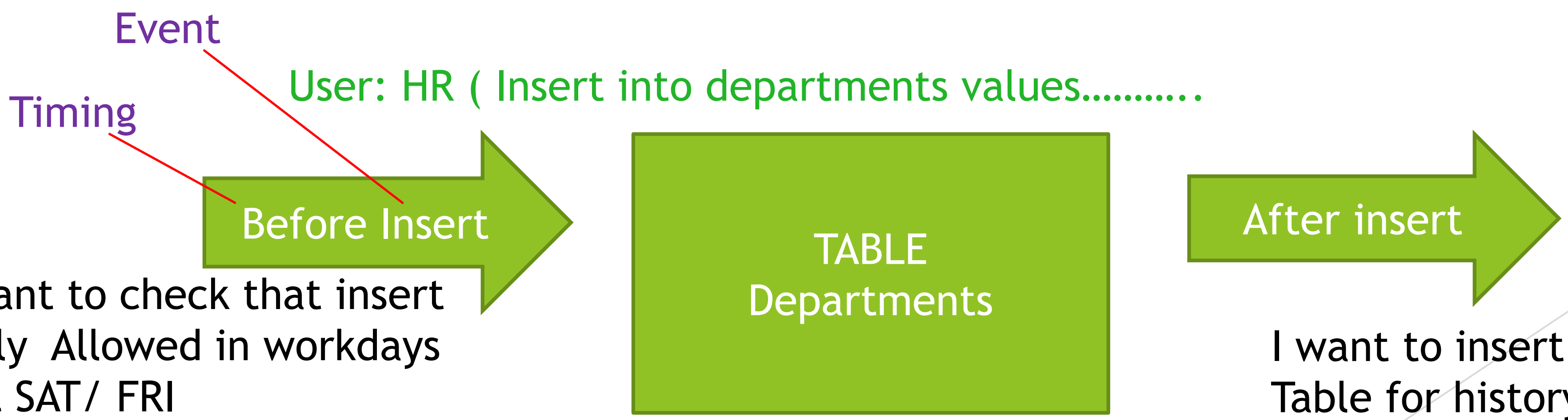
Creating Triggers

What Are Triggers?

- A trigger is a PL/SQL block that is stored in the database and fired (executed) in response to a specified event.
- The Oracle database automatically executes a trigger when specified conditions occur.



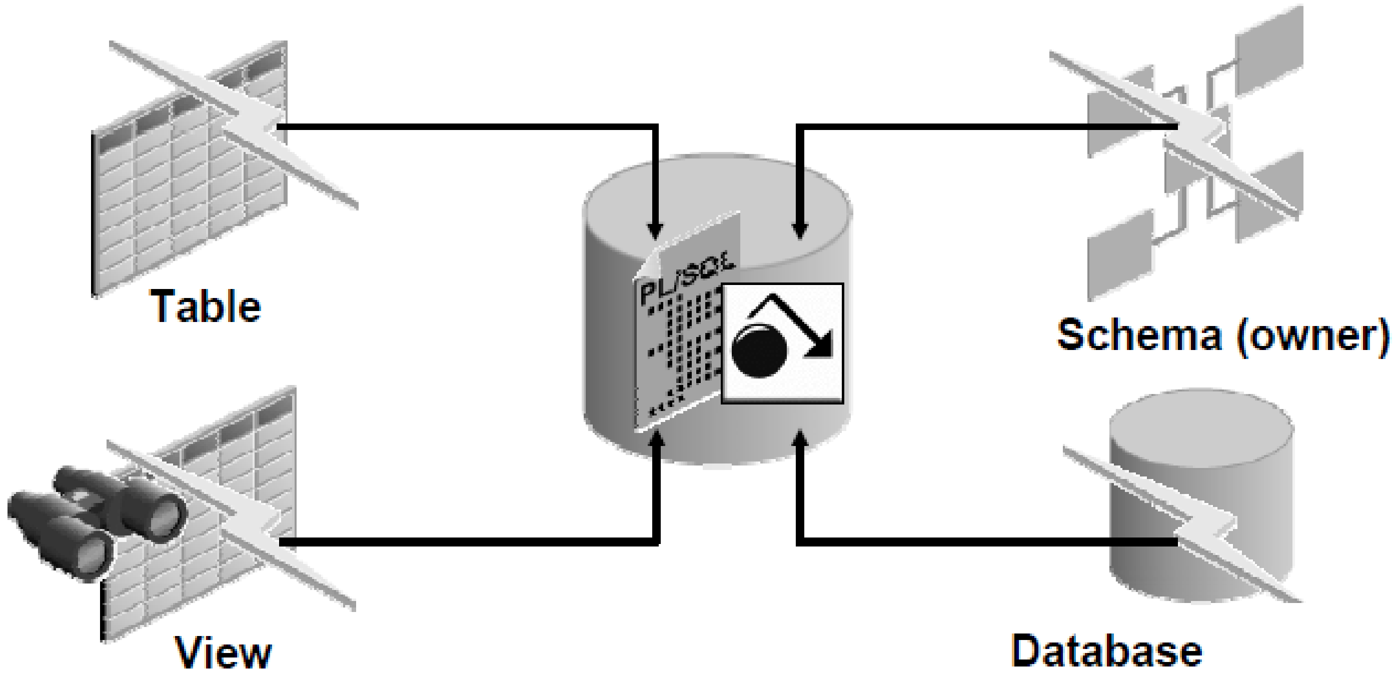
User: HR (Insert into departments values……….

I want to check that insert only Allowed in workdays Not SAT / FRI

I want to insert record for any insert to log Table for history Audit

Defining Triggers

A trigger can be defined on the table, view, schema (schema owner), or database (all users).



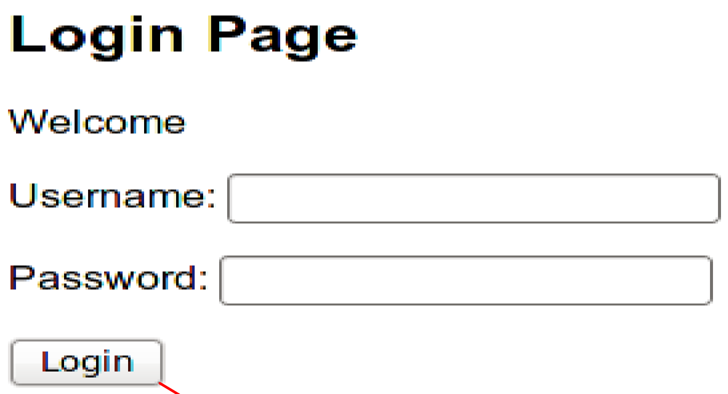
The Trigger Event Types

You can write triggers that fire whenever one of the following operations occurs in the database:

- A database manipulation (DML) statement (`DELETE`, `INSERT`, or `UPDATE`).
- A database definition (DDL) statement (`CREATE`, `ALTER`, or `DROP`).
- A database operation such as `SERVERERROR`, `LOGON`, `LOGOFF`, `STARTUP`, or `SHUTDOWN`.

Application and Database Triggers

- Database trigger (covered in this course):
  - Fires whenever a DML, a DLL, or system event occurs on a schema or database
- Application trigger:
  - Fires whenever an event occurs within a particular application


Login Page

Welcome

Username:

Password:

The code in LOGIN button fire when button pressed

Business Application Scenarios for Implementing Triggers

You can use triggers for:

- Security Ex: insert allowed only in working hours
- Auditing Ex: log all the transactions for specific tables
- Data integrity Ex: Complex integrity rules which not standard
- Referential integrity Ex: non standard referential
- Table replication Ex: synchronize a table
- Computing derived data automatically
- Event logging

The Available Trigger Types

- Simple DML triggers
  - `BEFORE`
  - `AFTER`
  - `INSTEAD OF`
- Compound triggers
- Non-DML triggers
  - DDL event triggers
  - Database event triggers

Trigger Event Types and Body

- A trigger event type determines which DML statement causes the trigger to execute. The possible events are:
  - `INSERT`
  - `UPDATE [OF column]`
  - `DELETE`
- A trigger body determines what action is performed and is a PL/SQL block or a `CALL` to a procedure

DML Triggers

Statement-Level trigger

Statement-Level Triggers
Is the default when creating a trigger
Fires once for the triggering event
Fires once even if no rows are affected

Ex: security check on (user, time,…)

ROW-level Triggers

Row-Level Triggers
Use the `FOR EACH ROW` clause when creating a trigger.
Fires once for each row affected by the triggering event
Does not fire if the triggering event does not affect any rows

Ex: log the transactions

DML Triggers

Statement-Level trigger
common cases

- When you want to check security before DML (Date, Time)
- When you want to check user profile before DML

So here no need to fire the trigger for each row , the trigger will fire only once

Update emp
Set sal=sal +10 where emp_id=1; (1 row)

Update emp
Set sal=sal +10; (all rows)

ROW-level Triggers
common cases

When you need the OLD and new values
For the DML

Here you should use row level trigger

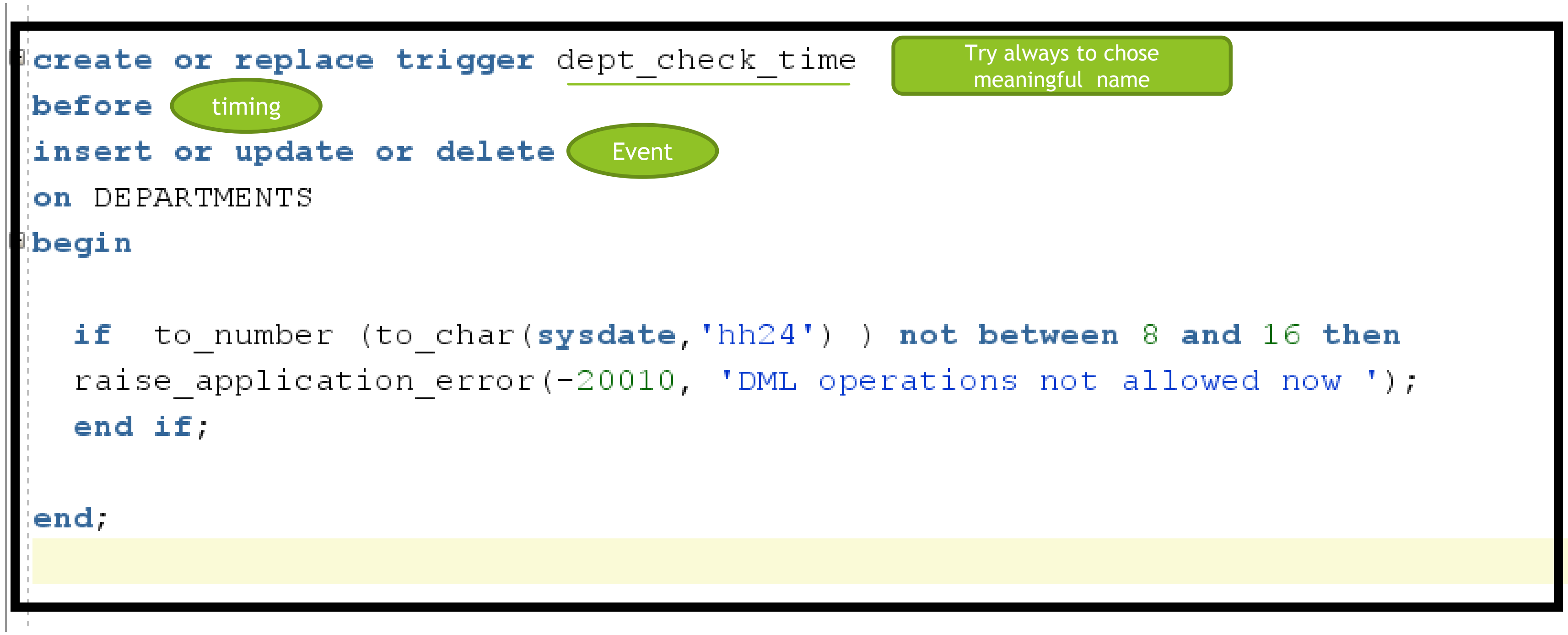
Update emp
Set sal=sal +10;

```
Name   old sal   new sal
Khaled 500       510
Ahmed 600        610
....
```

Statement -Level triggers

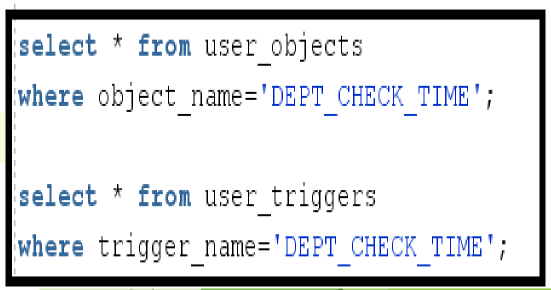
Trigger Event

Trigger body



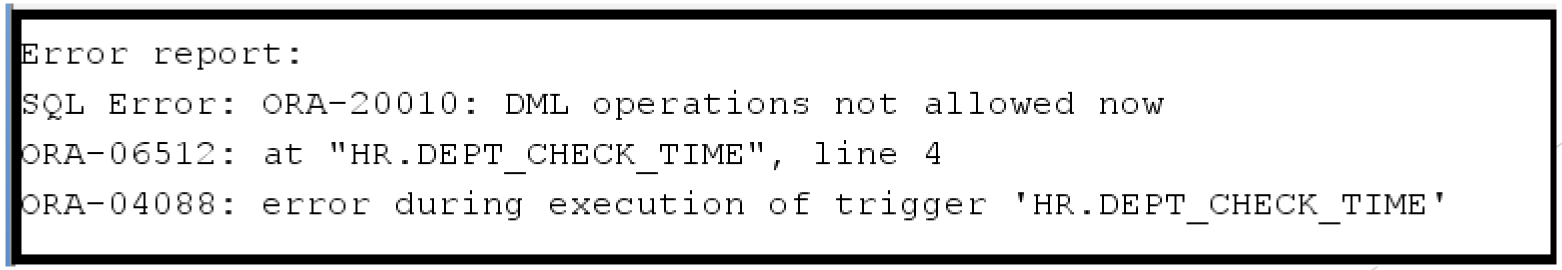
```
create or replace trigger dept_check_time
before
insert or update or delete
on DEPARTMENTS
begin

  if  to_number (to_char(sysdate,'hh24') ) not between 8 and 16 then
  raise_application_error(-20010, 'DML operations not allowed now ');
  end if;

end;
```

Dictionary views



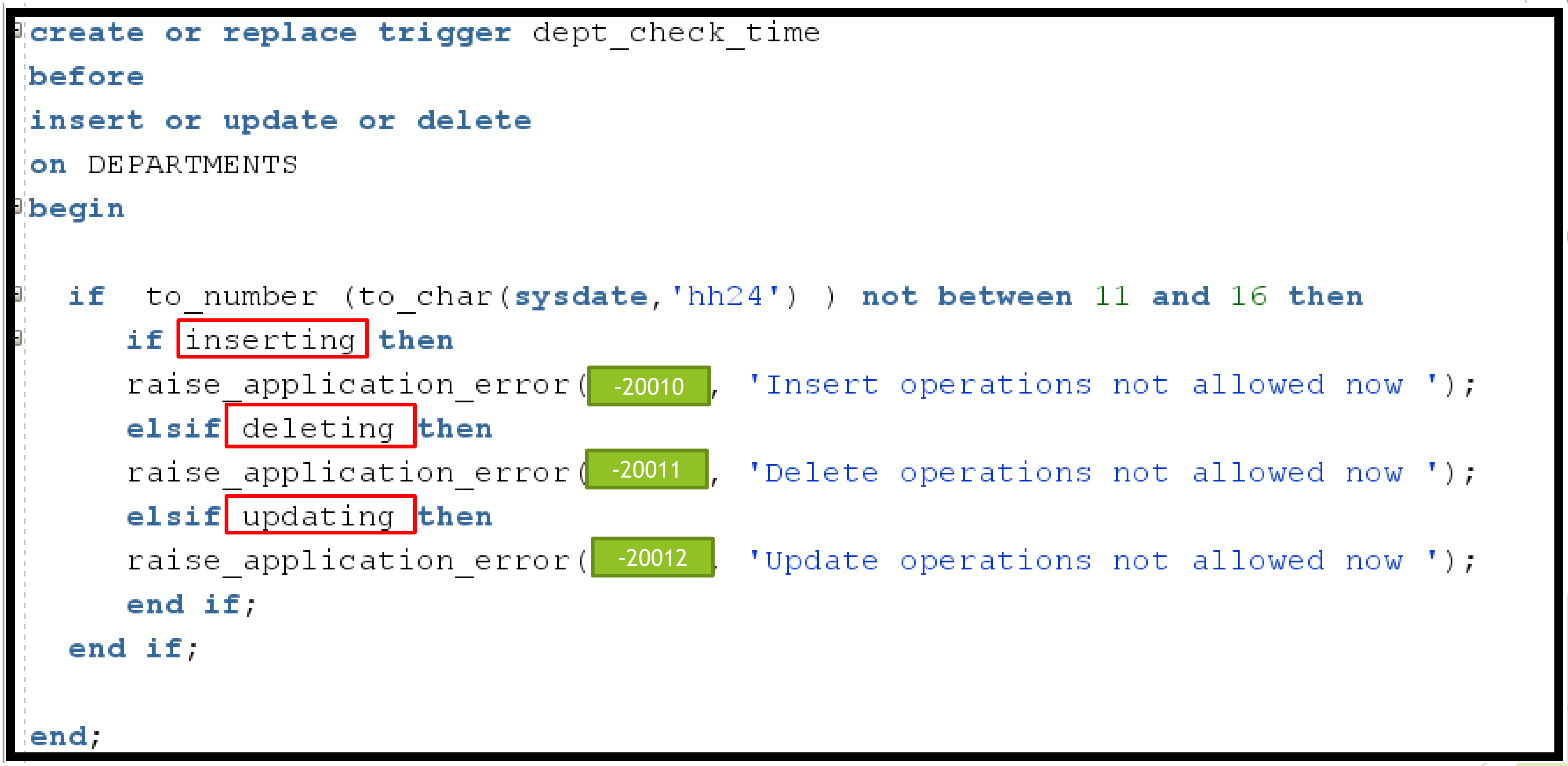
```
select * from user_objects
where object_name='DEPT_CHECK_TIME';

select * from user_triggers
where trigger_name='DEPT_CHECK_TIME';
```

If A user tried to do : delete from departments;   at 7:00 for example



```
Error report:
SQL Error: ORA-20010: DML operations not allowed now
ORA-06512: at "HR.DEPT_CHECK_TIME", line 4
ORA-04088: error during execution of trigger 'HR.DEPT_CHECK_TIME'
```

Using Conditional Predicates



```
create or replace trigger dept_check_time
before
insert or update or delete
on DEPARTMENTS
begin

  if  to_number (to_char(sysdate,'hh24') ) not between 11 and 16 then
     if inserting then
     raise_application_error( -20010 , 'Insert operations not allowed now ');
     elsif deleting then
     raise_application_error( -20011 , 'Delete operations not allowed now ');
     elsif updating then
     raise_application_error( -20012 , 'Update operations not allowed now ');
     end if;
  end if;

end;
```

OLD & New qualifiers

- Insert case :

Insert into dept (deptno, dname) values (1,'IT');

:new.deptno=1
:new.dname='IT'

- Update case :

:new.dname

Update dept
Set dname='IT dept'
Where depno=1

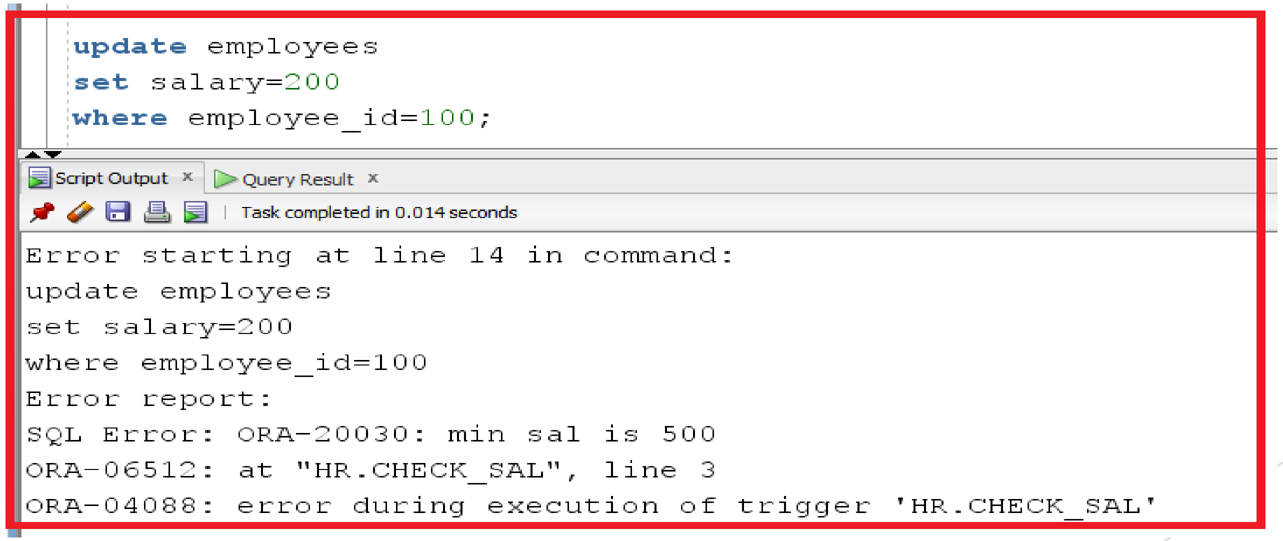
:old.deptno

Data Operations	Old Value	New Value
INSERT	NULL	Inserted value
UPDATE	Value before update	Value after update
DELETE	Value before delete	NULL

- delete case :
- All the columns are old values, there is no new

Row -Level triggers

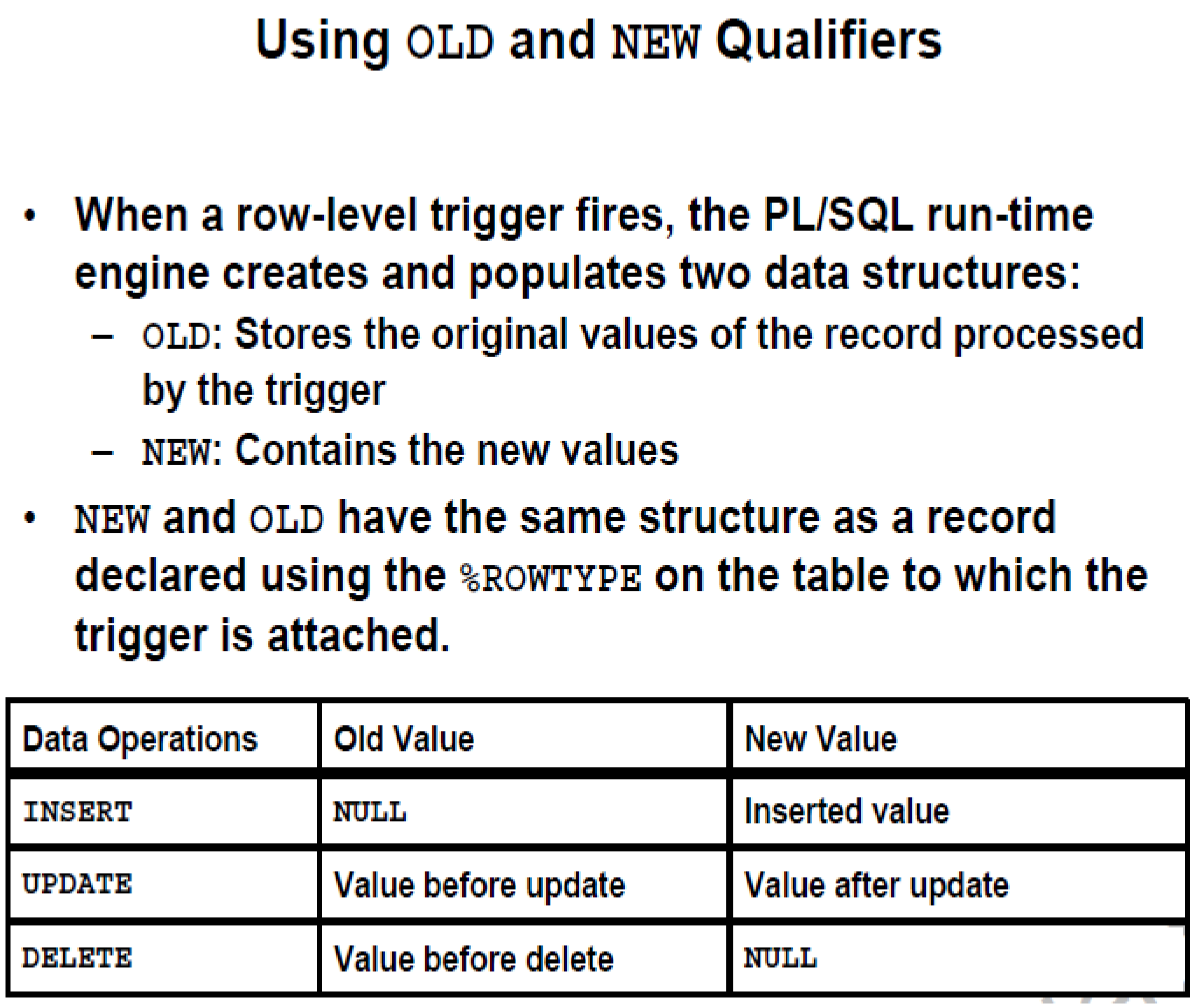
```
create or replace trigger check_sal
before
insert or update of salary
on
employees
for each row
begin
  if :new.salary<500 then
  raise_application_error(-20030, 'min sal is 500');
  end if;
end;
```



```
update employees
set salary=200
where employee_id=100;
```

```
Error starting at line 14 in command:
update employees
set salary=200
where employee_id=100
Error report:
SQL Error: ORA-20030: min sal is 500
ORA-06512: at "HR.CHECK_SAL", line 3
ORA-04088: error during execution of trigger 'HR.CHECK_SAL'
```

Row -Level triggers



Using `OLD` and `NEW` Qualifiers

- When a row-level trigger fires, the PL/SQL run-time engine creates and populates two data structures:
  - `OLD`: Stores the original values of the record processed by the trigger
  - `NEW`: Contains the new values
- `NEW` and `OLD` have the same structure as a record declared using the `%ROWTYPE` on the table to which the trigger is attached.

Data Operations	Old Value	New Value
`INSERT`	`NULL`	Inserted value
`UPDATE`	Value before update	Value after update
`DELETE`	Value before delete	`NULL`

Trigger-Firing Sequence:

→ BEFORE statement trigger

→ BEFORE row trigger

→ AFTER row trigger

→ AFTER statement trigger

Trigger-Firing Sequence:

Update x
Set y=???
Where ….

5 rows updated

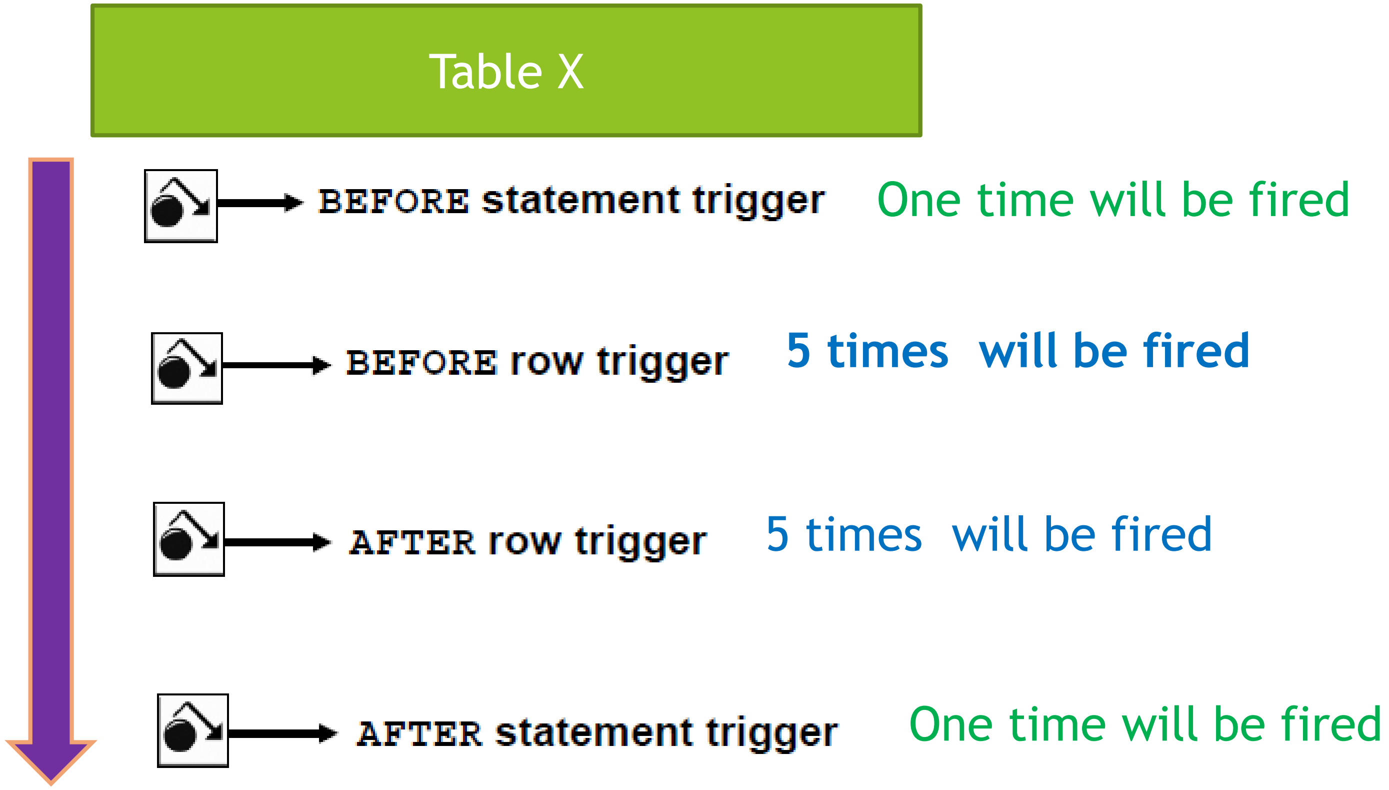


Table X

BEFORE statement trigger — One time will be fired

BEFORE row trigger — **5 times will be fired**

AFTER row trigger — 5 times will be fired

AFTER statement trigger — One time will be fired

Managing Triggers Using the ALTER and DROP SQL Statements

```
-- Disable or reenable a database trigger:

ALTER TRIGGER trigger_name DISABLE | ENABLE;
```

```
-- Disable or reenable all triggers for a table:

ALTER TABLE table_name DISABLE | ENABLE ALL TRIGGERS;
```

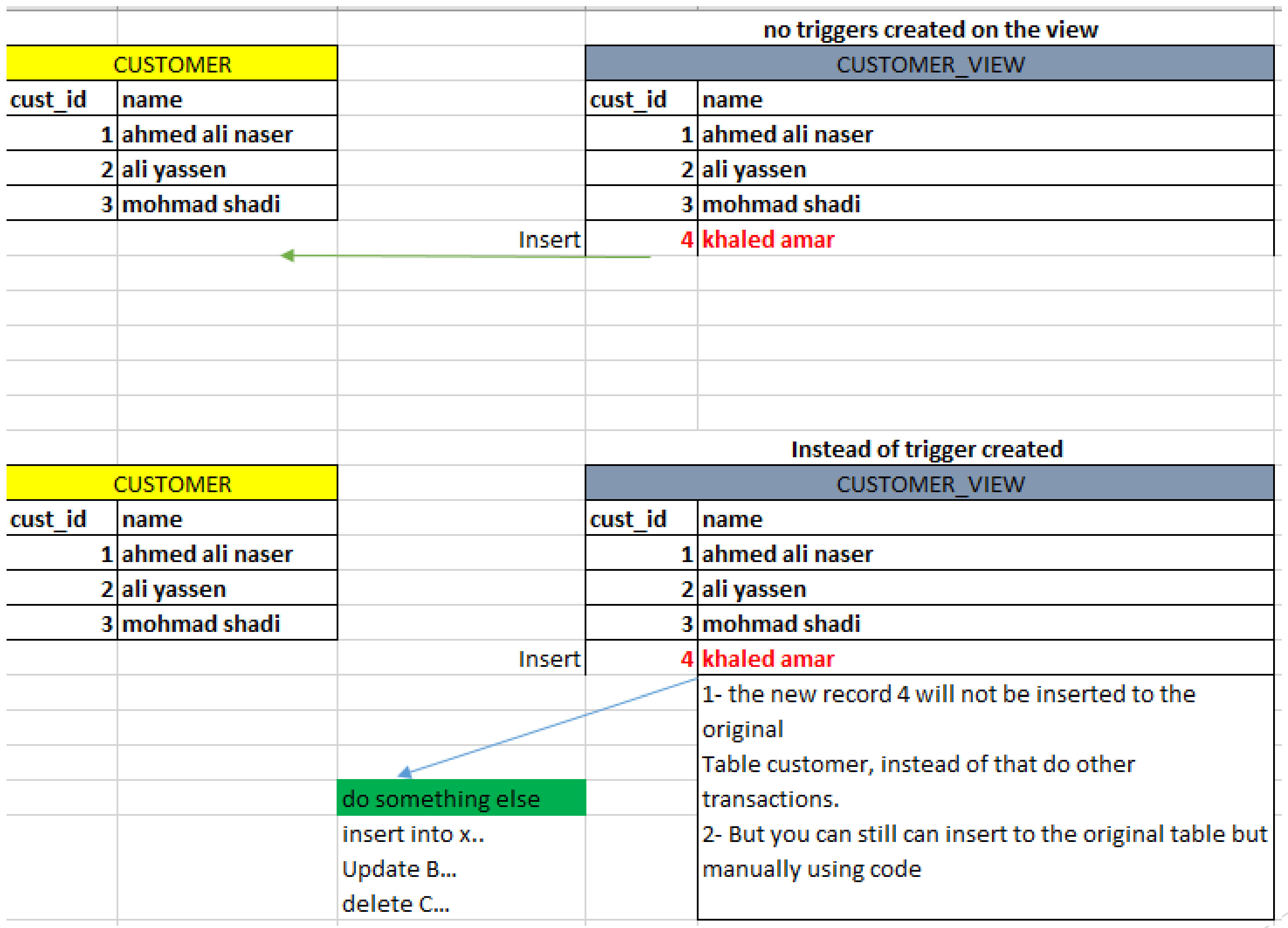
```
-- Recompile a trigger for a table:

ALTER TRIGGER trigger_name COMPILE;
```

```
-- Remove a trigger from the database:

DROP TRIGGER trigger_name;
```

Creating a Disabled Trigger

- Before Oracle Database 11*g*, if you created a trigger whose body had a PL/SQL compilation error, then DML to the table failed.
- In Oracle Database 11*g*, you can create a disabled trigger and then enable it only when you know it will be compiled successfully.

INSTEAD OF Triggers



no triggers created on the view

CUSTOMER	
cust_id	name
1	ahmed ali naser
2	ali yassen
3	mohmad shadi

CUSTOMER_VIEW	
cust_id	name
1	ahmed ali naser
2	ali yassen
3	mohmad shadi
4	khaled amar

Insert

Instead of trigger created

CUSTOMER	
cust_id	name
1	ahmed ali naser
2	ali yassen
3	mohmad shadi

CUSTOMER_VIEW	
cust_id	name
1	ahmed ali naser
2	ali yassen
3	mohmad shadi
4	khaled amar

Insert

do something else
insert into x..
Update B...
delete C...

1- the new record 4 will not be inserted to the original
Table customer, instead of that do other transactions.
2- But you can still can insert to the original table but manually using code

▶ Thank You