Databases Supervision 3 (srns2)

Question 1

Adding a movie is really straightforward in a document database. First, a movie object is created as follows:

```
movie = {
  'title': 'Movie title',
  'actors' : [...],
  '...': ...
}
```

✓ but what about keys & relations?

This can then simply be added to the database as follows:

```
movies['movie_id'] = movie
```

Then this can be written to the database file. The new movie has now been added to the database.

Question 2

This can be written as follows (using the `get_actor_by_id` and `get_actor_by_name` methods defined in the tutorial):

```
def closest_relationship(a1_id, a2_id):
  if a1_id == a2_id:
    return 0

  a1 = get_actor_by_id(a1_id)
  a2 = get_actor_by_id(a2_id)

  min_steps = None;

  for movie in a1.acted_in:
    for actor in movies[movie.id].actors:
      steps = closest_relationship(actor, a2)
      if min_steps is None or steps < min_steps:
        min_steps = steps

    return 1 + min_steps


def bacon_number(a):
  kevin_bacon = get_actor_by_name('Kevin Bacon')
  return closest_relationship(a, kevin_bacon.actor_id)
```

✓

This is far less efficient than the graph database version as it iterates through every movie an actor acted in and then every actor in each of those movies and will blow up very quickly. It is also far less readable. ✓ Find shortest path

Question 3

Yes, this can be done. First, the two entities of `movie` and `actor` are drawn. The first step is to identify the primary keys in each of these entity types. Once `movie_id` and `actor_id` have been identified, the next step is to find the relationship between then. This is done by looking for where in a `movie` object an `actor_id` appears, and vice versa, which is used to draw the relationship onto the E-R model. Finally, all other fields in each of these are added as properties. ✓

what are the challenges?

Question 4

Part (b)

Similarities	Differences
✓ They are both declarative, in that you specify what you want rather than specifying how to do it.	Relational algebra obeys set semantics, while SQL obeys multiset semantics, meaning that SQL can have duplicate records. ✓
✓ They both use set theory to organise their records. relational	Relational algebra is written in mathematical notation while SQL uses a programming syntax. ✓

Part (c)

These expressions are not equal. Consider the case where v = ~~3,~~ and R and S are as follows:

R

a	b
1	3
2	3
2	1
3	4

S

b	c
1	5
2	6
3	7
4	8

Finding $\sigma_{a = 3 \text{ or } b = 3}$ (R ⋈ S)

Natural joining R and S results in:

R ⋈ S

a	b	c
2	1	5
1	3	7
2	3	7
3	4	8

Therefore selecting the records where a = 3 or b = 3 results in:

$\sigma_{a = 3 \text{ or } b = 3} (R \bowtie S)$

a	b	c
1	3	7
2	3	7
3	4	8

Finding $(\sigma_{a = 3} (R)) \bowtie (\sigma_{b = 3} (S))$

Now selecting the records from R where a = 3:

$\sigma_{a = 3} (R)$

a	b
3	4

Now selecting the records from S where b = 3:

$\sigma_{b = 3} (S)$

b	c
3	7

Now natural joining these: ✓

$(\sigma_{a = 3} (R)) \bowtie (\sigma_{b = 3} (S)) = \varnothing$

This results in the empty set as there is no common b.

Conclusion

In this example, $\sigma_{a = 3 \text{ or } b = 3} (R \bowtie S)$ is not equal to $(\sigma_{a = 3} (R)) \bowtie (\sigma_{b = 3} (S))$. Therefore these expressions are not equal. ✓

Question 5

Part (a)

Online transaction processing is designed for day-to-day operations. It updates small amounts of data in the database many times by many users, and is therefore optimised for updates and doesn't lock data for very long. ✓

Part (b)

Online analytical processing is designed for analysis. It reads huge amounts of data but doesn't need to update the database much so it doesn't tend to lock the database much as nothing changes much. It is therefore optimised for reading data. ✓ historical

Part (c)

OLTP is designed for day-to-day operations, so is optimised for updates, as its transactions are mostly updates. OLAP on the other hand is designed for data analysis, and is therefore optimised for reads as most of its transactions will be reads. ✓

The data in OLTP is current while the data in OLAP is historical (for analysis). ✓

OLTP has low data redundancy while OLAP has high data redundancy. ✓

OLTP tends to have large databases but OLAP tends to be far larger. ✓

Part (d)

OLAP has a fixed, strict schema, while NoSQL doesn't require a strict schema and allows greater flexibility in how data is stored. Additionally, NoSQL may have additional data types such as lists. NoSQL may also have other features such as graphs that make it easier to interpret relationships between nodes. write vs. read? ✓

Part (e)

A social media website would likely use a combination of OLTP, OLAP and NoSQL. ✓

It will use OLTP to store data about users, posts to show them, their messages, friend requests, and most other things users will encounter on a daily basis. ✓

It will use OLAP to analyse information about its users as a whole. If its aim is to keep users on the site for as long as possible to maximise ad revenue, it will analyse huge amounts of data from all of its users to determine factors which maximise the amount of time users spend on the site. ✓

It could use NoSQL in various ways, such as with a graph database to store links and relationships between users and use this to suggest friends or people the user may know. ✓

To implement this, the OLTP system would likely use a version of SQL, such as MySQL or PostgreSQL. The NoSQL system would be something like Neo4j or another graph database. The OLAP system could use many different types of database, as it likely has a fixed schema so SQL would be appropriate, but it may have many relationships to analyse, meaning something like Neo4j would be better, but in any case it needs to be optimised for reads and not updates. ✓

Question 6

Part (i)

$\pi_{gender}(RockStar) \subseteq \{'Male', 'Female'\}$ ✓

Part (ii)

$\pi_{starname}(RockManager) \subseteq \pi_{name}(RockStar)$ ✓

Part (iii)

$size(\pi_{name}(RockStar)) = size(\pi_{address}(RockStar))$ (*This is assuming that address is not null so every record with name has at least one address.* bbv)