

Effects of Generation of Spurious Tuples during Joining of Relations in Case of BCNF based Relations

Dr.C.Bhanuprakash¹, Jeevan P.R² and TaraK.N³

¹Assistant Professor, Department of Master of Computer Applications, Siddaganga Institute of Technology, Tumkur, Karnataka, India

Email: bhanuprakashc@hotmail.com

²⁻³MCA Students, Department of Master of Computer Applications, Siddaganga Institute of Technology, Tumkur, Karnataka, India

Email: jeevan.1si19mca20@gmail.com, taranarasimhababu@gmail.com

Abstract—The intention behind this paper is to familiarize the database faculty fraternity regarding the actual meaning of spurious tuples, which are the occasion's with which these kind of tuples may be generated and various evaluation techniques along with the proper and right kind of meaningful tuples / rows / records. Because, many of the database experts may not know the actual meaning of spurious tuples and which are the instances with which these tuples may be generated. Apart from these issues, this paper is also highlighted on the problems to be faced with generation of these kinds of tuples, methods to solve it, methods to avoid it, various techniques to be used to find the presence of spurious tuples. Suggestions were also given by making minimum changes in the relation's /table's design to avoid spurious tuples.

Index Terms— Spurious tuples, Normalization, Table, Relation, Row, Record, Attribute, Field, Multi-joined Dependency, Normal Forms, Functional Dependency, Full/Partial FD, Transitive Dependency.

I. INTRODUCTION

Spurious tuples are nothing but additional / extra rows generated as a result of joining two or more relations/tables. These are the rows generated during joining of tables. This paper enlightens on this issue regarding its generation, causes of generation, effects and implications caused on the database operations. These types of the rows create problems during accessing the records from the database relations. i.e. no records will be accessed from the database relations. This is due to the problem of unmatched joining conditions. i.e. no record of one table will be matched with any record of the other table. Even though there may have multiple matched records across many tables, no results will come from any of the records. If relation R and its decomposed relations $R_1, R_2, R_3, \dots, R_n$, then, condition for not getting any Spurious Tuple is denoted by,

$$R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n = R$$

Whereas condition for getting Spurious Tuples is denoted by,

$$R \subset R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n$$

The generation of spurious tuples can be avoided by slightly changing the rule of BCNF during the designing of tables. BCNF is a stronger form of 3NF. 3NF states that every non-prime attribute must be non-transitively

dependent on prime attribute. BCNF states that every attribute (prime or non-prime) must be non-transitively dependent on every key. In other words, every determinant is a candidate key. If any type of the child table false under the category of BCNF by combining all foreign key attributes in to a prime attribute of candidate key type, definitely results in generation of spurious tuples during joining operation.

This can be avoided by doing following suggestions. In case of child tables, it needs to leave the foreign key attributes as it is. It should not be modified into combined candidate keys along with other foreign keys. Because, when it combines with multiple foreign keys into single combined primary key, it results in to a single value. i.e. multiple values are joined together to become a single value which results in a problem of un matched records during joining of tables. This ultimately ends up with no rows / zero rows selected. This leads to lot of conflicts about the data. Even though, there are multiple matched data records with matching criteria, it cannot be able to access these records, because of the wrong way of designing the tables. The intention of this paper is to avoid these drawbacks and highlighted many database experts about the pros and cons of spurious tuples.

II. RELATED WORK

There have been many approaches made by many researchers on generation of Spurious tuples as well as consequences of database normalization process.

“Teaching Tip – Clarifying Normalization” [1]. Mr Donald.A.Carpenter made an attempt to clarify on many doubts and ambiguities on normalization process. He has pointed out the discrepancies among textbooks regarding two important concepts of normalization. First, the goal of normalization should be standardized back to Codd’s original definition. Many potential outcomes and benefits of normalization should not be confused with the overriding goal, which is to create properly structured relations, as those contain fewer future anomalies. Focusing on benefits instead of the goal can mislead one to bypass normalization altogether since there are other ways to at least partially achieve some of those same benefits that are misstated as being goals.

“The Deluge of Spurious Correlations in Big Data”[2], Mr. Cristian Calude and Mr. Giuseppe Longo suggested that correlation supersedes causation and theorising is mathematically wrong. Scientists and philosophers have always worried about fallacies one commits by looking at correlations only. Their work confirms the intuition that the bigger the database which one mines for correlations, the higher is the chance to end recurrent regularities and the higher is the risk of committing such fallacies. In addition to looming threat of spurious correlations, they also presented many examples on successful use of data analytics in robust scientific studies. They also pointed out that a spurious correlation in the above sense is also spurious according to any possible definition, because, by construction, its values are chosen at random intervals with respect to all the data in the database. As a consequence, such a correlation cannot provide reliable information on future developments of any type of such behaviour.

“Spurious Factor Analysis” by Alexei Onatski and Chen Wang [3], draws parallels between the principal components analysis of factor less high-dimensional non stationary data and the classical spurious regression. They have shown that a few of the principal components of such data absorb nearly all the data variation. The corresponding screen plot suggests that the data contains a few factors, which is corroborated by the standard panel information criteria. Furthermore, the Dickey–Fuller tests of the unit root hypothesis applied to the estimated “idiosyncratic terms” often reject, creating an impression that a few factors are responsible for most of the non stationary in the data. They also warn empirical researchers of these peculiar effects and suggest to always comparing the analysis in levels with that in differences.

III. METHODOLOGY

A. Spurious Tuples

Spurious Tuples are nothing but rows/records in a relation/table, which generate as a result of joining two or more relations/tables in a wrong way. These are extra rows which creates a problem during matching of records. That is matching a record of one table with a corresponding record of another table.

If relation R and its decomposed relations $R_1, R_2, R_3, \dots, R_n$, then, condition for not getting any Spurious Tuple is denoted by,

$$R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n = R$$

Whereas condition for getting Spurious Tuples is denoted by,

$$R \subset R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n$$

B. 3NF (Third Normal Form)

3NF is based on the concept of transitive dependency. i.e. in a relation R, there should not be any transitive dependency among prime attribute and non-prime attributes. If it is there, we need to decompose the relation to satisfy the rules of 3NF.

Definition : A relation schema R is in 3NF if it satisfies 2NF and no non-prime attribute is transitively dependent on the primary key of R.

3NF means that each non-prime attribute value in any tuple is truly dependent on the primary key and not even partially on other attributes. 3NF prohibits transitive dependencies.

Example : Consider the relation with a name “Report”

Report { StudentNo, CourseNo, StudentName, CourseName, DOB, Duration, Marks, Grade}

Here, StudentNo, CourseNo is combined prime attribute, StudentName, CourseName, DOB, Duration, Marks and Grade are non-prime attributes.

Because of many violations with respect to full functional dependency, the above relation is decomposed in to 3 individual relations which are as follows.

Student { StudentNo, StudentName, DOB}

In the relation Student, StudentNo is the prime attribute, StudentName and DOB are the non-prime attributes which are fully functionally dependent on prime attribute. It satisfies the 2NF.

Course {CourseNo, CourseName, Duration}

In the relation Course, CourseNo is the prime attribute, CourseName and Duration are the non-prime attributes which are fully functionally dependent on prime attribute. It satisfies the 2NF.

Report {StudentNo,CourseNo, Marks, Grade}

In the relation Report, the non-prime attribute Marks is fully functionally dependent on prime-attribute StudentNo,CourseNo and one more non-prime attribute Grade is fully functionally dependent on Marks but not either on StudentNo nor on CourseNo which in turn results in transitive dependency which is against to the rule of 3NF. Therefore, the relation ‘Report’ is not satisfying the norms of 3NF. In order to bring the relation ‘Report’ to 3NF, we need to decompose the relation to further relation which is as follows:

Report { StudentNo,CourseNo,Marks},

StudentGrade { Marks, Grade},

Here, in the relation ‘Report’, the non-prime attribute ‘Marks’ is fully functionally dependent on the combined key StudentNo,CourseNo and in the relation ‘StudentGrade’, the non-prime attribute ‘Grade’ is fully functionally dependent on the attribute ‘Marks’. Because, amount of marks defines the value of the grade. The relation StudentGrade can also be modified as StudentGrade { InitialMarks, FinalMarks, Grade}, because, grade is specified always by considering the range of marks. Range lies between two range of values. Therefore, the attribute marks can be again divided into InitialMarks and FinalMarks.

C. BCNF (Boyce-Codd Normal Form)

This Normal form was proposed from two database experts Mr. Boyce and Mr. Codd in the year 1969. It was proposed as a simpler form of 3NF, but it was found to be stricter version than 3NF. That is every relation in BCNF is also in 3NF, however, a relation in 3NF is not necessarily in BCNF. In simple sense, relation with one column primary key can satisfies the rules of 3NF, but a relation with combined column primary key cannot satisfy the rules of 3NF. This time, we bring this relation to BCNF form.

Definition: A relation schema R is in BCNF if and only if, for every non-trivial functional dependency $A \rightarrow B$ in it, it is true that A is a super key of R. In other words, every determinant is a candidate key.

BCNF is a stronger form of 3NF

* 3NF states that every non-prime attribute must be non-transitively dependent on every key

* BCNF states that every attribute (prime or non-prime) must be non transitively dependent on every key.

Example :

StudentReport{StudentNo, CourseNo, SubjectNo, Marks}

Here, in the relation 'StudentReport', the non-prime attribute 'Marks' is fully functionally dependent on the combined key StudentNo, CourseNo, SubjectNo. The meaning is a student has scored some marks in a subject of particular course.

EmpProject{EmpNo, ProjectNo, NumberOfHours}

Here, in the relation 'EmpProject', the non-prime attribute 'NumberOfHours' is fully functionally dependent on the combined key EmpNo, ProjectNo. The meaning is an employee has worked some number of hours in a particular project.

DriverDrive { DriverNo, VehicleNo, NumberOfKms, DateOfDrive}

Here, in the relation 'DriverDrive', the non-prime attributes 'NumberOfKms' and 'DateOfDrive' are fully functionally dependent on the combined key DriverNo, VehicleNo. The meaning is a driver has driven a car with some number of kilometers on particular date.

Explanation of Generation of Spurious tuples: Consider the following relations Student, Course, Subject, Teacher, Student_Course_Subject and Teacher_Course_Subject.

TABLE I. STUDENT

StudentId	StudentName	DateofBirth	Gender	Address
STD1	Arunkumar	12-Mar-02	M	Tumkur
STD2	Ashok	21-Feb-02	M	Madhugiri
STD3	Akshatha	22-Jan-02	F	Tumkur
STD4	Arjun	13-Feb-02	M	Tumkur
STD5	Asha	24-Mar-02	F	Kunigal
STD6	Akaash	26-Feb-02	M	Madhugiri

In the relation Student, the attribute StudentId is the prime attribute, StudentName, DateofBirth, Gender, Address are the non-prime attributes and are fully functionally dependent on the prime attribute. Hence, it satisfies the 2NF. Since there is no transitivity dependency among any of the attributes, it also satisfies 3NF.

TABLE II. COURSE

CourseId	CourseName	CourseDuration	Intake
CO1	Computer Science	4 Years	60
CO2	Information Science	4 Years	60
CO3	Data Science	4 Years	60
CO4	Mechanical Engg	4 Years	90
CO5	Electronics Engg	4 Years	60
CO6	Civil Engg	4 Years	60
CO7	Architecture	5 Years	40

In the relation Course, the attribute CourseId is the prime attribute, CourseName, CourseDuration, Intake are the non prime attributes and are fully functionally dependent on the prime attribute. Hence, it satisfies the 2NF. Since there is no transitivity dependency among any of the attributes, it also satisfies 3NF.

TABLE III. SUBJECT

SubjectId	SubjectName	Class
S1	Physics	I Semester
S2	Chemistry	I Semester
S3	Drawing	II Semester
S4	Engg Maths	II Semester
S5	Engg Mechanics	III Semester

In the relation Subject, the attribute SubjectId is the prime attribute, SubjectName, Class are the non prime attributes and are fully functionally dependent on the prime attribute. Hence, it satisfies both 2NF and 3NF.

TABLE IV. STUDENT_COURSE_SUBJECT

StudentId	CourseId	SubjectId	Marks
STD1	CO1	S1	67
STD2	CO1	S1	78
STD3	CO2	S1	65
STD4	CO3	S2	67
STD5	CO1	S2	56
STD6	CO3	S2	78

In the relation Student_Course_Subject, the attribute StudentId is the foreign key attribute references from the relation Student, the attribute CourseId is the foreign key attribute references from the relation Course, the attribute SubjectId is the foreign key attribute references from the relation Subject, the attribute marks is the only non prime attribute and is fully functionally dependent on all the three foreign key attributes. That is, a student has scored amount of marks in some subject. If it is applied on the basis of BCNF on the relation Student_Course_Subject by combining foreign keys as a candidate key, it is created a candidate key by combining StudentId, CourseId and SubjectId. Later, the combination of values 'STD1', 'CO1' and 'S1' will become single value in the form of 'STD1CO1S1'. Similarly, remaining records will become 'STD2CO1S1', 'STD3CO2S1', 'STD4CO3S2', 'STD5CO1S2' and 'STD6CO3S2'. These single values will create a problem during joining of the tables.

For example, Let us display the values of the attributes StudentId, StudentName, CourseName and Marks from the relations Student, Course and Student_Course_Subject, it needs to join the relations by including number of joining conditions by using n-1 criteria. SQL query to fulfil the above requirement is as follows.

```
SQL> Select StudentId, StudentName, CourseName, Marks
From Student, Course, Student_Course_Subject
Where Student_Course_Subject.StudentId = Student.StudentId
And Student_Course_Subject.CourseId = Course.CourseId;
```

To select the matching records from the relations Student, Course and Student_Course_Subject, the evaluation of the values for the joining conditions are as follows.
For the first joining condition,

Student_Course_Subject.StudentId = Student.StudentId

the values will be matched according to the joining condition with corresponding attributes are as follows
STD1CO1S1 = STD1

For the second joining condition,
Student_Course_Subject.CourseId = Course.CourseId

the values will be matched according to the joining condition with corresponding attributes are as follows

$$\text{STD1CO1S1} = \text{CO1}$$

The above criterion is only meant for the first record. Similarly for the remaining records the evaluation of the values for the joining conditions are as follows.

For the second record :
STD2CO1S1 = STD2
STD2CO1S1 = CO1

For the third record :
STD3CO2S1 = STD3
STD3CO2S1 = CO2

For the fourth record :
STD4CO3S2 = STD4
STD4CO3S2 = CO3

For the fifth record :
STD5CO1S2 = STD5
STD5CO1S2 = CO1

For the Sixth record :
STD6CO3S2 = STD6
STD6CO3S2 = CO3

If it looks in to any of the above joining conditions, it will not get any of the matching records. Ultimately it results in no rows selected criteria, i.e. zero rows selected. This is due to the implementation of BCNF in one of the relations Student_Course_Subject. In this relation, the foreign key attributes StudentId, CourseId and SubjectId have been combined and made them as candidate key. Right now there are only six rows in the child relation Student_Course_Subject which results in zero rows selected. This situation will continue even if it contains more than 100 rows. This problem is also called as joined dependency which resulted in the generation of spurious tuples. This leads to serious situation in real time environment which badly needs a solution to avoid this problem.

Join Dependencies : A join dependency denoted by $JD(R_1, R_2, R_3, \dots, R_n)$ specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have non additive join decomposition into $R_1, R_2, \dots, R_n$.

i.e. for every such r, $*(\pi_{R_1}(r), \pi_{R_2}(r), \pi_{R_3}(r), \dots, \pi_{R_n}(r)) = r$

d. FIFTH NORMAL FORM (5NF) : A relation schema R is in 5NF (project join normal form, PJNF) with respect to a set F of functional, multivalued and join dependencies if, for every non-trivial join dependency $JD(R_1, R_2, \dots, R_n)$ in F^+ , every R_i is a super key of R.

In other words, A table is said to be in the 5NF if and only if every non-trivial join dependency in it is implied by the candidate keys.

e. A join dependency : $\{A, B, \dots, Z\}$ on R is implied by the candidate key(s) of R if and only if each of A, B, ..., Z is a super key for R.

To solve this typical problem, it needs to modify the rules of one of the above relations Student_Course_Subject which is against to the criteria's of BCNF. In the relation, the foreign key attributes StudentId, CourseId and

SubjectId have been combined and made them as candidate key. It should not be combined like this, instead of this, it would be better to leave all the foreign keys as it is, i.e. StudentId, CourseId and SubjectId become foreign keys. Then the tuples/rows will become separate values like 'STD1', 'CO1' and 'S1'. Similarly second record will be like 'STD2', 'CO1', 'S1', third record will be 'STD3', 'CO2', 'S1', fourth record will be like 'STD4', 'CO3', 'S2', fifth record will be like 'STD5', 'CO1', 'S2' and sixth record will be like 'STD6', 'CO3', 'S2'. During joining of these records with the previous SQL query,

```
SQL> Select StudentId, StudentName, CourseName, Marks
      From Student, Course, Student_Course_Subject
      Where Student_Course_Subject.StudentId = Student.StudentId
      And Student_Course_Subject.CourseId= Course.CourseId;
```

For the first joining condition,

Student_Course_Subject.StudentId = Student.StudentId

the values will be matched according to the joining condition with corresponding attributes are as follows
 'STD1' = 'STD1'

For the second joining condition,

Student_Course_Subject.CourseId = Course.CourseId

the values will be matched according to the joining condition with corresponding attributes are as follows
 'CO1' = 'CO1'

For the first joining condition,

'STD1' = 'STD1',

since one of the values 'STD1' is equal to 'STD1' and for the second joining condition,

'CO1' = 'CO1'

the value 'CO1' is equal to 'CO1', with these criteria's, it will get first matching record.

For the second record : 'STD2' = 'STD2',
 'CO1' = 'CO1'

with these criteria's, it will get second matching record.

For the third record : 'STD3' = 'STD3'
 'CO2' = 'CO2'

with these criteria's, it will get third matching record.

For the fourth record : 'STD4' = 'STD4'
 'CO3' = 'CO3'

with these criteria's, it will get fourth matching record.

For the fifth record : 'STD5' = 'STD5'
 'CO1' = 'CO1'

with these criteria's, it will get fifth matching record.

Similarly, for the sixth record : 'STD6' = 'STD6'
 'CO3' = 'CO3'

with these criteria's, it will get sixth matching record.

This is the result it gives, when it needs to join only three relations, i.e. Student, Course and Student_Course_Subject. Similarly when it needs to display subject related information in addition to student and course, then it is necessary to join four relations. i.e. Student, Course, Subject and Student_Course_Subject with the following SQL query.

```
SQL> Select StudentId, StudentName, CourseName, SubjectName, Class, Marks
      From Student, Course, Subject, Student_Course_Subject
Where Student_Course_Subject.StudentId = Student.StudentId
And Student_Course_Subject.CourseId= Course.CourseId
And Student_Course_Subject.SubjectId= Subject.SubjectId;
```

In this case, we have three joining conditions. For the first joining condition,

Student_Course_Subject.StudentId = Student.StudentId
the values will be matched according to the joining condition with corresponding attributes are as follows
'STD1' = 'STD1',

In this case, the relation Student_Course_Subject is joined with student relation with a common attribute StudentId.

For the second joining condition,

Student_Course_Subject.CourseId= Course.CourseId
the values will be matched according to the joining condition with corresponding attributes are as follows
'CO1' = 'CO1'

In this case, the relation Student_Course_Subject is joined with Course relation with a common attribute CourseId.

For the third joining condition,

Student_Course_Subject.SubjectId= Subject.SubjectId
the values will be matched according to the joining condition with corresponding attributes are as follows
'S1' = 'S1'

In this case, the relation Student_Course_Subject is joined with Subject relation with a common attribute SubjectId. With these criteria's, it will give first matching record.

The same strategy will continue for the remaining records.

For the second record : 'STD2' = 'STD2'
'CO1' = 'CO1'
'S1' = 'S1'

with these criteria's, it will give second matching record.

For the third record : 'STD3' = 'STD3'
'CO2' = 'CO2'
'S1' = 'S1'

with these criteria's, it will give third matching record.

For the fourth record : 'STD4' = 'STD4'
'CO3' = 'CO3'
'S2' = 'S2'

with these criteria's, it will give fourth matching record.

For the fifth record : 'STD5' = 'STD5'
'CO1' = 'CO1'
'S2' = 'S2'

with these criteria's, it will give fifth matching record.

Similarly, for the sixth record :

'STD6' = 'STD6'
'CO3' = 'CO3'
'S2' = 'S2'

with these criteria's, it will give sixth matching record.

This is how problems will occur when it is designed the database relations according to the rules specified by BCNF especially in the case of child tables. Because, of this practical problem, it is advisable to leave the foreign keys as it is. Even though, there is a provision to make it, it is giving problem during joining operation of multiple tables/relations. That is the main theme of this paper.

IV. CONCLUSIONS

This is the attempt made to resolve the conflicts and ambiguities of spurious tuples among the people of database fraternity and clarified the actual meaning of spurious tuples. There was also an effort has been put in easily convincing way to the experts on how to deal with these type of tuples in the real time scenario. There were very few articles published on these type of tuples. This has raised a sort of curiosity in publishing an article which includes the generation of such tuples, their effects and implications on day to day database transactional activities and various ways of avoiding such tuples without making major changes in the database design. On many occasions, experts felt that it would be very difficult to convince the audience in solving the doubts on spurious tuples. Keeping all these things in mind, a sincere efforts have been put to make the familiarity of what is spurious tuples?, what are the causes of its generation?, how to solve it ?. All these things have been presented by giving simple tables / relations along with easily understandable records. Lot of care has been taken to avoid the mistakes while it was presenting with many SQL queries with joining conditions. It would be highly appreciable with your valuable feedback, healthy comments and suggestions.

REFERENCES

- [1] Donald.A. Carpenter, “Teaching Tip Clarifying Normalization”,- Journal of Information Systems Education, Vol. 19(4), ISSN 1055-3096.
- [2] Cristian Calude, Giuseppe Longo “The Deluge of Spurious Correlations in Big Data” – October – 2015, Nantes, France, pp.1 – 18, HAL - 01380626
- [3] Spurious Factor Analysis - Alexei Onatski, Chen Wang, Econometrica 89(2) : 591 – 614, DOI – 10.3982/ECTA16703
- [4] LDBN - Learn DataBase Normalization. <http://ldbnonline.com/> (Last visited August - 2008).
- [5] “The theory of joins in relational databases” - C. Beeri A. V. Aho and J. D. Ullman, ACM Trans. Database Syst., 4(3):297{314, 1979.
- [6] “Achievements of Relational Database Schema Design Theory Revisited, Semantics in Databases”- J. Biskup, (1358):29{54, 1998. uni-dortmund.de/uploads/tx_ls6ext/Biskup_1998.ps.gz
- [7] “A Simple Guide to Five Normal Forms in Relational Database Theory” – Pages 66{71, 1989. <http://www.bkent.net/Doc/simple5.htm>, (Last visited August 2008).

AUTHORS PROFILES



Dr.C.Bhanuprakash,

received the B.E.in Mechanical Engineering and Master of Computer Applications degrees from Siddaganga Institute of Technology, Tumkur, of Bangalore University, in 1993 and 1998, respectively. Received Ph.D degree from Visveswaraya Technological University, Belagavi in the year 2021.

C.Bhanuprakash has 20 years of Teaching experience, 4 years of Industry experience and 8 years of Research experience. Presently he is working as Assistant Professor in the Department of Master of Computer Applications, Siddaganga Institute of Technology, Tumkur, Karnataka. His areas of specialization include Design and Development of Database applications, Operations Research, Optimization Techniques, Intelligent Data Analysis, Advanced Operating systems, Neural Networks, Data Mining and Soft Computing Techniques.

Mr. Jeevan P.R and Tara K.N

Presently studying in fifth semester MCA degree at Department of Master of Computer Applications, Siddaganga Institute of Technology, Tumkur – 572103. Their areas of interests include Data Mining, Data Analytics, Web based Project Development and Database applications.