CSCI-585 Fall 2022 Midterm Exam Rubrics

Q1

```
SELECT *
FROM SalesData AS S0
WHERE sales_amt IN (SELECT S1.sales_amt
FROM SalesData AS S1
WHERE S0.district_nbr = S1.district_nbr
AND S0.sales_amt <= S1.sales_amt
HAVING COUNT(*) <= 3)
ORDER BY S0.district_nbr, S0.sales_person, S0.sales_id,
S0.sales_amt;
```

```
SELECT DISTINCT district_nbr, sales_person
FROM SalesData AS S0
WHERE sales_amt <= (SELECT MAX(S1.sales_amt)
FROM SalesData AS S1
WHERE S0.district_nbr = S1.district_nbr
AND S0.sales_amt <= S1.sales_amt
HAVING COUNT(DISTINCT S0.sales_amt) <= 3);
```

Question: what does each of the two queries do, which makes them be alike and also distinct?

Output of both queries **(2 points)**
Query 1: Outputs utmost **top three sales** per district based on sales amount and sorts it by district number, sales person, sales_id and sales_amt
Query 2: Outputs distinct district_nbr, sales_person pairs [**top three salespeople**]

Similarity: Using Correlated Sub queries **(1.5 points)**
Difference: The subquery in second part is redundant **(1.5 points)**

For similarity, this is ok too - they both output 'top' results.
For diff, this is ok too: the first one is about amount (regardless of who is involved], whereas the second one is about people.

Q2 In computational structures (software, databases etc.), a strong dependence of any component of a system, on any other component, is in general, a 'bad' thing. Why?
Explain the above, when it comes to file formats for data storage.
Explain the above, in terms of an existence-dependent entity.
Note/hint - we went through all the above, in our lectures!

High dependency in a software/database system results in high maintenance requirements. This can result in various long term issues such as:

1. It raises security risk because some of the dependencies may be outdated
2. It becomes more difficult to disover bugs in the code or other performance issues
3. Modifications in dependencies can result in breaking of its dependent components
4. ALSO (a different way to state #3): independent modifications of the parts become impossible

Student can reply with similar opening answer to the first question. Any ONE reason for the first questions “why?” is enough.

When it comes to file formats for data storage, strong dependencies will result in data dependence. Access to information is dependent on the characteristics/format of its dependencies. Let’s say there is a change to the file format of one of the data storage files, this might result in its dependent component not being able to access the correct data.

For the second question they should be able to relate this scenario to data dependence and give one reason to why this is bad.

When it comes to existence-dependent entity, strong dependencies will result in structural dependence. Access to information is dependent on the structure/existence of its dependencies. Let’s say there is a modification in the data structure where we remove an entity from a table, any dependent component which relies on this entity will not be able to access the correct data.

Similarly, for the third question they should be able to relate this scenario to structural dependence and give one reason to why this is bad.

Q3 Where does data come from? In other words , what constitutes data? Note/hint - in the world, there is no such inherent thing as ‘data’ (but there is matter, for example!). Please explain carefully (the question is worth 5 points!).

Data is the intrinsic or extrinsic characteristics of matters or relation among groups of matters. It can be quantitative or categorical. It comes from a collection of measured facts and statistics collected together for reference or analysis.

It has to be identified, collected, observed, interpreted, or even generated by an intelligent agent(humans, animals / machines).

Student can reply with any similar concept, and optionally, some examples to elaborate it.
The conceptual high level definition of **Where does data come from** or **what constitutes data** worth **5 points.**

If the explanation is off (for instance, you give them 3 points), it is ok to use more specific answers as below, each can add 1 point until this question reaches 5 points capped. But please note that the answer should adhere to the fact that data does not inherently exist in nature.

1.measurement:
    e.g., the height of students
2.semantic categorization:
    e.g., genders of the students (gender is identified by human, society or biosystematics), the quality of product (good/bad) which is defined by industrial standard.
3. Quantized observation:
    e.g., image pixel color data, voxel data, mesh data, Lidar/Radar data, phaseorial data.
4. processed numerical or categorical values:
    e.g., the projected future GNP/GDP of a country. It can not be measured since it's not happening yet, but can be calculated and used as data.
5. associating qualities/characteristics/features... to somethng (an entity) - eg color, weight, CO2 level, salary... and quantifying them [the quantification's result is what we call 'data'].

Q4 Consider two tables T1 and T2. Symbolically, here are 4 potential operations on them:

T1 + T2
T1 - T2
T1 * T2
T1 / T2

Which three have we studied (ie. which one is 'fake news' and shouldn't be on the list)? Explain any two, of the valid three, using small examples.

T1 - T2, T1 * T2, T1 / T2 [T1+T2 is valid].
2 pts, if pointing out the 3 operations correctly or pointing out the "fake news". Deduct one point for each missing operation.

T1 - T2:

T1:

C1	C2

T2:

C1	C2

T1-T2:

C1	C2

1	a
2	b
3	c

1	a
4	d

2	b
3	c

T1 * T2:

T1:

C1	C2
1	a
2	b

T2:

C1	C2
1	a
4	d

T1*T2:

T1.C1	T1.C2	T2.C1	T2.C2
1	a	1	a
1	a	4	d
2	b	1	a
2	b	4	d

T1 / T2:

T1:

C1	C2
1	a
2	a
3	c

T2:

C1
1
2

T1/T2:

C2
a

Total 3 pts.
1 point for correct Difference example.
1 point for correct Product example.

1 point for correct Divide example.

Q5 **What is the point of normalization? Be specific in your answer; also, do NOT list the steps in the process (the question is not, ‘how does normalization work?’).**
What is the point of denormalization? Again, be specific.

Normalization:
Normalization aims to improve the database structure in order to create an appropriate database design. *The main goal of database normalization is to minimize data redundancies.*
By reducing redundancies in the database, data anomalies (e.g. insert/update anomalies) will be reduced as well.
3 pts. If answer mentions minimizing data redundancies
-1 if minimizing data redundancies is not mentioned in the answer, but the answer still mentions reducing data anomalies.

Denormalization:
Denormalization is performed on databases that have been normalized. *The main goal of database denormalization generally is to improve performance.*
For example: Improving performance on certain queries, in a normalized database it might be necessary to join many tables (which each might contain large amounts of data) to return a specific result. By denormalizing (i.e. creating redundant data, adding pre-computed values) it will be possible to obtain this result with simpler queries that require less processing time and power.
2 pts. If the answer mentions improving performance.

Q6. Come up with one new example (which we did NOT discuss at all), for each of these: a unary non-recursive relationship, a unary recursive relationship, a binary relationship, a ternary relationship, a quaternary relationship .

Below are some relationships discussed in class (re-using these examples will result in a penalty as described below - so good answers will NOT be any of these):

Unary recursive:
Employee-manager, Course-course, Academic paper-references

Unary non-recursive:
Student-mentor, Parent-child, Employee-spouse

Binary relationship:

Professor-course, Student-course, Professor-school, School-department, Department-course, Course-class, Professor-department, Professor-class, Student-class, Department-student, Professor-Student, Building-Room, Room-Class

<u>Terinary relationship:</u>
Doctor-drug-patient, Agent-Customer-car, Professor-department-courses

Any 1 valid example for each unary non-recursive relationship, unary recursive relationship, binary relationship, ternary relationship, and quaternary relationship. (1 mark for each example) -0.5 for reusing examples mentioned in class.(For eg: if all 5 answers are re-used examples, the student would be awarded 2.5 marks).

Q7 Generally speaking, why are NULLs bad, in relational modeling (ie. in tables)?
Discuss, with a small example each, two different ways (reasons) why NULLs are not desirable.
(1 point) Any valid explanation for why Nulls are bad
(1+1) Any Valid Explanation 1 with supporting example
(1+1) Any Valid Explanation 2 with supporting example
(-0.5) If any example is invalid or has errors
Some valid reasons and examples could be:

- NULLs consume space in a sparse table where most values are NULL and rows could be avoided altogether.
- NULLs hinder in creating 1NF
- Handling NULLs can get difficult in applications that extract data
- Creating foreign keys on columns containing these values is impossible
- It gets difficult to query the table when the inner query contains null values combined with a “not in” clause.
  Say for example: select * from table1 where id not in (select id from table2)
  Here, if table2 contains Null values, this translates to “where id not in [NULL, …other values]. The clause “id != Null” will be true for all values of id in table2 that are not null. Meaning, the whole query would never return any values.
- NULLs signify uncertainty in data, which is a bad thing - eg. a NULL value for ‘allergies’ for a patient might mean the patient has none, or that we don’t know (yet)

Q8 (1+2+2 = 5 points).

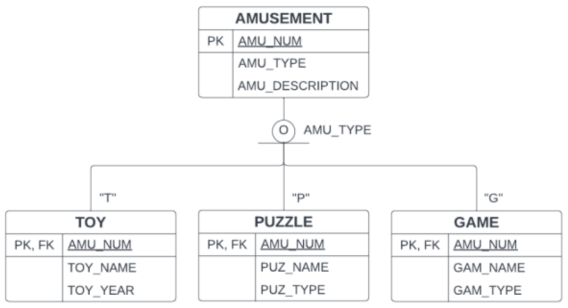
Why are EER diagrams useful?

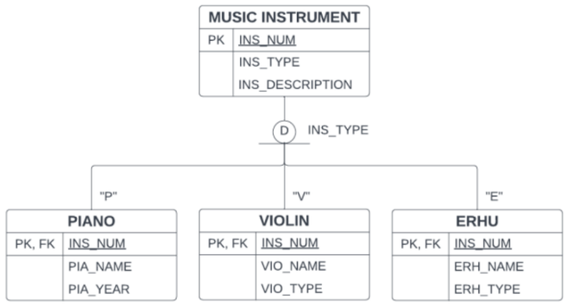
Flashback - your childhood, where you likely played with toys/games/puzzles (let’s call them all, ‘amusements’), and/or played with musical instruments.

Represent ‘amusements’ as an EER diagram, and musical instruments as another EER diagram. Make sure to include sample columns for each entity you list.

Sample answer:

- (1 point) Part 1: Anyone of the following:
    - EER diagrams (EERD) are useful, because EERD depicts arrangement of higher-level entity supertypes and lower-level entity subtypes.
    - ERRD provides the means to 1) support attribute inheritance, 2) define a special supertype attribute known as the subtype discriminator, and 3) define disjoint/overlapping constraints and complete/partial constraints.
    - EER models are helpful tools for designing databases with high-level models. With their enhanced features, you can plan databases more thoroughly by delving into the properties and constraints with greater precision.
- (2 + 2 points) Part 2: EER diagrams of amusements and musical instruments are presented as follows.





Rubric:
For the first part of the question, no need to be exactly the same as the sample answer; any similar answer is acceptable.
For the second part of the question, subtypes should be either the subtype of amusements or music instruments.

For musical instruments, the second level (the one below MUSICAL INSTRUMENTS) would (could) be wind instruments, string instruments, percussion instruments, electronic instruments etc - under each of THOSE will be flute, violin, bongos, drum machines, etc.

Major error for part 1 (Maximum -1 point)

- -1.0 point. If the answer does not mention why the EER diagram is useful.

Major error for part 2 (Maximum -4 points)

- -4.0 points. If the answer does not contain any diagram.
- -2.0 points. If the EERD for “amusements” is missed.
- -2.0 points. If the EERD for “music instruments” is missed.
- -1.5 points. If anyone tries to use some diagram to describe the relationship between supertype and subtypes for “amusements”, but the diagram in the answer has nothing to do with EERD.

- 1.5 points. If anyone tries to use some diagram to describe <u>the relationship between supertype and subtypes</u> for “music instruments”, but the diagram in the answer has nothing to do with EERD.
- -1.0 point. If the EERD for “amusements” does not contain attributes.
- -1.0 point. If the EERD for “music instruments” does not contain attributes.

Minor error for part 2 (Maximum -2 points)

- -0.5 points. If the answer misses or misuses the constraints between supertypes and subtypes in **one** EER diagram. <u>Maximum -1 point in this item</u>.
    - -0.5 points. If it misses the category symbol, or the character in the circle is neither “D” / “O”.
    - -0.5 points. If it misses the partial/complete constraint.
    - Generally, both scenarios can be complete or partial constraints. No deduction for either complete/partial constraints.
    - No deduction as long as the overlapping/disjoint is meaningful according to the relationships between supertypes and subtypes.
- -0.5 points. If the subtype discriminator is missed, or the subtype discriminator is not an attribute in supertype.
- -0.5 points. If the supertypes and subtypes do not have the same PK.
- -0.5 points. If the subtypes’ PKs are not FKs.
- -0.5 points. If there are some other errors.
- No need to deduct if the answer is in the following situation.
    - If the subtype does not have unique attributes.
    - If the subtype does not have the abbreviation as the example or slide.
- Repeated occurrences of the same error in the chart are only deducted once, unless otherwise specified.

Note:

- Specialization hierarchy refers to page 7 of EER slides.

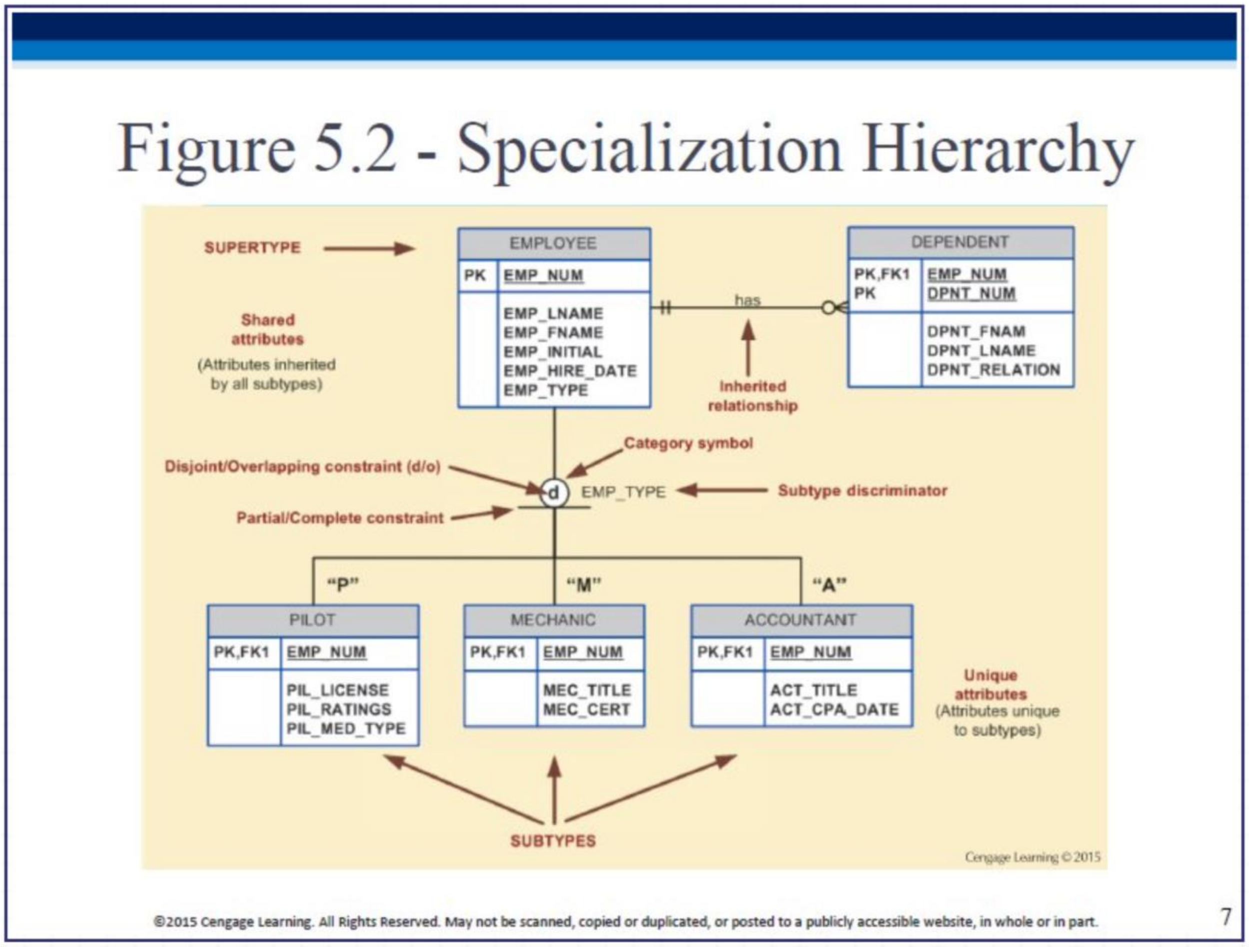
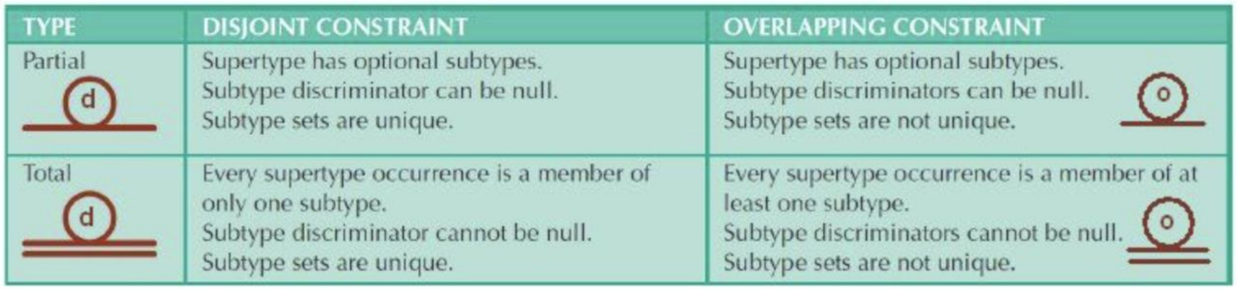


Figure 5.2 - Specialization Hierarchy

7

- Hierarchy constraint refers to page 14 of EER slides.



TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

Q9 Why care about 'transaction management' (TM) at all?

If you decide to NOT implement specific TM algorithms (eg. 2PL, deadlock detection), what are the two extreme ways in which you can operate? The answer does not need to take any sort of 'efficiency' into account (it is a conceptual question). List each extreme, and explain in a few lines, the ramifications (ie. implications) of each. Note - we went through these in class!

Answer: we care about TM, to ensure that multiple, simulatenous ('competing') operations do not end up clobbering each other, ie producing incorrect results (eg in a report) or corrupting data (via anomalies).

Two extremes:
1. make the transactions be strictly serial - when one is executing, no other one can (they need to wait in a time-ordered sequence)
2. make the transactions be entirely lock-free and simultaneous, potentially overwriting data [which we would fix later by analyzing logs and discovering and fixing any problems]

Q10 A table cell is not supposed to contain multiple values for an attribute (eg. 'Degree obtained', 'Coding language').
How would you get away with 'doing it anyway' in a cell, ie. if you can 'cheattu'? Provide an example, and mention how you would handle it (ie how you would access the multiple values).
How would you do it, if you cannot (aren't allowed to) 'get away' with cheating, ie how would you do it for real? Provide a small example.
In real life, such a need to represent multi-values ended up leading to a massive change in data modeling - what is the change? Your answer can be general, no need for specifics.
(2 points) If any valid explanation from the following
Store data with any delimiter (eg: comma, dollar sign, ampersand etc) separated string
OR
Store stringified JSON data
OR
Any other string type in which we can store multiple data and retrieve the same data again from string

(1+1 point)
Award 1 point if student have mentioned, create another database table for multiple values and do 1 to many mapping.
+1 point if there is a correct and proper example
Possible examples:
student => s_id, s_name; student_phone => s_id, phone
customer => c_id, c_name; customer_address => c_id, c_address
shop => s_id, s_same; shop_open => s_id, weekday

-0.5 point if the example is proper but not correctly presented

(1 point)
Databases evolved from structured SQL databases to unstructured NoSQL databases.