Min Thant Hein
ID: PIUS20230001

Consider the following three tables.

$S(\underline{sid},\ A)$

sid	A
s1	a1
s2	a2
s3	a3

$R(\underline{sid},\ \underline{tid},\ B)$

sid	tid	B
s1	t1	b1
s1	t2	b2
s2	t1	b4
s2	t3	b5

$T(\underline{tid},\ C)$

tid	C
t1	c1
t2	c2
t3	c3
t4	c4

(*a*) Represent the information in these three tables in a single table, using NULL values where needed. [4 marks]

(*b*) Represent the information in these three tables as three JSON objects, each associated with one of the values of the *sid* key. Is any information lost? [4 marks]

(*c*) Represent the information in these three tables as four JSON objects, each associated with one of the values of the *tid* key. Is any information lost? [4 marks]

(*d*) We now have three distinct ways of representing the same information (the original tables, one big table, and the collection of JSON objects from parts (*b*) and (*c*)). Carefully compare and contrast these approaches and discuss their related advantages and disadvantages. [8 marks]

(a)
Query lines:

```
SELECT S.sid, S.A, R.tid, R.B, T.C
FROM S
LEFT JOIN R
ON S.sid = R.sid
LEFT JOIN T
ON R.tid = T.tid;
```

Output result:

sid	A	tid	B	C
s1	a1	t1	b1	c1
s1	a1	t2	b2	c2
s2	a2	t1	b4	c1
s2	a2	t3	b5	c3
s3	a3	NULL	NULL	NULL

##########

(b)


```
{
  "s1": {
    "A": "a1",
    "R": [
      {"tid": "t1", "B": "b1"},
      {"tid": "t2", "B": "b2"}
    ]
  },
  "s2": {
    "A": "a2",
    "R": [
      {"tid": "t1", "B": "b4"},
      {"tid": "t3", "B": "b5"}
    ]
  },
  "s3": {
    "A": "a3",
    "R": [ ]
  }
}
```


Yes, we lost some information. For example, the C values from T are not included unless we nest them too. If we only use sid as key, tid-related information from T is missing.

##########

(c)


```
{
  "t1": {
    "C": "c1",
    "R": [
      {"sid": "s1", "B": "b1"},
      {"sid": "s2", "B": "b4"}
    ]
  },
  "t2": {
    "C": "c2",
    "R": [
      {"sid": "s1", "B": "b2"}
    ]
  },
  "t3": {
    "C": "c3",
    "R": [
      {"sid": "s2", "B": "b5"}
    ]
  },
  "t4": {
    "C": "c4",
    "R": [ ]
  }
}
```


Yes, we lost some information. For example, the A values from S table are missing unless we include them in the nested objects.

############

(d)

For the original tables, they are normalized, one single value per cell. They avoid redundancy, which means storing the same data in multiple places or cells. These are the advantages of the original tables. On the other hand, they need to join for the combination of data. This is the disadvantage of the original tables.

For one big table, it is easy to query without joining query lines! This is the advantage of one big table. On the other hand, redundancy, such as the repetition of A and C, having NULL values, and the UPDATE anomalies are the disadvantages of one big table.

For the collection of JSON objects from part (b) and part(c), they are easy to organize data which are similar to tree data structure. They are also good for APIs because they use a simple and text-base format which is easy to read, write and understand. Thus, it is more efficient for data exchange between client and server. These are the advantages of the collection of JSON objects. On the other hand, it is difficult to keep data integrity, which means data are accurate, complete and consistent. Thus, it can lead to duplication of data. This is the disadvantage of the JSON objects.

############