CIDR 2011 Paper
Database Access Control & Privacy: Is There A Common Ground?
Chaudhuri et al

Summary

The authors proposed an idea to incorporate database access control and differential privacy. Particularly, the idea is to rewrite user’s query in a way that it takes advantage of the user authorization and produces differentially private result while preserving as much utility as it can.

This is the first paper incorporating data access mechanism with differential privacy utilizing standard SQL. Another similar paper in this area was Airavat system utilizing mandatory access on a MapReduce query engine.

Key ideas in the paper:
- Access control is maintained via authorization policies (whether user has access to a particular record/row or column)
- The goal is to maintain the epsilon-differential privacy rule for user u where two database instances have the same authorization policy
- The main idea is to add or modify predicate into queries protecting privacy
- Differentially private view (DPV): incorporates both authorized subset and unauthorized subset preserving differential privacy
- Some computation has large sensitivity (sum, join and etc), while some has low sensitivity (count): so they must be treated differently.
- For authorized subset that user has access to, the returned result is accurate. For unauthorized subset, noise is added into result for that subset. Then they are aggregated together and returned to the user.

Positive/Strong Points
- S1 The authors provided detailed proofs for the differential private mechanism in database access.
- S2 There are many examples incorporated for different parts of SQL statements (selects, where, group by) providing as much utility as we can.
- S3 The final example deals with join, which is the computationally the most difficult part of the query showing that the authors’ method can work in a generic case.

Negative/Weak Points
- W1 The authorization policy might not be easily written into a predicate to incorporate into a SQL query
- W2 The authors did not state how to add noise into the query result (probably Laplacian or Gaussian noise).
- W3 Since the framework described here uses noisy DPV, this would add cost to maintain the database (time, space).
- W4: (also addressed in Research question 3): The current only-supported type of join is key-key, which excludes joins of more foreign key tables is a key missing element, since in practice, join of cardinality more than 2 on foreign key tables is very common.
Research Questions and Points for Discussion
- D1 How can we relax the assumption that there is no post-update actions by the database? (That is to say if a user updates a record, only the statement issued by the user is rewritten using the framework, but consecutive automatic updates by the database system is not subject to privacy preserving mechanism described here in the paper.)
- D2 Can we also apply the same technique into a spatial database query engine? What is the spatial support we can add to protect user’s privacy?
- D3 How can we overcome the limitation in join? (Currently only key-key join is supported, since deleting/changing one record on the primary table has an unbounded effect on foreign table; so there is a large gap to do more researches before the framework can be incorporated into a fully functionally system.)