**Project Planner**

**Triggers**

1. **Triggers to prevent entering any past date as start date of task**

CREATE TRIGGER verify\_task\_startdate1

BEFORE INSERT ON Tasks

FOR EACH ROW

BEGIN

IF NEW.StartDate < CURDATE() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Start date must be current or future date';

END IF;

END //

CREATE TRIGGER verify\_task\_startdate2

BEFORE UPDATE ON Tasks

FOR EACH ROW

BEGIN

IF NEW.StartDate < CURDATE() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Start date must be current or future date';

END IF;

END //

CREATE TRIGGER verify\_project\_startdate1

BEFORE INSERT ON Projects

FOR EACH ROW

BEGIN

IF NEW.StartDate < CURDATE() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Start date must be current or future date';

END IF;

END //

CREATE TRIGGER verify\_project\_startdate2

BEFORE UPDATE ON Projects

FOR EACH ROW

BEGIN

IF NEW.StartDate < CURDATE() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Start date must be current or future date';

END IF;

END //

1. **Trigger to verify if end date arrives after start date**

CREATE TRIGGER check\_task\_startenddates\_insert

BEFORE INSERT ON Tasks

FOR EACH ROW

BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'End date must be at least one day after start date';

END IF;

END //

CREATE TRIGGER check\_task\_startenddates\_update

BEFORE UPDATE ON Tasks

FOR EACH ROW

BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'End date must be at least one day after start date';

END IF;

END //

-- Trigger to ensure EndDate is at least one day after StartDate for Milestones

CREATE TRIGGER check\_milestone\_startenddates\_insert

BEFORE INSERT ON Milestones

FOR EACH ROW

BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'End date must be at least one day after start date';

END IF;

END //

CREATE TRIGGER check\_milestone\_startenddates\_update

BEFORE UPDATE ON Milestones

FOR EACH ROW

BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'End date must be at least one day after start date';

END IF;

END //

CREATE TRIGGER project\_startenddates\_insert

BEFORE INSERT ON projects

FOR EACH ROW BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'End date must be at least one day after start date';

END IF;

END

CREATE TRIGGER project\_startenddates\_update

BEFORE UPDATE ON projects

FOR EACH ROW BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'End date must be at least one day after start date';

END IF;

END

1. **Trigger to adjust due date of milestone based on tasks**

CREATE TRIGGER task\_insert AFTER INSERT ON tasks

FOR EACH ROW BEGIN

UPDATE Milestones

SET DueDate = (

SELECT MAX(EndDate)

FROM Tasks

WHERE MilestoneID = NEW.MilestoneID

)

WHERE MilestoneID = NEW.MilestoneID;

END //

CREATE TRIGGER task\_update AFTER UPDATE ON tasks

FOR EACH ROW BEGIN

UPDATE Milestones

SET DueDate = (

SELECT MAX(EndDate)

FROM Tasks

WHERE MilestoneID = NEW.MilestoneID

)

WHERE MilestoneID = NEW.MilestoneID;

END //

1. **Trigger to adjust start date of milestone based on tasks**

CREATE TRIGGER after\_task\_insert\_set\_milestone\_startdate

AFTER INSERT ON tasks

FOR EACH ROW BEGIN

UPDATE Milestones

SET StartDate = (

SELECT MIN(StartDate)

FROM Tasks

WHERE MilestoneID = NEW.MilestoneID

)

WHERE MilestoneID = NEW.MilestoneID;

END

CREATE TRIGGER after\_task\_update\_set\_milestone\_startdate AFTER UPDATE ON tasks

FOR EACH ROW BEGIN

UPDATE Milestones

SET StartDate = (

SELECT MIN(StartDate)

FROM Tasks

WHERE MilestoneID = NEW.MilestoneID

)

WHERE MilestoneID = NEW.MilestoneID;

END

CREATE TRIGGER after\_task\_delete\_set\_milestone\_startdate AFTER DELETE ON tasks

FOR EACH ROW BEGIN

UPDATE Milestones

SET StartDate = (

SELECT MIN(StartDate)

FROM Tasks

WHERE MilestoneID = OLD.MilestoneID

)

WHERE MilestoneID = OLD.MilestoneID;

END

1. **Trigger to update milestone status based on individual task statuses**

CREATE TRIGGER check\_taskstatus\_insert AFTER INSERT ON tasks

FOR EACH ROW BEGIN

-- If a single task is in progress, set the milestone status to 'In Progress'

IF NEW.Status = 'In Progress' THEN

UPDATE Milestones

SET Status = 'In Progress'

WHERE MilestoneID = NEW.MilestoneID;

END IF;

-- If all tasks are completed, set the milestone status to 'Completed'

IF (SELECT COUNT(\*) FROM Tasks WHERE MilestoneID = NEW.MilestoneID AND Status != 'Completed') = 0 THEN

UPDATE Milestones

SET Status = 'Completed'

WHERE MilestoneID = NEW.MilestoneID;

END IF;

END

CREATE TRIGGER check\_taskstatus\_update AFTER UPDATE ON tasks

FOR EACH ROW BEGIN

-- If a single task is in progress, set the milestone status to 'In Progress'

IF NEW.Status = 'In Progress' THEN

UPDATE Milestones

SET Status = 'In Progress'

WHERE MilestoneID = NEW.MilestoneID;

END IF;

-- If all tasks are completed, set the milestone status to 'Completed'

IF (SELECT COUNT(\*) FROM Tasks WHERE MilestoneID = NEW.MilestoneID AND Status != 'Completed') = 0 THEN

UPDATE Milestones

SET Status = 'Completed'

WHERE MilestoneID = NEW.MilestoneID;

END IF;

END

CREATE TRIGGER check\_taskstatus\_delete AFTER DELETE ON tasks

FOR EACH ROW BEGIN

-- If a single task is in progress, set the milestone status to 'In Progress'

IF (SELECT COUNT(\*) FROM Tasks WHERE MilestoneID = OLD.MilestoneID AND Status = 'In Progress') > 0 THEN

UPDATE Milestones

SET Status = 'In Progress'

WHERE MilestoneID = OLD.MilestoneID;

END IF;

-- If all tasks are completed, set the milestone status to 'Completed'

IF (SELECT COUNT(\*) FROM Tasks WHERE MilestoneID = OLD.MilestoneID AND Status != 'Completed') = 0 THEN

UPDATE Milestones

SET Status = 'Completed'

WHERE MilestoneID = OLD.MilestoneID;

END IF;

END

CREATE TRIGGER after\_task\_insert\_update\_milestone\_status AFTER INSERT ON tasks

FOR EACH ROW BEGIN

DECLARE total\_tasks INT;

DECLARE completed\_tasks INT;

-- Count total tasks and completed tasks for the milestone

SELECT COUNT(\*), COUNT(CASE WHEN Status = 'Completed' THEN 1 END)

INTO total\_tasks, completed\_tasks

FROM Tasks

WHERE MilestoneID = NEW.MilestoneID;

-- If all tasks are completed and a new task is inserted as Not Started, set milestone status to In Progress

IF total\_tasks = completed\_tasks + 1 AND NEW.Status = 'Not Started' THEN

UPDATE Milestones

SET Status = 'In Progress'

WHERE MilestoneID = NEW.MilestoneID;

END IF;

END

CREATE TRIGGER after\_task\_update\_update\_milestone\_status AFTER UPDATE ON tasks

FOR EACH ROW BEGIN

DECLARE total\_tasks INT;

DECLARE completed\_tasks INT;

-- Count total tasks and completed tasks for the milestone

SELECT COUNT(\*), COUNT(CASE WHEN Status = 'Completed' THEN 1 END)

INTO total\_tasks, completed\_tasks

FROM Tasks

WHERE MilestoneID = NEW.MilestoneID;

-- If all tasks are completed and a task status changes to Not Started, set milestone status to In Progress

IF total\_tasks = completed\_tasks + 1 AND NEW.Status = 'Not Started' THEN

UPDATE Milestones

SET Status = 'In Progress'

WHERE MilestoneID = NEW.MilestoneID;

END IF;

END

CREATE TRIGGER after\_task\_delete\_update\_milestone\_status AFTER DELETE ON tasks

FOR EACH ROW BEGIN

DECLARE total\_tasks INT;

DECLARE completed\_tasks INT;

-- Count total tasks and completed tasks for the milestone

SELECT COUNT(\*), COUNT(CASE WHEN Status = 'Completed' THEN 1 END)

INTO total\_tasks, completed\_tasks

FROM Tasks

WHERE MilestoneID = OLD.MilestoneID;

-- If all tasks are completed and at least one task exists, set milestone status to Completed

IF total\_tasks = completed\_tasks AND total\_tasks > 0 THEN

UPDATE Milestones

SET Status = 'Completed'

WHERE MilestoneID = OLD.MilestoneID;

ELSE

-- If all tasks are not completed, set milestone status to In Progress

UPDATE Milestones

SET Status = 'In Progress'

WHERE MilestoneID = OLD.MilestoneID;

END IF;

END

1. **Triggers to update project end date based on individual milestone due dates**

CREATE TRIGGER milestone\_insert AFTER INSERT ON milestones

FOR EACH ROW BEGIN

UPDATE Projects

SET EndDate = (

SELECT MAX(DueDate)

FROM Milestones

WHERE ProjectID = NEW.ProjectID

)

WHERE ProjectID = NEW.ProjectID;

END

CREATE TRIGGER milestone\_update AFTER UPDATE ON milestones

FOR EACH ROW BEGIN

UPDATE Projects

SET EndDate = (

SELECT MAX(DueDate)

FROM Milestones

WHERE ProjectID = NEW.ProjectID

)

WHERE ProjectID = NEW.ProjectID;

END

CREATE TRIGGER milestone\_delete AFTER DELETE ON milestones

FOR EACH ROW BEGIN

UPDATE Projects

SET EndDate = (

SELECT MAX(DueDate)

FROM Milestones

WHERE ProjectID = OLD.ProjectID

)

WHERE ProjectID = OLD.ProjectID;

END

1. **Triggers to prevent the entry of milestone start date which is before start date of project it belongs to**

CREATE TRIGGER check\_milestone\_startdate\_insert

BEFORE INSERT ON milestones

FOR EACH ROW BEGIN

DECLARE projectStartDate DATE;

SELECT StartDate INTO projectStartDate FROM Projects WHERE ProjectID = NEW.ProjectID;

IF NEW.StartDate < projectStartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Milestone start date must be on or after the project start date';

END IF;

END

CREATE TRIGGER check\_milestone\_startdate\_update BEFORE UPDATE ON milestones

FOR EACH ROW BEGIN

DECLARE projectStartDate DATE;

SELECT StartDate INTO projectStartDate FROM Projects WHERE ProjectID = NEW.ProjectID;

IF NEW.StartDate < projectStartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Milestone start date must be on or after the project start date';

END IF;

END

1. **Trigger to update project status based on individual milestone statuses**

CREATE TRIGGER check\_milestonestatus\_insert

AFTER INSERT ON Milestones

FOR EACH ROW

BEGIN

-- If a single milestone is in progress, set the project status to 'In Progress'

IF NEW.Status = 'In Progress' THEN

UPDATE Projects

SET Status = 'In Progress'

WHERE ProjectID = NEW.ProjectID;

END IF;

-- If all milestones are completed, set the project status to 'Completed'

IF (SELECT COUNT(\*) FROM Milestones WHERE ProjectID = NEW.ProjectID AND Status != 'Completed') = 0 THEN

UPDATE Projects

SET Status = 'Completed'

WHERE ProjectID = NEW.ProjectID;

END IF;

END //

CREATE TRIGGER check\_milestonestatus\_update AFTER UPDATE ON milestones

FOR EACH ROW BEGIN

-- If a single milestone is in progress, set the project status to 'In Progress'

IF NEW.Status = 'In Progress' THEN

UPDATE Projects

SET Status = 'In Progress'

WHERE ProjectID = NEW.ProjectID;

END IF;

-- If all milestones are completed, set the project status to 'Completed'

IF (SELECT COUNT(\*) FROM Milestones WHERE ProjectID = NEW.ProjectID AND Status != 'Completed') = 0 THEN

UPDATE Projects

SET Status = 'Completed'

WHERE ProjectID = NEW.ProjectID;

END IF;

END //

CREATE TRIGGER check\_milestonestatus\_delete AFTER DELETE ON milestones

FOR EACH ROW BEGIN

-- If a single milestone is in progress, set the project status to 'In Progress'

IF (SELECT COUNT(\*) FROM Milestones WHERE ProjectID = OLD.ProjectID AND Status = 'In Progress') > 0 THEN

UPDATE Projects

SET Status = 'In Progress'

WHERE ProjectID = OLD.ProjectID;

END IF;

-- If all milestones are completed, set the project status to 'Completed'

IF (SELECT COUNT(\*) FROM Milestones WHERE ProjectID = OLD.ProjectID AND Status != 'Completed') = 0 THEN

UPDATE Projects

SET Status = 'Completed'

WHERE ProjectID = OLD.ProjectID;

END IF;

END //

1. **Triggers to verify if the task is assigned to task completer and not any task assigner**

CREATE TRIGGER check\_assignedto\_user\_insert BEFORE INSERT ON tasks

FOR EACH ROW BEGIN

DECLARE userType INT;

SELECT UserType INTO userType FROM Users WHERE UserID = NEW.AssignedTo;

IF userType != 2 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'User cannot be assigned the task';

END IF;

END

CREATE TRIGGER check\_assignedto\_user\_update BEFORE UPDATE ON tasks

FOR EACH ROW BEGIN

DECLARE userType INT;

SELECT UserType INTO userType FROM Users WHERE UserID = NEW.AssignedTo;

IF userType != 2 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'User cannot be assigned the task';

END IF;

END

1. **Triggers to prevent entry of projects created by task completers**

CREATE TRIGGER restrict\_created\_by

BEFORE INSERT ON projects

FOR EACH ROW BEGIN

DECLARE user\_type INT;

-- Retrieve the UserType of the user attempting to create the project

SELECT UserType INTO user\_type FROM Users

WHERE UserID = NEW.Created\_By;

-- Check if the UserType is 1 (assuming UserType 1 is for the required user type)

IF user\_type != 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'User has no right to create project';

END IF;

END