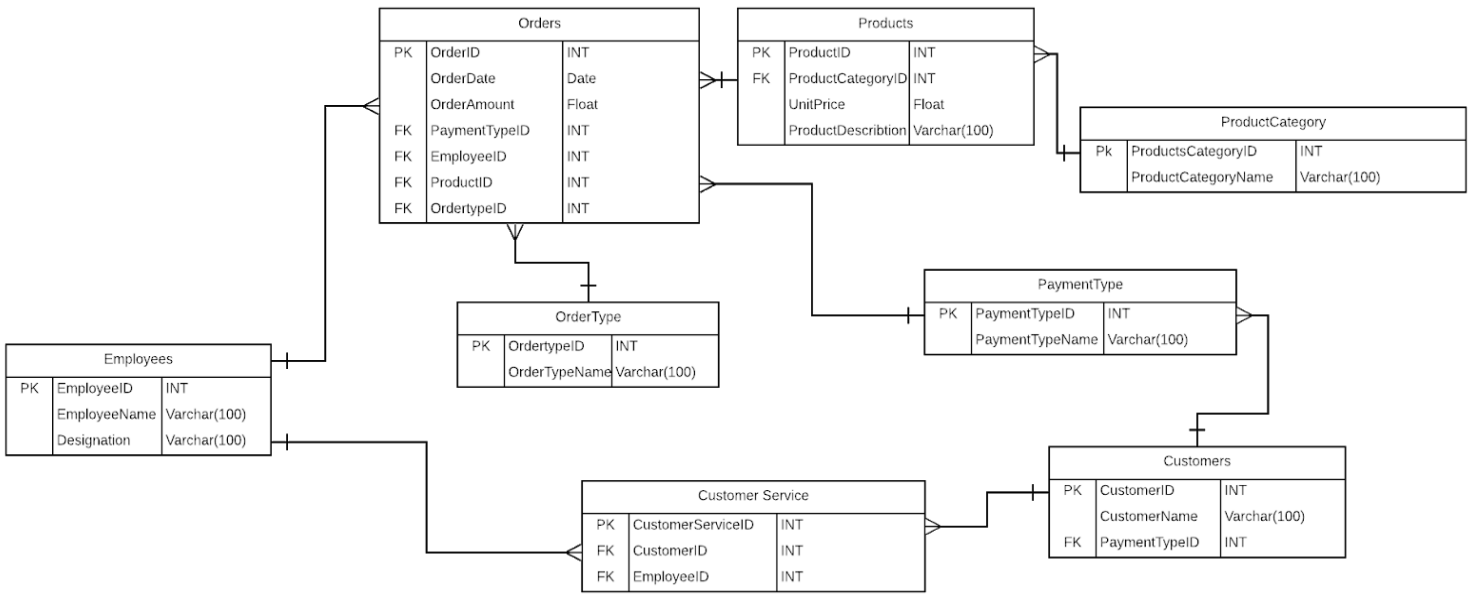
Problem

The database typically contains the crown jewels of any environment; it usually holds the most business sensitive information which is why it is a high priority target for an attacker. The purpose of the study is to understand the problems with the file-based database which are highly complex, tends to be inherently restrictive and have reached the point where a complete upgrade in the system becomes necessary. Understanding the relational database system in the security-based firm and analyzing the best type of relational database as the requirement of the business changed.

Solution: Such problems can be tackled with the help of RDBMS.

ERD DIAGRAM FOR Relational Database



CREATING TABLES

Input

#CREATE TABLE fp_Employees (EmployeeID int Primary key,EmployeeName varchar (100), Designation varchar (100));

Output:

fp_Employees	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
EmployeeID	INT	PK	NOT NULL		
EmployeeName	Varchar(100)				
Designation	Varchar(100)				

Input

#CREATE TABLE fp_Customers (CustomerID int Primary Key, CustomerName varchar(100), PaymentTypeID int, FOREIGN key (PaymentTypeID) REFERENCES fp_PaymentType (PaymentTypeID));

Output

fp_Customers	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
CustomerID	INT	PK	NOT NULL	Customer	
CustomerName	Varchar(100)				
PaymentTypeID	INT	FK	NOT NULL		fp_PaymentType

Input

#CREATE TABLE fp_ProductCategory (ProductCategoryID int Primary key, ProductCateggoryName varchar(100));

Output

fp_ProductCategory	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES

ProductCategoryID	INT	PK	NOT NULL		
ProductCategoryName	Varchar(100)				

Input

#CREATE TABLE fp_Orders (OrderID int Primary key, OrderDate Date, OrderAmount Float, PaymentTypeID int, FOREIGN key (PaymentTypeID) REFERENCES fp_PaymentType (PaymentTypeID), EmployeeID int, FOREIGN key (EmployeeID) REFERENCES fp_Employees (EmployeeID), ProductID int, FOREIGN key (ProductID) REFERENCEs fp_Products (ProductID), OrdertypeID int, FOREIGN key (OrdertypeID) REFERENCES fp_OrderType (OrdertypeID));

Output

fp_Orders	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
OrderID	INT	PK	NOT NULL		
OrderDate	Date				
OrderAmount	Float				
PaymentTypeID	INT	FK	NOT NULL		fp_PaymentType
ProductID	INT	FK	NOT NULL		fp_Products
OrderTypeID	INT	FK	NOT NULL		fp_OrderType
EmployeeID	INT	FK	NOT NULL		fp_Employees

Input

#CREATE TABLE fp_Products (ProductID int Primary key, ProductCategoryID int, FOREIGN KEY (ProductCategoryID) REFERENCES fp_ProductCategory(ProductCategoryID), Unitprice Float, ProductDescription varchar(200));

Output

fp_Products	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
ProductID	INT	PK	NOT NULL		
ProductCategoryID	INT	FK	NOT NULL		

UnitPrice	Float				
ProductDescribtion	Varchar(100)				

Input

#CREATE TABLE fp_OrderType (OrdertypeID int Primary key, OrderTypeName varchar (100));

Output

fp_OrderType	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
OrdertypeID	INT	PK	NOT NULL		
OrdertypeName	Varchar(100)				

Input

#CREATE TABLE fp_PaymentType (PaymentTypeID int Primary key, PaymentTypeName varchar(100));

Output

fp_PaymentType	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
PaymentTypeID	INT	PK	NOT NULL		
PaymentTypeName	VARCHAR(100)				

Input

#CREATE TABLE fp_CustomerService(CustomerServiceID int Primary Key, CustomerID int, FOREIGN key (CustomerID) REFERENCES fp_Customers (CustomerID), EmployeeID int, FOREIGN key (EmployeeID) REFERENCES fp_Employees (EmployeeID));

Output

fp_CustomerService	TYPE	KEY	NULL	DEFAULT VALUE	REFERENCES
CustomerServiceID	INT	PK	NOT NULL		
CustomerID	INT	FK	NOT NULL		fb_Customers

EmployeeID	INT	FK	NOT NULL		fb_Employees

INSERTING DATA

Input

#INSERT INTO fp_Employees VALUES (1,'Austin S.',1), (2,'Alex R.',1), (3,'Abbot H.',1), (4,'Aby A.',1), (5,'Amanda R.',1), (6,'Brad S.',2), (7,'Becky R.',2), (8,'Britney H.',2), (9,'Bill A.',2), (10,'Brianna R.',2), (11,'Chad S.',3), (12,'Caitlin R.',3), (13,'Cody H.',3), (14,'Courtney A.',3), (15,'Cate R.',3), (16,'Doug T.',4), (17,'Diana R.',4), (18,'Daily H.',4), (19,'David D.',4), (20,'Daisy R.',4);

Output

EmployeeID	EmployeeName	Designation
1	Austin S.	1
2	Alex R.	1
3	Abbot H.	1
4	Aby A.	1
5	Amanda R.	1
6	Brad S.	2
7	Becky R.	2
8	Britney H.	2
9	Bill A.	2
10	Brianna R.	2
11	Chad S.	3
12	Caitlin R.	3
13	Cody H.	3
14	Courtney A.	3
15	Cate R.	3
16	Doug T.	4
17	Diana R.	4
18	Daily H.	4
19	David D.	4
20	Daisy R.	4

Input

#INSERT INTO fp_ProductCategory VALUES (1,'Application security'), (2,'Anti Malware'), (3,'Cloud security'), (4,'Encryption');

Output:

ProductCategoryID	ProductCateggoryName
1	Application security
2	Anti Malware
3	Cloud security
4	Encryption

Input

#INSERT INTO fp_Products VALUES (1,1,1200, 'BlackstoneOne'), (2,1,1500, 'Contrast Security'), (3,1,1000, 'Cryptanium'), (4,1,1200, 'ThreadFix'), (5,1,1500, 'Waratek'), (6,2,100, 'Cyren WebSecurity'), (7,2,150, 'PSafe Total'), (8,2,120, 'Zemana AntiMalware'), (9,2,100, 'SiteLock INFINITY'), (10,2,120, 'Strongarm'), (11,3,150, 'Armor Anywhere'), (12,3,120, 'CipherCloud'), (13,3,100, 'GuardiCore'), (14,3,80, 'Dome9 Arc'), (15,3,100, 'Threat Stack'), (16,4,200, 'CipherCloud Active Encryption'), (17,4,180, 'CryptoMove'), (18,4,160, 'SafeNet KeySecure'), (19,4,180, 'SafeNet ProtectV'), (20,4,200, 'CipherCloud Active Encryption');

Output

ProductID	ProductCategoryID	Unitprice	ProductDescription
1	1	1200	BlackstoneOne
2	1	1500	Contrast Security
3	1	1000	Cryptanium
4	1	1200	ThreadFix
5	1	1500	Waratek
6	2	100	Cyren WebSecurity
7	2	150	PSafe Total
8	2	120	Zemana AntiMalware
9	2	100	SiteLock INFINITY
10	2	120	Strongarm
11	3	150	Armor Anywhere
12	3	120	CipherCloud
13	3	100	GuardiCore
14	3	80	Dome9 Arc
15	3	100	Threat Stack
16	4	200	CipherCloud Active Encryption
17	4	180	CryptoMove
18	4	160	SafeNet KeySecure
19	4	180	SafeNet ProtectV
20	4	200	CipherCloud Active Encryption

Input

#INSERT INTO fp_PaymentType VALUES (1,'Credit card'), (2,'Debit card'), (3,'pay pal');

Output

PaymentTypeID	PaymentTypeName
1	Credit card
2	Debit card
3	pay pal

Input

#INSERT INTO fp_Customers VALUES (1, 'Susan B.',1), (2, 'Cynthia C.',1), (3, 'Fiona B.',1), (4, 'Frank D.',1), (5, 'Gary B.',1), (6, 'Henna F.',1), (7, 'Henry T.',1), (8, 'Kristy B.',2), (9, 'Kayla D.',2), (10, 'Karen H.',2), (11, 'Kyle B.',2), (12, 'Koby T.',2), (13, 'Kim B.',2), (14, 'Lauren T.',2), (15, 'Lip F.',3), (16, 'Lary T.',3), (17, 'Lily K.',3), (18, 'Lisa G.',3), (19, 'Lokie F.',3), (20, 'Flokie B.',3);

Output

CustomerID	CustomerName	PaymentTypeID
1	Susan B.	1
2	Cynthia C.	1
3	Fiona B.	1
4	Frank D.	1
5	Gary B.	1
6	Henna F.	1
7	Henry T.	1
8	Kristy B.	2
9	Kayla D.	2
10	Karen H.	2
11	Kyle B.	2
12	Koby T.	2
13	Kim B.	2
14	Lauren T.	2
15	Lip F.	3
16	Lary T.	3
17	Lily K.	3
18	Lisa G.	3
19	Lokie F.	3
20	Flokie B.	3

Input

#INSERT INTO fp_CustomerService VALUES (1,1,1), (2,2,1), (3,3,2), (4,4,2), (5,5,3), (6,6,3), (7,7,4), (8,8,4), (9,9,5), (10,10,5), (11,11,6), (12,12,6), (13,13,7), (14,14,7), (15,15,8), (16,16,8), (17,17,9), (18,18,9), (19,19,10), (20,20,10), (21,1,11), (22,2,11), (23,3,12), (24,4,12), (25,5,13), (26,6,13), (27,7,14), (28,8,14), (29,9,15), (30,10,15), (31,11,16), (32,12,16), (33,13,17), (34,14,17), (35,15,18), (36,16,18), (37,17,19), (38,18,19), (39,19,20), (40,20,20);

Output

CustomerServiceID	CustomerID	EmployeeID
1	1	1
2	2	1
3	3	2
4	4	2
5	5	3
6	6	3
7	7	4
8	8	4
9	9	5
10	10	5
11	11	6
12	12	6
13	13	7
14	14	7
15	15	8
16	16	8
17	17	9
18	18	9
19	19	10
20	20	10

CustomerServiceID	CustomerID	EmployeeID
21	1	11
22	2	11
23	3	12
24	4	12
25	5	13
26	6	13
27	7	14
28	8	14
29	9	15
30	10	15
31	11	16
32	12	16
33	13	17
34	14	17
35	15	18
36	16	18
37	17	19
38	18	19
39	19	20
40	20	20

Input

#INSERT INTO fp_OrderType VALUES (1,'Rent'), (2,'Buy');

Output

OrdertypeID	OrderTypeName
1	Rent
2	Buy

Input

INSERT INTO fp_Orders VALUES (1, '19/02/02',2,1,1,2,1), (2, '19/02/03',1,1,2,1,1), (3, '19/02/04',2,1,3,4,2), (4, '19/02/05',3,2,4,3,1), (5, '19/02/06',2,2,5,1,1), (6, '19/02/07',3,1,6,6,2), (7, '19/08/02',1,1,7,5,1), (8, '19/02/09',2,1,8,7,1), (9, '19/02/10',3,3,9,8,1), (10, '19/02/11',2,2,10,9,1), (11, '19/02/12',4,3,11,10,2), (12, '18/03/01',3,1,12,11,1), (13, '18/03/02',2,1,13,13,1), (14, '18/03/03',1,2,14,12,2), (15, '18/03/04',2,3,15,14,1), (16, '18/03/05',3,1,16,13,1), (17, '18/02/06',3,1,17,16,1), (18, '18/02/01',2,2,18,15,2), (19, '18/02/02',2,3,19,18,1), (20, '18/02/03',3,2,20,17,1), (21, '18/02/04',2,2,1,16,1), (22, '18/02/05',2,1,2,1,1), (23, '18/02/06',1,1,3,2,2), (24, '18/02/07',1,1,4,3,1), (25, '18/02/08',2,2,5,4,1), (26, '18/02/09',2,1,6,5,1), (27, '18/02/10',1,3,7,6,1), (28, '8/02/11',3,1,8,7,1), (29, '18/02/12',3,3,9,8,2), (30, '19/03/02',4,2,10,9,1), (31, '19/03/03',4,1,11,10,1), (32, '19/03/04',2,3,12,11,1), (33, '19/03/05',2,2,13,12,1), (34, '19/03/06',1,3,14,13,2), (35, '19/03/07',1,2,15,14,2), (36, '19/03/08',2,2,16,15,1), (37, '19/03/09',2,3,17,16,1), (38, '19/03/10',4,1,18,17,2), (39, '19/03/11',4,2,19,20,1), (40, '19/03/12',3,3,20,1,2);

Output

OrderID	OrderDate	OrderAmount	PaymentTypeID	EmployeeID	ProductID	OrdertypeID
1	2019-02-02	2	1	1	2	1
2	2019-02-03	1	1	2	1	1
3	2019-02-04	2	1	3	4	2
4	2019-02-05	3	2	4	3	1
5	2019-02-06	2	2	5	1	1
6	2019-02-07	3	1	6	6	2
7	2019-08-02	1	1	7	5	1
8	2019-02-09	2	1	8	7	1
9	2019-02-10	3	3	9	8	1
10	2019-02-11	2	2	10	9	1
11	2019-02-12	4	3	11	10	2
12	2018-03-01	3	1	12	11	1
13	2018-03-02	2	1	13	13	1
14	2018-03-03	1	2	14	12	2
15	2018-03-04	2	3	15	14	1
16	2018-03-05	3	1	16	13	1
17	2018-02-06	3	1	17	16	1
18	2018-02-01	2	2	18	15	2
19	2018-02-02	2	3	19	18	1
20	2018-02-03	3	2	20	17	1

OrderID	OrderDate	OrderAmount	PaymentTypeID	EmployeeID	ProductID	OrdertypeID
21	2018-02-04	2	2	1	16	1
22	2018-02-05	2	1	2	1	1
23	2018-02-06	1	1	3	2	2
24	2018-02-07	1	1	4	3	1
25	2018-02-08	2	2	5	4	1
26	2018-02-09	2	1	6	5	1
27	2018-02-10	1	3	7	6	1
28	2018-02-11	3	1	8	7	1
29	2018-02-12	3	3	9	8	2
30	2019-03-02	4	2	10	9	1
31	2019-03-03	4	1	11	10	1
32	2019-03-04	2	3	12	11	1
33	2019-03-05	2	2	13	12	1
34	2019-03-06	1	3	14	13	2
35	2019-03-07	1	2	15	14	2
36	2019-03-08	2	2	16	15	1
37	2019-03-09	2	3	17	16	1
38	2019-03-10	4	1	18	17	2
39	2019-03-11	4	2	19	20	1
40	2019-03-12	3	3	20	1	2

ANALYSING THE DATA

TOP EMPLOYEES BY SALES (orderamount)

INPUT

```
SELECT
fp_Employees.EmployeeID,
fp_Employees.EmployeeName,
SUM( fp_Orders.OrderAmount )
FROM
fp_Orders
JOIN fp_Employees ON fp_Employees.EmployeeID = fp_Orders.EmployeeID
```

```
GROUP BY
EmployeeID,
EmployeeName
ORDER BY
SUM(
(
fp_Orders.OrderAmount )
)DESC
```

OUTPUT

EmployeeID	EmployeeName	SUM(fp_Orders.OrderAmount)
11	Chad S.	8
19	David D.	6
9	Bill A.	6
20	Daisy R.	6
18	Daily H.	6
10	Brianna R.	6
17	Diana R.	5
16	Doug T.	5
12	Caitlin R.	5
8	Britney H.	5
6	Brad S.	5
13	Cody H.	4
5	Amanda R.	4
1	Austin S.	4
4	Aby A.	4
2	Alex R.	3
15	Cate R.	3
3	Abbot H.	3
7	Becky R.	2
14	Courtney A.	2

TOP EMPLOYEES WHO SOLD APPLICATION SECURITY

INPUT

```
SELECT
fp_Employees.EmployeeID,
fp_Employees.EmployeeName,
SUM( fp_Orders.OrderAmount ),
fp_Products.ProductID,
fp_ProductCategory.ProductCategoryID,
fp_ProductCategory.ProductCateggoryName
 FROM
fp_Orders
JOIN fp_Employees ON fp_Employees.EmployeeID = fp_Orders.EmployeeID
JOIN fp_Products ON fp_Orders.ProductID = fp_Products.ProductID
JOIN fp_ProductCategory ON fp_Products.ProductCategoryID =
fp_ProductCategory.ProductCategoryID
 WHERe
fp_ProductCategory.ProductCategoryID = 1
 GROUP BY
EmployeeID,
EmployeeName,
ProductID,
ProductCateggoryName
 ORDER BY
SUM(
(
fp_Orders.OrderAmount )
)DESC
```

OUTPUT

+ Options

EmployeeID	EmployeeName	SUM(fp_Orders.OrderAmount)	ProductID	ProductCategoryID	ProductCateggoryName
4	Aby A.	4	3	1	Application security
20	Daisy R.	3	1	1	Application security
2	Alex R.	3	1	1	Application security
3	Abbot H.	2	4	1	Application security
6	Brad S.	2	5	1	Application security
5	Amanda R.	2	4	1	Application security
1	Austin S.	2	2	1	Application security
5	Amanda R.	2	1	1	Application security
3	Abbot H.	1	2	1	Application security
7	Becky R.	1	5	1	Application security

TOP EMPLOYEES WHO SOLD ANTI-MALWARE

INPUT

```
SELECT
fp_Employees.EmployeeID,
fp_Employees.EmployeeName,
SUM( fp_Orders.OrderAmount ),
fp_Products.ProductID,
fp_ProductCategory.ProductCategoryID,
fp_ProductCategory.ProductCateggoryName

FROM
fp_Orders
JOIN fp_Employees ON fp_Employees.EmployeeID = fp_Orders.EmployeeID
JOIN fp_Products ON fp_Orders.ProductID = fp_Products.ProductID
JOIN fp_ProductCategory ON fp_Products.ProductCategoryID =
fp_ProductCategory.ProductCategoryID

WHERE
fp_ProductCategory.ProductCategoryID = 2

GROUP BY
EmployeeID,
EmployeeName,
ProductID,
ProductCateggoryName
```

```
ORDER BY
SUM(
(
fp_Orders.OrderAmount )
)DESC
```

OUTPUT

+ Options

EmployeeID	EmployeeName	SUM(fp_Orders.OrderAmount)	ProductID	ProductCategoryID	ProductCateggoryName
11	Chad S.	8	10	2	Anti Malware
10	Brianna R.	6	9	2	Anti Malware
9	Bill A.	6	8	2	Anti Malware
8	Britney H.	5	7	2	Anti Malware
6	Brad S.	3	6	2	Anti Malware
7	Becky R.	1	6	2	Anti Malware

TOP EMPLOYEES WHO SOLD CLOUD SECURITY

INPUT

```
SELECT
fp_Employees.EmployeeID,
fp_Employees.EmployeeName,
SUM( fp_Orders.OrderAmount ),
fp_Products.ProductID,
fp_ProductCategory.ProductCategoryID,
fp_ProductCategory.ProductCateggoryName

FROM
fp_Orders
JOIN fp_Employees ON fp_Employees.EmployeeID = fp_Orders.EmployeeID
JOIN fp_Products ON fp_Orders.ProductID = fp_Products.ProductID
JOIN fp_ProductCategory ON fp_Products.ProductCategoryID =
fp_ProductCategory.ProductCategoryID

WHERe
fp_ProductCategory.ProductCategoryID = 3
```

```
GROUP BY
EmployeeID,
EmployeeName,
ProductID,
ProductCateggoryName


ORDER BY
SUM(
(
fp_Orders.OrderAmount )
)DESC
```

OUTPUT

+ Options

EmployeeID	EmployeeName	SUM(fp_Orders.OrderAmount)	ProductID	ProductCategoryID	ProductCateggoryName
12	Caitlin R.	5	11	3	Cloud security
15	Cate R.	3	14	3	Cloud security
16	Doug T.	3	13	3	Cloud security
18	Daily H.	2	15	3	Cloud security
13	Cody H.	2	13	3	Cloud security
16	Doug T.	2	15	3	Cloud security
13	Cody H.	2	12	3	Cloud security
14	Courtney A.	1	13	3	Cloud security
14	Courtney A.	1	12	3	Cloud security

TOP EMPLOYEES WHO SOLD ENCRYPTION

INPUT

```
SELECT
fp_Employees.EmployeeID,
fp_Employees.EmployeeName,
SUM( fp_Orders.OrderAmount ),
fp_Products.ProductID,
fp_ProductCategory.ProductCategoryID,
fp_ProductCategory.ProductCateggoryName

FROM
fp_Orders
```

```
JOIN fp_Employees ON fp_Employees.EmployeeID = fp_Orders.EmployeeID
JOIN fp_Products ON fp_Orders.ProductID = fp_Products.ProductID
JOIN fp_ProductCategory ON fp_Products.ProductCategoryID =
fp_ProductCategory.ProductCategoryID

WHERe
fp_ProductCategory.ProductCategoryID = 4

GROUP BY
EmployeeID,
EmployeeName,
ProductID,
ProductCateggoryName


ORDER BY
SUM(
(
fp_Orders.OrderAmount )
)DESC
```

OUTPUT

+ Options

EmployeeID	EmployeeName	SUM(fp_Orders.OrderAmount)	ProductID	ProductCategoryID	ProductCateggoryName
17	Diana R.	5	16	4	Encryption
19	David D.	4	20	4	Encryption
18	Daily H.	4	17	4	Encryption
20	Daisy R.	3	17	4	Encryption
19	David D.	2	18	4	Encryption
1	Austin S.	2	16	4	Encryption

Sales by Product Category

INPUT

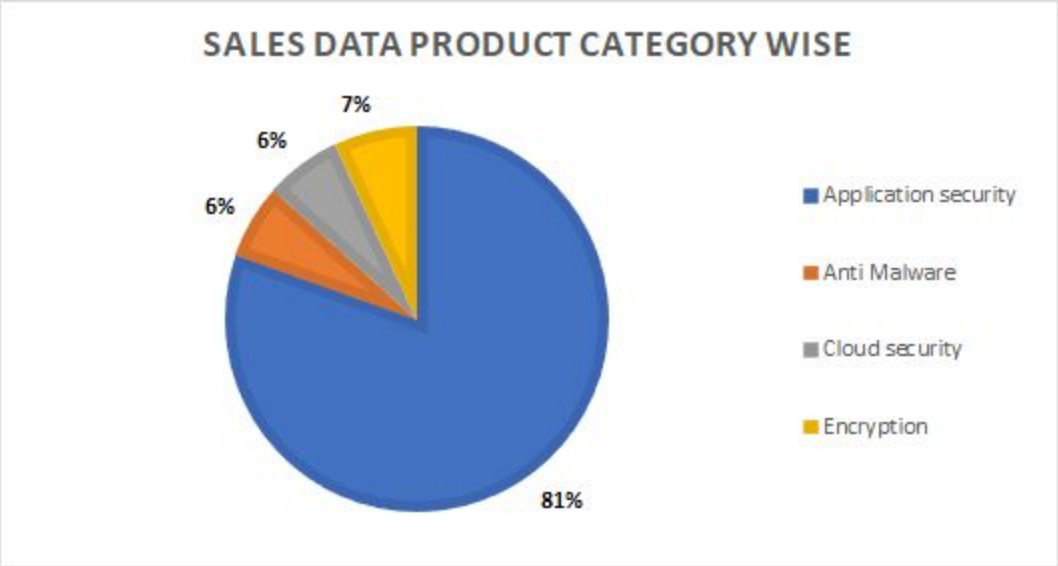
```
SELECT
SUM( fp_Orders.OrderAmount * fp_Products.Unitprice),
```

```
fp_ProductCategory.ProductCategoryID,
fp_ProductCategory.ProductCateggoryName

FROM
fp_Orders
JOIN fp_Products ON fp_Orders.ProductID = fp_Products.ProductID
JOIN fp_ProductCategory ON fp_Products.ProductCategoryID =
fp_ProductCategory.ProductCategoryID


GROUP BY
ProductCategoryID,
ProductCateggoryName


ORDER BY
SUM(
   ( fp_Orders.OrderAmount * fp_Products.Unitprice)
)DESC
```

OUTPUT

+ Options

SUM(fp_Orders.OrderAmount * fp_Products.Unitprice)	ProductCategoryID	ProductCateggoryName
27400	1	Application security
3780	4	Encryption
3430	2	Anti Malware
2350	3	Cloud security



CHOICE OF ORDER TYPE BY CUSTOMERS

INPUT

```
SELECT
fp_OrderType.OrdertypeID,
fp_OrderType.OrderTypeName,
SUM( fp_Orders.OrderAmount )

FROM
fp_Orders
JOIN fp_OrderType ON fp_OrderType.OrdertypeID = fp_Orders.OrdertypeID

GROUP BY
OrdertypeID,
OrderTypeName

ORDER BY
SUM(
(
fp_Orders.OrderAmount )
)DESC
```

OUTPUT

+ Options

OrdertypeID	OrderTypeName	SUM(fp_Orders.OrderAmount)
1	Rent	67
2	Buy	25

PRODUCT SALES DATA

INPUT

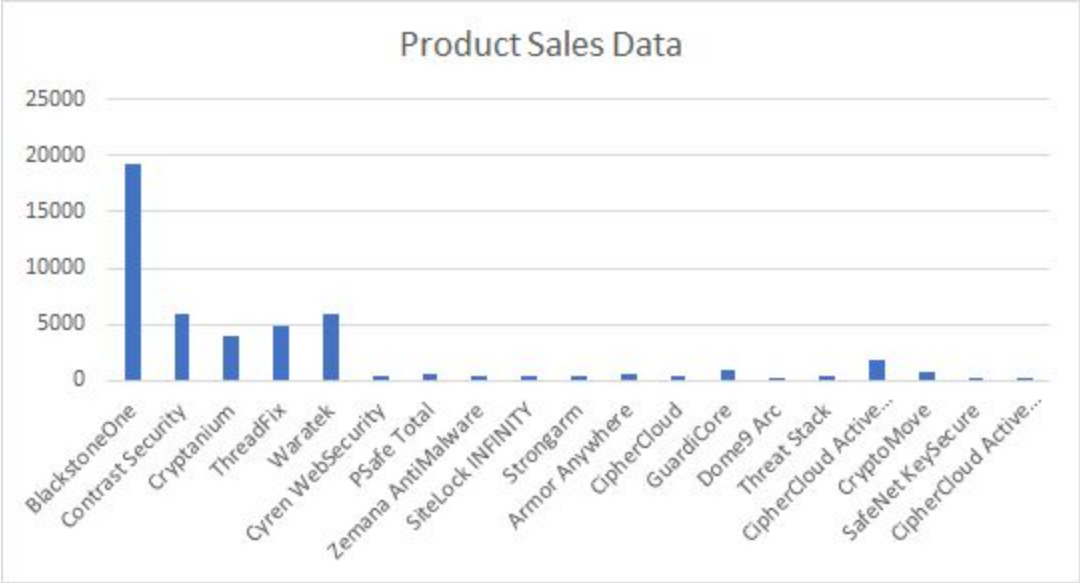
```
SELECT
fp_Products.ProductID,
fp_Products.ProductDescription,
SUM( fp_Orders.OrderAmount * fp_Products.Unitprice )
```

```
FROM
fp_Orders
JOIN fp_Products ON fp_Products.ProductID = fp_Orders.ProductID

GROUP BY
ProductID,
ProductDescription

ORDER BY
SUM(
(fp_Orders.OrderAmount * fp_Products.Unitprice
)
)DESC
```

OUTPUT

+ Options

ProductID	ProductDescription	SUM(fp_Orders.OrderAmount * fp_Products.Unitprice)
1	BlackstoneOne	9600
4	ThreadFix	4800
5	Waratek	4500
2	Contrast Security	4500
3	Cryptanium	4000
16	CipherCloud Active Encryption	1400
17	CryptoMove	1260
10	Strongarm	960
20	CipherCloud Active Encryption	800
7	PSafe Total	750
11	Armor Anywhere	750
8	Zemana AntiMalware	720
13	GuardiCore	600
9	SiteLock INFINITY	600
6	Cyren WebSecurity	400
15	Threat Stack	400
12	CipherCloud	360
18	SafeNet KeySecure	320
14	Dome9 Arc	240


Product Sales Data
25000
20000
15000
10000
5000
0
BlackstoneOne
Contrast Security
Cryptanium
ThreadFix
Waratek
Cyren WebSecurity
PSafe Total
Zemana AntiMalware
SiteLock INFINITY
Strongarm
Armor Anywhere
CipherCloud
GuardiCore
Dome9 Arc
Threat Stack
CipherCloud Active...
CryptoMove
SafeNet KeySecure
CipherCloud Active...

Conclusion

Relational databases support the concept of users and user rights, thus meeting the security needs of databases. The information of the organization can be captured, manipulated, managed and shared and the value the database brings to the organization is immense. For example, we were able to use the data from the relational database for analyzing key metrics and gather insights like below:

- Out of the four product categories, the best selling categories for the company is "Application Security' systems.
- Customers preferred to order products majorly on Rent than to purchase it at once.
- We were also able to analyze the performance of the employees by analyzing the sales data. For instance, we could identify the top employees who sold the maximum amount of products overall and basis of various categories too.

REFERENCES

Watt, A., Eng, N. (n.d.). *Database design- 2nd edition*. Retrieved from https://opentextbc.ca/dbdesign01/chapter/chapter-1-before-the-advent-of-database-systems/

Spencer, D. (2014, July 14). Top 10 common database security issues. Retrieved from https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2014/july/top-10-common-database-security-issues/

Tutorialink. (2019). DBMS. Retrieved from https://tutorialink.com/dbms/advantage-and-disadvantages-of-file-oriented-system.dbms

Lemahieu, W., Broucke, S. V., & Baesens, B. (2018). *Principles of database management: The practical guide to storing, managing and analyzing big and small data*. Cambridge, United Kingdom: Cambridge University Press.