IE2042 - Database Management Systems for Security

Assignment 01: 2020 Regular Intake

Title: Restaurant Management System

Group Members:

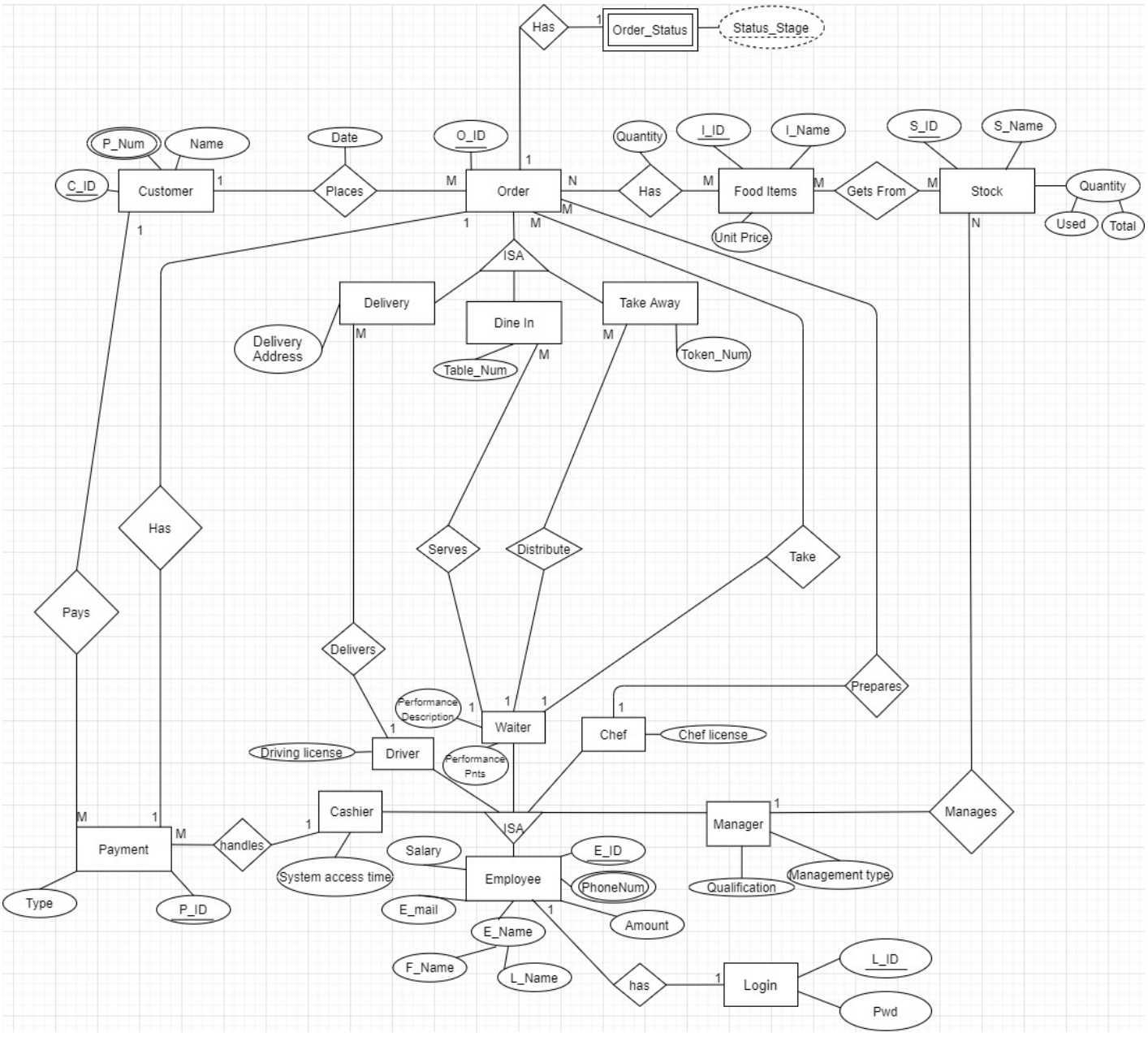
IT Number	Name
19017884	Yugandhree.S
19010236	Kajanthan.B
18197310	Sarangan.R

Contribution for the Project

IT Number	Contribution
19017884	• Contributed some ideas for ER Diagram • Identified one transaction • Created the tables • Contributed some insertion data queries and Inserted the Data • Implemented Access control Privileges • Contributed codes and Implemented the Security Mechanisms (Stored, View, Sequence and Triggers) • Contributed some information about web-based attacks and counter measures, Database recovery. • Presented the Presentation
19010236	• Contributed some ideas for ER Diagram • Identified one transaction • Contributed Table creation queries • Complete Relational Schema Drawing • Contributed some codes for Security Mechanisms • Contributed some information about web-based attacks and counter measures • Completed the major part of Database recovery.
18197310	• Contributed some ideas for ER Diagram • Identified one transaction • Contributed some insertion data • Complete ER Diagram drawing • Contributed some codes for Security Mechanisms • Contributed some information about Database recovery. • Completed the major part of web-based attacks and counter measures • Created the Presentation

Table of Contents

ER Diagram


Has
1
Order_Status
Status_Stage
P_Num
Name
C_ID
Customer
1
Date
Places
M
O_ID
Order
N
Quantity
Has
M
I_ID
I_Name
Food Items
M
Gets From
M
S_ID
S_Name
Stock
Quantity
Used
Total
N
Unit Price
1
1
M
M
ISA
Delivery
Delivery Address
M
Dine In
Table_Num
M
Take Away
M
Token_Num
Has
Pays
Serves
Distribute
Take
Delivers
Prepares
Performance Description
1
1
1
Waiter
1
Chef
Chef license
Driving license
Driver
1
Performance Pnts
M
1
Payment
M
handles
1
Cashier
ISA
Manager
1
Manages
Salary
E_ID
System access time
Employee
PhoneNum
Qualification
Management type
Type
P_ID
E_mail
Amount
1
E_Name
F_Name
L_Name
has
1
Login
L_ID
Pwd

Relational Schema

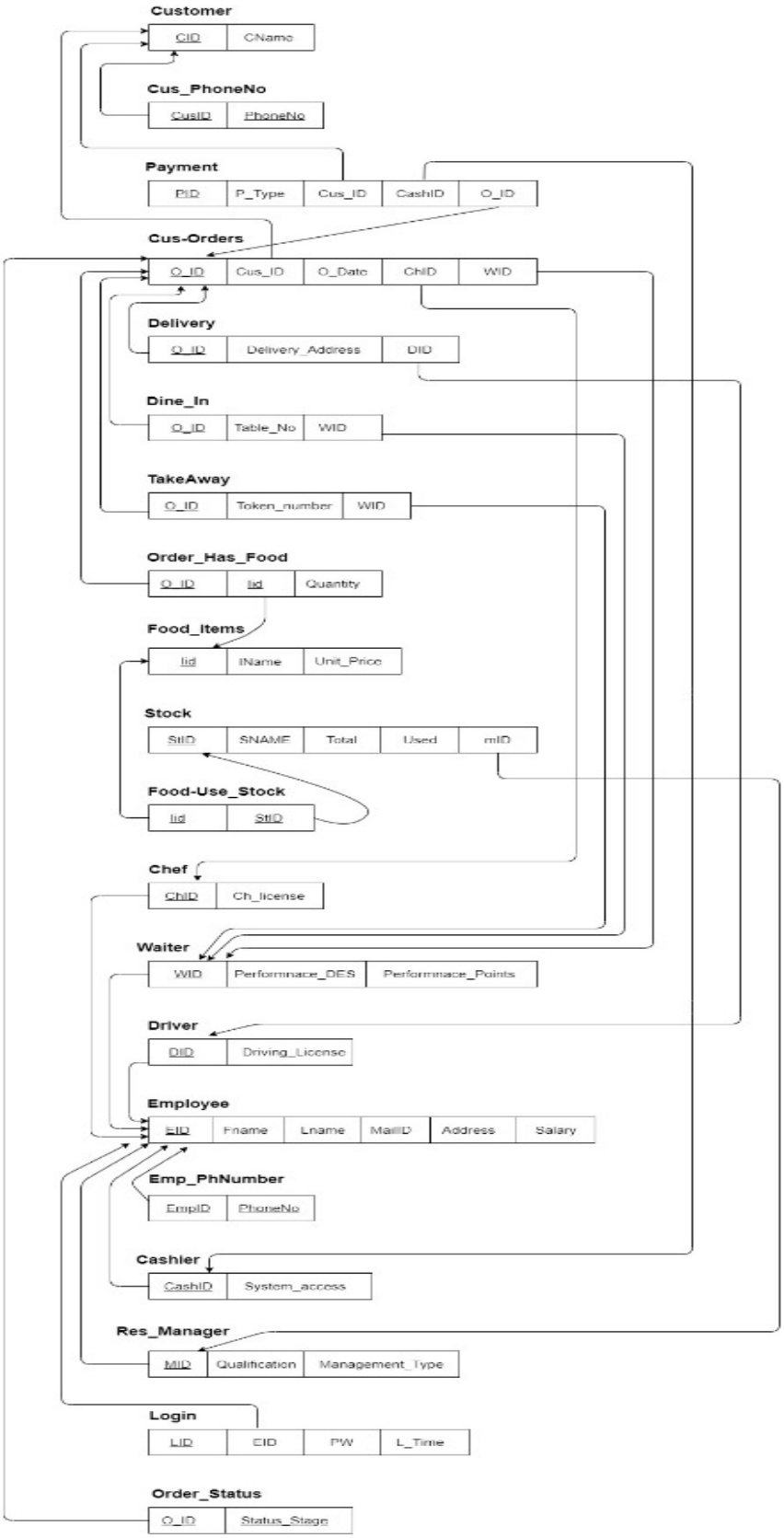


Table Creation Queries

Worksheet | Query Builder

```
/* ISA Relatiohship tables (Employee) */
CREATE TABLE Employee
(
    EID VARCHAR(5),
    Fname VARCHAR(25) NOT NULL, /* This field cannot be empty */
    Lname VARCHAR (25) NOT NULL,
    MailID VARCHAR (50),
    Address VARCHAR(80) NOT NULL,
    Salary REAL,

     CONSTRAINT pk_Employee PRIMARY KEY(EID)

);
/* Multivalued Attribute table */
CREATE TABLE Emp_PhNumber
(
    EmpID VARCHAR(5),
    PhoneNo CHAR(10),

    CONSTRAINT pk_Emp_PhNumber PRIMARY KEY(EmpID,PhoneNo),
    CONSTRAINT fk_Emp_PhNumber FOREIGN KEY(EmpID) REFERENCES Employee(EID),
    CONSTRAINT Chk_Emp_PhNumber CHECK(REGEXP_LIKE(PhoneNo, '^0\d{9}$'))


);
CREATE TABLE Res_Manager
(
    MID VARCHAR(5),
    Qualification VARCHAR(80) NOT NULL,
    Management_Type VARCHAR(50),

     CONSTRAINT pk_Res_Manager PRIMARY KEY(MID),
     CONSTRAINT fk_Res_Manager FOREIGN KEY(MID) REFERENCES Employee(EID)

);

CREATE TABLE Chef
(
    ChID  VARCHAR(5),
    Ch_license VARCHAR(20),

     CONSTRAINT pk_Chef PRIMARY KEY(ChID),
     CONSTRAINT fk_Chef FOREIGN KEY(ChID) REFERENCES Employee(EID)

);
```

Worksheet | Query Builder

```
/*payment table */
CREATE TABLE Payment
(
    PID VARCHAR(5),
    P_Type VARCHAR(10),
    Cus_ID VARCHAR(5),
    CashID VARCHAR(5),
    O_ID VARCHAR(5),


    CONSTRAINT pk_Payment PRIMARY KEY(PID),
    CONSTRAINT fk_Payment FOREIGN KEY(Cus_ID) REFERENCES Customer(CID),
    CONSTRAINT fk_Payment1 FOREIGN KEY(CashID) REFERENCES Cashier(CashID),
    CONSTRAINT fk_Payment2 FOREIGN KEY(O_ID) REFERENCES Cus_Orders(O_ID),
    CONSTRAINT Chk_Payment CHECK (P_Type IN ('Cash','Card'))

);

/*Food items table */
CREATE TABLE Food_Items
(
    Iid VARCHAR(5),
    IName VARCHAR (40),
    Unit_Price REAL,

    CONSTRAINT pk_Food_Items PRIMARY KEY(Iid)

);

/* Stock table */
CREATE TABLE  Stock
(
    StID VARCHAR (5),
    SNAME VARCHAR (50) NOT NULL,
    Total REAL,
    Used REAL,
    mID VARCHAR(5),

    CONSTRAINT pk_Stock PRIMARY KEY(StID),
    CONSTRAINT fk_Stock1 FOREIGN KEY(mID) REFERENCES Res_Manager(MID),
    CONSTRAINT Chk_Stock CHECK (Total > 0)

);
```

Worksheet | Query Builder

```
/* M-M Relationship (order and food item) table */
CREATE TABLE Order_has_Food
(
    O_ID  VARCHAR(5),
    Iid VARCHAR(5),
    Quantity INT,

    CONSTRAINT pk_Order_has_Food PRIMARY KEY(O_ID,Iid),
    CONSTRAINT fk_Order_has_Food FOREIGN KEY(O_ID) REFERENCES Cus_Orders(O_ID),
    CONSTRAINT fk_Order_has_Food1 FOREIGN KEY(Iid) REFERENCES Food_Items(Iid)

);

/* M-M Relationship (Stock and food item) table */
CREATE TABLE Food_use_Stock
(
   Iid VARCHAR(5),
   StID VARCHAR(5),

   CONSTRAINT pk_Food_use_Stock PRIMARY KEY(Iid,StID),
   CONSTRAINT fk_Food_use_Stock FOREIGN KEY(iID) REFERENCES Food_Items(Iid),
   CONSTRAINT fk_Food_use_Stock1 FOREIGN KEY(StID) REFERENCES Stock (StID)

);
```

Tables After Inserting the Data

```
SELECT *
FROM Cashier
```

Query Result — All Rows Fetched: 2 in 0.002 seconds

	CASHID	SYSTEM_ACCESS
1	E014	26-JAN-2020 08:30:23
2	E015	27-JAN-2020 13:30:15

```
SELECT *
FROM Chef
```

Query Result — All Rows Fetched: 3

	CHID	CH_LICENSE
1	E004	10266826
2	E005	10266957
3	E006	10267234

```
SELECT *
FROM Customer
```

Query Result — All Rows Fetched: 5

	CID	CNAME
1	C001	Varun
2	C002	Saru
3	C003	Alex
4	C004	Zahir
5	C005	Saman

```
SELECT *
FROM Cus_PhoneNo
```

Query Result — All Rows Fetch

	CUSID	PHONENO
1	C001	0769441022
2	C002	0715650054
3	C002	0725650000
4	C003	0775667049
5	C004	0759865064
6	C005	0719670094
7	C005	0777678763

```
SELECT *
FROM Delivery
```

Query Result — All Rows Fetched: 3 in 0.039 seconds

	O_ID	DELIVERY_ADDRESS	DID
1	O106	No.32, New Kandy Road, Malabe	E011
2	O107	No.45, Aadiyapaatham Road, Colombo	E013
3	O108	No.12, Galle Road, Colombo	E012

```
SELECT *
FROM Dine_In
```

Query Result — All Rows Fetched: 3 in

	O_ID	TABLE_NO	WID
1	O100	T001	E007
2	O101	T003	E008
3	O102	T007	E009

```
SELECT *
FROM Driver
```

Query Result — All Rows Fetched: 3 in

	DID	DRIVING_LICENSE
1	E011	B2748754
2	E012	B2747778
3	E013	B2776456

```
SELECT *
FROM Food_Items
```

Query Result — All Rows Fetched: 5 in 0.002 seconds

	IID	INAME	UNIT_PRICE
1	I001	Pittu Koththu	100
2	I002	Rotti Koththu	150
3	I003	Idiyaapa Koththu	120
4	I004	Noodles	140
5	I005	Rice and Curry	130

```
SELECT *
FROM Waiter
```

Query Result — All Rows Fetched: 4 in 0.042 seconds

	WID	PERFORMANCE_DES	PERFORMANCE_POINTS
1	E007	Quick serving,need to improve the langugae skills	6
2	E008	Quick serving, Cx interaction skills also better	7
3	E009	,must improve the langugae skills,time for serving	4
4	E010	Listening to cx is poor, language skills better	5

```
SELECT *
FROM Employee
```

Script Output | Query Result | All Rows Fetched: 15 in 0.002 seconds

	EID	FNAME	LNAME	MAILID	ADDRESS	SALARY
1	E001	Chaminda	Erangoda	chamin23@gmail.com	88/1,Gringots street,colombo-7	80000
2	E002	Sarujan	Ravi	saru@gmail.com	107A,Velivitta road,Malabe	55000
3	E003	Sandra	Gorge	sandrag@gmail.com	76/2, galle rd, colombo-3	55000
4	E004	Kasun	Molligoda	ksmolli@gmail.com	667, pitaheta mw, wellampitiya	40000
5	E005	Venkatesh	Murugan	venkim@gmail.com	93,2/3,Span tower,colombo-4	40000
6	E006	Meera	Mithun	mmeera@gmail.com	76B,Pennyquick lane,col-06	40000
7	E007	Chrish	Jhon	chrish@gmail.com	111,poorvarama rd,colombo-4	30000
8	E008	Jhony	Premkumar	pkjhony12@gmail.com	95,2/1,fedric road,col-6	30000
9	E009	Asoka	Jayasena	asokjs@gmail.com	12C,Fransis rd,colombo-05	30000
10	E010	Stella	Roy	sroy35@gmail.com	18/3,borupana rd, ratmalana	30000
11	E011	Tony	Watson	tonyw12@gmail.com	79,bandaranayaka mw,col 12	10000
12	E012	Rizvi	Ziyan	rziyan11@gmail.com	22,dias place,colombo-1	10000
13	E013	Kamal	Chnadran	kamalc34@gmail.com	34,pitugala rd,col-08	10000
14	E014	Nirmala	John	njohn67@gmail.com	128/2,mallika rd,col-3	47000
15	E015	Nimal	Jayaweera	nweera7@gmail.com	67A,lilly avenue,colombo-6	47000

```
SELECT *
FROM Res_Manager
```

Query Result | SQL | All Rows Fetched: 3 in 0.007 seconds

	MID	QUALIFICATION	MANAGEMENT_TYPE
1	E001	Bsc in Management	Overoll Management
2	E002	HND in Management	Stock Managemnet
3	E003	HND in Business Management	Stock Managemnet

```
SELECT *
FROM Emp_PhNumber
```

Query Result | SQL | All Rows Fetched:

	EMPID	PHONENO
1	E001	0712345678
2	E001	0762345678
3	E002	0722345678
4	E002	0782345678
5	E003	0752345678
6	E003	0772345678
7	E004	0771234567
8	E005	0781234567
9	E006	0761234567
10	E007	0711234567
11	E008	0701234567
12	E009	0751234567
13	E010	0721234567
14	E011	0765678345
15	E012	0775678345
16	E013	0715678345
17	E013	0755678345
18	E014	0725678345
19	E015	0785678345

```
SELECT *
FROM Food_use_Stock
```

Query Result | SQL | All Rows Fetched:

	IID	STID
1	I001	S002
2	I001	S004
3	I001	S006
4	I001	S008
5	I002	S001
6	I002	S002
7	I002	S004
8	I002	S006
9	I003	S001
10	I003	S002
11	I003	S004
12	I003	S006
13	I004	S002
14	I004	S003
15	I004	S004
16	I004	S006
17	I005	S002
18	I005	S004
19	I005	S006
20	I005	S007

```
SELECT *
FROM Order_Status
```

Query Result | SQL | All Rows Fetche

	O_ID	STATUS_STAGE
1	O100	Completed
2	O101	Completed
3	O102	Completed
4	O103	Completed
5	O104	Completed
6	O105	Completed
7	O106	Completed
8	O107	Completed
9	O108	Completed
10	O109	Cancelled

```
SELECT *
FROM Order_has_Food
```

Query Result | SQL | All Rows Fetche

	O_ID	IID	QUANTITY
1	O100	I001	2
2	O100	I002	1
3	O101	I003	2
4	O102	I005	3
5	O102	I004	2
6	O103	I002	1
7	O104	I001	1
8	O105	I004	1
9	O106	I002	1
10	O107	I005	2
11	O107	I004	2
12	O108	I002	1

```
288 SELECT *
289 FROM Login
```

Query Result

SQL | All Rows Fetched: 15 in 0.003 seconds

	EID	LID	PW	L_TIME
1	E001	ML001	Gjhtyiknel23	26-JAN-2020 08:30:00
2	E002	ML002	Gjhknel23	26-JAN-2020 08:30:10
3	E006	ChL001	Ajhtgrknel3	26-JAN-2020 08:30:15
4	E004	ChL002	Jjhtfghnel	26-JAN-2020 08:30:20
5	E014	CashL001	TYhtygjnel23	26-JAN-2020 08:30:21
6	E007	WL001	fhghjgGHfg45	26-JAN-2020 08:35:15
7	E011	DL001	fhghjSDfg23	26-JAN-2020 09:00:00
8	E008	WL002	tfgjgGHfg895	26-JAN-2020 09:00:15
9	E003	ML003	fhghjgGHfg45	27-JAN-2020 08:15:00
10	E009	WL003	fhgtyfHfg455	27-JAN-2020 08:20:15
11	E012	DL002	fhgfhgAA789	27-JAN-2020 08:27:00
12	E010	WL004	ffhgfT7098	27-JAN-2020 08:30:15
13	E015	CashL002	ffQWgdhgf8	27-JAN-2020 08:40:00
14	E005	ChL003	ffhgfhHj7098	27-JAN-2020 08:45:15
15	E013	DL003	ffhghgWER098	27-JAN-2020 09:00:03

Worksheet | Query Builder

```
268
269 SELECT *
270 FROM Payment
271
```

Query Result

SQL | All Rows Fetched: 9 in 0.001 seconds

	PID	P_TYPE	CUS_ID	CASHID	O_ID
1	1	Card	C003	E014	O100
2	2	Cash	C001	E014	O102
3	3	Cash	C004	E014	O101
4	4	Card	C004	E015	O103
5	5	Cash	C002	E015	O108
6	6	Card	C001	E015	O107
7	7	Cash	C005	E014	O106
8	8	Cash	C003	E015	O104
9	9	Card	C002	E014	O105

```
282 SELECT *
283 FROM Stock
284
```

Query Result

SQL | All Rows Fetched: 7 in 0.003 seconds

	STID	SNAME	TOTAL	USED	MID
1	S001	Wheat Flour	8	5	E002
2	S002	Coconut Oil	7	4	E002
3	S003	Noodle Packets	15	8	E003
4	S004	Masala powder	3	1	E003
5	S006	Salt	4	1	E002
6	S007	Rice	8	3	E002
7	S008	Rice Flour	8	4	E002

```
270 SELECT *
271 FROM Cus_Orders
```

Query Result

SQL | All Rows Fetched: 10 in 0.059 seconds

	O_ID	CUS_ID	O_DATE	CHID	WID
1	O100	C003	26-JAN-2020...	E004	E007
2	O101	C004	27-JAN-2020...	E005	E008
3	O102	C001	26-JAN-2020...	E004	E009
4	O103	C004	27-JAN-2020...	E005	E008
5	O104	C003	26-JAN-2020...	E004	E010
6	O105	C002	27-JAN-2020...	E005	E009
7	O106	C005	27-JAN-2020...	E006	E010
8	O107	C001	27-JAN-2020...	E005	E007
9	O108	C002	27-JAN-2020...	E006	E010
10	O109	C005	27-JAN-2020...	E005	E008

```
276 SELECT *
277 FROM TakeAway
278
```

Query Result

SQL | All Rows Fetched: 3 in 0.046 sec

	O_ID	TOKEN_NUMBER	WID
1	O103	001	E008
2	O104	002	E010
3	O105	003	E009

```
73  /*Calculate Availabe stock*/
74  SELECT StID,SNAME,(Total - Used) AS Available
75  FROM Stock
76
```

Query Result | Query Result 1

SQL | All Rows Fetched: 7 in 0.004 seconds

	STID	SNAME	AVAILABLE
1	S001	Wheat Flour	3
2	S002	Coconut Oil	3
3	S003	Noodle Packets	7
4	S004	Masala powder	2
5	S006	Salt	3
6	S007	Rice	5
7	S008	Rice Flour	4

Note: We used this query to find the available stock balance

```
62  /* Calculating total amount to pay*/
63  SELECT p.PID, p.Cus_ID,f.O_ID,SUM(i.Unit_Price * f.Quantity) AS "Total Amount"
64  FROM Cus_Orders o, Payment p, Order_has_Food f, Food_Items i
65  WHERE o.O_ID = p.O_ID AND o.O_ID = f.O_ID AND i.Iid = f.Iid
66  GROUP BY p.PID, p.Cus_ID,f.O_ID
67  ORDER BY PID ASC
68
```

Query Result

SQL | All Rows Fetched: 9 in 0.027 seconds

	PID	CUS_ID	O_ID	Total Amount
1	P001	C003	O100	350
2	P002	C001	O102	670
3	P003	C004	O101	240
4	P004	C004	O103	150
5	P005	C002	O108	150
6	P006	C001	O107	540
7	P007	C005	O106	150
8	P008	C003	O104	100
9	P009	C002	O105	140

Note – We use this query to calculate the total bill amount

Transaction and Operations

A transaction is a work unit which is performed against a database. Transactions are units or work sequences that are performed in a logical order, whether in a user's manual fashion or automatically through some kind of database program.

There are some basic controls for the transactions, such as

- Commit – Save the Changes
- Rollback – Roll back the Changes
- Savepoint - creates points within the groups of transactions in which to ROLLBACK
- SET Transaction – Place a name on transaction

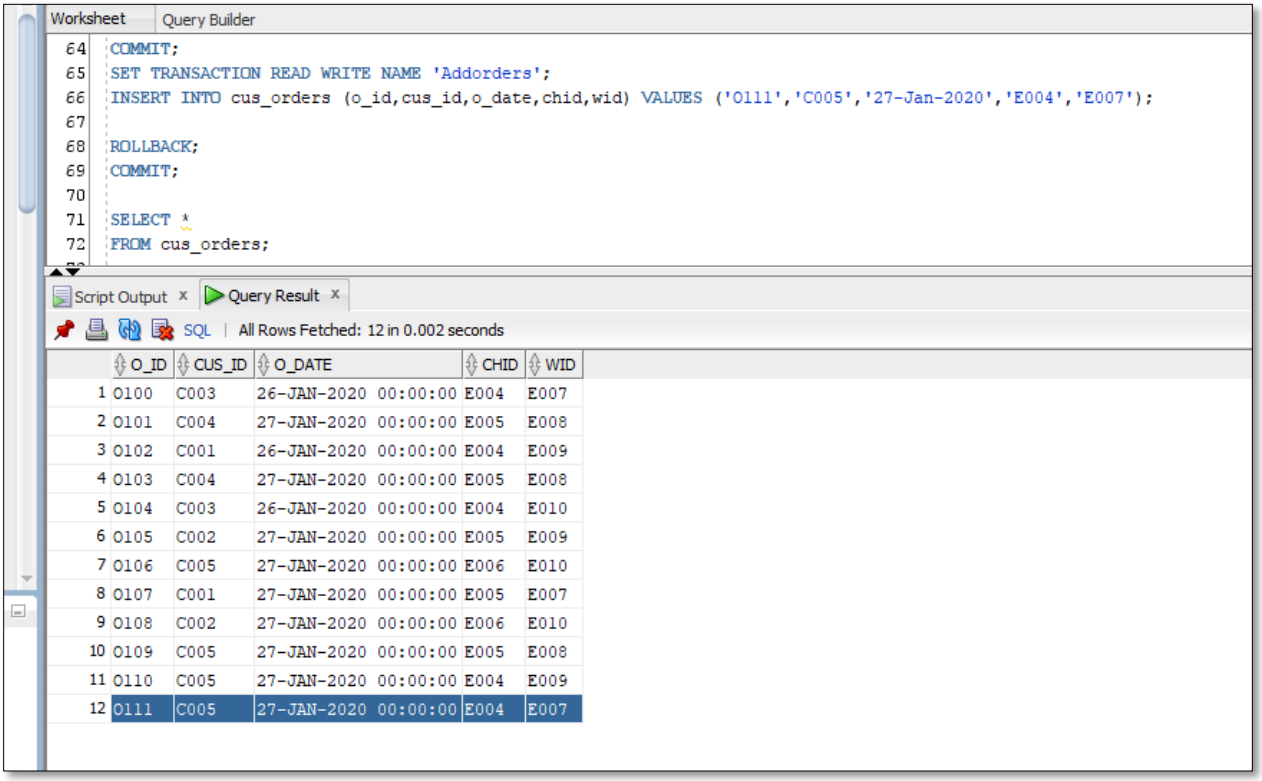
```
65  SET TRANSACTION READ WRITE NAME 'Addorders';
66  INSERT INTO cus_orders (o_id,cus_id,o_date,chid,wid) VALUES ('O111','C005','27-Jan-2020','E004','E007');
67
68  ROLLBACK;
69  COMMIT;
70
71  SELECT *
72  FROM cus_orders;
73
```

Script Output | Query Result | SQL | All Rows Fetched: 11 in 0.001 seconds

	O_ID	CUS_ID	O_DATE	CHID	WID
1	O100	C003	26-JAN-2020 00:00:00	E004	E007
2	O101	C004	27-JAN-2020 00:00:00	E005	E008
3	O102	C001	26-JAN-2020 00:00:00	E004	E009
4	O103	C004	27-JAN-2020 00:00:00	E005	E008
5	O104	C003	26-JAN-2020 00:00:00	E004	E010
6	O105	C002	27-JAN-2020 00:00:00	E005	E009
7	O106	C005	27-JAN-2020 00:00:00	E006	E010
8	O107	C001	27-JAN-2020 00:00:00	E005	E007
9	O108	C002	27-JAN-2020 00:00:00	E006	E010
10	O109	C005	27-JAN-2020 00:00:00	E005	E008
11	O110	C005	27-JAN-2020 00:00:00	E004	E009

In the above shown figure we set the transaction by the command ‘SET TRANSACTION’. Then by INSERT command, inserting the data into the cus_orders table. When we run the selected command, we can observe that new data added

in the new row. After the rollback command and commit command those details were gone. That image is submitted below here

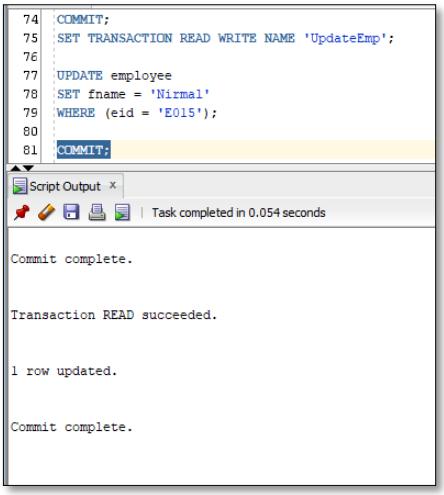


```
Worksheet | Query Builder
64 COMMIT;
65 SET TRANSACTION READ WRITE NAME 'Addorders';
66 INSERT INTO cus_orders (o_id,cus_id,o_date,chid,wid) VALUES ('O111','C005','27-Jan-2020','E004','E007');
67
68 ROLLBACK;
69 COMMIT;
70
71 SELECT *
72 FROM cus_orders;
```

Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.002 seconds

	O_ID	CUS_ID	O_DATE	CHID	WID
1	O100	C003	26-JAN-2020 00:00:00	E004	E007
2	O101	C004	27-JAN-2020 00:00:00	E005	E008
3	O102	C001	26-JAN-2020 00:00:00	E004	E009
4	O103	C004	27-JAN-2020 00:00:00	E005	E008
5	O104	C003	26-JAN-2020 00:00:00	E004	E010
6	O105	C002	27-JAN-2020 00:00:00	E005	E009
7	O106	C005	27-JAN-2020 00:00:00	E006	E010
8	O107	C001	27-JAN-2020 00:00:00	E005	E007
9	O108	C002	27-JAN-2020 00:00:00	E006	E010
10	O109	C005	27-JAN-2020 00:00:00	E005	E008
11	O110	C005	27-JAN-2020 00:00:00	E004	E009
12	O111	C005	27-JAN-2020 00:00:00	E004	E007

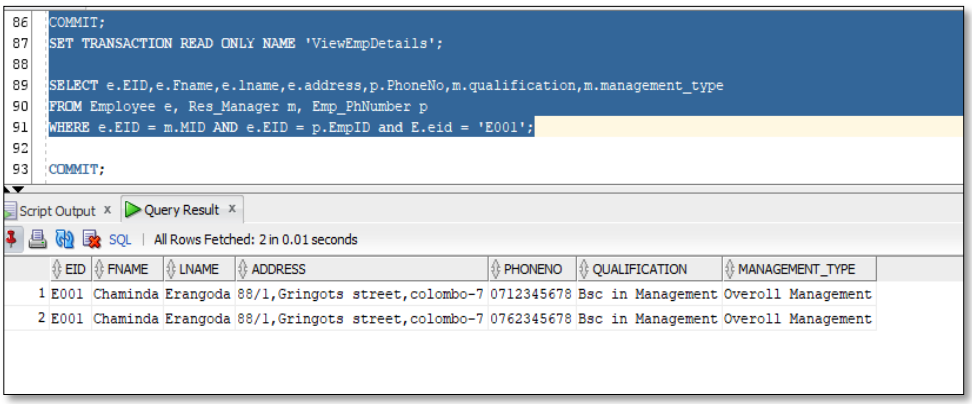


```
74 COMMIT;
75 SET TRANSACTION READ WRITE NAME 'UpdateEmp';
76
77 UPDATE employee
78 SET fname = 'Nirmal'
79 WHERE (eid = 'E015');
80
81 COMMIT;
```

Script Output

Task completed in 0.054 seconds

```
Commit complete.

Transaction READ succeeded.

1 row updated.

Commit complete.
```



```
86 COMMIT;
87 SET TRANSACTION READ ONLY NAME 'ViewEmpDetails';
88
89 SELECT e.EID,e.Fname,e.lname,e.address,p.PhoneNo,m.qualification,m.management_type
90 FROM Employee e, Res_Manager m, Emp_PhNumber p
91 WHERE e.EID = m.MID AND e.EID = p.EmpID and E.eid = 'E001';
92
93 COMMIT;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.01 seconds

	EID	FNAME	LNAME	ADDRESS	PHONENO	QUALIFICATION	MANAGEMENT_TYPE
1	E001	Chaminda	Erangoda	88/1,Gringots street,colombo-7	0712345678	Bsc in Management	Overoll Management
2	E001	Chaminda	Erangoda	88/1,Gringots street,colombo-7	0762345678	Bsc in Management	Overoll Management

Access Control Privilege

For any cyber-secure system or network, securing user accounts and helping avoid abuse of privileged accounts is crucial. User accounts, particularly those with special access privileges (e.g. administrative accounts), should only be assigned to approved persons, efficiently controlled, and minimal access to software, computers and networks should be provided. [1]

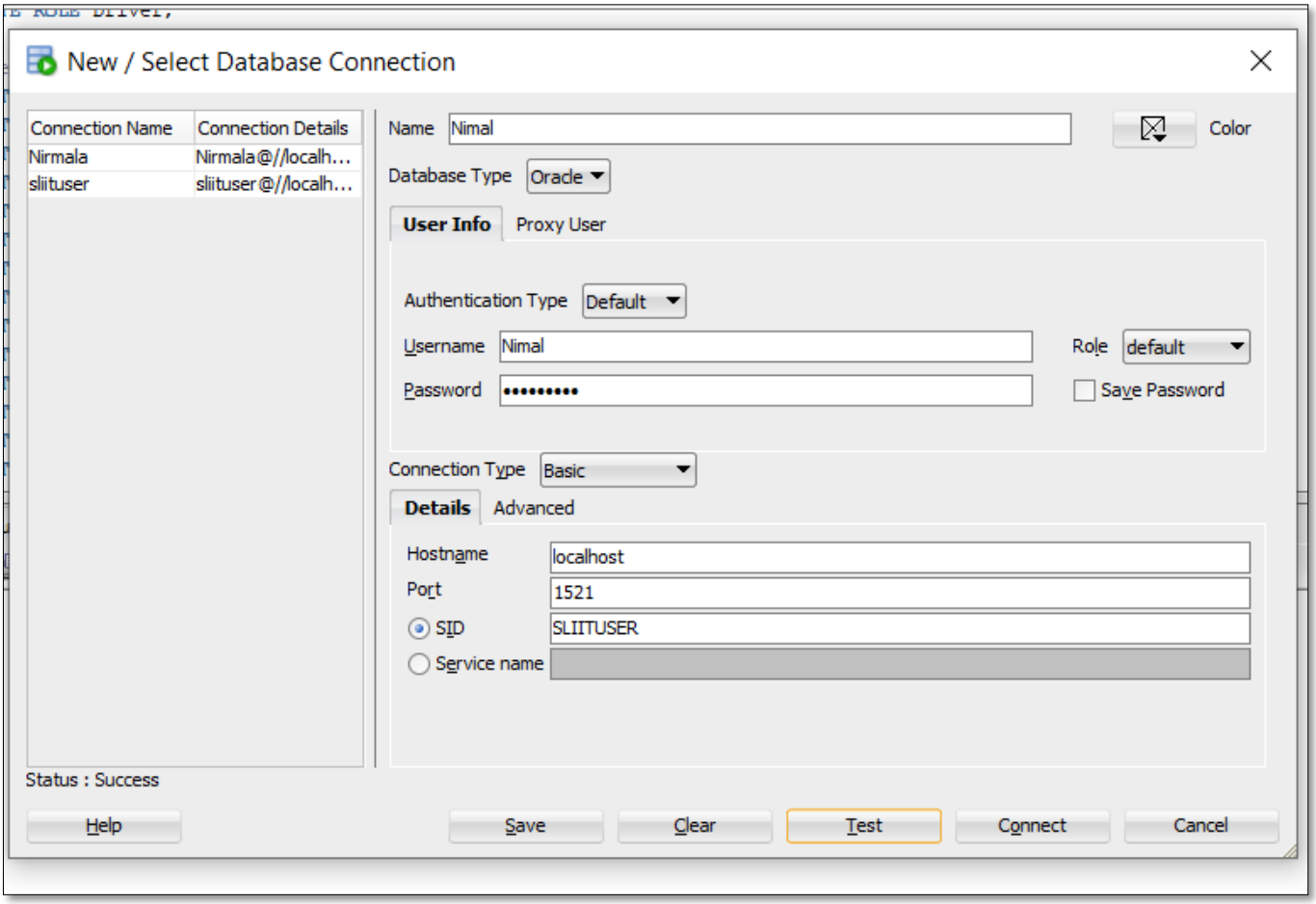
Worksheet | Query Builder

```
1  /*Enable scripting*/
2  ALTER SESSION SET "_ORACLE_SCRIPT"=TRUE;
3
4  /*Creating roles*/
5  CREATE ROLE Overoll_Manager;
6  CREATE ROLE Stock_Manager;
7  CREATE ROLE Cashier;
8  CREATE ROLE Chef;
9  CREATE ROLE Waiter;
10 CREATE ROLE Driver;
11
12 /*Creating Users*/
13 CREATE USER Chaminda  IDENTIFIED BY ChaminPW123;
14 CREATE USER Sarujan   IDENTIFIED BY SaruPW123;
15 CREATE USER Sandra    IDENTIFIED BY SandraPW123;
16 CREATE USER Nirmala   IDENTIFIED BY NirmaPW123;
17 CREATE USER Nimal     IDENTIFIED BY NimaPW123;
18 CREATE USER Kasun     IDENTIFIED BY KasunPW123;
19 CREATE USER Venkatesh IDENTIFIED BY VenkaPW123;
20 CREATE USER Meera     IDENTIFIED BY MeerPW123;
21 CREATE USER Chrish    IDENTIFIED BY ChriPW123;
22 CREATE USER Jhony     IDENTIFIED BY JhonPW123;
23 CREATE USER Asoka     IDENTIFIED BY AsokPW123;
24 CREATE USER Stella    IDENTIFIED BY StellPW123;
25 CREATE USER Tony      IDENTIFIED BY TonPW123;
26 CREATE USER Rizvi     IDENTIFIED BY RizPW123;
27 CREATE USER Kamal     IDENTIFIED BY KamPW123;
28
```

```
/*Granting Access Privileges to Roles*/
GRANT ALL Privileges TO Overoll_Manager;

/*Granting Permissions to Stock manager 1*/
GRANT CONNECT TO Stock_Manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON Stock TO Stock_Manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON Food_Items TO Stock_Manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON Order_has_Food TO Stock_Manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON Food_use_Stock TO Stock_Manager;


/*Granting Permissions to Cashiers*/
GRANT CONNECT TO Cashier;
GRANT SELECT, INSERT, UPDATE, DELETE ON Payment TO Cashier;
GRANT SELECT, INSERT, UPDATE ON Order_Status TO Cashier;
GRANT SELECT ON Customer TO Cashier;
GRANT SELECT ON Cus_Orders TO Cashier;
GRANT SELECT ON Food_Items TO Cashier;
GRANT SELECT ON Order_has_Food TO Cashier;

/*Granting Permissions to Chef*/
GRANT CONNECT TO Chef;
GRANT SELECT ON Food_Items TO Chef;
GRANT SELECT ON Order_has_Food TO Chef;
GRANT SELECT ON Order_Status TO Chef;

/*Granting Permissions to Waiter*/
GRANT CONNECT TO Waiter;
GRANT SELECT, INSERT ON Cus_Orders TO Waiter;
GRANT SELECT, INSERT ON Dine_In TO Waiter;
GRANT SELECT, INSERT ON TakeAway TO Waiter;
GRANT SELECT, INSERT ON Delivery TO Waiter;
GRANT SELECT, INSERT ON Order_Status TO Waiter;

/*Granting Permissions to Driver*/
GRANT CONNECT TO Driver;
GRANT SELECT ON Delivery TO Driver;
GRANT SELECT ON Order_Status TO Driver;
```

Worksheet | Query Builder

```
69
70  /*Grant Users for Roles*/
71  GRANT Overoll_Manager TO Chaminda;
72  GRANT Stock_Manager   TO Sarujan;
73  GRANT Stock_Manager   TO Sandra;
74  GRANT Cashier         TO Nirmala;
75  GRANT Cashier         TO Nimal;
76  GRANT Chef            TO Kasun;
77  GRANT Chef            TO Venkatesh;
78  GRANT Chef            TO Meera;
79  GRANT Waiter          TO Chrish;
80  GRANT Waiter          TO Jhony;
81  GRANT Waiter          TO Asoka;
82  GRANT Waiter          TO Stella;
83  GRANT Driver          TO Tony;
84  GRANT Driver          TO Rizvi;
85  GRANT Driver          TO Kamal;
```



Worksheet | Query Builder

```
1  SELECT *
2  FROM SLIITUSER.payment
```

Query Result

SQL | All Rows Fetched: 9 in 0.004 seconds

	PID	P_TYPE	CUS_ID	CASHID	O_ID
1	P001	Card	C003	E014	O100
2	P002	Cash	C001	E014	O102
3	P003	Cash	C004	E014	O101
4	P004	Card	C004	E015	O103
5	P005	Cash	C002	E015	O108
6	P006	Card	C001	E015	O107
7	P007	Cash	C005	E014	O106
8	P008	Cash	C003	E015	O104
9	P009	Card	C002	E014	O105

Worksheet | Query Builder

```
1  SELECT *
2  FROM SLIITUSER.Employee
```

Query Result

SQL | Executing:SELECT *FROM SLIITUSER.Employee in 0 seconds

```
ORA-00942: table or view does not exist
00942. 00000 -  "table or view does not exist"
*Cause:
*Action:
Error at Line: 2 Column: 16
```

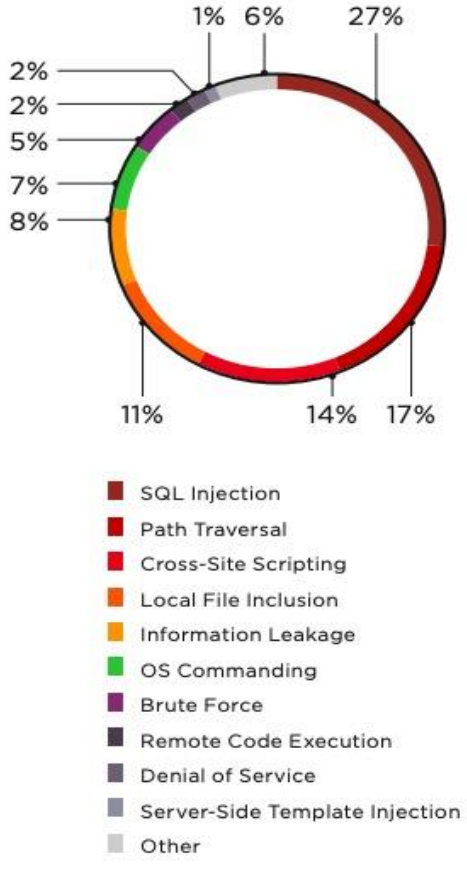
Attacks and Countermeasures

Web Application Attack

The hacking of web applications is one of the most common attacks on individuals and organisations. Hacked sites can be used for a number of things: to spread malware, to steal data, to post advertising or forbidden information, to commit fraud or to infiltrate an internal network. [2] Serious flaws or vulnerabilities allow criminals to access databases directly and publicly to churn confidential data – this is known as web application attack. Many of these databases contain sensitive information that makes them a regular target of attacks (e.g. personal data and financial details). Some faults and loopholes during database creation or implementation make the database vulnerable. Such bugs can easily be used by attackers / hackers to manipulate the database and steal, erase or change the data in your database. Reducing the effect of these events Organizations must identify and follow a strong security policy supported by a strong commitment of high management and policy must include corporate database security.

There are some most common vulnerabilities and web database attacks such as

- SQL Injection Attack
- Brute Force Attack
- Security Misconfiguration
- Privilege Escalation
- DoS Attack
- etc



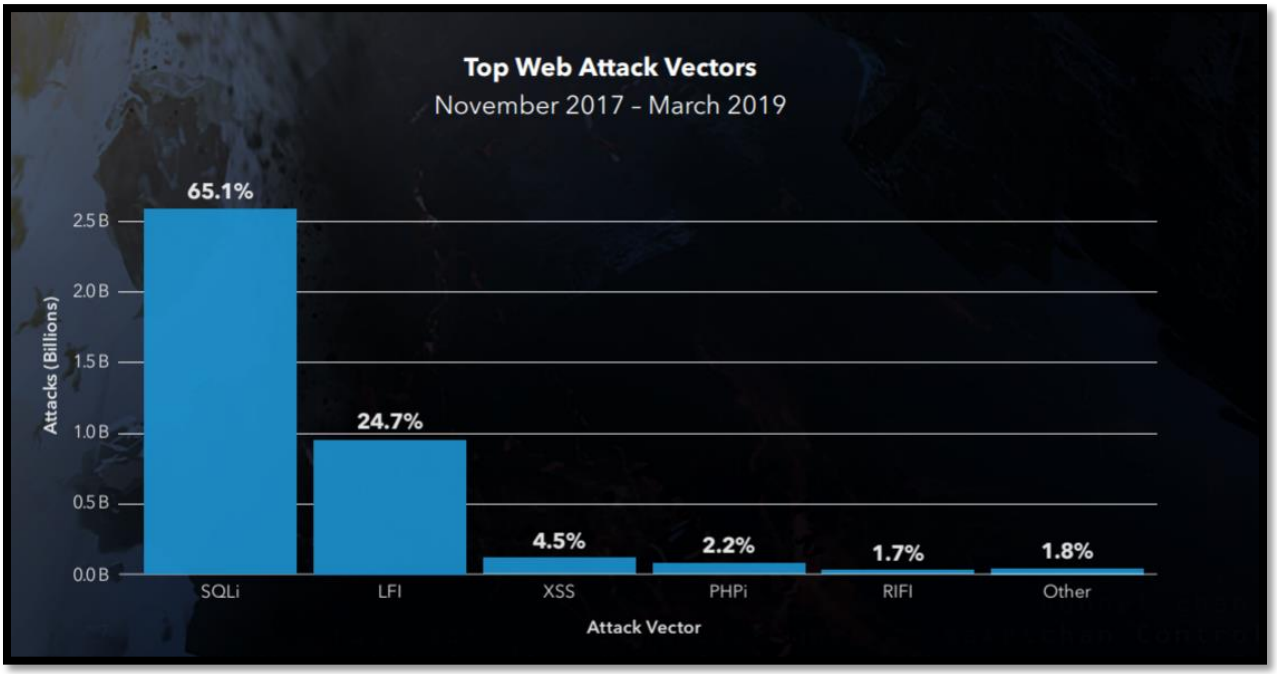
SQL Injection Attacks

What is SQL Injection?

SQL injection is a type of attack which, by inserting arbitrary SQL code into a database query, can give an adversary complete control over your web application database. There are various types of SQL injection but all include an intruder injecting arbitrary SQL into a database query for a web application. The simplest type of SQL injection is through input from the user. Web applications usually accept user input via a form, and for processing, the front-end transfers user input to the back-end database. When user feedback is not sanitized by the Web program, an attacker can inject SQL into the backend database of their choice and delete, copy or change database contents. [3] SQL injection attacks accounted for 65 percent of web based attack vectors from November 2017 to March 2019.

Let's see the countermeasures for these types of SQL injection attacks

- Don't trust anyone, take more careful when entering data into a database
- Give the web application the minimum privileges it needs to run.
- Apply Patches and Updates.
- Continuously monitor SQL statements from database-connected applications
- Use strong password credentials
- Use firewalls



Denial of Service Attacks

A "denial of service" or DoS assault is used to tie up the resources of a website, so that users who wish to access the site are unable to do so. DoS attacks have focussed on many major companies. Since a DoS attack can be easily conceived from almost any place, it can be extremely difficult to locate those responsible. Unlike a virus or malware, a DoS attack isn't based on running a specific program. Instead, it exploits an inherent flaw in how computer networks interact. [4]

Main Classes of DoS attack are

1. Abuse of functions
2. complex queries
3. Bugs and defects in the database
4. Application usage

There are two varieties of DoS attack

1. Flooding Attack
2. Crash Attack

How to prevent database from the DoS attack (Countermeasures) [5]

- Intrusion Detection Systems (IDS) and an Intrusion Protection Systems (IPS).
- Strong anti-virus and anti-spyware software on all systems with Internet connectivity.
- File and folder hashes on system files and folders to identify if they have been compromised.
- Reverse DNS lookup to verify the source address.
- External firewalls with the following filters:
    - Ingress filters that specify any inbound frame must have a public IP address from outside of the organization's LAN.
    - Egress filters that specify any outbound frame must have a private IP address within the organization's LAN.
    - Address filter to prevent traffic from specific attackers (if known).
- Once a DoS attack begins, you can minimize its effects by implementing filters to block unwanted traffic. You can also contact your ISP to implement filtering closer to the source and reduce the bandwidth used by the attack.

- Hardening practices on all machines, especially publicly exposed servers and directory and resource servers.

Brute Force Attack

A brute force attack is a common method of cracking: brute force attacks accounted for 5 per cent of reported breaches of security by some accounts. A brute force attack involves obtaining unauthorized access to a network by 'guessing' username and passwords. Brute force is a straightforward form of attack, which has a high rate of effectiveness. Some attackers use programs and scripts as instruments of brute force. Such software seeks out various variations of passwords to counteract authentication processes. In other instances, attackers try to access web applications by searching for the right session ID. Attacker motive can involve information theft, malware infecting sites or disrupting service.

There are some types of Brute Force Attacks, such as

- Simple Brute Force Attack
- Hybrid Brute Attack
- Dictionary Attacks
- Rainbow Table Attack
- Reverse Brute Attack
- Credential Stuffing

Some popular Brute Force Attack tools are used to identify the vulnerabilities such as

- THC- Hydra
- Aircrack-ng
- John the Ripper
- L0phtCrack
- Hashcat
- DaveGrohl
- Ncrack

Countermeasures for these types of brute force password cracking are

- Lockout policy where you can lock accounts after several failed login attempts and then unlock it as the administrator

- Progressive delays where you can lock out accounts for a limited amount of time after failed login attempts. Each attempt makes the delay longer.
- Tools like reCAPTCHA require users to complete simple tasks to log into a system. Users can easily complete these tasks while brute force tools cannot.
- You can force users to define long and complex passwords. You should also enforce periodical password changes. Ensure the passwords are in the highest encryption rates.
- You can use multiple factors to authenticate identity and grant access to accounts. Nowadays people are using two factor authentications.

Security Misconfiguration

Security misconfiguration encompasses several types of vulnerabilities all centered on a lack of maintenance or a lack of attention to the web application configuration. For the application, frameworks, application server, web server, database server and platform a stable configuration has to be specified and deployed. Misconfiguration of protection helps hackers to access private data or apps, and can result in a complete system compromise.

Sometimes, attackers may try to exploit unpatched bugs or access default accounts, inactive pages, insecure files and folders, etc. in order to obtain unauthorized access or device information. Security errors may occur at any level of an application stack, including network infrastructure, interface, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Automated scanners are useful for detecting misconfigurations, using default accounts or settings, redundant facilities, outdated options, etc. These bugs also offer attackers unauthorized access to data or functionality of some device. Occasionally these vulnerabilities lead to a complete compromise in the system. The market effect is based on the application's and data security needs.

How to prevent this issue,

Secure installing process by including, [6]

- A repeatable hardening process which makes it quick and easy to deploy another properly locked environment. Creation, QA, and production environments should all be identically configured, with different credentials

used in each environment. To reduce the effort needed to set up a new, safe environment, this process should be automated.

- A minimal platform without any unnecessary functionality, modules, samples and documentation. Delete the unused functionality and frameworks, or do not.
- A duty to review and update all security reports, fixes and patches correct configurations as part of the patch management process. In particular, check permissions for cloud storage.
- A segmented application architecture with segmentation, containerization, or cloud security groups (ACLs), providing reliable, safe separation between components or tenants.

Privilege Escalation

Privilege escalation occurs when a malicious user exploits an application or operating system bug, design defect, or configuration error to obtain privileged access to resources that would ordinarily be inaccessible to that user. The attacker will then use the new privileges to steal confidential data, execute administrative commands or deploy malware- and potentially seriously hurt the OS, server software, organization and credibility. We'll look at common privilege escalation situations in this blog post and learn how to secure user accounts in your systems and applications to maintain a good security posture. [7]

Privilege escalation take place in two forms such as [8]

- Vertical privilege escalation, also known as privilege elevation, where a lower privilege user or application accesses functionality or content reserved for higher privilege users or applications (e.g. Internet banking users may access administrative web functions or bypass the mobile password);
- Horizontal elevation of privilege where a regular user accesses functions or information reserved for other normal users (e.g. Internet Banking User A accesses User B's Internet Bank account)

How to prevent access control vulnerabilities

Vulnerabilities in access management can usually be avoided by taking a defence-in-depth approach and applying the following principles:

- Never depend on obfuscation alone to control access.
- Refuse access by default, unless a tool is meant to be publicly available.
- Using a single application-wide system to implement access controls, whenever possible.
- Consider it compulsory for developers to announce the access that is enabled for each resource at the code level, and refuse access by default.
- Access controls are regularly audited and reviewed to ensure they operate as expected.

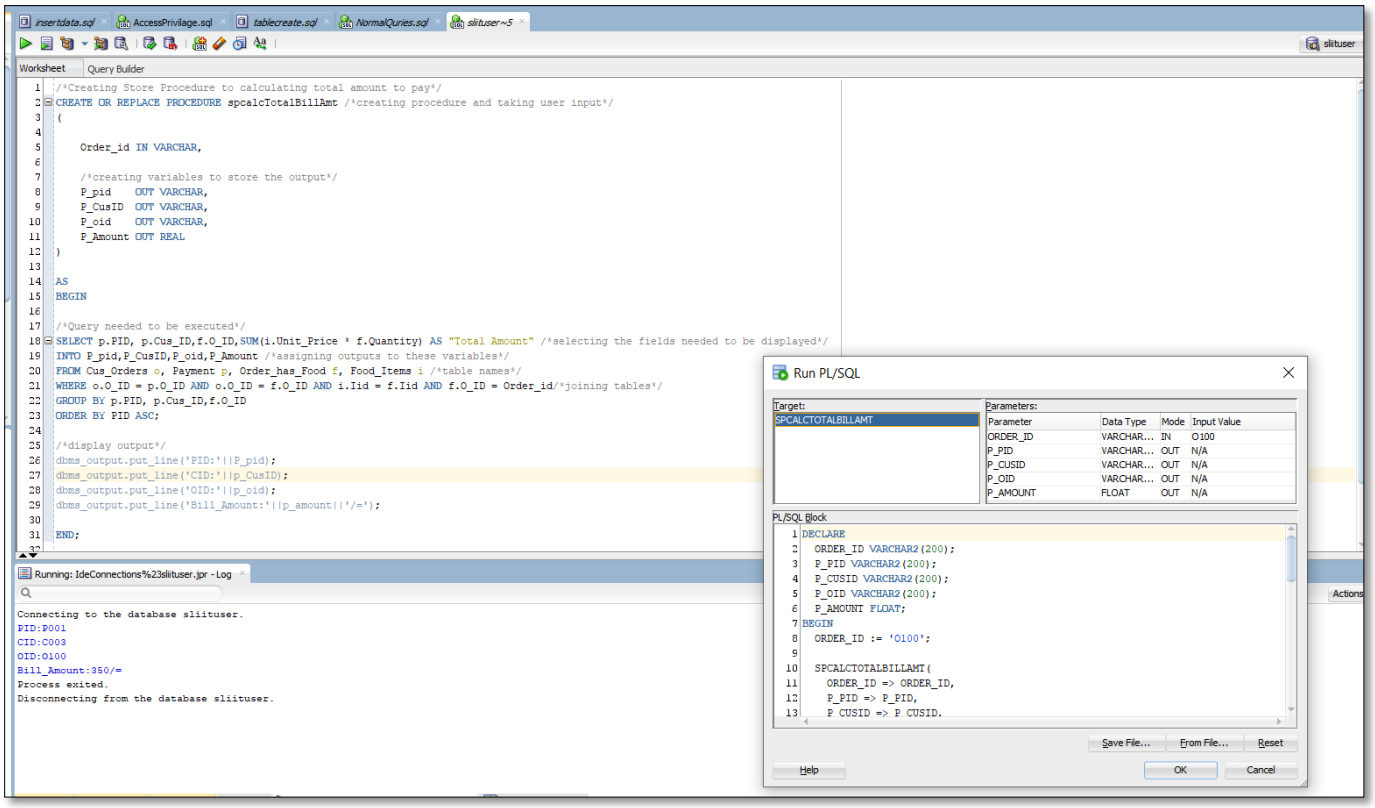
Implementing Countermeasures

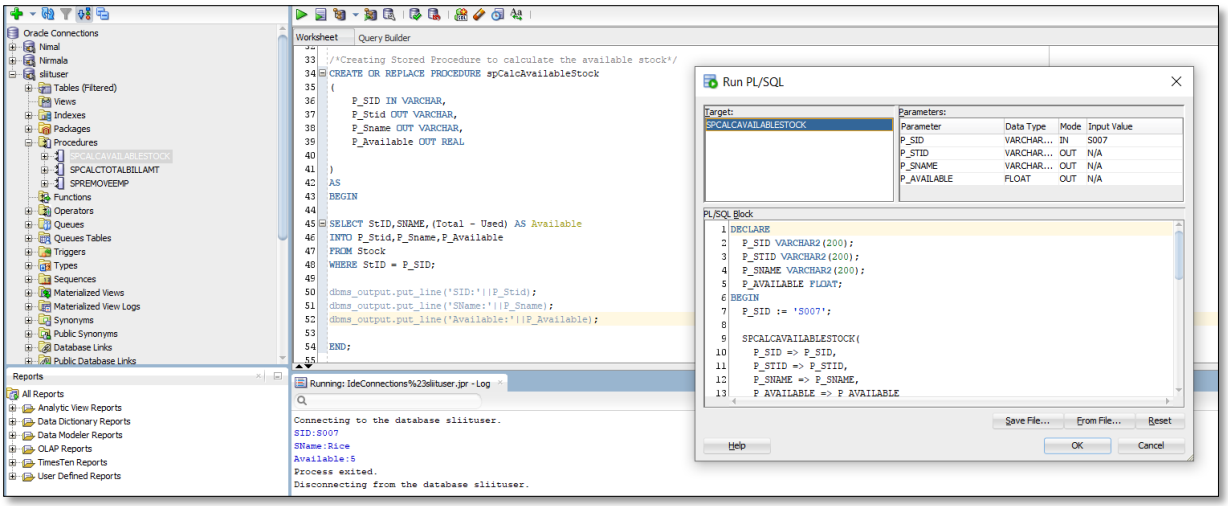
(View and Stored Procedures)

Stored procedures are pre-compiled database queries that improve the usability, productivity and security of data base server or client applications. Developers can define inputs and output variables, and compile them. The major benefits of this function are gaining significant output from pre-compiled execution, minimizing client or server traffic, gaining productivity in development by minimizing the reuse of codes and granting permissions to users on a particular security control method.

A View is a virtual screen. In one view, you can combine several tables and use the view to show the data as if the data came from a single table. A stored method uses a function's parameters. Either data is modified and reused, or single values or data sets are returned.

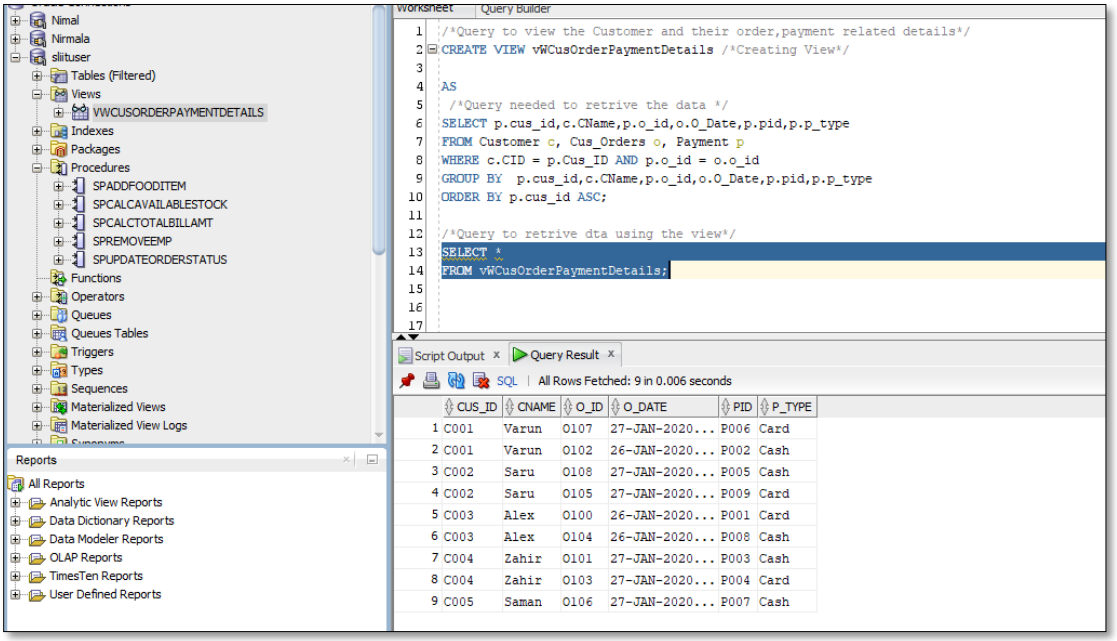
Shown Below images are the stored procedures

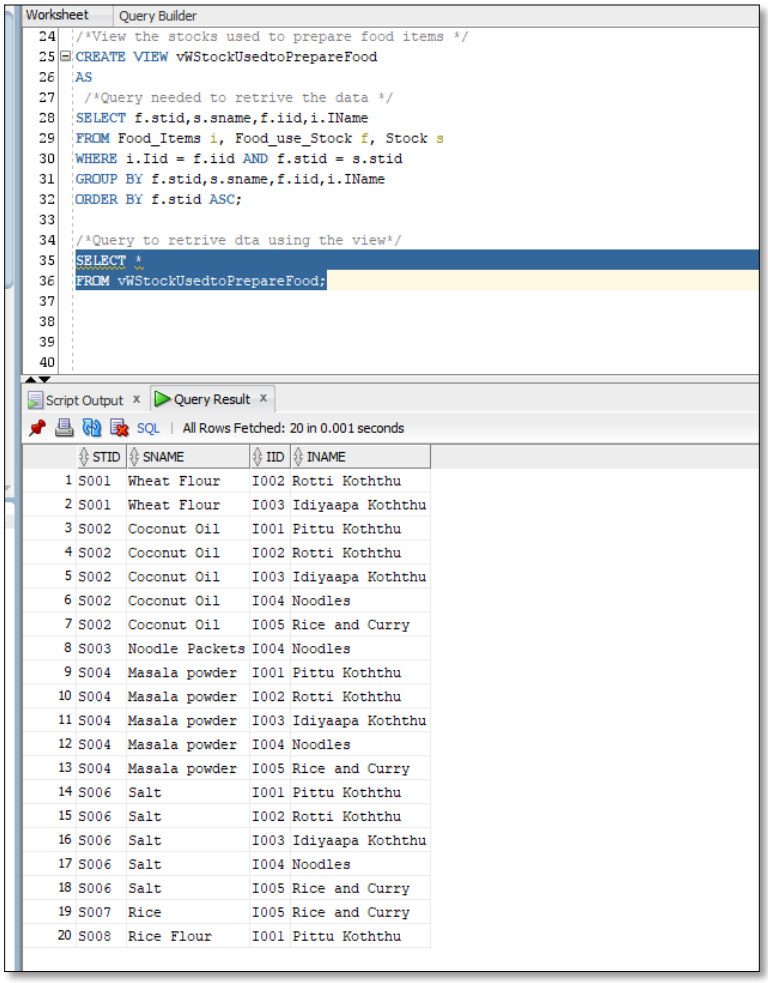





```
61
62  /*Creating Stored Procedure to INSERT a new food item*/
63  CREATE OR REPLACE PROCEDURE spAddfooditem
64  (
65      P_ItID IN VARCHAR,
66      P_ITName IN VARCHAR,
67      P_UPrice IN REAL
68  )
69  AS
70  BEGIN
71
72  INSERT INTO Food_Items(Iid,IName,Unit_Price)
73  VALUES (P_ItID,P_ITName,P_UPrice);
74
75  END;
76
77  /*Creating Stored Procedure to UPDATE the order sttaus*/
78  CREATE OR REPLACE PROCEDURE spUpdateOrderStatus
79  (
80      P_Oid IN VARCHAR,
81      P_Sstage IN VARCHAR
82
83  )
84  AS
85  BEGIN
86
87  UPDATE Order_Status
88  SET Status_stage = P_Sstage
89  WHERE (O_ID = P_Oid);
90
91  END;
92
93  /*Creating Stored Procedure to DELETE employees*/
94  CREATE OR REPLACE PROCEDURE spRemoveEmp
95  (
96      EmpID IN VARCHAR
97
98  )
99  AS
100 BEGIN
101
102 DELETE FROM Employee
103 WHERE (EID = EmpID);
104
105 END;
106
```

Below Shown Pictures are the view procedure.





```
1  CREATE SEQUENCE Payment_seq /*Creating sequence*/
2    MINVALUE 1
3    START WITH 10 /*We have already entered 9 values into the table manually*/
4    INCREMENT BY 1
5    CACHE 20;
6
7
8    CREATE OR REPLACE TRIGGER Test_Seq /*Creating trigger to check before inserting a new row into the payment table*/
9    BEFORE INSERT ON SLIITUSER.Payment
10   FOR EACH ROW
11   BEGIN
12     IF:new.PID IS NULL THEN
13     SELECT Payment_seq.Nextval INTO: new.PID FROM DUAL;
14     END IF;
15
16
17   END;
18
19
20
21   INSERT INTO Payment (PID,P_Type,Cus_ID,CashID,O_ID) VALUES (Payment_seq.NEXTVAL,'Card','C005','E014','O110');
22
23
24   SELECT *
25   FROM Payment
26
```

Script Output × | Query Result ×

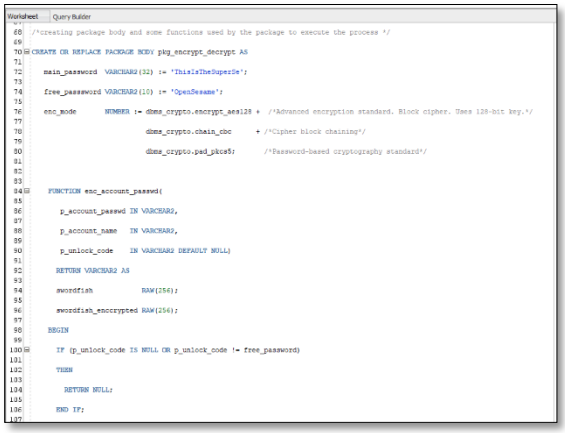
SQL | All Rows Fetched: 10 in 0.002 seconds

	PID	P_TYPE	CUS_ID	CASHID	O_ID
1	1	Card	C003	E014	O100
2	2	Cash	C001	E014	O102
3	3	Cash	C004	E014	O101
4	4	Card	C004	E015	O103
5	5	Cash	C002	E015	O108
6	6	Card	C001	E015	O107
7	7	Cash	C005	E014	O106
8	8	Cash	C003	E015	O104
9	9	Card	C002	E014	O105
10	10	Card	C005	E014	O110

Above shown figure is explaining the sequence and trigger procedure

Additional Security Implementation

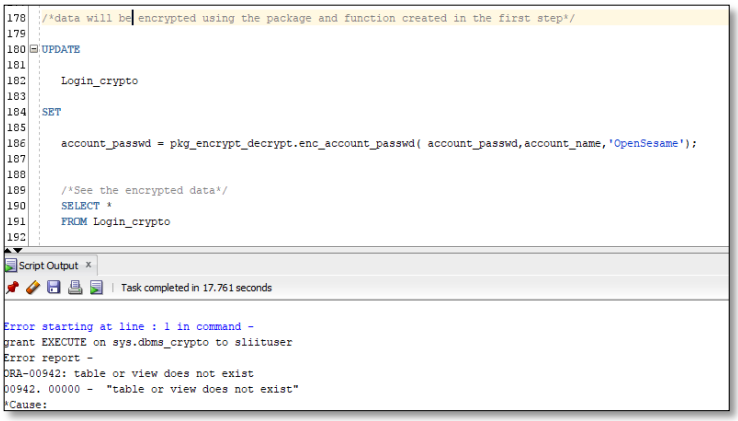
Since we do not have the access to use some functions, we could not able to get the output but we have submitted our idea in here,



```
/*creating package body and some functions used by the package to execute the process */
CREATE OR REPLACE PACKAGE BODY pkg_encrypt_decrypt AS
    main_password  VARCHAR2(32) := 'ThisIsTheSuperSe';
    free_password VARCHAR2(10) := 'OpenSesame';
    enc_mode       NUMBER := dbms_crypto.encrypt_aes128 +  /*Advanced encryption standard. Block cipher. Uses 128-bit key.*/
                             dbms_crypto.chain_cbc      + /*Cipher block chaining*/
                             dbms_crypto.pad_pkcs5;       /*Password-based cryptography standard*/


      FUNCTION enc_account_passwd(
        p_account_passwd IN VARCHAR2,
        p_account_name   IN VARCHAR2,
        p_unlock_code    IN VARCHAR2 DEFAULT NULL)
        RETURN VARCHAR2 AS
        swordfish           RAW(256);
        swordfish_encrypted RAW(256);
      BEGIN
        IF (p_unlock_code IS NULL OR p_unlock_code != free_password)
        THEN
          RETURN NULL;
        END IF;
```

```
Query Builder
    swordfish := dbms_crypto.randombytes(16); /*This function encrypts raw data using a stream or block cipher with a user supplied key*/


    swordfish_encrypted := dbms_crypto.encrypt(swordfish,enc_mode,utl_i18n.string_to_raw(main_password, 'al32utf8'));


  /*Inserting the account name, and swordfish, encrypted using the main_password as key, in secrets table*/
  INSERT INTO
    Login_crypto_secrets
  VALUES
    (p_account_name, swordfish_encrypted);
  RETURN
    utl_encode.base64_encode( dbms_crypto.encrypt( utl_i18n.string_to_raw(p_account_passwd,'al32utf8'),enc_mode,swordfish));
  END;
   FUNCTION dec_account_passwd(
   p_account_passwd IN VARCHAR2,
   p_account_name   IN VARCHAR2,
   p_unlock_code    IN VARCHAR2 DEFAULT NULL)
  RETURN VARCHAR2 AS
  swordfish RAW(256);
```

```
Worksheet  Query Builder
    BEGIN
      IF (p_unlock_code is null or p_unlock_code != free_passsword)
      Then
        RETURN NULL;
      END IF;


      SELECT
        dbms_crypto.DECRYPT(value2,enc_mode,utl_i18n.string_to_raw(main_password,'al32utf8'))
      INTO
        swordfish
      FROM
        Login_crypto_secrets
      WHERE
        value1 = p_account_name;
      RETURN utl_i18n.raw_to_char(dbms_crypto.DECRYPT(utl_encode.base64_decode( p_account_passwd),enc_mode,swordfish),'al32utf8');
    END;
  END;
```



```
/*data will be encrypted using the package and function created in the first step*/

UPDATE
    Login_crypto
SET
    account_passwd = pkg_encrypt_decrypt.enc_account_passwd( account_passwd,account_name,'OpenSesame');


    /*See the encrypted data*/
    SELECT *
    FROM Login_crypto
```

Script Output

Task completed in 17.761 seconds

```
Error starting at line : 1 in command -
grant EXECUTE on sys.dbms_crypto to sliituser
Error report -
ORA-00942: table or view does not exist
00942. 00000 -  "table or view does not exist"
*Cause:
```

```
update
   tab_dbms_crypto
set
   account_passwd = pkg_encrypt_decrypt.enc_account_passwd(
                       account_passwd,
                       account_name,
                       'OpenSesame')
/

3 rows updated

commit
/

Commit complete
```

Finally, the encrypted data can be seen.

```
select
   *
from
   tab_dbms_crypto
/

ACCOUNT_NAME   ACCOUNT_PASSWD
-------------- ------------------------------------------------
user1          6E42464477424C7145733576626F666F766D79344E773D3D
user2          7159616849675256554B6D6B31736E2F4E4C747639513D3D
user3          4B5742784C6F385778334645E58346A58396A5338673D3D
```

Recovery Mechanism

Database systems, like any other computer system, are subject to failures but the information stored in it must be accessible as and when necessary. When a server fails it must have the facilities for quick recovery. Both for backup of data and for recovery from malfunction situations there are automatic and non-automatic mechanisms. Database recovery methods are the strategies used to retrieve the missing data due to server failures, transaction errors, bugs, catastrophic failure, incorrect commands, etc. And data loss recovery methods can be used based on delayed updates and immediate updating or backup data.

"The process of restoring the database to a correct state in the event of a failure"

Recovery strategies rely heavily on the presence of a special file called as system log. It includes information on the start and end of every transaction and any transaction updates. The log tracks all transaction transactions that impact database items values. This knowledge is important to recover from failure of the transaction.

- The log is kept on disk start_transaction(T): This log entry records that transaction T starts the execution.
- read_item(T, X): This log entry records that transaction T reads the value of database item X.
- write_item(T, X, old_value, new_value): This log entry records that transaction T changes the value of the database item X from old_value to new_value. The old value is sometimes known as a before an image of X, and the new value is known as an afterimage of X.

- commit(T): This log entry records that transaction T has completed all accesses to the database successfully and its effect can be committed (recorded permanently) to the database.
- abort(T): This records that transaction T has been aborted.
- checkpoint: Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

A transaction T reaches its commitment point when all of its procedures which access the storage have been successfully completed. The transaction has reached the point that it does not abort (end without completion). The transaction will be registered permanently in the database until committed. Commitment is always to make a commit insert into the log and write the log to your drive. If a device crash happens, item is retrieved from the log for all transactions T with start transaction(T) entries but not yet commit(T), which will have to be retrieved in the database during the recovery process.

1.Undoing - If a transaction fails, the recovery manager will undo transactions, i.e. reverse a transaction's operations. This includes inspecting a transaction for the write item (T, x, old value, new value) log entry and changing the value of item x in the database to old-value. There are two main recovery methods for non-catastrophic transaction failures: delayed updates and immediate updates.

2.Deferred update -This method does not update the database physically on the disk until a transaction has reached its point of commit. All transaction updates are

registered in the workspace of the local transaction before the commit is reached. If a transaction fails before it reaches its commit stage, the database will not have been updated in any way so that UNDO is not required. The effect of the operations reported in the local transaction workspace will need to be REDO, as their effect may not have been written in the database yet. Therefore, a delayed update is also known as the algorithm No-undo / redo.

3.Immediate update - In the immediate update, some operations of a transaction can update the database before it reaches its commit point. These operations are, however, documented in a disk log prior to being added to the database, making recovery still possible. If a transaction does not hit its commit stage, its effect must be reversed. Example, the transaction must be rolled back, so we need both undo and redoing. It is known as the undo / redo algorithm.

4.Caching / Buffering - One or more disk pages containing modified data objects are cached into main memory buffers and then restored to memory until they are written back to disk. For keeping those buffers, a list of in-memory buffers called the DBMS cache is held under DBMS power. A directory is used to keep track of which objects in the buffer are in the database. Every buffer is associated with a dirty bit, which is 0 if the buffer is not updated rather than 1 if changed.

5.Shadow paging - Atomicity and durability are given. A directory is constructed with n entries, where the ith entry points to the connection page of the ith database. When a transaction begins to run the current directory will be copied to a shadow directory. When a page is to be changed, a shadow page is allocated where changes are made and all pages that refer to the original are revised to refer to the replacement page when it's ready to become permanent.

Some of the backup strategies are as follows.

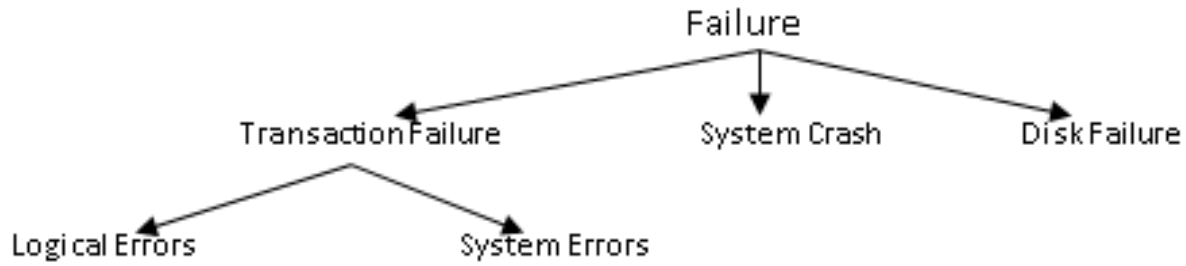
- Full database backup – Meta information needed to recover the entire database, including full-text catalogues, is backed up in a predefined time series in this complete database including data and database.
- Differential Backup – It only records the changes in data that have occurred since the last complete backup of the database. If the same data has changed several times since the last full database backup, the most current version of updated data is processed by a differential backup. We need to restore a complete backup of the database for this first.
- Transaction log backup – All actions that have happened in the database are backed up in this, like a record of any single statement that has been executed. This is the backup of transaction log entries which includes all the transactions that the database has happened to. Via this, it is possible to restore the database to a particular time point. A backup from a transaction log may also be done if the data files are destroyed and not even a single committed transaction is lost.

Reasons for Database Failure

There are several different forms of failure which can impact the operation of databases, each of which needs to be dealt with differently. Some failures impact only the primary memory while others require non-volatile storage (secondary). Among the causes of failure are:

- Device Crashes
- User Error
- Carelessness
- Sabotage (intentional data corruption)
- Declaration Failure
- Network Failure
- Media Failure
- Natural Physical Disasters

Classification of failures to see where the problem has arisen, we generalize a failure into various groups, as follows.



1.Transaction failure: A transaction must be terminated if it does not proceed, or if it reaches a stage from which it cannot go any further. This is called transaction failure when there are just a few transactions or processes that get hurt.

Reasons for transaction failure may be

- Logical errors − Where a transaction is unable to complete because it has some code error or some internal error.
- System Errors − Where the database system itself terminates an active transaction because the DBMS cannot execute it, or because of some system state, it has to interrupt it. For example, the program aborts an active transaction in the event of a deadlock or a resource unavailability.

2.System Crash: There are problems external to the system which can cause the system to suddenly stop and cause the system to crash. For example, power supply disturbances may cause the underlying hardware or software failure to fail.

Examples may involve errors in the operating system.

3.Disk Failure: In early days of technology development, a common issue existed where hard-disk drives or optical drives frequently failed.

Disc failures include bad sector creation, disk unavailability, disk head crash or some other failure that destroys all or part of the disk storage.

To help with the recovery mechanism, each DBMS should provide the following facilities

- Backup mechanism allows backup copies for the database at a given interval.
- Logging facilities help to monitor the existing transaction status and any modifications made to the database.
- Checkpoint facility allows database upgrades to make the new fixes permanent and to keep vulnerability secure.
- Recovery manager after any malfunction, helps the database system to restore the database to a safe and secure state.

Oracle Backup and Recovery Solution.

Oracle Backup and Recovery Solutions The following tools are available when implementing a backup and recovery strategy:

1.Recovery Manager (RMAN) Recovery Manager is completely integrated with the Oracle database to conduct a variety of backup and recovery tasks including the maintenance of an RMAN archive of backup historical data. You can access RMAN via the command line, or via Oracle Enterprise Manager.

- Incremental backups: Only blocks updated in incremental backup stores after a previous backup. They therefore have more portable backups and faster recovery, reducing the need to redo during recovery of data file media. If you allow tracking block shift, then you can boost performance

by avoiding full scans of any data file input. To perform incremental backups, you use the BACKUP INCREMENTAL button.

- Block media recovery: With only a small number of damaged blocks of data you can restore a data file without taking it offline or restoring it from backup. You use the command Restore Button to do network media recovery.
- Binary compression: A binary compression mechanism built in with Oracle Database reduces backup size.
- Encrypted backups: Encrypted RMAN backups use built-in backup encryption capabilities in Oracle Database to store backup sets in encrypted format. The data base will use the Advanced Protection Option to build encrypted backups on the disk. RMAN must use the Oracle Secure Backup SBT interface to make encrypted backups directly on the tape but does not require the Advanced Security Option.
- Automated database replication: Easily build a copy of your database which supports different storage configurations, including direct replication between ASM databases.

2.User-managed backup and recovery: In this approach, you perform backup and recovery using a mixture of host operating system commands and SQL*Plus recovery commands. You are responsible for deciding all aspects of when and how backups and recovery are performed.

The concept of data concept Data is the layout of the system's memory. It is divided primarily into two categories

- Volatile Memory: These are the system's primary memory devices and are mounted alongside the CPU. These memories can only store small amounts of data but are very fast. For example: main memory, cache memory etc. Such memories are unable to survive device failures- the data in such memories would be lost on failure.
- Non-volatile memory: These are secondary memories and are immense in size, but processing sluggish. Such memories are designed to survive device crashes, e.g.: Flash drive, hard disk, magnetic tapes etc ..
- Stable Memory: This is considered to be the third type of memory structure which is the same as non-volatile memory structure. Copies of same non-volatile memories are processed in different locations in this case. That is because data can be retrieved from other copies, in the event of any accident and data loss.

Assumptions

In our database we have used the units kg(kilogram), g(gram) and l(liters), but we have noted those units as integers in our tables.

We are calculating the total bill of the orders through the queries.

If a delivery order assigned in system that will very firstly identified by the waiter and receive that order and inform to chef. Here waiters and chef will be taking part in updating the orders. Drivers will never get the update access they will only get the delivery details according to the orders.

And also, delivery will take place only for larger orders which were ordered 24hours prior. So, we are having part time drivers only.

References

[1] C. E. A. C. &. A. P. Management, "itgovernance," [Online]. Available: https://www.itgovernance.co.uk/access-control-and-administrative-privilege.

[2] P. Tech, "ptsecurity," 26 June 2019. [Online]. Available: https://www.ptsecurity.com/ww-en/analytics/web-application-attack.

[3] J. Porup, "CSO," 2 October 2018. [Online]. Available: https://www.csoonline.com/article/3257429/what-is-sql-injection-how-sqli-attacks-work-and-how-to-prevent-them.html.

[4] Norton, "Norton," [Online]. Available: https://us.norton.com/internetsecurity-emerging-threats-dos-attacks-explained.html.

[5] G. Sites, "Sites," [Online]. Available: https://sites.google.com/a/pccare.vn/it/security-pages/dos-attacks-and-countermeasures.

[6] QWASP, "A6:2017-Security Misconfiguration," [Online]. Available: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A6-Security_Misconfiguration.

[7] Z. Banach, "Web Security Readings," 02 August 2019. [Online]. Available: https://www.netsparker.com/blog/web-security/privilege-escalation/.

[8] Wiki, "Wikipedia," 29 April 2020. [Online]. Available: https://en.wikipedia.org/wiki/Privilege_escalation.