set by one or more columns. Note - important to mention **aggregate** functions - without one, the GROUP BY clause is meaningless (so the answer is only partially correct).

Q9 (10 points)
Here is a 'pets' table and a 'owners' table:

```
+---------+---------+--------+-----------+
| pets_id | animal  | name   | owners_id |
+---------+---------+--------+-----------+
|       1 | fox     | Rusty  |         2 |
|       2 | cat     | Fluffy |         2 |
|       3 | cat     | Smudge |         3 |
|       4 | cat     | Toffee |         3 |
|       5 | dog     | Pig    |         3 |
|       6 | hamster | Henry  |         1 |
|       7 | dog     | Honey  |         1 |
+---------+---------+--------+-----------+
```

```
+-----------+-------+
| owners_id | name  |
+-----------+-------+
|         1 | Susie |
|         2 | Sally |
|         3 | Sarah |
```

What would be the output (please indicate using a table) of the following **SQL query**?

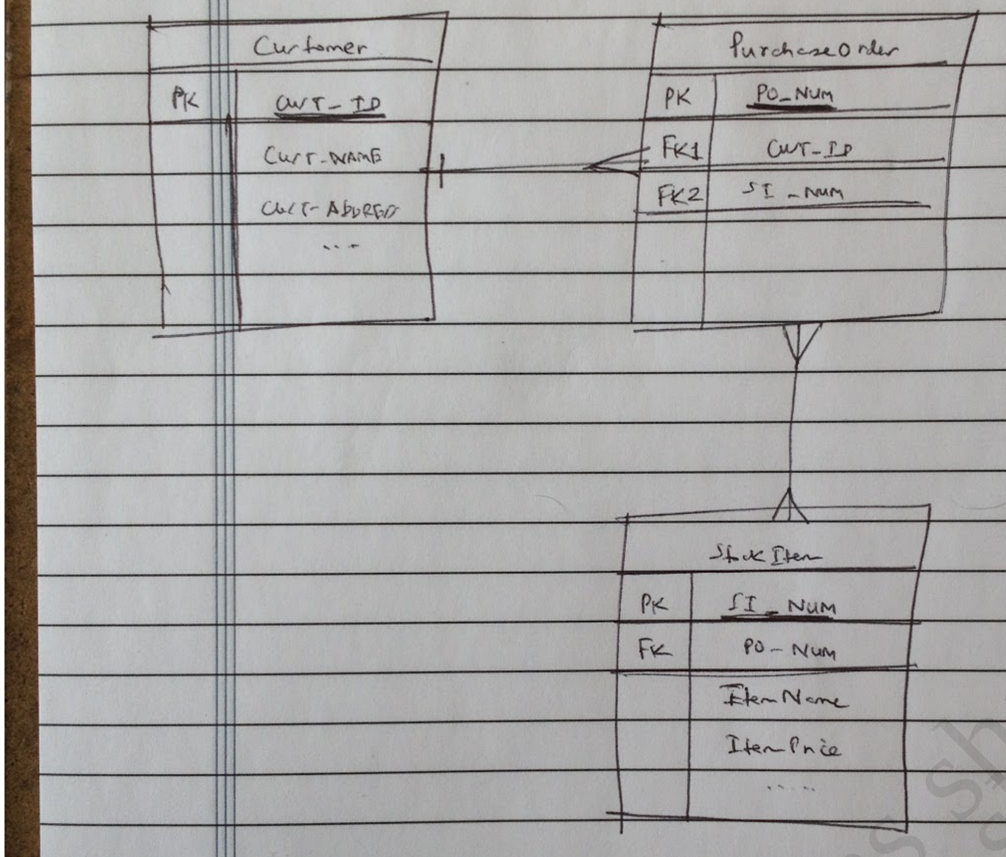
```
SELECT owners.name as owner, pets.name as pet, pets.animal
FROM owners JOIN pets ON (pets.owners_id = owners.owners_id);
```

```
+-------+--------+---------+
| owner | pet    | animal  |
+-------+--------+---------+
| Sally | Rusty  | fox     |
| Sally | Fluffy | cat     |
| Sarah | Smudge | cat     |
| Sarah | Toffee | cat     |
| Sarah | Pig    | dog     |
| Susie | Henry  | hamster |
| Susie | Honey  | dog     |
+-------+--------+---------+
```

Q10 (10 points)

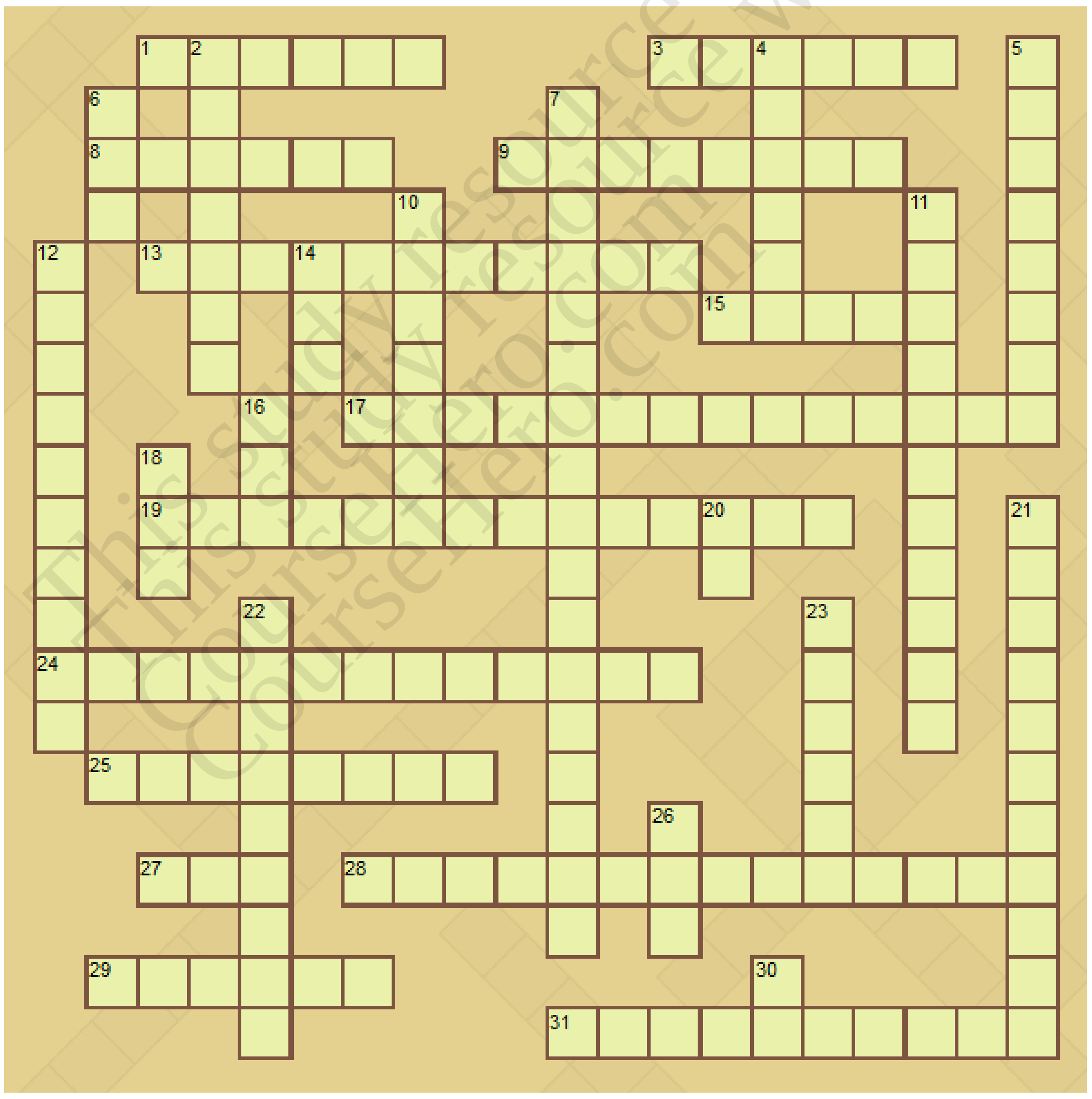
Consider these entities: Customer, PurchaseOrder, StockItem. A Customer can place several PurchaseOrders, and each PurchaseOrder comes from just one Customer. A PurchaseOrder can list many StockItems, and a StockItem could be found in (listed in) many PurchaseOrders.

Draw an **implementation-ready E-R diagram** below (use Crow's Foot notation) that models the above relationships. Implementation-readiness means that your diagram would contain adequate info that can lead to tables creation and data insertion via SQL statements.


Customer
PK
CUST_ID
CUST_NAME
CUST_ADDRESS
...
PurchaseOrder
PK
PO_NUM
FK1
CUST_ID
FK2
SI_NUM
Stock Item
PK
SI_NUM
FK
PO_NUM
Item Name
Item Price
.....

Bonus (0.5*10=5 points)

Solve the **SQL-themed** crossword puzzle below - 10 hints are provided, for the 10 words we seek (ignore the rest!). Due to its incompleteness, it's more like fill-in-the-letter-blanks rather than a crossword puzzle, really :)



Across
1. The command that permanently saves changes to a db
3. A comparison operator that checks whether a subquery returns any rows
8. A command that allows attributes to be changed in one or more rows of a table
9. A clause that specifies that output values should all be different from each other (no repetitions)

Down

2. A clause that is useful for organizing (eg. for presentation) the output of a SELECT query
4. A comparison operator used to check if an attribute has a value
5. A query that is embedded ('nested') in another query
6. An aggregate function that yields the total of all values in a column or expression
7. The name given to a symbol that can be used as a substitute for one or more characters in a 'LIKE' clause
10. A special comparison operator used to check if a value lies within a range

Answers: COMMIT, EXISTS, UPDATE, DISTINCT, GROUPBY, ISNULL, EMBEDDED, SUM, WILDCARD, BETWEEN