Fundamentals of Database Systems

[Normalization – I]

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
Indian Statistical Institute, Kolkata

September, 2019

Redundancy in databases

Redundancy in a database denotes the repetition of stored data

Redundancy might cause various anomalies and problems pertaining to storage requirements:

- Insertion anomalies: It may be impossible to store certain information without storing some other, unrelated information.
- Deletion anomalies: It may be impossible to delete certain information without losing some other, unrelated information.
- Update anomalies: If one copy of such repeated data is updated, all copies need to be updated to prevent inconsistency.
- Increasing storage requirements: The storage requirements may increase over time.

Redundancy in databases

Redundancy in a database denotes the repetition of stored data

Redundancy might cause various anomalies and problems pertaining to storage requirements:

- Insertion anomalies: It may be impossible to store certain information without storing some other, unrelated information.
- Deletion anomalies: It may be impossible to delete certain information without losing some other, unrelated information.
- Update anomalies: If one copy of such repeated data is updated, all copies need to be updated to prevent inconsistency.
- Increasing storage requirements: The storage requirements may increase over time.

These issues can be addressed by decomposing the database – normalization forces this!!!

Insertion anomaly – An example

Consider the following table (the attributes are `not null`) detailing some of the cars available in the Kolkata market.

Company	Country	Make	Distributor
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

Suppose RAV4 is a new make of Tesla, a company from US, which is soon to arrive in Kolkata market with no distributor announced yet.

This insertion is not possible in the above table as the `Distributor` cannot be null.

Deletion anomaly – An example

Consider the following table (the attributes are `not null`) detailing some of the cars available in the Kolkata market.

Company	**Country**	**Make**	**Distributor**
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

Suppose CarTrade is no more a distributor for the make X1 of BMW, a company from Germany.

This deletion from the above table would result in the car record being deleted.

Update anomaly – An example

Consider the following table (the attributes are `not null`) detailing some of the cars available in the Kolkata market.

Company	**Country**	**Make**	**Distributor**
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

Suppose Maruti is no more an Indian company due to its 100% procurement by Suzuki Motor Corporation, a company from Japan.

This update is to be made in multiple records in the above table resulting into atomicity challenges.

An overview of different normal forms in the literature

Normal Form	Details	Reference
1NF (Codd (1970), Date (2006))	Domains should be atomic/At least one candidate key	[1, 5]
2NF (Codd (1971))	No non-prime attribute is functionally dependent on a proper subset of any candidate key	[2]
3NF (Codd (1971), Zaniolo (1982))	Every non-prime attribute is non-transitively dependent on every candidate key	[2, 4]
BCNF (Codd (1974))	Every non-trivial functional dependency is a dependency on a superkey	[3]

Motivations behind normalization

Normal Form	Basic Motivation
1NF	Removing non-atomicity
2NF	Removing partial dependency (Part of key attribute $\rightarrow$ Non-key attribute)
3NF	Removing transitive dependency (Non-key attribute $\rightarrow$ Non-key attribute)
BCNF	Removing any kind of redundancy

Problems with normalization

Denormalization

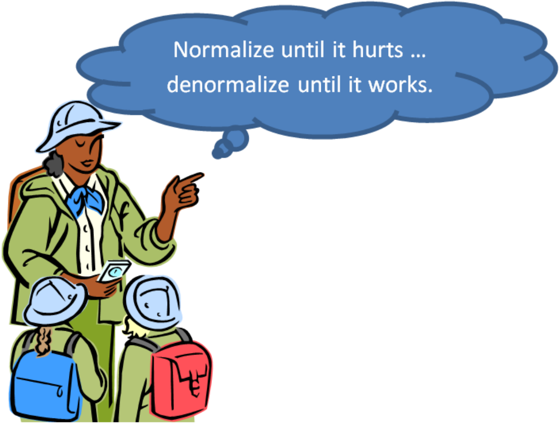
Denormalization is the process of converting a normalized schema to a non-normalized one

Denormalization

Denormalization is the process of converting a normalized schema to a non-normalized one

Note: Designers use denormalization to tune performance of systems to support time-critical operations. They assess the cost, benefit, and risk to identify the right normalization level with respect to the data, its use and its quality requirements.

Normalization versus denormalization



Applications

Normalization:

1. Use of normalization to minimize the impact of various anomalies created with database modification.
2. Use of normalization to reduce the data integrity problems.

Denormalization:

1. Use of denormalization in case the data is not going to be updated after being created.
2. Use of denormalization results into the performance gain.

<u>Note</u>: There is no “ideal” normal form for a table or the data as a whole.

References

- E. F. Codd (1970) *CACM*, 13(6):377-387.
- E. F. Codd (1971) *IBM Research Report*, RJ909.
- E. F. Codd (1974) *IBM Research Report*, RJ1385.
- C. Zaniolo (1982) *ACM TDS*, 7(3), 489-499.
- C. J. Date (2006) *Date on Database: Writings 2000-2006*, Springer-Verlag.