Security Analysis in Multilevel Databases

Aniket Kuiri(163050059)[1]

[1]Department of Computer Science
IIT Bombay

June, 2018

Outline

Importance

- Database contains lots of information
- Users need them for different purposes
- Giving more information to an user than required can pose threat.

Instances of Threat

- Military Databases contains information like location of military bases, details of army personnel, location of artillery, etc. that are very sensitive.
- Medical databases contain medical records of individuals which a person would nor like to show to others.
- Transportation databases contain information of arrival, departure, refuelling stations, etc. that can pose danger if given people who hijack gets them.
  So it is necessary to control information flow.

Hippocratic Database

- Each row of data is owned by a data-owner.
- Access to data to user us based on the purpose for which the user wants the data.

Let us consider the following database of a company:

Table name	Attributes
Customer	Customer-Id, Name, Shipping Address, e-mail, Credit-card info
Order	Customer-Id , Transaction-id, product info, Status

Example of table customer

Customer-ID	Name	Shipping-address	e-mail	Credit-card info
A1	Alex	Washington	alex@gmail.com	7895-4578-4512-7896
A2	Jim	Texas	jim@hotmail.com	8974-6524-1452-3254
A3	Olly	Boston	olly@ymail.com	7896-2541-3256-2541
A4	Brian	New York	brain@gmail.com	2145-9874-6589-4785

Table: Table customer

Here Alex, Jim, Olly and Brian are the data owners.

Example of table order

Customer-Id	Transaction	Product info	Status
A1	T123	Phone	Delivered
A2	T234	Laptop	Not Delivered
A3	T213	Books	Dispatched
A4	T324	Clothes	Delivered

Table: Table Order

Understanding Hippocratic Databases

- The company has various departments that needs data from these 2 tables for various purposes.
- Shipping-address column for data-owners can be used for various purposes by the company:
    1. It can be used by the Shipping department to deliver the product.
    2. It can be used by Analytics department to determine type of products purchased in a particular region.
- Similarly the e-mail column for data-owners can be used for various purposes:
    1. It can be used by the Advertisement department to give update of new products
    2. It can be used by Customer-service department to answer any inquiry.
- Similarly the Purchase-Product department needs the name and credit-card info for purchase of product.

Understanding Hippocratic Databases

- So we see that the data of data-owners can be used by Company for various purposes.
- But there may be Customers (data owners) who would not like to share their data for all purposes.
- There can a customer who would only allow his Shipping-address column to be used for delivery of product and not to determine statistical analysis.
- There can be another customer who would not like to share his e-mail address for advertisement purposes.
- In general it should be under the control of the Data owners that is the customers for what purpose their data is being used by the comapny.

Understanding Hippocratic Databases

- So each customer based on their choice of purpose gives their data to different departments.
- These kind of databases where the data-owners controls their data to be shared based on purposes are known as Hippocratic Databases.

Policy and Privacy Authorization Table

- Let us consider each department of the company as a subject that wants to access data.
- Initially the data of the data-owners is not shared to any one.
- To give access to a subject by a data-owner the data-owner executes a policy.
- The policy defines which column a data-owner is sharing to which subject and for what purpose.
- The policy enters data to a Privacy Authorization Table.
- Each data owner has his own privacy authorization table.
- The authorization table captures the access controls that support the privacy policy.

Policy Enforcement

A **policy** is a rule to give access (read/write) to a database element of a table to a person (subject in this case).

Syntax of policy:
Create restriction <**policy-name**>
On <**table-name**>
For user <**subject-name**>
To cell <**attribute-name**>
For purpose <**purpose-name**>
Restricting access to <**access-type**>
Example:let data-owner Alice creates the policy as below: Create restriction R1
On **customer**
For user **shipping**
To cell **shipping-address**
For purpose **delivery**
Restricting access to **select**

Example

Privacy authorization table of Alex:

Purpose	**Attribute**	**Authorized users**
Purchase	name	Purchase-Product
Purchase	shipping-address	Purchase-Product
Purchase	credit-card info	Purchase-Product
Delivery	name	Shipping
Delivery	shipping-address	Shipping
Inquiry	name	Customer-Service
Inquiry	shipping-address	-
Inquiry	e-mail	Customer-Service
Statistical-Analysis	shipping-address	Analytics

Example

Customer-ID	Name	Shipping-address	e-mail	Credit-card info
A1	Alex	Washington	alex@gmail.com	7895-4578-4512-7896
A2	Jim	Texas	jim@hotmail.com	8974-6524-1452-3254
A3	Olly	Boston	olly@ymail.com	7896-2541-3256-2541
A4	Brian	New York	brain@gmail.com	2145-9874-6589-4785

Table: Table customer

Let subject **Shipping** execute the following query:
Select name,shipping-address from customer for purpose delivery;
Output:

Name	Shipping-Address
Alex	Washington

Example

Let role **Shipping** execute the following query:
Select e-mail from customer for purpose delivery;
Output: In privacy authorization table Alex has not given access to attribute email for purpose delivery. So he will not get access to the data.

Example of Policy Enforcement

Let us consider the following privacy authorization table of data owner Alex:

Purpose	**Attribute**	**Authorized users**
Purchase	name	Purchase-Product
Purchase	shipping-address	Purchase-Product
Purchase	credit-card info	Purchase-Product
Delivery	name	Shipping
Delivery	shipping-address	Shipping
Inquiry	name	Customer-Service
Inquiry	shipping-address	-
Inquiry	e-mail	Customer-Service
Statistical-Analysis	shipping-address	Analytics

Example of Policy Enforcement

Now we assume Alex wants to share his e-mail for purpose advertisement to sales department:
Create restriction r1 on **customer**
for user **sales**
to cells **e-mail**
for purpose **advertisement**
restricting access to **select**

Example of Policy Enforcement

Privacy authorization table of Alex after implementing the policy:

Purpose	**Attribute**	**Authorized users**
Purchase	name	Purchase-Product
Purchase	shipping-address	Purchase-Product
Purchase	credit-card info	Purchase-Product
Delivery	name	Shipping
Delivery	shipping-address	Shipping
Inquiry	name	Customer-Service
Inquiry	shipping-address	-
Inquiry	e-mail	Customer-Service
Statistical-Analysis	shipping-address	Analytics
Advertisement	e-mail	Sales

Detecting policy consistency

- Suppose there are 2 policies P1 and P2 that can be applied on a table.
- We want to determine whether the 2 policies are consistent or not.
- Here consistency means there should be access defined for a subject such that:
  1. After applying P1 the subject gets access to data elements X
  2. After applying P2 the subject gets access to data elements Y
  3. $X = Y$
- To show it we convert the model into equivalent RWFM model.

RWFM model

RWFM(Readers Writers Flow Model) can overcome the drawbacks of the previous implementations . Following are the basic points of RWFM model :

- Each role(user) is classified as a **subject**.
- Each unit of data is classified as an **object**.
- Each subject or object is defined by a triplet (owner,permissible readers, permissible writers or influencers).

    1. A subject is granted access rules on an object o iff $s \epsilon R(o) \wedge R(o) \supseteq R(s) \wedge W(o) \subseteq W(s)$

- As operations are restricted to only Select so we need only the reader-sets.

Converting policy in labels

So in RWFM model each of the data will have reader-writer label in them. As we consider only select operations here, so we need to keep only the reader sets of objects and subjects.

Name	**R_Name**	**Shipping-address**	**R_Shipping-address**	**e-mail**	**R_e-mail**
-	-	-	-	-	-

Table: Table customer

Converting policy in labels

Let us consider the following privacy authorization table of customer Alex:

Purpose	**Attribute**	**Authorized users**
Purchase	name	Purchase-Product
Purchase	shipping-address	Purchase-Product
Purchase	credit-card info	Purchase-Product
Delivery	name	Shipping
Delivery	shipping-address	Shipping
Inquiry	name	Customer-Service
Inquiry	shipping-address	-
Inquiry	e-mail	Customer-Service
Statistical-Analysis	shipping-address	Analytics

Converting policy into labels

Reader-set for attributes of customer Alex:

- **Name:** Alex
- **r-Name:** {purchase:Purchase-product}, {Inquiry:customer-service}
- **Shipping-address:** Washington
- **r-Shipping-address:** {purchase:Purchase-product}, {delivery:Shipping}, {statistical-analysis:Analytics}
- **E-mail:** alex@gmail.com
- **r-E-mail:** {inquiry:customer-service}
- **Credit-card info:** 7895-4578-4512-7896
- **r-Credit-card info:** {purchase:Purchase-Product}

Access

- Whenever a subject wants to access data in a column from the database, he needs to state the purpose.
- It is checked in the reader-set of the column whether the subject name is present or not for that particular purpose.
- If present then the subject gets access to the data, else not.

Consistency

- Similar to previous model purpose is checked first
- Similar to original model authorized subject for the attribute is then checked.
- Subjects have the purposes clubbed with the subject name which is same as defining purpose in original model to access data.

Converting policy into RWFM label

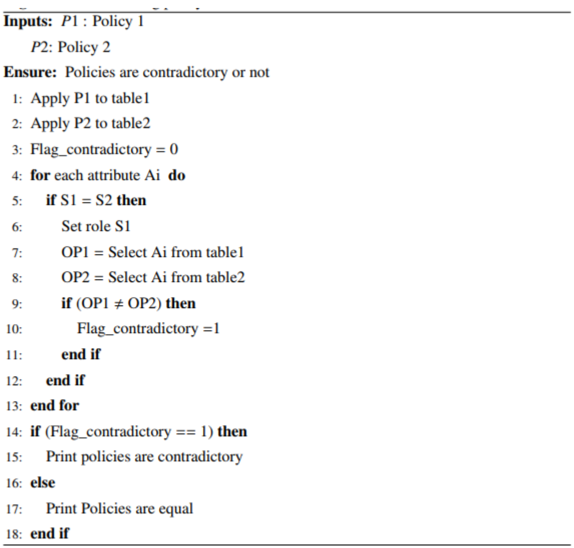
Policy structure:
Create restriction <**restriction-name**>
On <**table-name**>
For user <**subject-name**>
To cell <**attribute-name**>
For purpose <**purpose-name**>
Restricting access to <**access-type**> ;
String s=Policy
Array Arr = words of string S
Restriction-name=Arr[2]
Table-name=Arr[4]
Subject-name=Arr[7]
Attribute-name=Arr[10]
Purpose-name=Arr[13]
Access-type=Arr[17]

- Let us consider we want to apply a policy for customer Alex.
- Update <Table-name> set r_<Attribute-name> = r_<Attribute-name> $\|$ < Purpose-name > : <subject-name> where name = 'Alex';

Detecting Policy Contradictions

There can be a set of policies P1 and another set of policies P2 that can be applied to a database table. We want to find out whether P1 is same as P2 or different. If they are different then there is a policy contradiction. To determine so we need to have two identical tables with same set of data and same reader-set for each of the data in the table initially. Let the two tables be table1 and table2. Now apply P1 on table1 and P2 on table2 to check whether the policies are contradictory or not.

Detecting Policy Contradictions-implementation



```
Inputs: P1 : Policy 1
        P2: Policy 2
Ensure: Policies are contradictory or not
 1: Apply P1 to table1
 2: Apply P2 to table2
 3: Flag_contradictory = 0
 4: for each attribute Ai do
 5:     if S1 = S2 then
 6:         Set role S1
 7:         OP1 = Select Ai from table1
 8:         OP2 = Select Ai from table2
 9:         if (OP1 ≠ OP2) then
10:             Flag_contradictory =1
11:         end if
12:     end if
13: end for
14: if (Flag_contradictory == 1) then
15:     Print policies are contradictory
16: else
17:     Print Policies are equal
18: end if
```

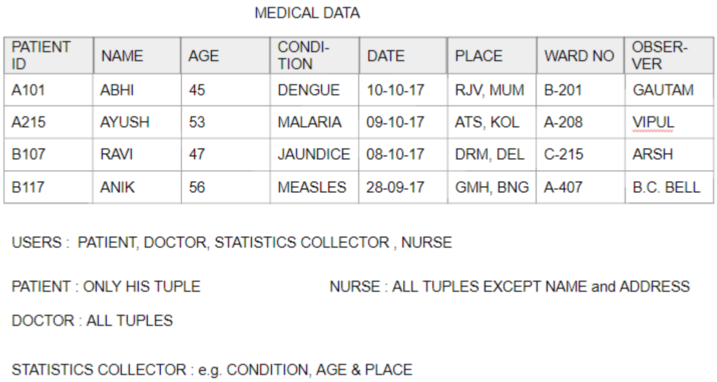
Security in Database System

- Uses Views for access control.
- Uses Grant/Revoke statements.

Implementation using views

- Database created by admin.
- Users for the database are created using SQL statement **CREATE ROLE** <**user-name**> **;**
- Initially no user except admin has access to the database.

Example using Medical Data



MEDICAL DATA

PATIENT ID	NAME	AGE	CONDI-TION	DATE	PLACE	WARD NO	OBSER-VER
A101	ABHI	45	DENGUE	10-10-17	RJV, MUM	B-201	GAUTAM
A215	AYUSH	53	MALARIA	09-10-17	ATS, KOL	A-208	VIPUL
B107	RAVI	47	JAUNDICE	08-10-17	DRM, DEL	C-215	ARSH
B117	ANIK	56	MEASLES	28-09-17	GMH, BNG	A-407	B.C. BELL

USERS : PATIENT, DOCTOR, STATISTICS COLLECTOR , NURSE

PATIENT : ONLY HIS TUPLE

NURSE : ALL TUPLES EXCEPT NAME and ADDRESS

DOCTOR : ALL TUPLES

STATISTICS COLLECTOR : e.g. CONDITION, AGE & PLACE

SQL Statements

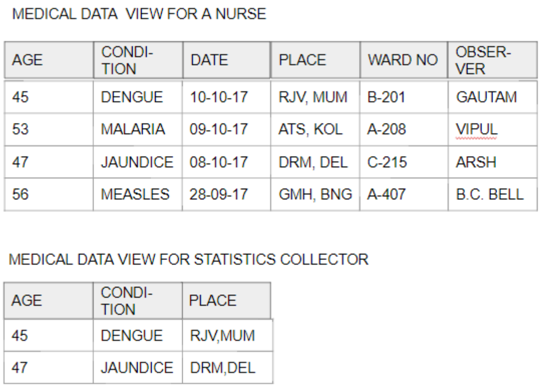
Let us define the Views for 2 users: Nurse and Statistics Collector.

- **Nurse :** CREATE VIEW view1 AS SELECT age, condition, data, place, ward no, observer FROM medical_data ;
- **Statistics Collector :** CREATE VIEW view2 AS SELECT age, condition, place FROM medical_data ;

Access granted to the users as :

- GRANT SELECT ON view1 TO nurse ;
- GRANT SELECT ON view2 TO statistics_collector ;

Output



MEDICAL DATA VIEW FOR A NURSE

AGE	CONDI-TION	DATE	PLACE	WARD NO	OBSER-VER
45	DENGUE	10-10-17	RJV, MUM	B-201	GAUTAM
53	MALARIA	09-10-17	ATS, KOL	A-208	VIPUL
47	JAUNDICE	08-10-17	DRM, DEL	C-215	ARSH
56	MEASLES	28-09-17	GMH, BNG	A-407	B.C. BELL

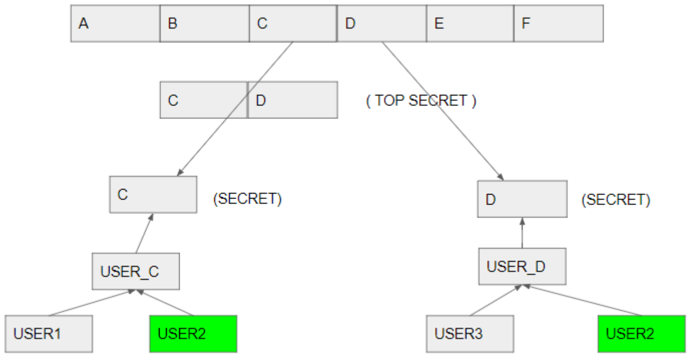
MEDICAL DATA VIEW FOR STATISTICS COLLECTOR

AGE	CONDI-TION	PLACE
45	DENGUE	RJV,MUM
47	JAUNDICE	DRM,DEL

Disadvantages

- For each user we need to create views. If number of users are very large than use of views is not feasible.
- If someone is able to bypass views, he gets direct access to the database.

Example of Indirect Inference



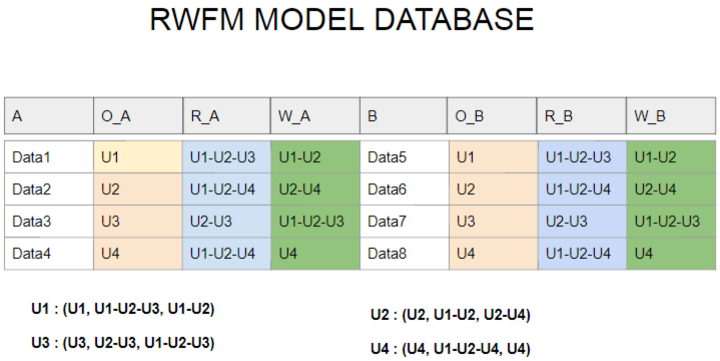
Here we can see that User2 can get access to higher level data by combining A and B which it is getting from 2 different users. It is undesirable.

RWFM Model

RWFM(Readers Writers Flow Model) model can overcome the drawbacks of the previous implementations . Following are the basic points of RWFM model :

- Each role(user) is classified as a **subject**.
- Each unit of data is classified as an **object**.
- Each subject or object is defined by a triplet (owner,permissible readers, permissible writers or influencers).
- Each object created by a subject has RWFM labels equal to that of the subject.
- Labels on the objects can also be upgraded and downgraded

Example Table



RWFM MODEL DATABASE

A	O_A	R_A	W_A	B	O_B	R_B	W_B
Data1	U1	U1-U2-U3	U1-U2	Data5	U1	U1-U2-U3	U1-U2
Data2	U2	U1-U2-U4	U2-U4	Data6	U2	U1-U2-U4	U2-U4
Data3	U3	U2-U3	U1-U2-U3	Data7	U3	U2-U3	U1-U2-U3
Data4	U4	U1-U2-U4	U4	Data8	U4	U1-U2-U4	U4

U1 : (U1, U1-U2-U3, U1-U2)

U2 : (U2, U1-U2, U2-U4)

U3 : (U3, U2-U3, U1-U2-U3)

U4 : (U4, U1-U2-U4, U4)

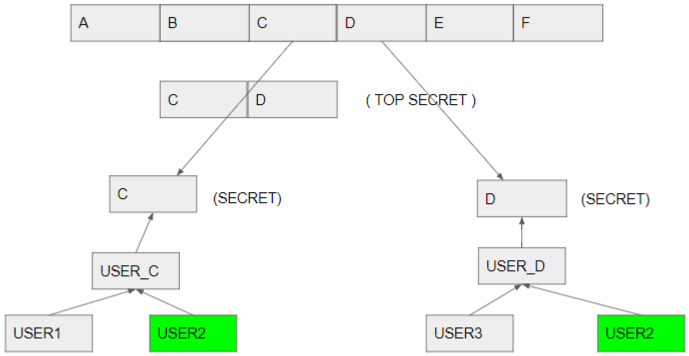
Access Rules

1. A subject is granted access rules on an object o iff
   s ϵ R(o) $\wedge$ R(o) $\supseteq$ R(s) $\wedge$ W(o) $\subseteq$ W(s)

According to the RWFM rule we can see that :

- User U3 can read Data1, Data5, Data3 and Data7.
- User U4 can update Data2, Data6, Data4, Data8.

Solution of the Problem



Lets define RWFM triplet for the users and data.

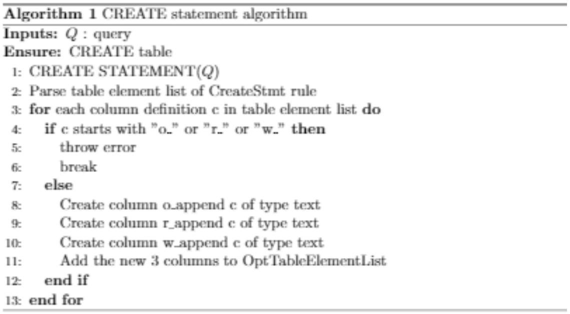
- Column C : (User0, USER_C - USER1, USER0)
- Column D : (User0, USER_D - USER1-USER3, USER0)
- user USER_C : (USER_C,USER_C - USER1-USER2 , USER_C)
- user USER_D : (USER_D,USER_D - USER1-USER2 , USER_D)

Using RWFM access rules we can see that USER_D cannot access attribute D as reader-set of USER_D is not a subset of reader-set of attribute D. Thus USER2 cannot get any data of attribute D from USER_D.

Implementation in PostgreSQL

- Each cell in a database is assigned a label (RWFM triplet).
- Read on each cell is done based on the labels.
- To implement RWFM model following changes are made in PostGreSQL source code :
  1. A table named main_table is created that has 4 attributes - user_id, o_attr, r_attr, w_attr. User _id attribute stores each of the subjects and rest 3 attributes stores the corresponding RWFM triplet for the subject. It means r_attr stores the reader set for subjects and w_attr stores the writer set.
  2. Change in **CREATE** Statement
  3. Change in **INSERT** statement
  4. Change in **SELECT** statement

Create Statement



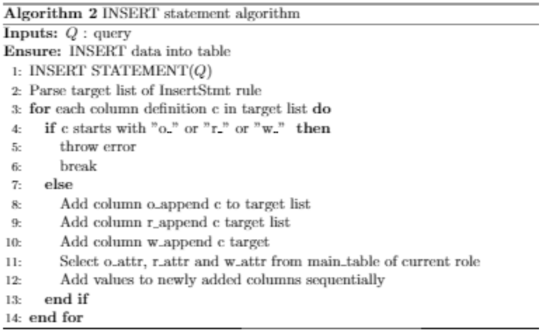
Algorithm 1 CREATE statement algorithm

Inputs: Q : query

Ensure: CREATE table

```
 1: CREATE STATEMENT(Q)
 2: Parse table element list of CreateStmt rule
 3: for each column definition c in table element list do
 4:    if c starts with "o_" or "r_" or "w_" then
 5:       throw error
 6:       break
 7:    else
 8:       Create column o_append c of type text
 9:       Create column r_append c of type text
10:       Create column w_append c of type text
11:       Add the new 3 columns to OptTableElementList
12:    end if
13: end for
```

Insert Statement



Algorithm 2 INSERT statement algorithm

Inputs: Q : query

Ensure: INSERT data into table

```
 1: INSERT STATEMENT($Q$)
 2: Parse target list of InsertStmt rule
 3: for each column definition c in target list do
 4:     if c starts with "o_" or "r_" or "w_"  then
 5:         throw error
 6:         break
 7:     else
 8:         Add column o_append c to target list
 9:         Add column r_append c target list
10:         Add column w_append c target
11:         Select o_attr, r_attr and w_attr from main_table of current role
12:         Add values to newly added columns sequentially
13:     end if
14: end for
```

Select Statement

Algorithm 3 SELECT statement algorithm

Inputs: Q : query

Ensure: desired data

```
SELECT STATEMENT(Q)
Parse table element list of SelectStmt rule
Add main_table to from_clause
for each column definition c in Select element list do
    if c starts with o_ or r_ or w_ then
        throw error
        break
    else
        Add column r_ append c to opt_target_list
        Add column w_ append c to opt_target_list
        Add condition r_ append c LIKE "current_user" to where_clause
    end if
end for
Add column r_attr of main_table to opt_target_list
Add column w_attr of main_ table to opt_target_list
Add condition o_attr LIKE "current_user" to where_clause
Flag=0
for each column definition c in Select element list do
    if r_attr ⊆ r_ append c and w_ append_ c ⊆ w_attr then
        continue
    else
        Flag=1
        break
    end if
end for
if Flag == 0 then
    print all columns
else
    Do nothing
    Flag=0
end if
```

HotCRP

- HotCRP is a widely used conference management system.
- It is used as a paper reviewing system and uses security policies to control information flow within the system.
- Authors add their papers and Reviewers review the papers and give score to them.
- The whole process is controlled by an administrator known as Program Chair.
- After reviews for all the papers are finalized Program Chair then declassifies the paper reviews to all reviewers and respective authors.

Rules in HotCRP

Main security policies in the system are as follows :

1. Authors cannot review papers i.e., Auth $\cap$ Rev $= \phi$
2. Program Chair cannot review any paper i.e, PC $\cap$ Rev $= \phi$
3. A paper P when added can be viewed by all reviewers and also by the Program-Chair.
4. Each paper also has an Assigned Set (AS). An assigned set is also a mapping from a paper P to a set of Reviewers Rev. P -> $2^{|\text{Rev}|}$ Only the reviewers present in the AS are allowed to review the paper.
5. A reviewer is allowed to access reviews of other reviewers for a particular paper only if the reviewer has submitted his own review.

Modelling HotCRP as RWFM

- Let the program chair be C.
- Let us consider two authors as A1 and A2. Let there be 2 reviewers as RR1 and RR2. So the total subjects in the system are
  {A1,A2,RR1,RR2,C}
- Let there be 2 papers P1 and P2 of authors A1 and A2 respectively.
- Let R11 be the review of reviewer RR1 for paper P1 and R12 be the review of reviewer RR2 of paper P1.
- Hence the set of objects in the system are { P1,P2,R11,R12 }

Modelling HotCRP as RWFM

Now we need to define RWFM labels for the objects and the subjects.

- For Paper P1 Author A1 is the author, hence A1 is the owner of P1. According to the HotCRP rule initially a paper can be read by the author, program chair and all the reviewers. So the reader set would be { A1,RR1,RR2,C }. As author A1 has written it so the writer set would be A1 itself. The RWFM labels for the paper P1 would be **({A1}, {A1,RR1,RR2,C}, {A1})**
- Similarly for P2 the RWFM labels would be: **({A2}, {A2,RR1,RR2,C}, {A2})**

Modelling HotCRP as RWFM

Now we need to define RWFM labels for the objects and the subjects.

- In HotCRP a review once submitted can be read only by the Program Chair first. So initially the reviewer is the owner of a review. As only the reviewer and program chair has access to the review the reader-set for a review would be the reviewer and program-chair. For the review author of the paper influences the review hence the author has to be in the writer set along with the reviewer. So RWFM labels for R11 would be:
  (RR1, {C,RR1}, {A1,RR1})
  RWFM labels for R12 would be:
  (RR2, {C,RR2}, {A1,RR2})
- For program chair, it can only read its own data. So in the reader-set only C would be present. Same for the writer set as well. So label for program chair would be
  (C, {C}, {C})

Modelling HotCRP as RWFM

There is a table Review-Decision owned by the program chair where all the reviewers submit their reviews for the papers. Structure of the Review-Decision table:

Paper	**Author**	**RR1**	**I-RR1**	**RR2**	**I-RR2**	**Decision**
-	-	-	-	-	-	-

Table: Table Review-Decision

Column decision is the label of the table that determines the access to the table.

Modelling HotCRP as RWFM

- Let us suppose first R11 is submitted by reviewer RR1 for paper P1 to program-chair C.
- So review of RR1 is added in the table review-Decision.
- Access to the review would be according to the RWFM rules.
- As program-chair accessed the review the label of R11 would be least upper bound of label of C and label of R11.
- This equals to **(C, $\{$C,RR1$\}$ $\cap$ $\{$C $\}$, $\{$A1,RR1$\}$ $\cup$ $\{$ C$\}$)** which equals to **(C, $\{$C$\}$, $\{$A1, RR1, C$\}$)**

Modelling HotCRP as RWFM

Review-Decision table:

Paper	**Author**	**RR1**	**I-RR1**	**RR2**	**I-RR2**	**Decision**
P1	A1	R11	({C}, {C}, {A1,RR1,C})	-	-	({C}, {C}, {A1,RR1,C})
.	.	.		.	.	

Table: Table Review-Decision

Modelling HotCRP as RWFM

- As RR1 has submitted his review he can get access to his review.
- So Program-Chair C declassifies the review R11 to RR2.
- According to RWFM a subject present in the writer set of a label can be added to the reader-set of the label so that the subject can get access to the data.
- As RR1 is present in the writer set of label of decision he can be added to the reader-set of decision label.
- After adding to the reader-set RR1 can now get access to his review.

Modelling HotCRP as RWFM

Review-Decision table:

Paper	**Author**	**RR1**	**I-RR1**	**RR2**	**I-RR2**	**Decision**
P1	A1	R11	({C}, {C}, {A1,RR1,C})	-	-	({C}, {C,RR1}, {A1,RR1,C})
.	.	.		.	.	

Table: Table Review-Decision

Modelling HotCRP as RWFM

- Here we see that RR2 is not in the reader-set of decision label currently.
- It is because he has not submitted his review which satisfies the condition of HotCRP.
- So now RR2 submits his review R12 for paper P1.
- label for RR2 is updated in review decision table as least upper bound of $\mathbf{(C, \{C,RR1\} \cap \{C\}, \{A1,RR1\} \cup \{C\})}$ which equals to $\mathbf{(C, \{C\}, \{A1, RR1, C\})}$
- Corresponding Decision label is updates as lub of its previous label and the label of RR2 in the table.

Modelling HotCRP as RWFM

Paper	Auth	RR1	I-RR1	RR2	I-RR2	Decision
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C}
-	-	-	{A1, RR1, C})	-	{A1, RR2,C})	{A1, RR1, RR2,C})

Table: Table Review-Decision

Modelling HotCRP as RWFM

- As RR2 has now submitted the review R12, now program chair can declassify both the review to both the reviews.
- Also Program-Chair can now declassify the review to author A1 as both the reviews are submitted.

Paper	**Auth**	**RR1**	**I-RR1**	**RR2**	**I-RR2**	**Decision**
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C,A1,RR1,RR2}
-	-	-	{A1, RR1, C})	-	{A1, RR2,C})	{A1, RR1, RR2,C})

Table: Table Review-Decision

Modelling HotCRP as RWFM

- Let P2 be now submitted by author A2.
- label for P2 would be **(A2, {A2,RR1,RR2,C} , {A2})**
- Then RR1 and RR2 submit their reviews as R21 and R22. Decision label will perform lub operation with the new labels.
- Similar to R11 label of R21 would be **(RR1, {C,RR1} , {A2,RR1})**
- Finally we will get the following table.

Paper	**Auth**	**RR1**	**I-RR1**	**RR2**	**I-RR2**	**Decision**
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C}
-	-	-	{A1, RR1, C})	-	{A1, RR2,C})	{A1, RR1, RR2,C})
P2	A2	R21	({C}, {C},	R22	({C}, {C},	({C}, {C}
-	-	-	{A2, RR1, C})	-	{A2, RR2,C})	{A1,A2, RR1, RR2,C})

Table: Table Review-Decision

Modelling HotCRP as RWFM

- Program-Chair can now declassify the reviews to RR1, RR2 and A2.

Paper	Auth	RR1	I-RR1	RR2	I-RR2	Decision
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C}
-	-	-	{A1, RR1, C})	-	{A1, RR2,C})	{A1, RR1, RR2,C})
P2	A2	R21	({C}, {C},	R22	({C}, {C},	({C}, {C,RR1,RR2,A2}
-	-	-	{A2, RR1, C})	-	{A2, RR2,C})	{A1,A2, RR1, RR2,C})

Table: Table Review-Decision

Conclusion

we focused on 3 things:

- Detecting whether 2 policies are contradictory in Hippocratic Databases.
- Comparison of classical databases and database implementing RWFM.
- Modelling HotCRP as RWFM.

Future Work

- Current Implementation of postgreSQL with RWFM deals with the basic SQL queries like CREATE, INSERT, DELETE, UPDATE, SELECT and VIEW. There are other functionality in SQL as transaction, concurrency ,etc. where security needs to be implemented.
- Also functionality to handle triggers must be included as triggers play a vital role in database management.

Bibliography I

Rakesh Agrawal, Paul Bird, Tyrone Grandison, Jerry Kiernan, Scott Logan, Walid Rjaibi.
Extending Relational Database Systems to Automatically Enforce Privacy Policies.
21st International Conference on Data Engineering (ICDE'05)

Dorothy E Denning, Selim G. Akl, Mark Heckman, Teresa F. Lunt, Metthew Morgenstern, Peter G. Neumann and Roger R. Schell.
Views for Multilevel Database Security.
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-13, NO. 2, FEBRUARY 1987

Bibliography II

N.V.Narendra Kumar and R.K.Shyamasundar .
Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels.
Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on 3-5 Dec, 2014

THANK YOU