CS585 Fall ’15 Test 1

Common Mistakes

Q1

Explain the difference between logical and physical data independence.

Background / Recap

- Three-schema approach in DBMS architecture:
  - External (user, application, query)
  - Conceptual / Logical (tables, columns)
  - Internal / Physical (data structures, hardware)

- Sometimes broken down into even more detailed levels
  - but for this question, just the three layers is sufficient

Q1

Definitions

- Physical Data Independence

  Requires that changes at the physical level (like data structures) have no impact in the applications that consume the database. For example, let's say you decide to stop using a Hash Index in your table and decide to use a B-Tree Index instead: Your application that executes queries against this table doesn't have to change at all.

- Logical Data Independence

  States that changes at the logical level (tables, columns, rows) will have no impact in the applications that access the database.. This feature is harder to implement that Physical Data Independence but there are still cases when this feature works. For example, if you add tables, columns or rows to your current scheme the already working queries aren't affected at all.

Q1

Common Mistakes

- Stating “software” instead of “logical”
- Misunderstood “internal” to mean “logical”
- Misunderstood “logical data independence” to be between internal/physical and logical/conceptual, instead of between logical and external/user.
- Discussing software vs. hardware. “Software” can span across all of the layers.
- Talking about functional dependencies (unrelated subject!)

Q2

Relational algebra operations

Background / Recap

- Relational algebra is not SQL.
  (No, you don't type these queries into DBMS)

- Pure sets, no duplicates
- null = no value
  - if you have a whole row of only nulls, that row doesn't actually exist!

Q2

a) Projection

Common Mistakes

- Include “Paris” duplicate row
- Include null row

city
Paris
Berne
Oslo

b) Union

Common Mistakes

- Include null row
- Other errors propagated from (a)

city
Paris
Berne
Oslo
Rome

Q2

c) Set Difference

city
Oslo

Common Mistakes

- Include “Rome” (from RHS of A-B)
- Include “Paris” (from to duplicate row in (a)?)

d) Intersection

Common Mistakes

- Include “Rome”, “Paris” or null row

city
Oslo

Q2

e) Natural Join

Common Mistakes

- Include two “city” columns
  - **Natural** Join eliminates the duplicate column
- Joining when both P.city and J.city are null
  - Only join when the specified fields have the **same value**!
  - “no value” can’t “have the same value”
- Missing P# and J# columns

P#	city	J#
P1	Paris	J1
P3	Rome	J2
P3	Rome	J5

Q2

f) Full Outer Natural Join

Common Mistakes

- Include two “city” columns
- Joining when both S.city and P.city are null
- Missing S# and P# columns

S#	city	P#
S1	Paris	P1
S2	Berne	P2
S3	Oslo	.null.
S5	Paris	P1
.null.	Rome	P3

Q3

Writing SQL

- Generally few problems =)
  - Still fresh from doing HW2

a) Find the flight numbers of all the flights originating from Vancouver which depart **after** “13:00”

Common Mistakes

- depart_time **=** “13:00”

  - Careless mistake!

Q3

c) List the aircraft ids of aircraft that have sufficient range to cover the distance from Vancouver to Tokyo (using any existing flight plan / route).

```
SELECT aircraft_id FROM aircraft
WHERE range > ANY (
    SELECT distance FROM flights
    WHERE from_city = 'Vancouver'
      AND to_city = 'Tokyo'
)
```

Common Mistakes

- Did not specify range > ANY or select MIN in nested query
  - Should not compare single value to a naked set. Bad practice. Even if DBMS allows query to run, results may be unpredictable!

Q3

c) List the aircraft ids of aircraft that have sufficient range to cover the distance from Vancouver to Tokyo (using any existing flight plan / route).

```
SELECT aircraft_id FROM aircraft
WHERE range > ANY (
    SELECT distance FROM flights
    WHERE from_city = 'Vancouver'
      AND to_city = 'Tokyo'
)
```

Common Mistakes

- Used range > ALL or select MAX in nested query
  - The condition is: as long as aircraft can fly *at least one* of the existing flight plan or route!

Q4

Interpreting SQL

- Some difficulty with complex parts of the queries
- Surprisingly easy to omit something if you don't carefully and systematically explain the whole query thoroughly!
- Important to ***explain*** and not just transcribe the SQL

Common Mistakes

- Need to be explicit about what fields are being returned in the result! (partial credit was given)
  - e.g. "Give the titles of books", not just "Give the books"

Q4

b)

```
SELECT name FROM borrowers P
WHERE member_since >= 1998 AND NOT EXISTS (
   SELECT * FROM books B WHERE NOT EXISTS (
     SELECT * FROM copies C, loans L
     WHERE B.book_id = C.book_id AND C.copy_id = L.copy_id
       AND L.borrower_id = P.borrower_id))
```

Find the names of borrowers who have been members since 1998, and who have borrowed every book before (or: for which there is no book that they have not borrowed before).

Common Mistakes

- Omitted “1998” part ← careless!
- The nested query was challenging to unwrap…
  - Innermost: gives records of book B borrowed by person P
  - Next: gives books, <u>for which we cannot find any</u> (book was borrowed by P)
    - i.e. gives books that have never been borrowed by P
  - Finally: give person, <u>for which we cannot find any</u> (books that they have never borrowed before)

Q4

c)

```
SELECT L.branch_id, L.branch_ name, AVERAGE(num_pages) AS pagecount
FROM branches L, books B
WHERE EXISTS (SELECT * FROM copies C
              WHERE L.branch_id = C.branch_id
                AND C.book_id = B.book_id)
GROUP BY L.branch_id
ORDER BY pagecount
```

List every branch. For each branch, give its ID, name, and the average number of pages across all books available in that branch (not counting multiple copies). Sort the results in increasing order of the average page count.

Common Mistakes

- Did not mention "**list every** branch" (or similar wording)
- Order is mentioned, but did not explicitly state "**ascending**" (partial credit)
- Average page count of which books? Those "available **in that branch**"
- Omitted detail: multiple copies of the same book not counted in averaging

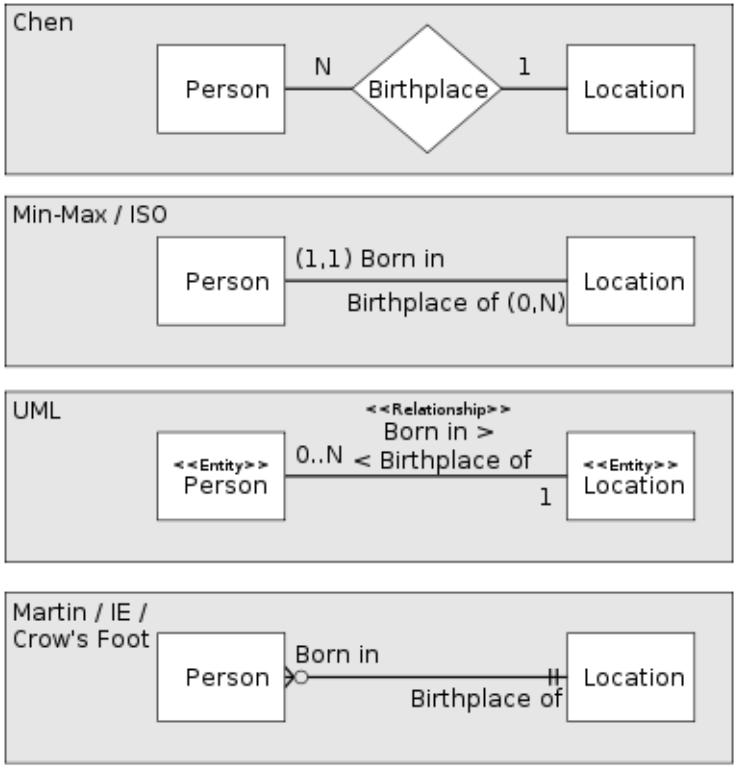
Q5

ER diagram

- Follow instructions! Explicitly instructed to use Chen's notation, and do not use crow's foot.

There are many different ER diagram notation conventions

- Always be mindful of which you are using / is being used
- Advice: use one consistently
  - mix and match = get confused



(diagrams: wikipedia)

Q5

Common Mistakes

- Using (M,N) cardinality notation, wrote it on the wrong side
  - It's the "odd one out", opposite side compared to the other notations

- Extra (redundant) relationships
  - "athlete(s) – belongs to – team" or "team – consists of – athlete(s)"
  - Either one is enough, no need to have both!
  - Even worse, cardinalities of the relationships are contradictory
  - Possible cause: reading the business rules ***far too literally*** sentence-for-sentence into entities, attributes, relationships ***without understanding*** the big picture

Q5

Common Mistakes

- Inappropriate (not meaningful) names
  - If I take away the question and read only your ER diagram, can I still understand what the entity/attribute/relationship means?
  - e.g.: “type” / “boolean” / “yes/no” / etc…
    instead of “isOlympic” (attribute)

Q6

Common Mistakes

- Reversed the two entities
  Description clearly gives “telephone number” vs. “name**s** of actor**s**”
  which should indicates actors’ names is the multi-valued attribute A4.

- Putting attributes that look/behave like “foreign keys”
  Foreign keys don’t exist in ER diagram, only primary keys.
  Relationship in ER diagram, when converted into relational tables,
  cause you to need to make foreign keys (to refer to PK of other table).

- Included “borrowing habits” or “heavy borrower”
  No, we store raw data (i.e. all past & current DVD loans).
  Things like “habits”, “heavy borrower”, “don’t return quickly” are analysis
  at the application level (applying some business logic).

Q6

Common Mistakes

- "Status" (borrow/returned) attribute
  (under DVD entity, or under borrow relationship)
  Unnecessary. If there is a borrowed_date but null returned_date then we know the DVD is currently loaned out.

- A5&6 are "borrow/returned" and "date"
  You get table ***Loan***(friend,dvd,b/r,date) instead of ***Loan***(friend,dvd,b_date,r_date)
  Very poor design; loses coupling of borrow+return as a single transaction.

- System can handles only "current status" and not "full history"
  aka: C1max=1 "because a DVD can only be physically loaned out to one friend at a time"
  Poor design. The problem statement indicates a need to have historical loan info to study and analyze borrowing habits / trends

Q6

Common Mistakes

- Min cardinalities set as “1”
  (when without valid justifying assumption) – (partial credit given)
  In general, unpopular DVD might never be borrowed; and a friend can become “library member” with initially no loans.

- Invalid cardinality values (partial credit given)
  The given diagram is using the “M..N” cardinality notation.
  Valid values are 0,1,2,...,N,M
  Not valid: “many” or “∞” (infinity sign)

Q7

Persons *(SSN, Name, StreetAddr, City, State, Zip, Phone, Employer, EmployerHQAddr)*

- Important: you are told that the table is unnormalized!

Common Mistakes

a)

- "SSN+employer" for primary key (partial credit given)
  - No. The wording in the question is to keep the table unnormalized, so SSN is enough for the primary key
  - Even if you assume one person can work for multiple companies, employer is just a multi-valued attribute in the unnormalized table. "SSN+employer" only becomes PK after normalizing into 1NF.

Q7

Common Mistakes

b)

- Invalid or incorrect notation for functional dependencies
- Functional dependency arrows pointing the wrong way

- Missing functional dependencies
  - need to be careful in order to exhaustively list all FDs
- Wrong functional dependencies
  - trying too hard … listing things that aren't actually FDs

- Remember: X→Y means "X uniquely determines Y"

Q7

c) If this relation were *used as defined (without normalization)*, describe an update anomaly that could arise.

Common Mistake:

- Stated something that isn't *an anomaly arising from using the unnormalized table* as the answer.
  - i.e. things that can't be blamed on leaving the table unnormalized
  - ask yourself: will the problem go away if I normalized the table?
  - e.g.: Company changes name. "Google" for many people needs to be changed to "Alphabet", and thus forgot to update all of the rows.
  - even after normalize to 3NF, this problem will still remain!

Q8

Common Mistakes

- Misunderstanding the table given in the question.
  - The advisor (e.g. Flor in our CS Dept.) is not the instructor for the course. She can be considered as the department academic advisor.
  - By inspecting the data, it should be clear that the advisor and the course numbers are not correspond or related in any way.
  - Each student has one advisor, and has some “RegisteredCourses”. So it is all about the students’ registered courses.
    Not “course taught by instructor”.

Q8

Common Mistakes

- Splitting the registered class column into two columns
  - Generally, when you have two separate columns, they have separate semantic meaning.
  - The question gives, e.g. student 123 has registered classes 102-8 and 104-9, which have equivalent meaning
  - As opposed to column1 = 102-8 and column2 = 104-9 which implies some ordering or different business rules handling for each one.