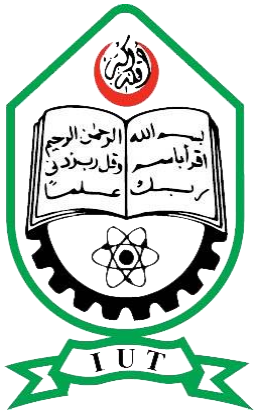
CSE 4308
Database Management Systems Lab
Lab 08
“Cascading Delete”, “Views” and “Roles”



Department of Computer Science and Engineering
Islamic University of Technology, OIC

Mohammad Anas Jawad
Lecturer, IUT CSE

Cascading Delete

A foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted.

Phone_Number

Phone_No	Opened	Outgoing	Customer
02134	12-06-2018	50,000	2
05468	21-01-2010	12,500	1
05698	01-12-2019	25,520	3
01234	22-04-2014	6,120	4

Foreign key referencing Cust_ID

Customer

Cust_ID	Name	Age	Gender	Occu.
1	A	123	M	Doctor
2	B	213	F	Teacher
3	C	321	O	Engineer
4	D	221	M	-----

Cascading Delete Syntax

```
--General Syntax
CREATE TABLE TABLE_NAME
(… … …,
… … …,
CONSTRAINT constraint_name FOREIGN KEY(…) REFERENCES REFERENED_TABLE(…) ON DELETE CASCADE
);
```

```
CONNECT Owner/test123;

CREATE TABLE CUSTOMER
(
        Cust_ID INT,
        Name VARCHAR2(20),
        Age INT,
        Gender VARCHAR2(1),
        Occupation VARCHAR2(10),
        CONSTRAINT customer_pk PRIMARY KEY(Cust_ID)
);

CREATE TABLE PHONE_NUMBER
(
        Phone_No INT,
        Opened DATE,
        Outgoing NUMBER(10,5),
        Customer INT,
        CONSTRAINT phone_number_pk PRIMARY KEY(Phone_No),
        CONSTRAINT phone_number_fk1 FOREIGN KEY(Customer) REFERENCES CUSTOMER(Cust_ID) ON DELETE CASCADE
);
```

Cascading Delete Demonstration

The CUSTOMER Table

```
SQL> CREATE TABLE CUSTOMER
  2  (
  3  Cust_ID INT,
  4  Name VARCHAR2(20),
  5  Age INT,
  6  Gender VARCHAR2(1),
  7  Occupation VARCHAR2(10),
  8  CONSTRAINT customer_pk PRIMARY KEY(Cust_ID)
  9  );

Table created.

SQL> INSERT INTO CUSTOMER VALUES(1,'A',25,'M','Doctor');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(2,'B',30,'F','Teacher');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(3,'C',35,'0','Engineer');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(4,'D',18,'M',NULL);

1 row created.
```

Cascading Delete Demonstration

The PHONE_NUMBER Table

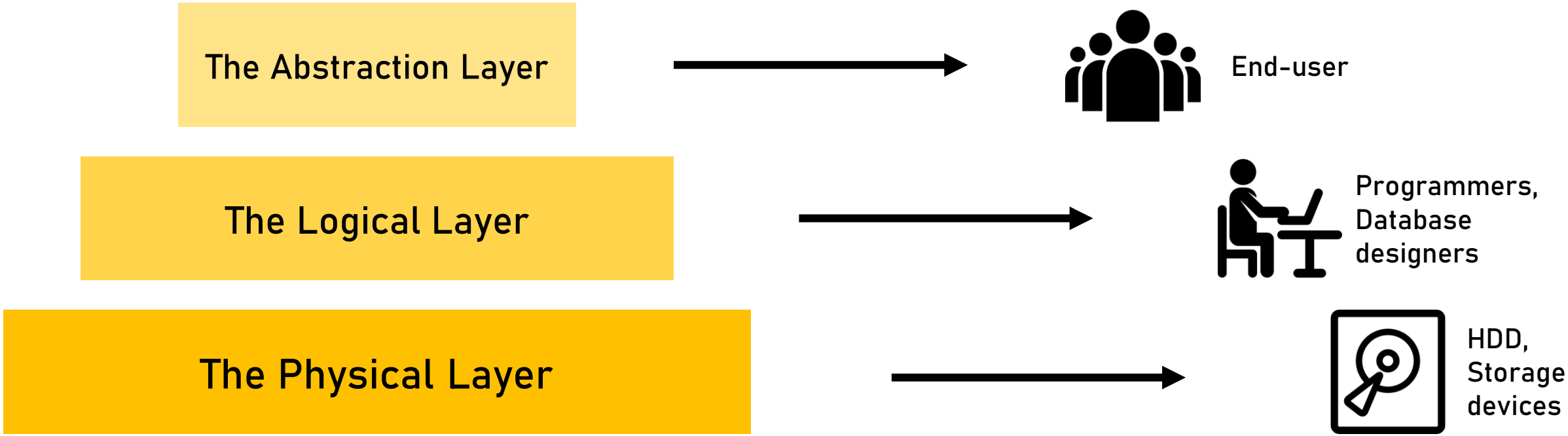
```
SQL> CREATE TABLE PHONE_NUMBER
  2  (
  3  Phone_No INT,
  4  Opened DATE,
  5  Outgoing NUMBER(10,5),
  6  Customer INT,
  7  CONSTRAINT phone_number_pk PRIMARY KEY(Phone_No),
  8  CONSTRAINT phone_number_fk1 FOREIGN KEY(Customer) REFERENCES CUSTOMER(Cust_ID) ON DELETE CASCADE
  9  );

Table created.

SQL> INSERT INTO PHONE_NUMBER VALUES(2134,TO_DATE( '12_Jun_2018' , 'DD_MON_YYYY' ),50000,2);

1 row created.

SQL> INSERT INTO PHONE_NUMBER VALUES(5468,TO_DATE( '21_Jan_2010' , 'DD_MON_YYYY' ),12500,1);

1 row created.

SQL> INSERT INTO PHONE_NUMBER VALUES(5698,TO_DATE( '01_Dec_2019' , 'DD_MON_YYYY' ),25520,3);

1 row created.

SQL> INSERT INTO PHONE_NUMBER VALUES(1234,TO_DATE( '22_Apr_2014' , 'DD_MON_YYYY' ),6120,4);

1 row created.
```

Final Outcome

```
SQL> SELECT * FROM CUSTOMER;

   CUST_ID NAME                      AGE G OCCUPATION
---------- -------------------- ---------- - ----------
         1 A                            25 M Doctor
         2 B                            30 F Teacher
         3 C                            35 0 Engineer
         4 D                            18 M

SQL> SELECT * FROM PHONE_NUMBER;

  PHONE_NO OPENED      OUTGOING   CUSTOMER
---------- --------- ---------- ----------
      2134 12-JUN-18      50000          2
      5468 21-JAN-10      12500          1
      5698 01-DEC-19      25520          3
      1234 22-APR-14       6120          4

SQL> DELETE FROM CUSTOMER WHERE CUST_ID=4;

1 row deleted.

SQL> SELECT * FROM CUSTOMER;

   CUST_ID NAME                      AGE G OCCUPATION
---------- -------------------- ---------- - ----------
         1 A                            25 M Doctor
         2 B                            30 F Teacher
         3 C                            35 0 Engineer

SQL> SELECT * FROM PHONE_NUMBER;

  PHONE_NO OPENED      OUTGOING   CUSTOMER
---------- --------- ---------- ----------
      2134 12-JUN-18      50000          2
      5468 21-JAN-10      12500          1
      5698 01-DEC-19      25520          3

SQL>
```

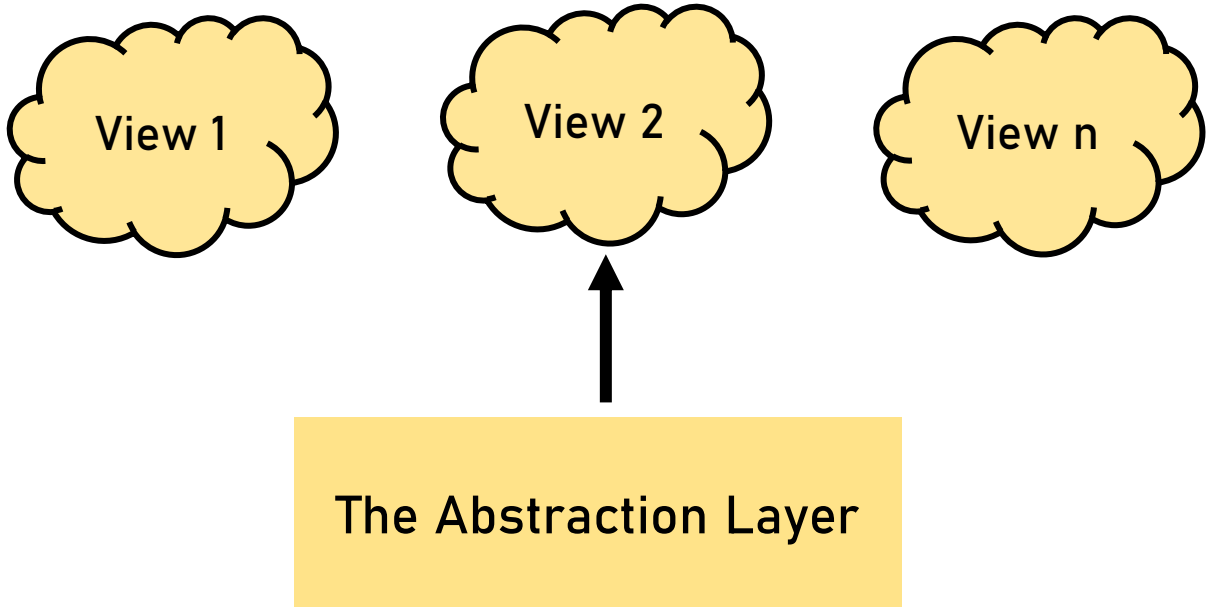
Views

-A virtual relation or table defined by a query, that essentially contains the results of the query.
-NOT precomputed and stored, rather, the view is computed by executing the query every time the view is used.



Views

-A virtual relation or table defined by a query, that essentially contains the results of the query.
-NOT precomputed and stored, rather, the view is computed by executing the query every time the view is used.



Views Syntax

Phone_Number

Phone_No	Opened	Outgoing	Customer
02134	12-06-2018	50,000	2
05468	21-01-2010	12,500	1
05698	01-12-2019	25,520	3
01234	22-04-2014	6,120	4

Customer

Cust_ID	Name	Age	Gender	Occu.
1	A	123	M	Doctor
2	B	213	F	Teacher
3	C	321	O	Engineer
4	D	221	M	-----

Views Syntax

```
--General Syntax for dropping a View

DROP VIEW VIEW_NAME;

--General Syntax for creating a View

CREATE OR REPLACE VIEW VIEW_NAME AS
<query>

CREATE VIEW VIEW_NAME AS
<query>
```

```
--Create a new view named CUSTOMER_INFO

CREATE OR REPLACE VIEW CUSTOMER_INFO AS
SELECT Name, Gender, Occupation, Phone_No, Opened FROM CUSTOMER, PHONE_NUMBER
WHERE PHONE_NUMBER.Customer = CUSTOMER.Cust_ID;


--Example query on views

SELECT NAME, OCCUPATION FROM CUSTOMER_INFO;
```

Views Demonstration

```
SQL> CREATE OR REPLACE VIEW CUSTOMER_INFO AS
  2  SELECT Name, Gender, Occupation, Phone_No, Opened FROM CUSTOMER, PHONE_NUMBER
  3  WHERE PHONE_NUMBER.Customer = CUSTOMER.Cust_ID;

View created.

SQL> SELECT * FROM CUSTOMER_INFO;

NAME                 G OCCUPATION   PHONE_NO OPENED
-------------------- - ---------- ---------- ---------
B                    F Teacher          2134 12-JUN-18
A                    M Doctor           5468 21-JAN-10
C                    0 Engineer         5698 01-DEC-19

SQL> SELECT NAME, OCCUPATION FROM CUSTOMER_INFO;

NAME                 OCCUPATION
-------------------- ----------
A                    Doctor
B                    Teacher
C                    Engineer

SQL> UPDATE CUSTOMER
  2  SET NAME = 'X'
  3  WHERE CUST_ID = 1;

1 row updated.

SQL> SELECT * FROM CUSTOMER_INFO;

NAME                 G OCCUPATION   PHONE_NO OPENED
-------------------- - ---------- ---------- ---------
B                    F Teacher          2134 12-JUN-18
X                    M Doctor           5468 21-JAN-10
C                    0 Engineer         5698 01-DEC-19

SQL>
```

Role-Based Access Control

-Create roles
-Grant specific privileges to those roles
-Grant roles to other roles
-Grant roles to specific users

Requirements

1. Customers should be able to view their information.
2. An operator should be able to update customer information.

Phone_Number

Phone_No	Opened	Outgoing	Customer
02134	12-06-2018	50,000	2
05468	21-01-2010	12,500	1
05698	01-12-2019	25,520	3
01234	22-04-2014	6,120	4

Customer

Cust_ID	Name	Age	Gender	Occu.
1	A	123	M	Doctor
2	B	213	F	Teacher
3	C	321	O	Engineer
4	D	221	M	-----

Role-based Access Control Syntax

```
--General Syntax for creating and deleting a role

CREATE ROLE ROLE_NAME;
DROP ROLE ROLE_NAME;

--General Syntax for granting privileges on a specific table to a role
GRANT PRIVILEGE_NAME ON TABLE_NAME TO ROLE_NAME;

--General Syntax for granting a role to another role
GRANT ROLE_NAME_1 TO ROLE_NAME_2;
```

```
--Connect to the server as the owner of the whole database
CONNECT Owner/testPassword;
--Create an example customer and an example operator
CREATE USER EXAMPLE_CUSTOMER_1 IDENTIFIED BY PASSWORD;
CREATE USER OPERATOR_1 IDENTIFIED BY test123;
GRANT CREATE SESSION TO EXAMPLE_CUSTOMER_1;
GRANT CREATE SESSION TO OPERATOR_1;
--Create a read-only role and grant it permission to view the Customer table
CREATE ROLE ROLE_READ_ONLY_CUSTOMER;
GRANT SELECT ON CUSTOMER TO ROLE_READ_ONLY_CUSTOMER;
--Create a role that can modify Customer table information
CREATE ROLE ROLE_MODIFY_CUSTOMER;
GRANT ROLE_READ_ONLY_CUSTOMER TO ROLE_MODIFY_CUSTOMER;
GRANT INSERT ON CUSTOMER TO ROLE_MODIFY_CUSTOMER;
GRANT DELETE ON CUSTOMER TO ROLE_MODIFY_CUSTOMER;
GRANT UPDATE ON CUSTOMER TO ROLE_MODIFY_CUSTOMER;
--Grant the created roles to the users
GRANT ROLE_READ_ONLY_CUSTOMER TO EXAMPLE_CUSTOMER_1;
GRANT ROLE_MODIFY_CUSTOMER TO OPERATOR_1;
```

Demonstration

```
SQL> CREATE USER EXAMPLE_CUSTOMER_1 IDENTIFIED BY PASSWORD;

User created.

SQL> CREATE USER OPERATOR_1 IDENTIFIED BY test123;

User created.

SQL> GRANT CREATE SESSION TO OPERATOR_1;

Grant succeeded.

SQL> GRANT CREATE SESSION TO EXAMPLE_CUSTOMER_1;

Grant succeeded.

SQL> GRANT CREATE SESSION TO OPERATOR_1;

Grant succeeded.

SQL> CREATE ROLE ROLE_READ_ONLY_CUSTOMER;

Role created.

SQL> GRANT SELECT ON CUSTOMER TO ROLE_READ_ONLY_CUSTOMER;

Grant succeeded.
```

Demonstration

```
SQL> CREATE ROLE ROLE_MODIFY_CUSTOMER;

Role created.

SQL> GRANT ROLE_READ_ONLY TO ROLE_MODIFY_CUSTOMER;

Grant succeeded.

SQL> GRANT INSERT ON CUSTOMER TO ROLE_MODIFY_CUSTOMER;

Grant succeeded.

SQL> GRANT DELETE ON CUSTOMER TO ROLE_MODIFY_CUSTOMER;

Grant succeeded.

SQL> GRANT UPDATE ON CUSTOMER TO ROLE_MODIFY_CUSTOMER;

SQL> GRANT ROLE_READ_ONLY TO EXAMPLE_CUSTOMER_1;

Grant succeeded.

SQL> GRANT ROLE_MODIFY_CUSTOMER TO OPERATOR_1;

Grant succeeded.
```

Demonstration

```
SQL> CONNECT EXAMPLE_CUSTOMER_1/PASSWORD;
Connected.

SQL> SELECT TABLE_NAME, OWNER FROM ALL_TABLES WHERE OWNER IN 'EXAMPLE_CUSTOMER_1';

no rows selected

SQL> SELECT * FROM CUSTOMER;
SELECT * FROM CUSTOMER
              *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> SELECT * FROM Owner.CUSTOMER;

   CUST_ID NAME                      AGE G OCCUPATION
---------- -------------------- ---------- - ----------
         1 X                            25 M Doctor
         2 B                            30 F Teacher
         3 C                            35 0 Engineer

SQL> INSERT INTO Owner.CUSTOMER VALUES (4,'D',55,'F','Doctor');
                    *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Thank You!