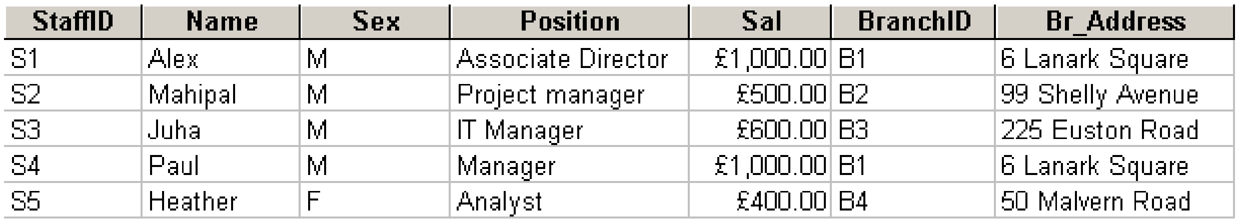
Homework #5

14.2 Discuss insertion, deletion, and modification anomalies. Why are they considered bad? Illustrate with examples.

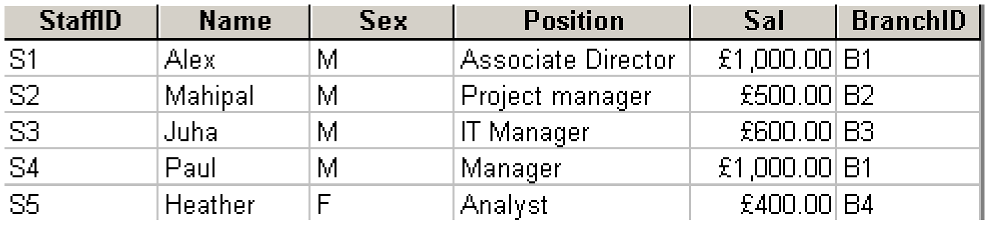
They are considered bad because, they lead to:

- It is difficult to maintain consistency of data in the database
- It leads to redundant data
- It causes unnecessary updates of data
- Memory space will be wasted at the storage level



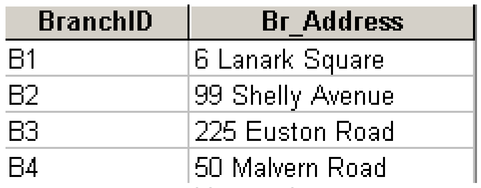
StaffID	Name	Sex	Position	Sal	BranchID	Br_Address
S1	Alex	M	Associate Director	£1,000.00	B1	6 Lanark Square
S2	Mahipal	M	Project manager	£500.00	B2	99 Shelly Avenue
S3	Juha	M	IT Manager	£600.00	B3	225 Euston Road
S4	Paul	M	Manager	£1,000.00	B1	6 Lanark Square
S5	Heather	F	Analyst	£400.00	B4	50 Malvern Road

Tbl_Staff_Branch



StaffID	Name	Sex	Position	Sal	BranchID
S1	Alex	M	Associate Director	£1,000.00	B1
S2	Mahipal	M	Project manager	£500.00	B2
S3	Juha	M	IT Manager	£600.00	B3
S4	Paul	M	Manager	£1,000.00	B1
S5	Heather	F	Analyst	£400.00	B4

Tbl_Staff



BranchID	Br_Address
B1	6 Lanark Square
B2	99 Shelly Avenue
B3	225 Euston Road
B4	50 Malvern Road

Tbl_Branch

The Problems resulting from data redundancy in an un-normalized database table are collectively known as update anomalies. So any database insertion, deletion or modification that leaves the database in an inconsistent state is said to have caused an update anomaly. They are classified as

- **Insertion anomalies:** To insert the details of a new member of staff located at branch B1 into the Tbl_Staff_Branch Table shown above, we must enter the correct details of branch numner B1 so that the branch details are consistent with the values for branch B1 in other rows.
  To insert the details of a new branch that currently has no members of staff into the Tbl_Staff_Branch table, it is necessory to enter nulls for the staff details which is not allowed as staffID is the primary key. But if you normalize Tbl_Staff_Branch, which is in Second Normal Form (2NF) to Third Normal Dorm (3NF), you end up with Tbl_Staff and Tbl_Branch and you shouldn't have the problems mentioned above.

- **Deletion anomalies:** If we delete a row from the Tbl_Staff_Branch table that represents the last member of staff located at that branch, (for e.g. row with Branch numbers B", B3 or B4) the details about that branch are also lost from the Database.

- **Modification anomalies:** Should we need to change the address of a perticular branch in the Tbl_Staff_Branch table, we must update the rows of all staff located at that branch. If this modification is not carried out on all the relevent rows, the database will become inconsistent.

14.3 Why should NULLs in a relation be avoided as much as possible? Discuss the problem of spurious tuples and how we may prevent it.

- Memory space will be wasted at the storage level
- When aggregate operations such as SUM, AVG etc. are performed on the attribute which has null value, the result will be incorrect
- When JOIN operation involves an attribute with null values, the result may be unpredictable
- The NULL value has different meaning. It may be unknown, not applicable or absent

Spurious tuples are generated as the result of bad design or improper decomposition of the base table.

- Spurious tuples are the tuples generated when a JOIN operation is performed on badly designed relations. The resultant will have more tuples than the original set of tuples
- The main problem with spurious tuples is that they are considered invalid as they do not appear in the base tables.

Spurious tuples can be avoided by taking care while designing relation schemas:

- The relations should be designed in such a way that when a JOIN operation is performed, the attributes involved in the JOIN operation must be a primary key in one table and foreign key in another table
- While decomposing a base table into two tables, the tables must have a common attribute. The common attribute must be primary key in one table and foreign key in another table

14.24 Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B}→{C}, {A}→{D, E}, {B}→{F}, {F}→{G, H}, {D}→{I, J}}. What is the key for R? Decompose R into 2NF and then 3NF relations.

A minimal set of attributes whose closure includes all the attributes in R is a key. Since the closure of {A, B}, {A, B}+ = R, one key of R is {A, B} (in this case, it is the only key).

To normalize R intuitively into 2NF then 3NF, we take the following steps

First, identify partial dependencies that violate 2NF. These are attributes that are functionally dependent on either parts of the key, {A} or {B}, alone. We can calculate the closures {A}+ and {B}+ to determine partially dependent attributes:
{A}+ = {A, D, E, I, J}. Hence {A} -> {D, E, I, J} ({A} -> {A} is a trivial dependency)
{B}+ = {B, F, G, H}, hence {A} -> {F, G, H} ({B} -> {B} is a trivial dependency)

To normalize into 2NF, we remove the attributes that are functionally dependent on part of the key (A or B) from R and place them in separate relations R1 and R2, along with the part of the key they depend

on (A or B), which are copied into each of these relations but also remains in the original relation, which we call R3 below:
R1 = {A, D, E, I, J}, R2 = {B, F, G, H}, R3 = {A, B, C}

The new keys for R1, R2, and R3 are underlined. Next, we look for transitive dependencies in R1, R2, R3. The relation R1 has the transitive dependency {A} -> {D} -> {I, J}, so we remove the transitively dependent attributes {I, J} from R1 into a relation R11 and copy the attribute D they are dependent on into R11. The remaining attributes are kept in a relation R12. Hence, R1 is decomposed into R11 and R12 as follows: R11 = {D, I, J}, R12 = {A, D, E} the relation R2 is similarly decomposed into R21 and R22 based on the transitive dependency {B} -> {F} -> {G, H}:
R2 = {F, G, H}, R2 = {B, F}
The final set of relations in 3NF are {R11, R12, R21, R22, and R3}

14.25 Repeat Exercise 14.24 for the following different set of functional dependencies G = {{A, B}→{C}, {B, D}→{E, F}, {A, D}→{G, H}, {A}→{I}, {H}→{J}}.
At first we may try to find out the closures of all single attributes
{A}+ -> {A, I}, {B}+ -> {B}, {C}+ -> {C}, {D}+ -> {D}, {E}+ -> {E}, {F}+ -> {F}, {G}+ -> {G}, {H}+ -> {H, J}, {I}+ -> {I}, {J}+ -> {J}
Here none of the single attributes is the key. So next, we calculate the closures of pairs of attributes that are possible keys:
{A,B}+ -> {A, B, C, I}, {B, D}+ -> {B, D, E, F}, {A, D}+ -> {A, D, G, H, I, J}
None of the pairs are keys since none of the closures includes all attributes. But the union of the three closures includes all the attributes:
{A, B, D}+ -> {A, B, C, D, E, F, G, H, I, J}
So, here {A, B, D} is the key
Now we decompose the relation R as R = {A, B, C, D, E, F, G, H, I, J}
In the first level partial dependencies on the key which is in violation of 2NF that are:
{A, B} -> {C, I}, {B, D} -> {E, F}, {A, D} + -> {G, H, I, J}
So R is decomposed into R1, R2, R3, and R4:
R1 = {A, B, C, I}
R2 = {B, D, E, F}
R3 = {A, D, G, H, I, J}
R4 = {A, B, D}
Additional Partial dependencies exist in R1 and R3 because {A} -> {I}. So, we remove {I} into R5, so the following relations are the results of 2NF decomposition:
R1 = {A, B, C}
R2 = {B, D, E, F}
R3 = {A, D, G, H, J}
R4 = {A, B, D}
R5 = {A, I}

Now, we check for transitive dependencies in each of the relations which violate 3NF.
Only R3 has transitive dependency {A, D} -> {H} -> {J}, so it is decomposed into R31 and R32 as follows:
R31 = {H, J}

R32 = {A, D, G, H}
The final set of 3NF relations is {R1, R2, R31, R32, R4, R5}

14.26 Consider the following relation:

A	B	C	TUPLE#
10	b1	c1	1
10	b2	c2	2
11	b4	c1	3
12	b3	c4	4
13	b1	c1	5
14	b3	c4	6

a. Given the previous extension (state), which of the following dependencies may hold in the above relation? If the dependency cannot hold, explain why by specifying the tuples that cause the violation.
i. A → B, ii. B → C, iii. C → B, iv. B → A, v. C → A

A -> B does not hold good in current state of relation as attribute B has two values corresponding to value 10 of attribute A
B -> C this dependency can hold well in current state of relation
C -> B does not hold good in current state of relation as attribute B has two values corresponding to value c1 of attribute C
B -> A does not hold well in current state of relation as attribute A has two values corresponding to value b1 and b3 of attribute B
C -> A does not hold well in current state of relation as attribute A has two values corresponding to value c1, c4 of attribute C

b. Does the above relation have a potential candidate key? If it does, what is it? If it does not, why not?
If value of the attribute – TUPLE# remains different for all tuples in relation it can act as candidate key

14.30 Consider the following relation:
CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)
Assume that a car may be sold by multiple salespeople, and hence {Car#,Salesperson#} is the primary key. Additional dependencies are
Date_sold → Discount_amt and
Salesperson# → Commission%
Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?

According to the first normal form, the relation or table should contain only atomic values i.e. the relation should not contain any repeating groups. In the above relation as there will not be any repeating groups, the relation is in first normal form.

According to the second normal form, the relation must be in first normal form and each non key attribute must depend on primary key. There should not be any partial dependency. In the relation, Commission_percent is functional dependent on Salesperson_id, which is only part of the primary key. Hence the relation is not in second normal form.

According to third normal form, the relation must be in second normal form and any non-key attribute should not describe any non-key attribute. As the relation is not in the second normal form and also there is a non-key attribute date_sold that describe another non-key attribute Discount_amt, the relation is not in third normal form

So, the relation satisfies only first normal form

As the relation is not in second normal form, so decompose the relation CAR_SALE into two relations as shown below:
CAR_SALES (Car_id, Date_sold, Salesperson_id, Discount_amt)
SALES_COMM (Salesperson_id, Commission_percent)

Now the tables CAR_SALES and SALES_COMM are in second normal form

According to the third normal form, the relation must be in second normal form and any non-key attribute should not describe any other non-key attribute
SALES_COMM relation is in third normal form but CAR_SALES is not in third normal form as there is a non-key attribute Date_sold that describes another non-key attribute Discount_amt.
So, decompose the relation into two relations as shown below:
CAR_SALE_DATE (car_id, salesperson_id, Date_sold)
DATE_DIS(Date_sold, discount_amt)

The relations CAR_SALE_DATE, SALES_COMM, DATE_DIS are in third normal form.