12-2010

Analysis of a database insider threat model

Andrea Samuel
University of Arkansas, Fayetteville

Analysis of a Database Insider Threat Model

Undergraduate Honors Thesis

Andrea Samuel

University of Arkansas

Department of Computer Science and Computer Engineering

Abstract

According to Silicon.com's *CIO Insight – Beware the Insider Security Threat,* insiders are bigger threats to corporate security than external threats such as denial of service attacks or malware. Statistics show that 70% of fraud is perpetrated by staff and that the main data security threat comes from poorly trained or disgruntled employees who are authorized to have access to data and file stores [4]. This research project focuses specifically on the problem of insider threat in relational database systems. The project involves simulating research conducted in Qussai Yaseen and Brajendra Panda's research paper, *Predicting and Preventing Insider Threat in Relational Database Systems*. The objective of this project is to develop the knowledgebase for an insider as they request access to attributes in transactions. The generated knowledge base for a given user or insider is then used to develop a Threat Prediction Graph that can be used to predict and prevent insider threat.

Generating the knowledge graph and threat prediction graph, which will issue warnings if insiders have the ability to infer values of data items to which they do not have authorized access, provides an effective solution to the insider threat problem in relational database systems. Conducting this test across different relational database schemas gives an idea of how long it takes to obtain unauthorized knowledge of data items for various types of relational databases and reveals which areas are most susceptible to insider threat.

Table of Contents

1. Introduction

The objective of this paper is analyze the process of providing a simulation of the solution proposed in Qussai Yaseen and Dr. Brajendra Panda's paper *Predicting and Preventing Insider Threat in Relational Database Systems.* The underlying premise that led to this study was that insiders caused 52% of breaches in 2004, more than the number of external threats posed to companies and organizations [5]. Security issues are becoming increasingly crucial, especially with regard to ensuring the protection of data from "interruption, modification, and fabrication" [1]. While extensive study has been done in preventing outsiders' attacks and increasingly more research has gone into the issue of insider attacks, there has been relatively little study in comparison to handle the issue of insider threat in relational databases. An insider has authorized access and privileges but can pose a threat by violating the security policy of the system through legitimate information access. This occurs through information that can be inferred from existing knowledge of other system units. Consequently, insider threat in relational databases is primarily influenced by the dependencies that exist in a given database.

The paper by Yaseen and Dr. Panda investigates the problem of knowledge acquisition by an unauthorized insider using dependencies between objects in relational databases. In proposing solutions to prevent insider threat and access to information, the paper introduces mechanisms such as the Constraint and Dependency Graph (CDG) and the Dependency Matrix that are used to represent dependencies and constraints between objects [2]. Based on these graphs, an insiders' knowledge graph can be constructed to show the knowledgebase of a user. The simulation that is the primary objective of this paper, takes the methods and process proposed in determining the dependencies and constraints to determine threats and prevent access to confidential information by unauthorized users.

The first step in predicting and preventing insider threat in relational database systems is to determine the dependencies that exist between data items. This is because insider threat in relational databases depends mainly on the dependencies that exist among tables. Dependencies as defined in this report are semantic relationships that exist among attributes. This goes beyond typical functional dependencies although it includes them. Determining dependencies within the

context of primary keys and foreign keys are the first step in creating the Dependency matrix. However, tracing the dependencies also requires a conceptual understanding of the schema that includes understanding business rules and regulations of a given organization. Dependencies are used as they tend to change infrequently. Few changes occur to the table structure, moreover, once the business rules have been established and the data model is created. As a result, mapping the dependencies and constraints among tables provides a reliable and consistent way to trace the threat conditions and sensitive information that exist for any given database.

For the purposes of this paper, the example database schema used is that of a generic Payroll System. The constraints and dependencies that exist in the data model are used to generate the Dependency Matrix, which in turn will be used to construct the knowledge graph of the insider.

2. Background & Motivation

Research was conducted on insider threat in relational database systems to prevent "insiders [who] may use their privileges or knowledge of various system units to infer about other system units to which they lack access" [1]. The primary goal of the research outlined in Yaseen and Panda's paper was to identify a strategy to predict and prevent insider threat in relational database systems by keeping track of a given users' overall knowledge acquisition. The strategies were developed in such a way that unauthorized access to information could be prevented without affecting the overall productivity of a given user.

Through the knowledge graphs, the amount of information that an insider can infer can be determined. This in turn can assist system administrators in determining an effective balance between the security and sensitivity of a transaction requested when they assign user permissions. Assigning permissions are critical in protecting the security of any system, including relational database systems. Being aware of the security issues that exist and having an idea of the threat prediction graph will allow administrators to assign permissions more efficiently. Consequently, users can maintain high productivity levels as they experience fewer rejected transactions.

The simulations that are the primary focus of this paper serve the purpose of identifying the critical areas for security breaches in a relational database system based on the dependencies and constraints that exist among tables for a given database. By running the simulation, administrators will have a better idea of how best to assign permissions that allow users access to all necessary information but prevent them from being able to infer unauthorized information. The multiple variables that can be manipulated in the simulation from the number of users and transactions to the number of data items being accessed allow administrators to test multiple scenarios of user access to the database. This information can be used to determine the best balance between enough access to data that is vital to productivity and too much access to sensitive information.

The Payroll System was used as the data model for the example database schema as it provides a system that contains several instances of sensitive information. Moreover, the Payroll database is likely to be found in a similar form at any organization or company and thus provides a model that can be easily understood and applied. The schema also offers multiple, fairly obvious dependencies and constraints among the tables which proved to be extremely useful in going through the steps of creating the Constraint and Dependency Graph (CDG) and the Dependency Matrix.

A second generic schema was produced through random generation. The generic schema showed that the simulation can be executed on any database schema, provided the dependencies and constraints that lead to the acquisition of sensitive information for the schema can be obtained. The generic schema differs from the Payroll schema in that it contains fewer overall attributes and consequently has fewer threat conditions. Having the second schema provided the opportunity to compare the two schemas against each other and determine how differences in constraints, dependencies, and threat conditions affect the overall knowledge acquisition and threat potential posed by a given user.

The simulations were run under the assumption that users did not have any special permissions or authorizations set. Dependencies between tables and threat conditions were determined at an attribute level to obtain the greatest level of detail in determining potential threats. The simulation takes a pro-active approach in that users are allowed to access whatever information they want to until the access has the potential for them to infer sensitive or confidential information to which they do not have access. At that point, any transaction requests by the user that have the potential to violate secure information will be rejected. The dependencies and constraints that exist for the Payroll and generic schema are detailed below along with the threat conditions that exist for the schemas, respectively.

3. Schema Analysis

3.1 Payroll Schema

The Payroll schema used was designed in an effort to be as generic as possible and therefore have similarities and applications similar to the Payroll data model of any typical organization or company. The threat conditions that were determined assumed that the users accessing the information did not have special permissions or qualifications such as being employees of the HR department. Therefore, information that could be inferred about base pay, salary, etc. was considered to be confidential. Any user requests that would allow either direct information or information to be inferred regarding these details were rejected.

The schema of the Payroll Data Model is shown in Appendix A. A brief description of the schema including the tables and their attributes are given below:

T_1 – Employee
T_2 – Position_Title
T_3 – Employee_Salary
T_4 – Pay_Period_Calendar
T_5 – Employee_Pay_Adjustment
T_6 – Adjustment_Type

In going through the process of determining the dependencies and constraints, the first step involved creating the dependency matrix. The dependency matrix shows dependencies between different tables as well as the constraints on such dependencies. For this project, dependencies were considered at the attribute level in addition to the table level. As stated in Yaseen and Panda's paper, "all types of dependencies are observed at the table level since a table inherits the dependencies present at its attribute levels, that is, a dependency between two tables is basically a dependency between attributes that belong to them. Therefore, two tables may have more than one type of dependency" [2].

Because the table level is at the highest level of granularity, it is the easiest to construct. This increased granularity is needed to express fully the relationships between attributes and also provides a more realistic representation of queries. Often users access only the attributes they are interested in seeing and do not view whole tables in a query. From this, the dependencies between attributes among tables can be more easily constructed.

There are several dependency relationships that exist among attributes in a relational database system. The two most common dependencies that will be discussed throughout this paper are strong and weak dependencies. The definitions of these dependencies are taken from Yaseen and Panda's paper [2]. Two data items A and B have a dependency relationship between them if one of them depends on the other or if they depend on each other. A dependency between A and B is represented by the notation A→B, which means that B depends on A. A dependency relationship is classified according to a number of categories, such as the strength, direction, and the transitivity. The strength of a dependency relationship is classified into two types: weak and strong, which are defined as follows.

Strong dependency: Given the dependency A→B, where A and B are two data items, if a change in A results in a change in B, then it is a strong dependency.
Weak dependency: The dependency A→B is called weak, if a change in A may not result in a change in B.

The values that are generated from establishing the dependencies and constraints among attributes in the schema are taken as inputs in the actual simulation.

The dependency matrix constructed for the payroll data model is shown below:

3.1.1 Dependency Matrix for Payroll Data Model

Table 1 Dependency Matrix for Payroll Schema

	T_1	T_2	T_3	T_4	T_5	T_6
T_1	-	0	(c_1, 2)	0	(c_2, 2)	0
T_2	(c_3, 2)	-	(c_4, 2)	0	(c_5, 1)	0
T_3	0	0	-	0	(c_6, 2)	0
T_4	0	0	0	-	0	0
T_5	(c_9, 1)	(c_{10}, 1)	(c_{11}, 1)	0	-	0
T_6	0	0	0	0	(c_{12}, 2)	-

** The notation (c_x, 1) indicates the constraint number and the degree of dependency. A value of 1 indicates a strong dependency while a value of 2 indicates a weak dependency. The descriptions below correspond to the constraint values above.

3.1.2 Description of Constraints on Dependencies

(c_1, 2) – Direct dependency; employee_id in Employee_Salary is a foreign key to employee_id in Employee. Any changes to the job_title_code, marital_status_code, and dependents of Employee will affect pay_period_id, net_pay, and gross_pay of Employee_Salary. Knowing the marital_status will affect net_pay in Employee_Salary.

(c_2, 2) – Direct dependency; employee_id in Employee_Pay_Adjustment is a foreign key to employee_id in Employee. The adjustment_type_code and adjustment_amount will be affected by pay_per_period, marital_status_code, and dependents. Knowing the marital_status will allow information about adjustment_amount in Employee_Pay_Adjustment to be known.

(c_3, 2) – Direct dependency; job_title_code in Employee is a foreign key to job_title_code in Position_Title. The job_title_code and base_pay of Position_Title will affect the pay-per-period of Employee. The job_title_code will only be useful if the job_title is known.

(c_4, 2) – Transitive dependency; base_pay in Position_Title corresponds to gross_pay in Employee_Salary.

(c_5, 1) – base-pay of Position_Title will affect the adjustment_amount for taxes etc. in Employee_Pay_Adjustment

(c_6, 2) – The gross_pay of Employee_Salary will affect the Employee_Pay_Adjustment

(c_9, 1) – Based on the adjustment_type_code, adjustment_amount, and adjustment_desc of Employee_Pay_Adjustment, the marital_status, pay_per_period, and number of dependents of an Employee can be determined.

(c_{10}, 1) – Based on adjustment_amount due to tax brackets etc. of the Employee_Pay_Adjustment, the base_pay and consequently the position of an individual can be inferred.

(c_{11}, 1) – The net_pay of Employee_Salary is affected by Employee_Pay_Adjustment

(c_{12}, 2) – Direct dependency; adjustment_type_code in Employee_Pay_Adjustment is a foreign key to adjustment_type_code in Adjustment_Type

3.1.3 Constraint and Dependency Graph

Based on the constraints determined, a set of attributes corresponding to threat conditions have been constructed which outline sensitive information that will be revealed to unauthorized users if all nodes of the graph are accessed.

For the purposes of the simulation, every attribute of the Payroll Data Schema has been numbered as shown in Appendix B. These numberings provide the necessary information to run the simulation that will create the knowledgebase of insiders and prevent any access to sensitive information. Since it is assumed that the users do not have any previous special authorizations,

they will not be allowed to execute any transactions that allow them to obtain corresponding information to any of the threats listed above.

The format for listing the dependencies and constraints that reveal sensitive information is shown below. This is what will be entered in the simulation to represent the schema and potential threats.

Dependency Constraints on Sensitive Information

Condition 1: Base salary of an employee
Attributes: 1, 2, 10, 4, 5
Condition 2: Gross salary based on start date
Attributes: 1, 2, 10, 4, 5, 7
Condition 3: Number of dependents of a given employee
Attributes: 17, 22, 23, 10, 1, 4, 5, 8, 18
Condition 4: Net pay of an employee for a given month
Attributes: 17, 19, 10, 21, 16, 1, 4, 5, 20, 7
Condition 5: Marital status of employee
Attributes: 1, 4, 5, 3, 17, 18, 19, 22, 23
Total number of attributes for schema: 23

Additional Notes on Threat Conditions
Knowledge of the number of dependents can be used to determine the insurance amount for a given individual.
Knowledge of the marital status of an individual will provide information about that person's tax bracket, which can be used in turn to infer the gross pay for that employee.

3.2 Generic Schema

A generic schema was randomly generated after specifying values for the total number of attributes and the number of threat conditions. The generic schema was generated as a sample schema to compare simulation results against the Payroll Data Schema. The generic schema in comparison to the Payroll Schema has only 12 attributes.

The dependency constraints for this schema are shown below:

Threat Attribute Values:

Condition 1: 1, 3, 5, 6, 7, 12

Condition 2: 2, 6, 7, 11, 12

Condition 3: 4, 5, 7, 1, 2, 9

In this paper, the simulations executed are similar for both the Payroll and the generic schema. The only differences that occur are that the range of values tested change according to the size of the schema. For instance, there is a smaller range set in the number of attributes accessed for a given transaction in the generic schema, given the relatively small number of attributes.

4. Simulation

4.1 Overview

The simulation has been designed in an effort to allow administrators to have a practical way to apply the research done on predicting and preventing threats to identify the areas where threats are most likely to occur. Based on this information, they can then provide access rights that give users as much freedom as possible while maintaining the security of the system. The simulation also provides an idea of how long it takes to obtain unauthorized knowledge of data items for various types of relational database schemas and reveals which areas or conditions are most susceptible to insider threat.

The simulation has been designed in such a way that it allows for multiple schemas to be run and saved for further analysis. The simulation also allows multiple parameters to be changed for testing purposes, including specifying the total number of attributes. Anyone using the simulation has the option to change the number of users that execute transactions, determine the number of overall transactions that are allowed, and specify a min and max for the number of attributes that are accessed per given transaction. This is randomly generated based on the min and max values specified.

The simulation can be accessed via the following link:

http://eventfinderbeta.com/InsiderThreat/publish.htm

4.2 Screen Layout

4.2.1 Schema Entry

The schema entry tab allows users to provide details relating to the schema that they run the simulations on. The user first enters the total number of attributes that are in the tables for a given database schema. The attribute threat combination corresponds to the list of attributes which if an insider without special permissions has access to, can reveal unauthorized sensitive information. To provide the attribute threat combinations, all attributes for a given schema should first be numbered. Based on the dependencies and constraints that exist, the attributes which when aggregated provide sensitive information are then entered as shown below with each threat condition corresponding to one line in the list box of the Insider Threat Schema.

Users are required to save the schema after entering it so that they do not have to re-enter the schema when they want to run further simulations on the specified list of threat conditions.

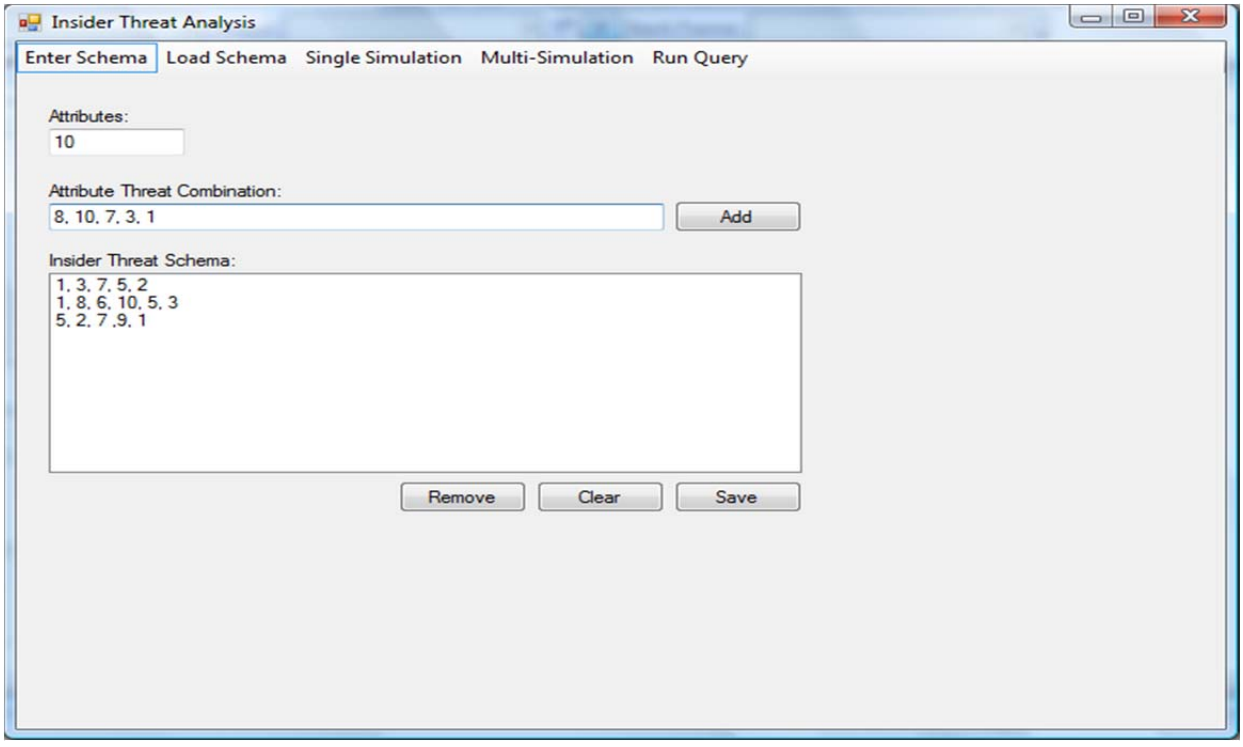


Figure 1 Schema Entry Example

4.2.2 Schema Loading

The saved schemas that were entered in the previous tab can then be opened in the Schema Loading tab. The opened schema is displayed in order to make sure that the threat conditions were as specified. The first line shows the total number of attributes and the threat conditions are listed by comma separated values in the lines following. The value provided for the total number of attributes is used to randomly generate transaction reads for users in the simulation. A schema has to be opened and loaded before a user can proceed to execute either a single run simulation or a multi run simulation.

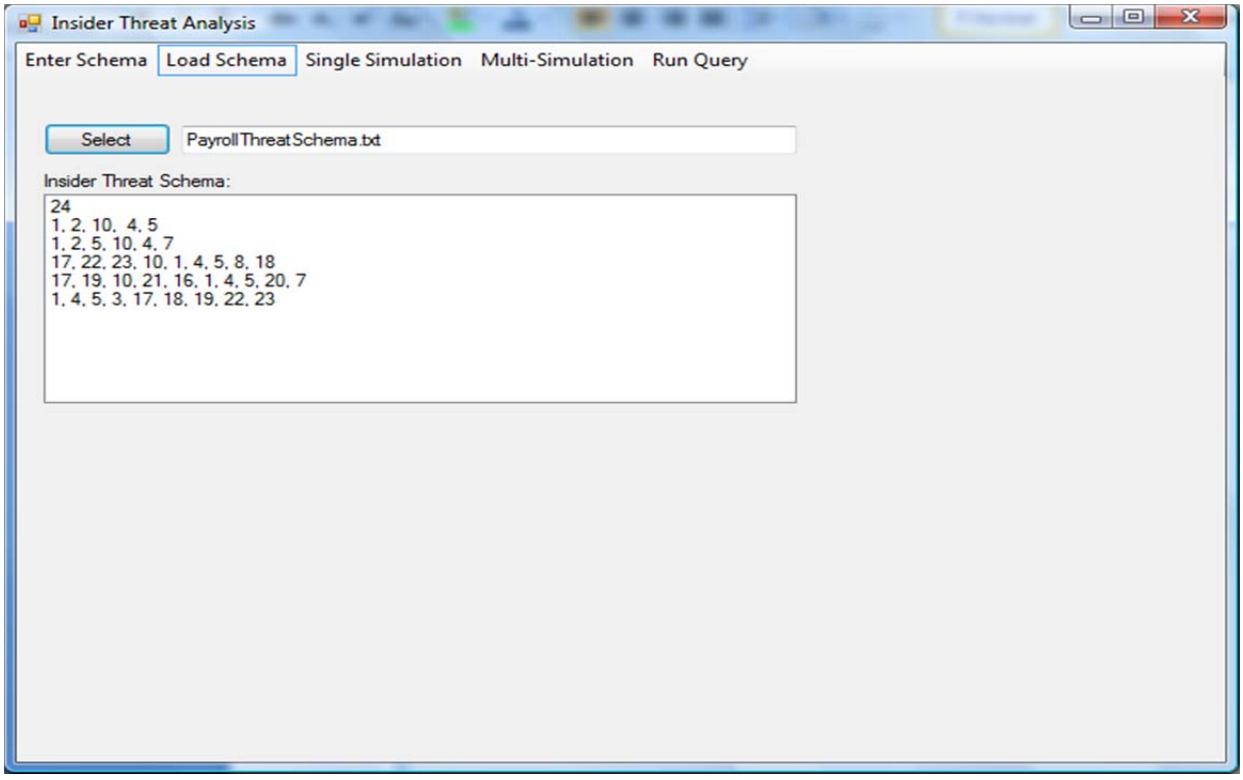


Figure 2 Loading Schema Example

4.2.3 Single Simulation

After the user specifies a schema to simulate, a single simulation run can be executed. The following parameters namely, users, transactions, and a range for the number of attributes per transaction need to be specified. The number of users that is provided is used to randomly simulate the overall number of specified transactions. The number of attributes per transaction is also randomly generated based on the given range. Therefore, as shown in the example below, the total number of reads that occur for each transaction for any given user will access between 5 to 7 items. For the purposes of the simulation, we are only considering transaction with reads and not writes to avoid dealing with additional complexities such as issues involving updates, etc. which are beyond the scope of this paper.

When a single simulation is executed, the data grid on the left shows the overall knowledge base of a given user based on accumulating the access information from all of the transactions for that user. The data grid on the right shows the random simulation of the total number of transactions distributed randomly across the number of users. The column labeled 'Allowed' shows whether a given transaction was approved or rejected. If a transaction was rejected, the column labeled 'Violates', shows which threat condition was violated that prevented the transaction from executing.

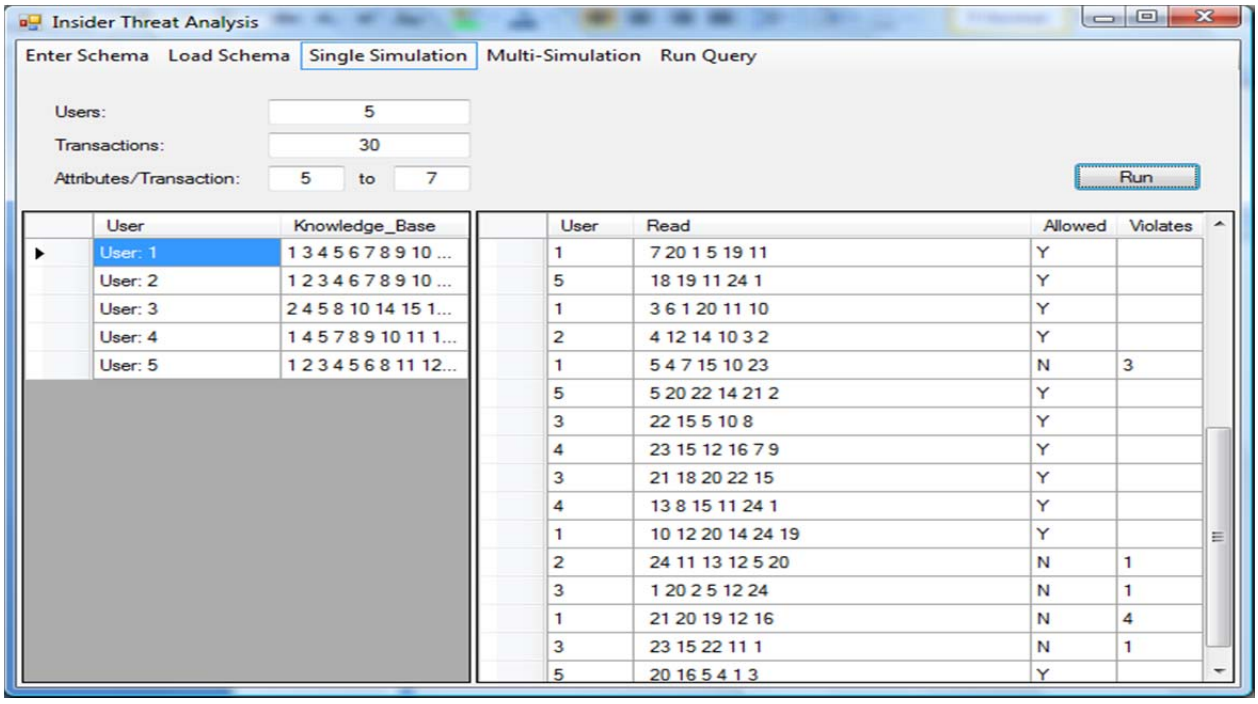


Figure 3 Example of Single Simulation Results

4.2.4 Multi Simulation

The multi simulation takes the same parameters that are found in the single simulation, except the multi simulation introduces additional complexity in allowing the three parameters to be executed against each other with two variables changing via the x-axis and series while the third variable remains constant. For instance, in the example below, the number of users is set to be x axis changing in increments of 5 and ranging from 5 to 25. Transactions is set as the variable manipulated through the series. In this example, the overall number of transactions is set to vary from 50 to 200 in increments of 50. Finally, the third variable, the number of attributes per transaction, is set as before ranging from 5 to 7. The text box labeled 'Runs Per Data Point' is used to specify the number of executions for each given set of parameters. The average is taken from all of the runs for a given set to obtain the greatest consistency and the most accurate results. The results are expected because as the number of transactions increases for a given number of users, the more likely that there will be more rejected transactions.

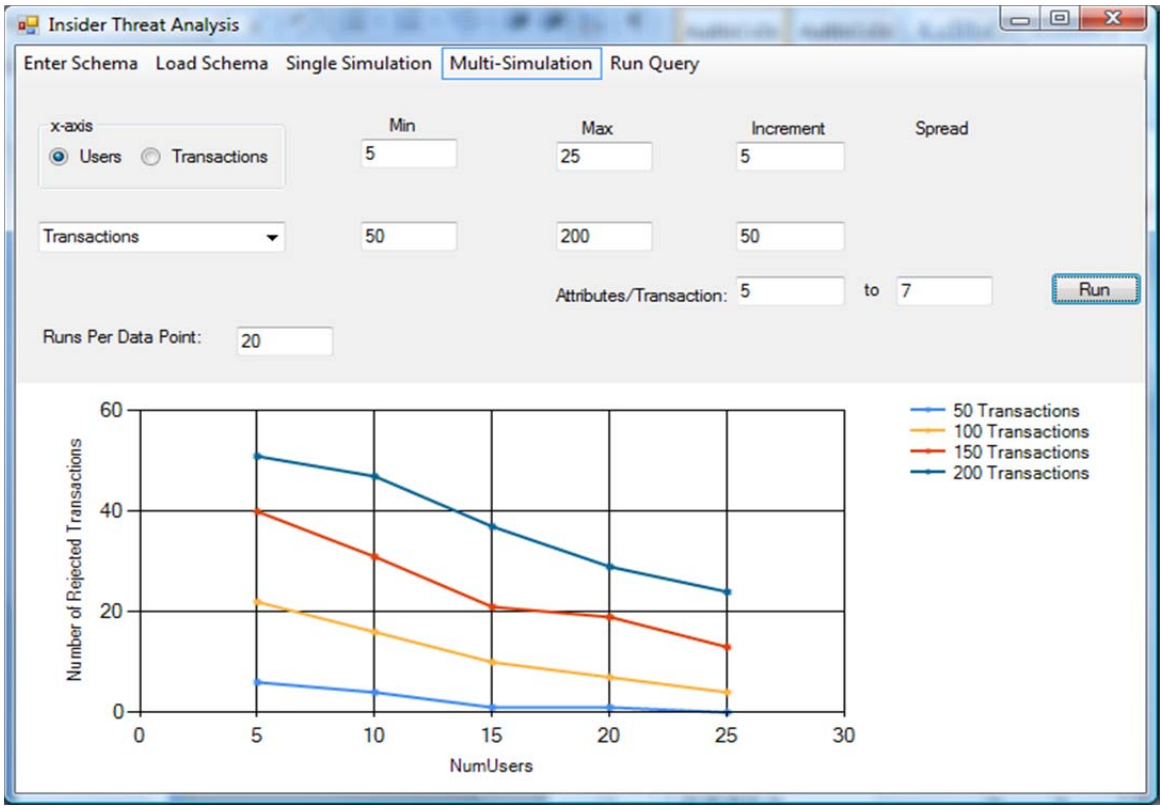


Figure 4 Example of Multi-Simulation Run

4.3 Example Simulation Runs

4.3.1 Single Run of Payroll Schema

The following data shows a single execution of the Threat Simulation Application in greater detail using the Payroll Schema. The following parameters were specified in running the simulation:

Users: 5

Transactions: 30

Items to be accessed per transaction: 5-7

The results generated in running the above simulation were as follows:

Table 2 Example of Transactions Executed Per User

User	Read	Allowed	Violates
2	19 14 18 4 24	Y	
2	18 1 22 4 13	Y	
5	13 15 12 8 5 14	Y	
5	12 9 15 2 18	Y	
4	17 15 4 6 12 13	Y	
3	7 12 18 6 19 11	Y	
2	19 2 10 5 1	N	1
1	7 19 2 20 10	Y	
3	18 8 6 13 9	Y	
3	1 18 24 21 8	Y	
2	18 22 6 24 23 9	Y	
3	24 6 5 18 20	Y	
5	3 6 13 11 7 24	Y	
4	8 10 4 7 17 14	Y	
2	23 4 15 14 20 18	Y	
2	4 8 7 22 15 20	Y	
4	12 20 6 22 5	Y	
5	9 8 16 13 5 11	Y	
4	13 17 18 12 19 8	Y	
5	13 11 2 1 16 18	Y	
2	21 22 3 23 17 9	Y	
5	10 16 18 17 2	Y	
3	11 3 23 19 4	Y	
2	10 16 5 21 8	N	3
5	16 9 14 19 21 8	Y	
3	10 17 1 5 20 11	Y	
3	14 19 20 12 22 2	N	1
2	17 24 3 2 19 1	Y	
4	22 16 7 15 14	Y	
3	12 2 5 14 6	N	1

The knowledgebase of the users that was built as they read multiple transactions and accessed more attributes is shown below:

Table 3 Example of User Knowledge Base

User: 1	2 7 10 19 20
User: 2	1 2 3 4 6 7 8 9 13 14 15 17 18 19 20 21 22 23
User: 3	1 3 4 5 6 7 8 9 10 11 12 13 17 18 19 20 21 23
User: 4	4 5 6 7 8 10 12 13 14 15 16 17 18 19 20 22
User: 5	1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 21

The results show that sensitive information from threat condition 1 is the most likely to be discovered through random access. This agrees with the constraints and dependencies imposed on the payroll data model. Threat condition 1 is composed of only 5 attributes, indicating that critical information can be obtained if access to all 5 attributes is obtained. The other threat conditions require having knowledge of more than five attributes. Thus, threat conditions that can only be obtained through the aggregation of information from several attributes have a lesser likelihood of being violated through random accesses.

4.3.2 Number of Users versus Rejected Transactions

The Payroll schema and the generic schema were both simulated in multiple runs to determine the number of rejected transactions while keeping the number of transactions the same and changing the number of users. Multiple runs allowed the results to be compared across multiple numbers of transactions executed.

The results of each run were obtained by taking the average of 100 runs per data point to ensure consistency. The number of attributes accessed per transaction was held constant between the range of 3 to 5 and were randomly generated among those values for every transaction in this simulation run. The results for the Payroll Schema are shown below.

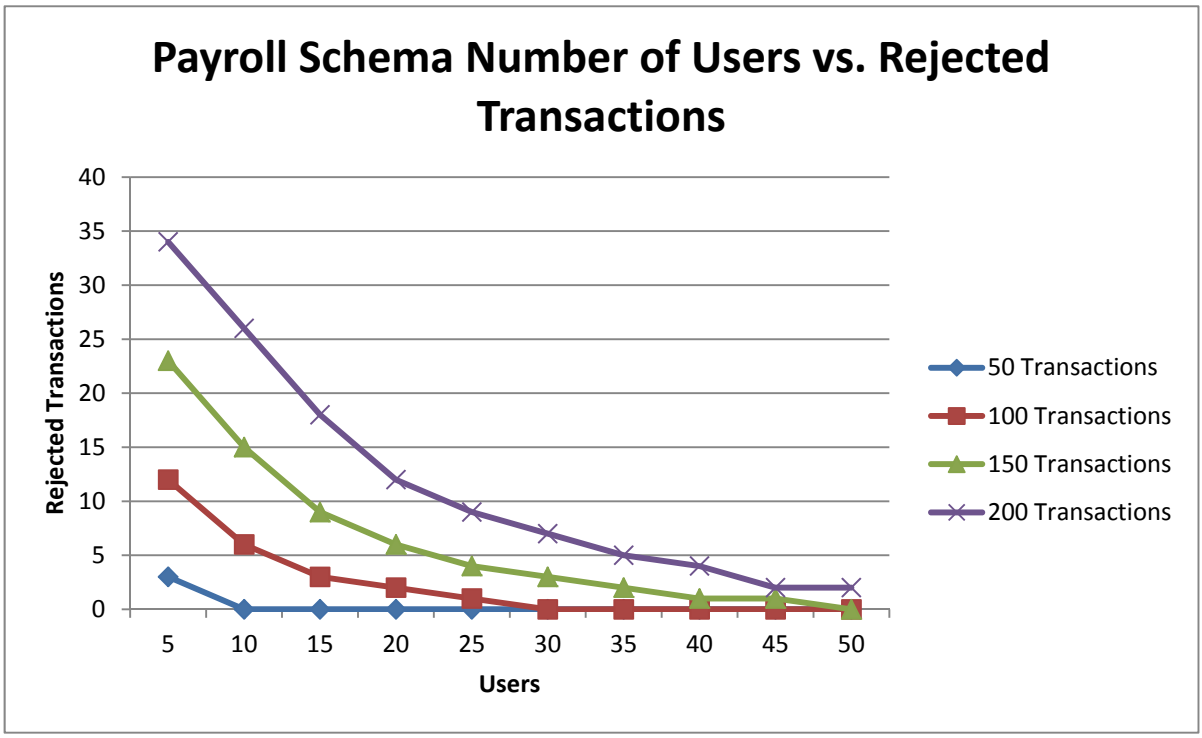


Figure 5 Graph of Payroll Schema, Number of Users vs. Rejected Transactions

There is a correlation seen in the percentages of rejected transactions as the total number of transactions increase relative to the number of users. For instance, the number of rejected transactions for 5 users regardless of increasing the overall number of transactions was approximately 1/6 of all transactions. Running 150 transactions for 5 users resulted in an average of 23 rejected (23/150 = 15%) while running 200 transactions resulted in 34 rejected (17%).

A possible explanation for this is that once most of the information from the schema has been added to the knowledge base, our criteria to reject any transaction, which will result in the discovery of confidential information, results in one attribute from each threat condition being excluded. This roughly corresponds to the total number of transactions that are rejected. Since there were a total of 24 attributes for the payroll data schema with 4 threat conditions, (4/24 = 16%) provides a relatively accurate number of rejected transactions relative to the number of transactions executed.

This information can be useful in creating a threshold value for user accesses to the database. Given knowledge that 16% of transactions are rejected because sensitive information can be

obtained, users should be restricted to access only (100% - 16%) = 84% in random accesses to ensure that they are not able to discover any sensitive information. The results for the generic schema are shown below.

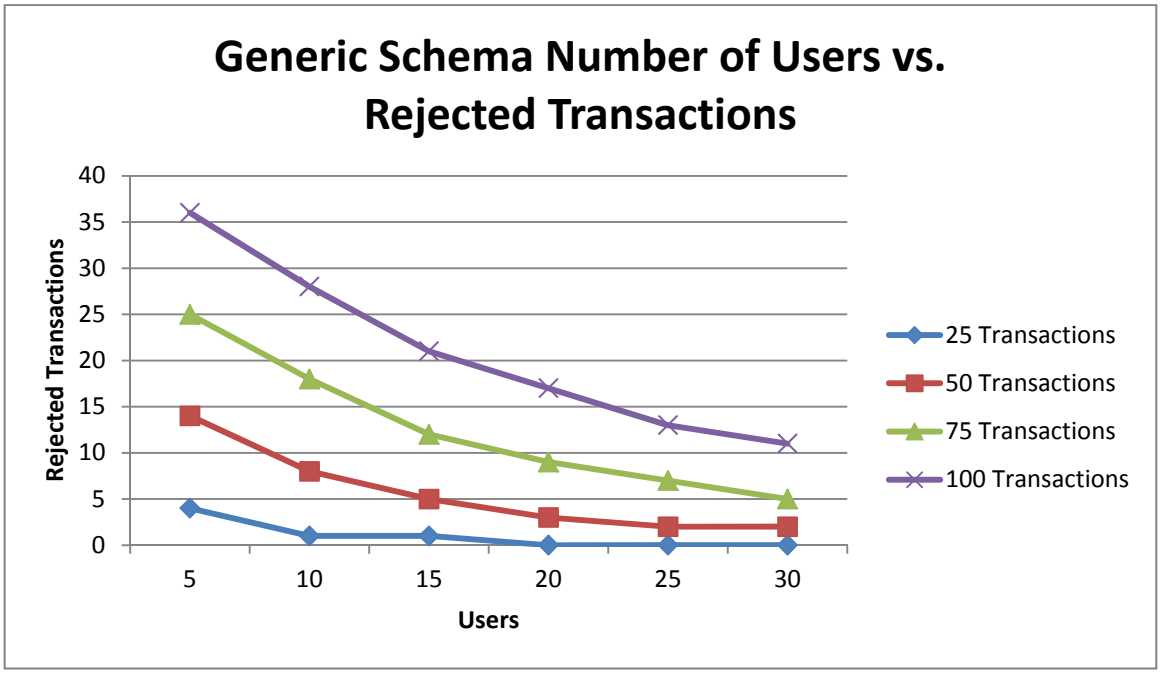
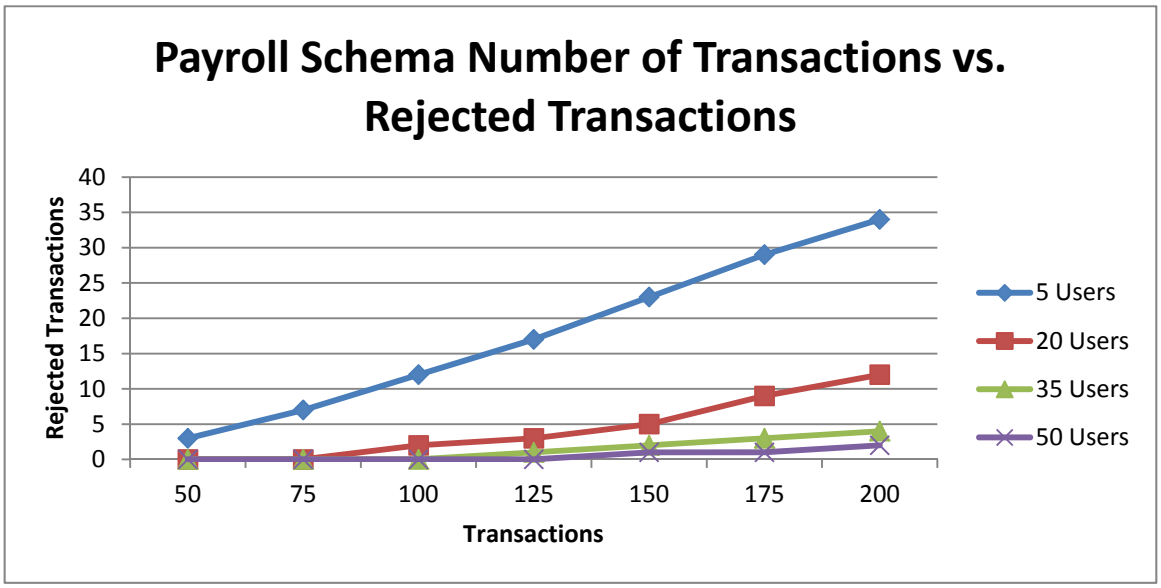


Figure 6 Graph of Generic Schema, Number of Users vs. Rejected Transactions

Generalizations that can be obtained in comparing the results using the generic schema to the payroll data schema show that the fewer the number of users that access the database, the more likely they are to obtain sensitive information quickly. This can be seen in Figure 6 where when the number of users ranges between 5 and 15, a greater number of transaction requests are likely to be rejected. This number evens out more as the number of transactions executed is increased relative to the number of users.

4.3.3 Number of Transactions versus Rejected Transactions

Simulations were run to test changing the number of transactions while keeping the number of users constant. The expected results were that it would be inverse of the above graphs for the two schemas respectively. As before, the results of each run were obtained by taking the average of 100 runs per data point and the number of attributes accessed per transaction was randomly generated between the range of 3 to 5. The results of running the simulation for the Payroll Schema is shown below.



The results of running the simulation for the generic schema is shown below:

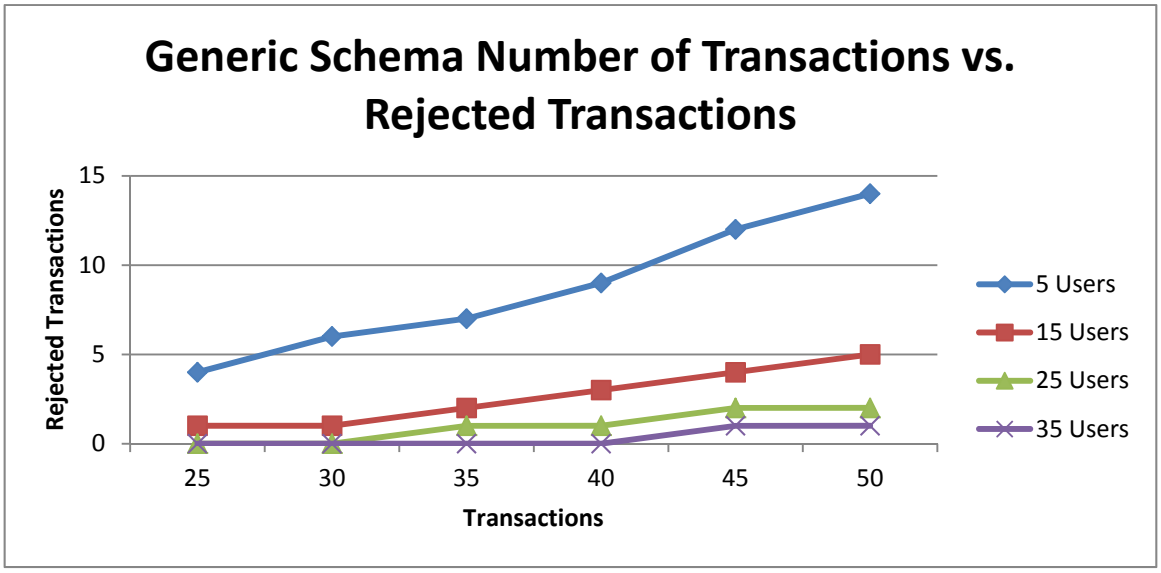


Figure 7 Graph of Generic Schema, Number of Transactions vs. Rejected Transactions

As expected, an analysis of the values showed that the simulations run comparing the number of transactions to the overall number of rejected transactions had a direct inverse relationship to the number of users and the number of rejected transactions. This can be understood intuitively that as the number of transactions increases for a fixed number of users, the more likely they will

develop their knowledgebase through random accesses and be more likely to request transactions that will allow them to decipher confidential information.

4.3.4 Number of Items Accessed versus Rejected Transactions

A test simulation was run to determine if there was any correlation between the number of items accessed at random within a given range to the number of rejected transactions overall. This simulation was run, keeping all other parameters of the number of users vs. rejected transactions the same except changing the range of the attributes accessed per transaction to 5-7 from 3-5.

The following graph was generated from running the simulation:

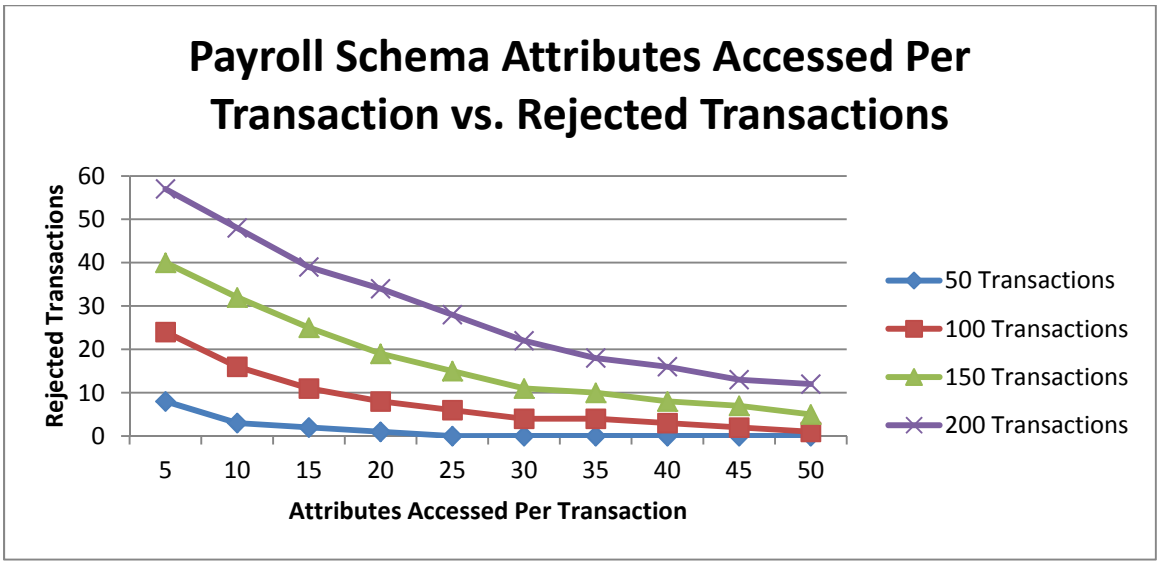


Figure 8 Graph of Payroll Schema, Attributes/Transaction vs. Rejected Transactions

Comparing the results of the graph to the initial graph that had only 3-5 items accessed at random per transactions showed that more transactions were rejected for the same number of users and number of overall transactions when the number of data items accessed were between the range of 5-7. This agrees with the premise that the more number of attributes that users are allowed to access in a given transaction, the more likely they are to build up their knowledge base at a faster rate and consequently increase their likelihood to determine sensitive information.

4.4 Query Simulation

A query simulation was developed that parses, analyzes, and accepts or rejects a user's SQL statement. Based on the existing knowledgebase of the user, the statement either executes and returns the expected results or fails to execute if sensitive information can be obtained. If the transaction request is allowed, it is passed to the database and the SQL statement is executed. The results are displayed in the query simulation window. Thus far, SELECT is the only SQL statement that is supported. Future developments may include support for update statements. This provides a pro-active method to prevent any insider breaches of unauthorized information.

The query simulation works in a similar fashion to the single simulation and multi simulation in that it builds up the knowledgebase of the user as access to data is provided. Transaction execution is blocked if the fields requested can allow the user to infer any sensitive information. The query simulation uses General SQL Parser, a commercially available SQL Parser that identifies the attributes accessed in the SELECT and WHERE clauses of a SQL statement. The current implementation uses SQL Server. At present, the query simulation is a prototype, but it can be implemented as a network service, which would allow applications to access it rather than merely providing a UI.

4.4.1 Query Simulation Demonstration

The following shows a basic SELECT statement.

The results shown by the executed SQL statement show that there is no confidential information that can be obtained thus far for the requested transaction.

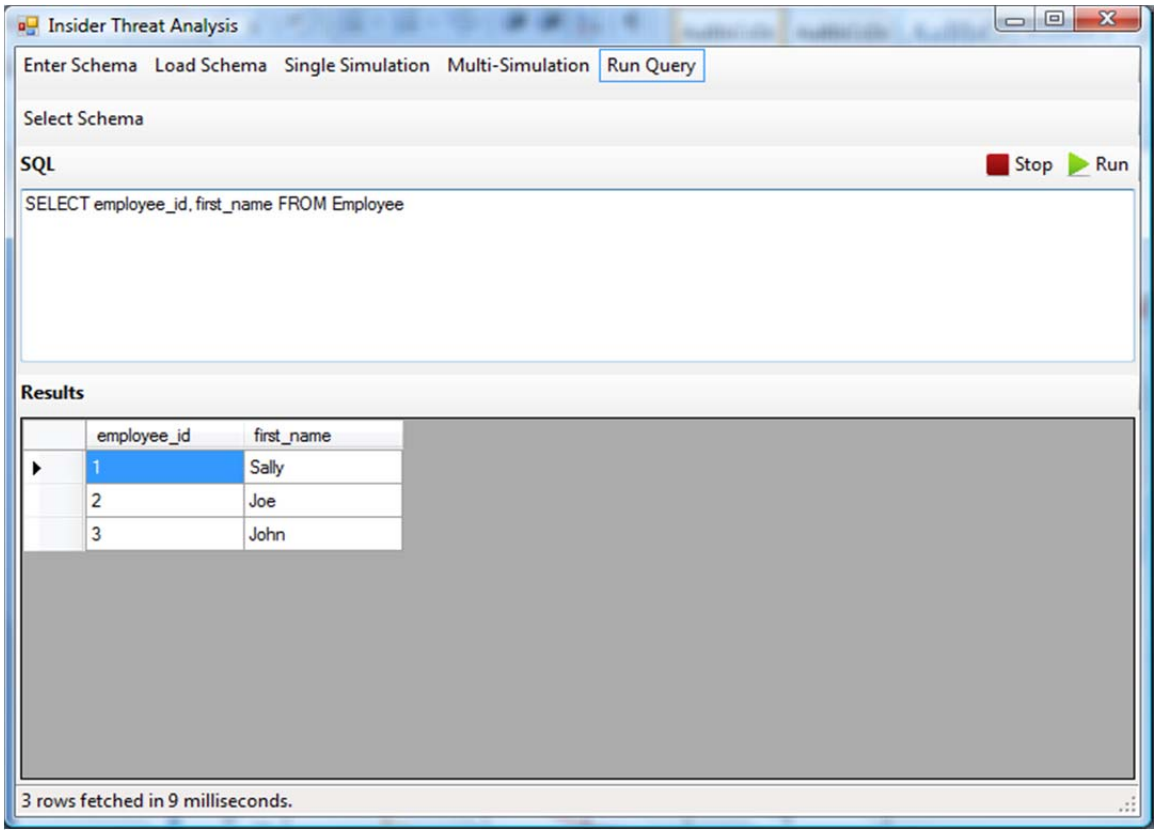


Figure 9 Query Simulation Example 1

The following diagram shows another SELECT statement. Note that the knowledgebase now includes all of the attributes selected so far including those selected in the previous statement.

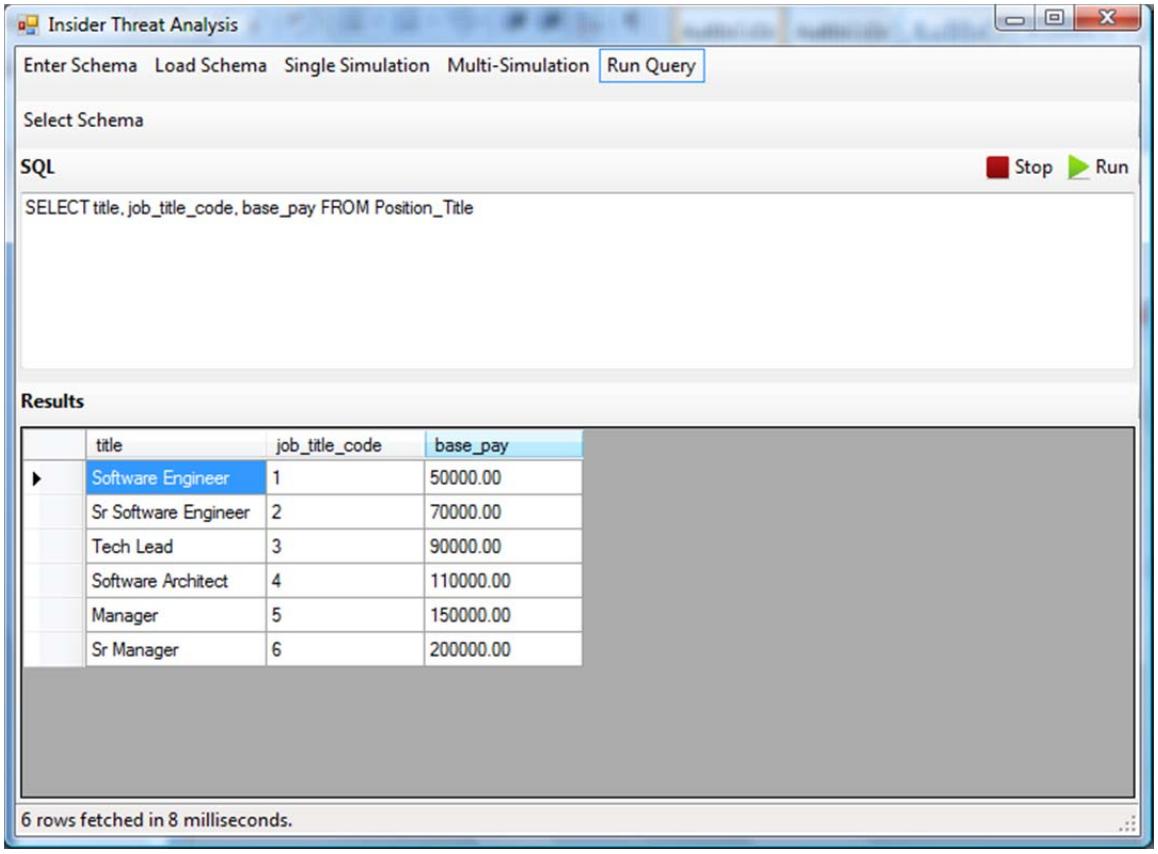


Figure 10 Query Simulation Example 2

Given the knowledgebase being built up thus far, the following query results in the request being rejected because sensitive information can be obtained. The transaction request is rejected because based on the information obtained so far, the user can establish a relationship between a given employee and their base pay which is strictly confidential.

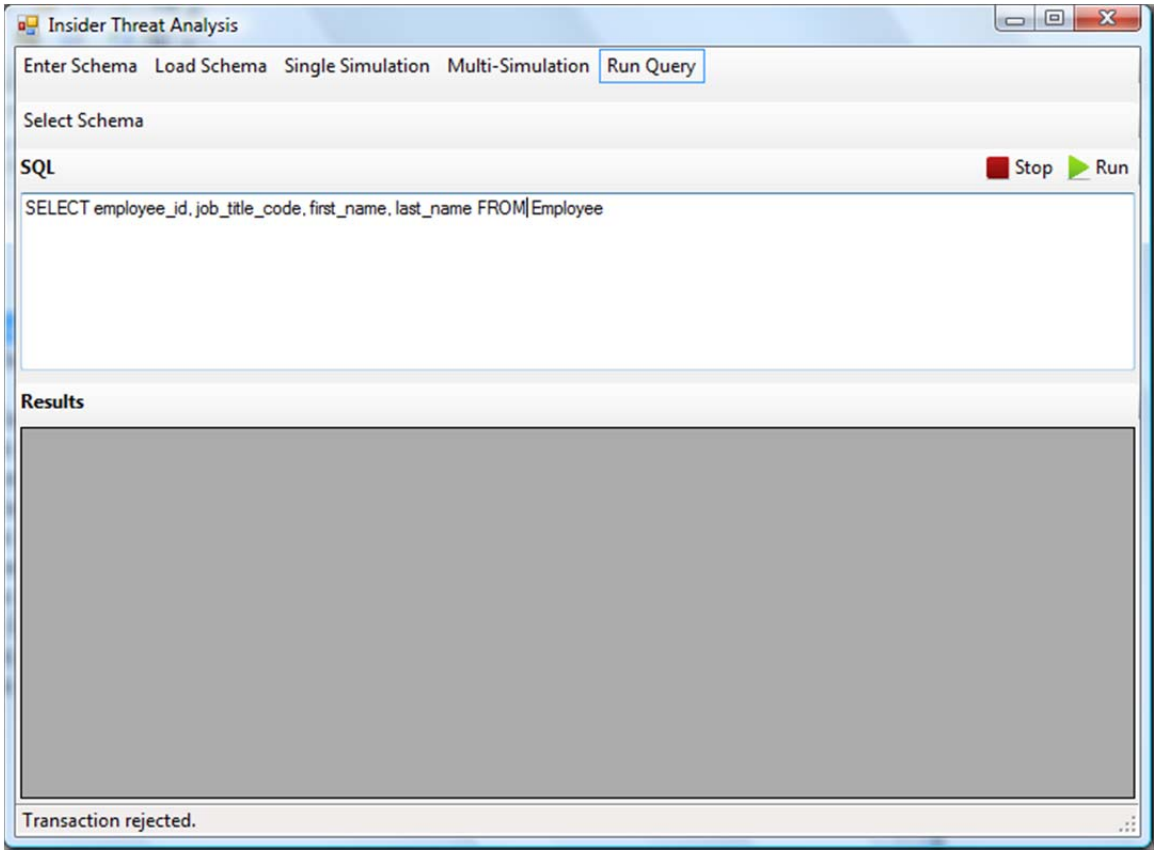


Figure 11 Query Simulation Example 3

5. Application

The simulation can be used in various ways to obtain useful information that can assist in enhancing the security and avoiding malicious insider attacks in relational database systems. Given a schema that has been broken down into various threat conditions, the simulation can be used to determine how to grant permission rights that reveal minimal confidential information. Moreover, having knowledge of threats that are most likely to be violated provides the system administrator with information to identify the best balance between providing access to as much data as possible and restricting the ability of users to infer unauthorized information.

At present, the simulation is pro-active to prevent insider threat before a breach occurs. Another development would be to have the option of scanning through logs of a specific database to develop the knowledgebase built up by users through their accesses. This can then be compared against the input threat conditions for the schema to see which users could have accessed which pieces of confidential information. If certain users did not have the authorization to such information, permissions and restrictions could be put in place to ensure that they no longer had access to the information.

6. Conclusion

An insider of a relational database system has been defined as "someone who has authorized access, privileges or knowledge of the relational database system he/she uses, and is familiar with the dependencies between data objects as well as the related mappings, and is motivated to violate the security policy of the system through authorized access" [1]. The objective of the paper by Yaseen and Panda was to provide a method that would allow insiders to perform their tasks as efficiently as possible without having potential threats. In the case where insider threat would be handled by extensively restricting permissions, the availability of information would be limited and users would not be able to work as effectively or efficiently as possible. The strategies and methods recommended in the paper stress the importance of prediction and prevention which allow users to access as much information as they can until it interferes with any sensitive information to which they do not have access.

Schemas are used under the assumption that attributes are far less likely to change on a frequent basis and will thus provide a reliable source to generate dependencies and constraints. The dependencies and constraints on dependencies that exist in a given schema can then be used to generate the user's knowledge graph which can be used to predict and prevent the threat that insiders pose. The Threat Prediction Graph and an insider's knowledge base that is built as the user requests transactions can be used to determine a threshold value as to the maximum amount of information a single insider can obtain regarding a given attribute. Once the threshold value is reached or exceeded, a user can be blocked access pro actively or a warning can be issued to the administrator to either revoke access or grant access to certain attributes.

At present, from Yaseen and Panda's paper, it is possible to create the knowledgebase of a user. That is given a list of what the user has permission to access, the knowledge base is constructed using the Constraint and Dependency graph to track how much information about other attributes can be inferred. This however, does not provide a calculation for the threat conditions that exist. As discovered in going through the simulation, knowledge about critical items can be obtained only through an aggregation of knowledge of different sets. The CDG cannot be obtained from

the schema as the conceptual constraints cannot be determined without knowledge of the business rules and the organization. A knowledge base can be built by a user based on inferences that require knowledge of relationships that go beyond foreign key relationships, etc.

The simulation provides a practical application of the methodology proposed in Dr. Panda and Qussai Yaseen's paper and offers a pro active solution to predict and prevent the insider threat problem in relational database systems. Generating the knowledge graph and knowledge base of users allows the system to keep track of the amount of information obtained. Warnings can be issued if insiders have the ability to infer values of data items to which they do not have authorized access. The simulation demonstrates a model of how the threat prediction and prevention solutions can be implemented for any database schema and consequently shows the potential for application in industry.

7. References

[1] Yaseen, Q., Panda, B., "Knowledge Acquisition and Insider Threat Prediction in Relational Database Systems," In: *Proceedings of the International Workshop on Software Security Processes*,. Vancouver, Canada, August, 2009, pp. 450-455

[2] Yaseen, Q., Panda, B., "Predicting and Preventing Insider Threat in Relational Database Systems," In the process of publication. Springer Link. 2010 http://www.springerlink.com/content/e43j7q17534hmk47/

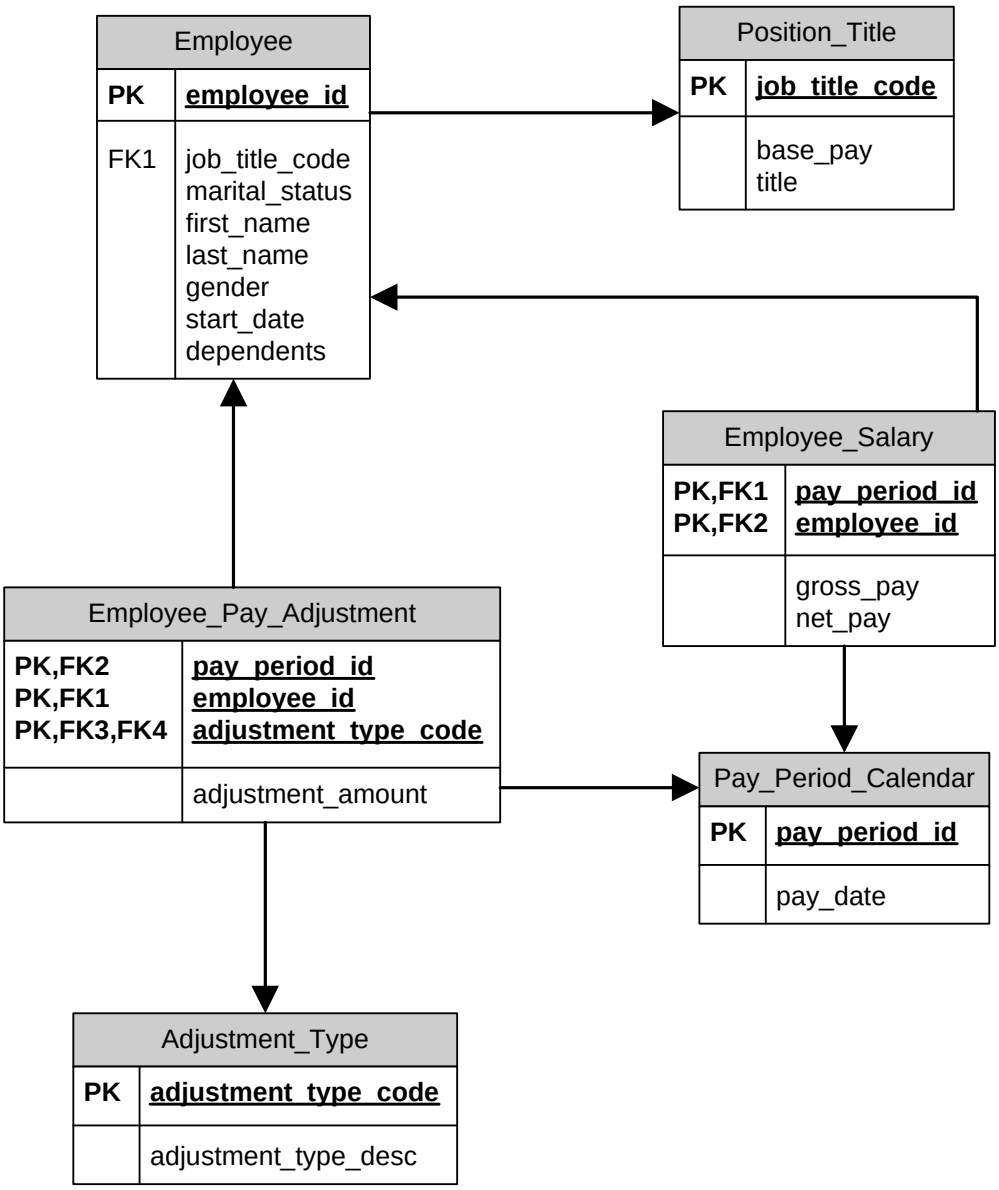
[3] Brackney, R., and Anderson, R., "Understanding the Insider Threat," In: Proceedings of the March 2004 Workshop. Technical Report, RAND Corporation. Santa Monica, CA, March, 2004.

[4] McCue, Andy. "Beware the Insider Security Threat*,"*. *CIO Jury*, April 17, 2008, Silicon.com. http://www.silicon.com/management/cio-insights/2008/04/17/beware-the-insider-security-threat-39188671/#comments

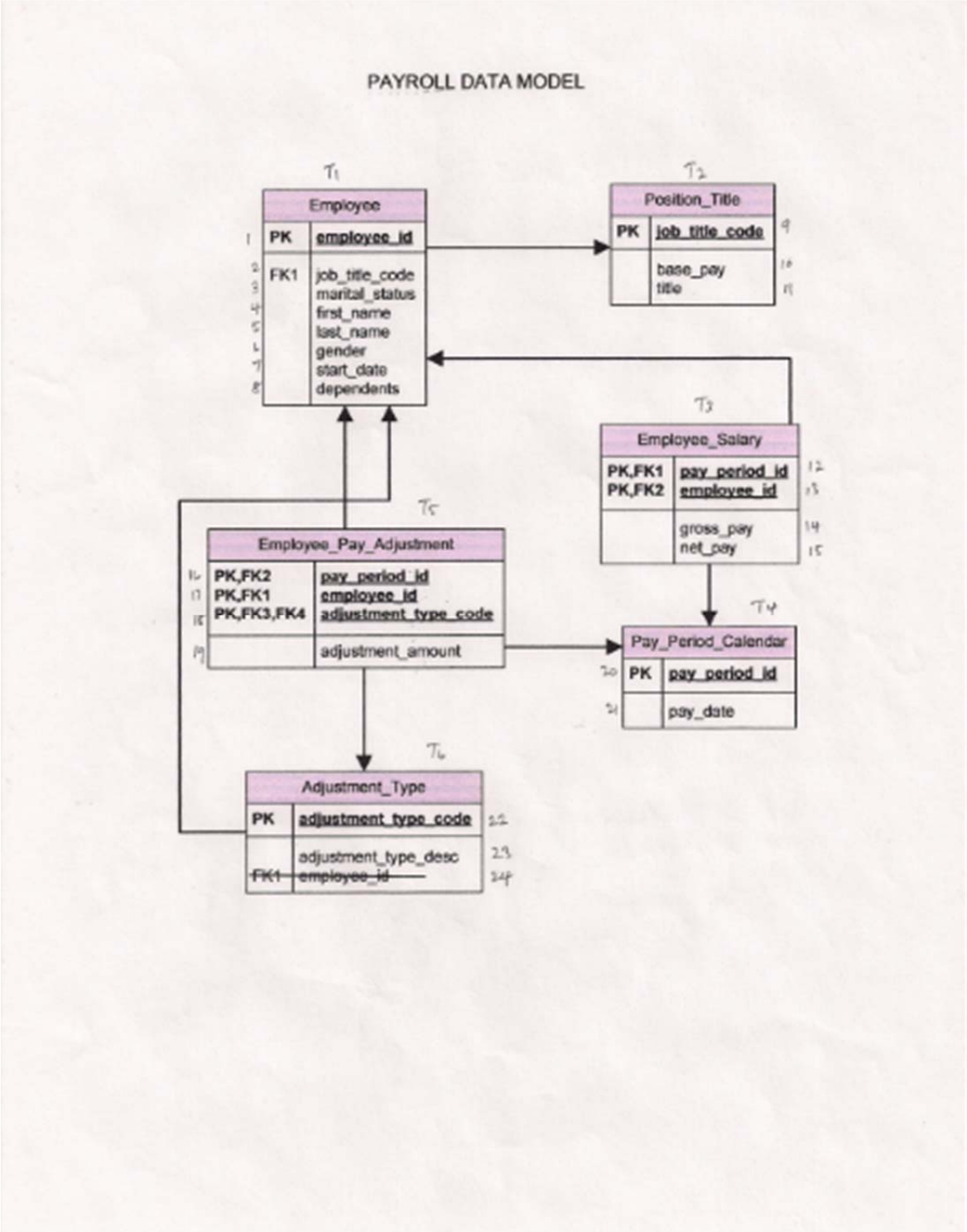
[5] Gordon, L., Loeb, M., Lucyshyn, W., Richardson, R., "Computer Crime and Security Survey." Available at http://www.cpppe.umd.edu/Bookstore/Documents/2005CSISurvey.pdf

7. Appendix A

PAYROLL DATA MODEL



8. Appendix B


PAYROLL DATA MODEL
Employee
PK employee_id
FK1 job_title_code
marital_status
first_name
last_name
gender
start_date
dependents
Position_Title
PK job_title_code
base_pay
title
Employee_Salary
PK,FK1 pay_period_id
PK,FK2 employee_id
gross_pay
net_pay
Employee_Pay_Adjustment
PK,FK2 pay_period_id
PK,FK1 employee_id
PK,FK3,FK4 adjustment_type_code
adjustment_amount
Pay_Period_Calendar
PK pay_period_id
pay_date
Adjustment_Type
PK adjustment_type_code
adjustment_type_desc
FK1 employee_id