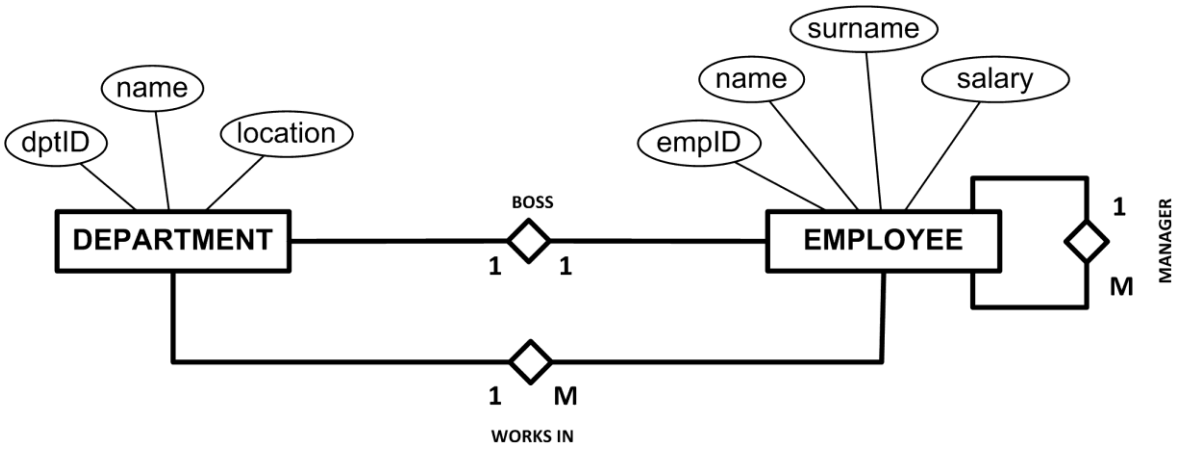
9 ТРИГЕРИ

ЗАДАЧА: Даден е ЕР дијаграмот за организационата шема на некоја фирма.



а) Да се дефинираат релациите за релациониот модел на дадениот ЕР.

б) Да се напише тригер TR_update_manager кој ќе се активира пред промената на шеф на некоја оддел, а преку кој ќе се постави шифрата на новиот шеф за менаџер на сите вработени кои работат во тој оддел.

в) Да се напише тригер кој ќе го мониторира атрибутот локација во табелата DEPARTMENT, односно пред да се промени неговата вредност, новата вредност треба да биде составена само од големи букви.

Додадете атрибут во релацијата DEPARTMENT со име totalSalary во кој ќе се чува вкупната плата од сите вработени во тој оддел. Напишете ги тригерите за следните настани и одржување на соодветната вредност за атрибутот totalSalary:

г) Вметнување на нови торки за вработени

д) Промена на платата на постоечки вработени

ѓ) Прераспределба на постоечки вработени од еден во друг оддел

е) Бришење на торки за вработени

Тригерите може да се користат за одржување на референтниот интегритет на базата на податоци.

ж) Да се напише тригер кој ќе го одржува референтниот интегритет на базата и ќе извршува каскадно бришење на сите информации поврзани со одделите кои се бришат од базата.

***РЕШЕНИЕ*:**

Општата синтакса за дефинирање на тригер е следната:

```
CREATE TRIGGER TriggerName
<triggerTiming> <triggerEvent> [OR <triggerEvent>]
[OF <TableName.Attribute>[,<TableName.Attribute>,…]]
ON <TableName>
[REFERENCING OLD as <alias_for_old_values>, NEW as <alias_for_new_values>]
FOR EACH {ROW | STATEMENT}
[WHEN (triggerCondition)]
BEGIN
     <triggerBody>
END;
```

Деловите ограничени во < > се пополнуваат со конкретни вредности во зависност од барањата за тригерот.

Деловите ограничени во [ ] се опционални и може да се изостават.

<triggerTiming> = **BEFORE | INSTEAD OF | AFTER**

<triggerEvent> = **INSERT | DELETE | UPDATE**

<TableName> е табела за која се однесува <triggerEvent>

<TableName.Attribute> е конкретен атрибут од <TableName> за кој се однесува <triggerEvent>

OLD и **NEW** се објекти од типот <TableName> кој ги содржат вредностите пред да се изврши <triggerEvent> (старите вредности во **OLD**) односно после извршувањето на <triggerEvent> (новите вредности во **NEW**)

Тригерот може да биде **ROW** или **STATEMENT**. Подразбираната варијанта е **STATEMENT**. Тригерот треба да биде **ROW** тригер доколку делот <triggerBody> треба да се изврши за секој ред од табелата <TableName> кој е засегнат од <triggerEvent>. Тригерот е **STATEMENT** доколку се извршува само еднаш на ниво на табела доколку се случи <triggerEvent> за табелата <TableName>.

WHEN делот се однесува на дополнителни услови кои треба да важат за да се извршат операциите од <triggerBody>.

а) **DEPARTMENT** (dptID, name, location, boss*)
EMPLOYEE (empID, name, surname, salary, manager*, dptID*)

б)
```
CREATE TRIGGER TR_update_manager
  BEFORE UPDATE OF boss ON DEPARTMENT
  FOR EACH ROW
  WHEN NEW.boss IS NOT NULL
  BEGIN
    UPDATE EMPLOYEE
    SET EMPLOYEE.manager = NEW.boss
    WHERE EMPLOYEE.dptID = OLD.dptID
  END;
```

в)
```
CREATE TRIGGER TR_caps_location
  BEFORE UPDATE OF location ON DEPARTMENT
  FOR EACH ROW
  WHEN NEW.location IS NOT NULL
  BEGIN
    SET NEW.location = upper(NEW.location)
  END;
```

г)
```
CREATE TRIGGER TR_total1
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  WHEN NEW.dptID IS NOT NULL
  BEGIN
    UPDATE DEPARTMENT
    SET totalSalary = totalSalary + NEW.salary
    WHERE dptID = NEW.dptID
  END;
```

д)
```
CREATE TRIGGER TR_total2
  AFTER UPDATE OF salary ON EMPLOYEE
  FOR EACH ROW
  WHEN NEW.dptID = OLD.dptID
  BEGIN
    UPDATE DEPARTMENT
    SET totalSalary = totalSalary + NEW.salary - OLD.salary
    WHERE dptID = NEW.dptID
  END;
```

ѓ)
```
CREATE TRIGGER TR_total3
AFTER UPDATE OF dptID ON EMPLOYEE
FOR EACH ROW
WHEN NEW.dptID IS NOT NULL
BEGIN
  UPDATE DEPARTMENT
  SET totalSalary = totalSalary + NEW.salary
  WHERE dptID = NEW.dptID;
  UPDATE DEPARTMENT
  SET totalSalary = totalSalary - OLD.salary
  WHERE dptID = OLD.dptID;
END;
```

е)
```
CREATE TRIGGER TR_total4
AFTER DELETE ON EMPLOYEE
FOR EACH ROW
BEGIN
  UPDATE DEPARTMENT
  SET totalSalary = totalSalary - OLD.salary
  WHERE dptID = OLD.dptID;
END;
```

ж)
```
CREATE TRIGGER TR_cascade_delete
AFTER DELETE ON DEPARTMENT
FOR EACH ROW
BEGIN
  DELETE FROM EMPLOYEE
  WHERE dptID = OLD.dptID;
END;
```