s

SCHOOL OF COMPUTING, ENGINEERING AND MATHEMATICS

SEMESTER TWO 48 HOUR COURSEWORK 2019/2020
ANSWER SHEET
CI330

Data Management

Student Number: ____17806554_____

QUESTION 1: DISTRIBUTED DATABASES

1 a)

Bookings are kept on local servers as a worker in any other branch than their respective one does need to be able to access booking from other branches. For examples, someone working in Brighton Welcome Hotel would not need to access any other Welcome Hotel branches, they are and should be only concerned with bookings in their respective hotel.

Moreover, a booking holds personal sensitive information; such as, a gusted which is linked to the guest table that holds personal sensitive information; e.g. Name, address and contact number.

1 b)

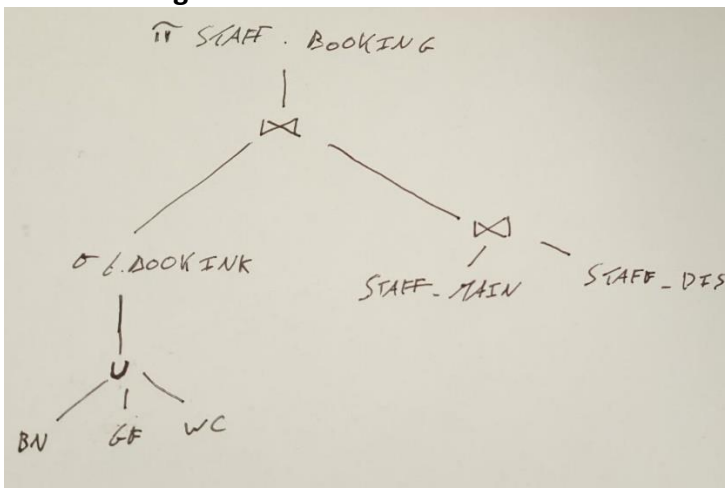
```
SELECT      staffID
INTO        #Staff(Brighton)
FROM        Staff_MAIN
WHERE        hotelID = 'BN'
```

1 c)

Horizontal vs Vertical distribution:
A horizontal distribution stores records as rows or tuples; i.e. a record or row. This process splits *client* or *server* information *logically* in a *physical manor*, resulting in parts that operate on their data set that is *complete* and *shared equalling* in a *balance load*. The booking table is distributed horizontally meaning its record are stored in rows.

A vertical distribution stores records as columns or attributes; i.e. a data item is an attribute. This process places *different components logically* on *different machines*. The staff table is distributed vertically meaning its information is stored as attributes vertically.

Relational Algebra:



1 d)

Security:
Data that is not required for local applications cannot be accessed and is not available for unauthorized users. This extend control and security is apparent in the Welcome Hotel Case Study as Staff members can only access room information from their local hotel. Moreover, staff information is only stored and accessed in the relative regional head office. These two features mean that only the appropriate people can access the information; for example, someone working in America will not be able to access staff or booking information from England's branch.

Fragmentation:
Fragmentation is the allocation & definition of *fragments* that are carried out in a strategic manor to achieve improved *availability* and *reliability* of resources. Fragmentation will also improve overall performance via a balance of storage capabilities with minimal cost of communication. In the case of this study with 400 hotels across 32 different countries fragmentation is vital and will be more so as the company grows as fragmentation will allow for a minimal cost of communication; i.e. rather than searching 400 hotels for a booking a staff will only search their hotel.

QUESTION 2: SECURITY AND LEGAL ISSUES

2a)

The GDPR covers any processing of personal data within the E.U. and U.K.

2b)

Personal data is considered to be information related to a identifiable or already identified individual.

In the instance of the case study, personal information would be the staff members name, their national insurance (NI) number, any disability information. Moreover, a guest's name or address.

2c)

Special category data is categorised as data that would require some extra measures of protection due to its sensitive nature; for example, a criminal record or disabilities. Moreover, for this case study disability information would fall under that category.

2d)

They must report to the supervising authority that is relevant this breach of data, the report must be filled by no later than 72 hours that they became conscious of the breach. Any and all individual's information that was breached must be contacted and informed as soon as possible. Moreover, a accurate and complete record of any and all personal data breaches must be kept. Although not every breach must be reported, this breach contains sensitive personal data and therefore must be reported.

QUESTION 3: Non-relational Databases, Database Selection

3a)

Columnar data storage:
A column approach stores / organises data in columns and rows, this approach holds columns that logically relate to each other and are manipulated or retrieved as a unit. Moreover, this method stores data in their key order rather than computing hash. All the columns that belong to a group are put together within the same file, each file containing a number of those rows.

For the booking table this method would present the data as follows:

roomID*	column family: room
	bookingStartDate: noOfNights: specialRequirements:

roomID*	column family: booking
	hotelID*: guestID*:

3b)

Non-relational databases are by design more flexible and scalable when compared to their counterparts; this is due to individual rows not requiring the same columns. This becomes an advantage in this case study as for a staff member if they are searching for only the room information, they can access it with much greater ease. The above-mentioned approach is great for storing large amount of data and if there are major changes to the database (which is likely to happen as the database hasn't been amended in a decade), with such a large database this becomes more apparent.

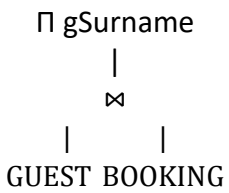
3c)

A dimensional database is harder to maintain in the case of large data warehouses compared to the relational database they have.
Moreover, this relational database hides physical storage details from users; i.e. physical data can be provided in a manner that reflect independent from each other.

Question 4 – Query Optimisation and Transaction Management

4a.

Π gSurname (*σ* guestSurname = “Samuels”) × ((GUEST) ⋈ (BOOKING))



This is assuming that they are running the query

4b.

Here are relational algebra symbols that you may wish to use in your optimised RAT.

Π *σ ρ* ⋈ ⋉ ⋊ *θ* ▷ **∧ ∨ ∪ ∈**

Π (hotelName, ROOM,roomID, roomRate£, bookingStartDate, guestID, noOfNights

|

|

σ (hotelCountry = “UK” ^ BOOKING.bookingStartDate = “13-04-2020” ^ ROOM.roomType=”double”)

|

|

BOOKING ⋈ Hotel ⋈ ROOM

4c.

i)

There is one room left, the problem is serializability.

ii)

Time:	booking1 (making booking)	booking 2 (cancelling booking)	number of remaining rooms for tonight (rr)
T1	begin transaction		2
T2	write_lock(rr)	begin transaction	2
T3	read (rr)	write_lock (rr)	2
T4	rr = rr – 2	wait	0
T5	write (rr)	wait	0
T6	commit & unlock (rr)	wait	0
T7		read (rr)	0
T8		rr = rr + 1	0
T9		write rr	1
T10		commit	1