A spurious tuple is a record in database that get created when ***two tables are joined badly***.

Spurious tuples are created when two tables are joined on attributes that are ***neither primary keys nor foreign*** keys.

This could lead to an integrity issue since primary keys are guaranteed to be unique. If you use a field which is not a primary key you cannot be sure that the outcome will produce the number of records you expect.

One of the main objectives for a ***decomposition*** is to ***estimate seasonal effects*** that can be used to create and present ***seasonally adjusted values***.  A seasonally adjusted value ***removes*** the seasonal effect from a value so that ***trends can be seen more clearly***.  For instance, in many regions of the U.S. unemployment tends to decrease in the summer due to increased employment in agricultural areas.  Thus, a drop in the unemployment rate in June compared to May doesn’t necessarily indicate that there’s a trend toward lower unemployment in the country.  To see whether there is a real trend, we should adjust for the fact that unemployment is always lower in June than in May.

**Basic Structures**

The following two structures are considered for basic decomposition models:

1. Additive:  *xt* = Trend + Seasonal + Random

2. Multiplicative:  *xt* = Trend \* Seasonal \* Random

The “Random” term is often called “Irregular” in software for decompositions.

**How to Choose Between Additive and Multiplicative Decompositions**

* The additive model is useful when the seasonal variation is relatively ***constant*** over time.
* The multiplicative model is useful when the seasonal variation increases over time.

Functional Dependent: -

If X -> Y then for every unique value of X there should be a Y which will not change. That is if X is repeated in any time then value of Y should remain same.

***For instance, if X -> Y & X1 -> Y1, then every time X1 is appeared in a table its value should be Y1.***

***If there exist a tuple for which X1 -/-> Y1 then then X -/-> Y,***

***You can have X1 -> Y1 and X2 -> Y1.***

Fully Functional Dependent: -

If X -> Y and no subset or proper subset of X alone can determine Y, then Y is said to be fully FD on X.

***For instance, if AB -> C & A -/-> C; B -/-> C, then we say C is Fully FD on AB***

Partially Functional Dependent: -

If X -> Y and there exists a subset of X on which Y in dependent, then we say Y is partially FD on X.

***For instance, if ABC -> D & A --> D, then we say D is Partially FD on ABC***

A+ = {A, D}; also, called as ***A closure***. (All attributes depending on A)

Transitive Functional Dependent: -

If X -> Y and Y -> Z then X -> Z

Trivial Functional Dependent: -

If Y is subset of X, then we can say Y is FD on X.

***For instance, if AB ABC then ABC -> AB***

***thus, if Y X then X -> Y***

***Also, A -> A is trivial FD***

***In Relational Database we cannot remove redundancy, we can reduce it to some extent.***

**Normalization** is the process of decomposition of bigger relation into smaller relation to reduce the redundancy and improve the efficiency.

***Normalization is theory and process to evaluate and improve relational database design.***

In 1-NF -> relation in which there is no multivalued, composite attribute that is every tuple has an atomic value for every attribute

In 2-NF -> all non-prime attribute is fully FD on each (candidate) key attribute

any dependency between non-prime attributes is allowed.

In 3-NF -> No non-prime attribute should be transitively dependent on (candidate) key attribute

No non-prime attribute should be determined by another non-prime attribute.

**Structural Constraints: -**

If an entity does not exist unless it appears with an entity in a relationship, the participation is ***total*** (existence dependency). Else, ***partial***.

**Specialization/Generalization: -**

Only ***subset*** of entities within a type have ***certain attributes*** or participate in ***certain relationships***

**Approaches to Conceptual Design: -**

***Centralized*** – single authority responsible, reasonable for smaller application

***View Integration*** – Each implements local, integrate into global schema

If R -> N and N -> O; N -> P then key is only **R**

***Objectives of Normalization***

• Make the schema informative  • Minimize information duplication

• Avoid modification anomalies • Disallow spurious tuples

***Modification Anomalies***

An undesired side-effect resulting from an attempt to modify a table [that has not been sufficiently normalized]

***candidate keys (minimal subsets of attributes that FD all attributes)***

***derived attribute can never be in the candidate keys***