TABLES:

CREATE TABLE Account

(

AccountId INT IDENTITY(1,1) CONSTRAINT pk\_User PRIMARY KEY,

Name VARCHAR(256) NOT NULL,

PhoneNumber VARCHAR(256) NOT NULL,

Email VARCHAR(256) UNIQUE NOT NULL CONSTRAINT check\_Email CHECK (Email LIKE '%@%.%'),

Login VARCHAR(256) UNIQUE NOT NULL,

PasswordHash VARCHAR(MAX) NOT NULL

);

CREATE TABLE Role

(

RoleId INT IDENTITY(1,1) CONSTRAINT pk\_Role PRIMARY KEY,

Name VARCHAR(256) UNIQUE NOT NULL

);

CREATE TABLE Conference

(

ConferenceId INT IDENTITY(1,1) CONSTRAINT pk\_Conference PRIMARY KEY,

Title VARCHAR(200) NOT NULL,

Description VARCHAR(MAX),

Place VARCHAR(400) NOT NULL,

BeginDate DATE NOT NULL,

EndDate DATE NOT NULL,

CONSTRAINT checkDatesInterval2 CHECK (beginDate <= endDate)

);

CREATE TABLE ConferenceStaff

(

AccountId INT NOT NULL,

FOREIGN KEY (AccountId) REFERENCES Account(AccountId),

RoleId INT NOT NULL,

FOREIGN KEY (RoleId) REFERENCES Role(RoleId),

ConferenceId INT NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

CONSTRAINT pk\_ConferenceStaff PRIMARY KEY (AccountId, RoleId, ConferenceId)

);

CREATE TABLE Message

(

MessageId INT IDENTITY(1,1) CONSTRAINT pk\_Message PRIMARY KEY,

SenderId INT NOT NULL FOREIGN KEY (SenderId) REFERENCES Account(AccountId),

ReceiverId INT NOT NULL FOREIGN KEY (ReceiverId) REFERENCES Account(AccountId),

Content VARCHAR(170) NOT NULL,

Date DATETIME NOT NULL DEFAULT GETDATE(),

CONSTRAINT ck\_MessageToYourselfAreNotAllowed CHECK (SenderId <> ReceiverId)

);

CREATE TABLE LastMessages (

PairID INT identity(1,1) PRIMARY key,

FirstId INT NOT NULL FOREIGN KEY (FirstId) REFERENCES Account(AccountId),

SecondId INT NOT NULL FOREIGN KEY (FirstId) REFERENCES Account(AccountId),

LastDate DATETIME NOT NULL,

LastMessage VARCHAR(280) NOT NULL,

firstIdReceived BIT DEFAULT 0 NOT NULL ,

secondIdReceived BIT DEFAULT 0 NOT NULL,

CONSTRAINT ck\_distinctPairOnly CHECK (FirstId <> SecondID)

);

CREATE TABLE Building

(

BuildingID INT IDENTITY(1,1) CONSTRAINT pk\_Building PRIMARY KEY,

Name VARCHAR(200) NOT NULL,

Address VARCHAR(200) UNIQUE NOT NULL

);

CREATE TABLE Room

(

RoomID INT IDENTITY(1,1) CONSTRAINT pk\_Room PRIMARY KEY,

Code VARCHAR(200) UNIQUE NOT NULL,

BuildingID INT NOT NULL CONSTRAINT fk\_RoomBuilding FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)

);

CREATE TABLE Task(

TaskId INT IDENTITY(1,1) CONSTRAINT pk\_Task PRIMARY KEY,

ConferenceId INT NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

EmployeeId INT NOT NULL,

FOREIGN KEY (EmployeeId) REFERENCES Account(AccountId),

Title VARCHAR(150) NOT NULL,

Description VARCHAR(2000) NOT NULL,

BeginDate DATETIME NOT NULL CONSTRAINT checkTskBegin CHECK (DATEDIFF(MONTH, BeginDate, SYSDATETIME()) < 6),

EndDate DATETIME NOT NULL,

CONSTRAINT checkDatesInterval CHECK (DATEDIFF(minute, beginDate, endDate) > 10 AND DATEDIFF(hour, beginDate, endDate) < 12),

ManagerId INT NOT NULL,

FOREIGN KEY (ManagerId) REFERENCES Account(AccountId)

);

CREATE TABLE Author

(

AuthorId INT IDENTITY(1,1) CONSTRAINT pk\_Author PRIMARY KEY,

AccountId INT UNIQUE NOT NULL,

FOREIGN KEY (AccountId) REFERENCES Account(AccountId),

FirstName VARCHAR(256) NOT NULL,

LastName VARCHAR(256) NOT NULL,

Title VARCHAR(10) CONSTRAINT check\_Title CHECK (Title IN ('Bachelor', 'Master', 'Doctor', 'Professor')),

FieldOfStudy VARCHAR(256) NOT NULL

);

CREATE TABLE EmergencyInfo

(

EmergencyInfoId INT IDENTITY(1,1) CONSTRAINT pk\_EmergencyInfo PRIMARY KEY,

ConferenceId INT UNIQUE NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

EmergencyNum VARCHAR(100),

EmergencyInfo VARCHAR(MAX)

);

CREATE TABLE AccommodationInfo

(

AccommodationInfoId INT IDENTITY(1,1) CONSTRAINT pk\_AccommodationInfo PRIMARY KEY,

ConferenceId INT NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

PlaceName VARCHAR(100),

Description VARCHAR(500),

Currency VARCHAR(5),

City VARCHAR(100),

CityDesc VARCHAR(500)

);

CREATE TABLE TravelInfo

(

TravelInfoId INT IDENTITY(1,1) CONSTRAINT pk\_TravelInfo PRIMARY KEY,

ConferenceId INT NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

Title VARCHAR(100),

AirportRoad VARCHAR(1000),

AirportRoadTime INT,

RailwayRoad VARCHAR(1000),

RailwayRoadTime INT,

TaxiNum VARCHAR(15)

);

CREATE TABLE Event

(

EventId INT IDENTITY(1,1) CONSTRAINT pk\_Event PRIMARY KEY,

ConferenceId INT NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

RoomId INT,

FOREIGN KEY (RoomId) REFERENCES Room(RoomId),

BeginDate DATETIME NOT NULL,

EndDate DATETIME NOT NULL,

Title VARCHAR(200) NOT NULL,

Description VARCHAR(2000) NOT NULL,

CONSTRAINT date\_ck CHECK (DATEDIFF(minute, beginDate, endDate) > 0 AND DATEDIFF(hour,beginDate,endDate)<=5)

);

CREATE TABLE WelcomePackReceiver

(

WelcomePackReceiverId INT IDENTITY(1,1) CONSTRAINT pk\_WelcomePackReceiver PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(150),

Type VARCHAR(300) NOT NULL CONSTRAINT check\_Type\_Guest CHECK (Type IN ('Guest', 'Special guest','Author', 'Reviewer')),

GetGift BIT NOT NULL

);

CREATE TABLE ConferenceBuilding

(

ConferenceId INT NOT NULL,

FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

BuildingId INT NOT NULL,

FOREIGN KEY (BuildingId) REFERENCES Building(BuildingId),

CONSTRAINT pk\_ConferenceBuilding PRIMARY KEY (BuildingId, ConferenceId)

);

CREATE TABLE Article

(

ArticleId INT IDENTITY(1,1) CONSTRAINT pk\_Article PRIMARY KEY,

Topic VARCHAR(256) NOT NULL,

PresentationId INT,

SpecialSessionId INT,

Status VARCHAR(10) DEFAULT 'submitted' CONSTRAINT check\_status CHECK(Status IN('submitted', 'rejected', 'accepted')) NOT NULL,

AcceptanceDate DATE,

ConferenceID INT NOT NULL,

FOREIGN KEY (ConferenceID) REFERENCES Conference(ConferenceID)

);

CREATE TABLE ArticleAuthor

(

ArticleId INT NOT NULL,

FOREIGN KEY (ArticleId) REFERENCES Article(ArticleId),

AuthorId INT NOT NULL,

FOREIGN KEY (AuthorId) REFERENCES Author(AuthorId),

CONSTRAINT pk\_ArticleAuthor PRIMARY KEY (ArticleId, AuthorId)

);

CREATE TABLE Review

(

ReviewId INT IDENTITY(1,1) CONSTRAINT pk\_Review PRIMARY KEY,

ArticleId INT NOT NULL,

FOREIGN KEY (ArticleId) REFERENCES Article(ArticleId),

ReviewerId INT NOT NULL,

FOREIGN KEY (ReviewerId) REFERENCES Account(AccountId),

ReviewDate DATE NOT NULL CONSTRAINT check\_RevDate CHECK (GETDATE()<= ReviewDate),

Grade DEC(2,1) CHECK (Grade IN (0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0)) NOT NULL,

Title VARCHAR(100) NOT NULL,

Comment VARCHAR(1000)

);

CREATE TABLE Session

(

SessionId INT IDENTITY(1,1) CONSTRAINT pk\_SpecialSession PRIMARY KEY,

ConferenceId INT NOT NULL CONSTRAINT fk\_SpecialSessionConference FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

ChairId INT NOT NULL CONSTRAINT fk\_SpecialSessionChair FOREIGN KEY (ChairId) REFERENCES Account(AccountId),

RoomId INT NOT NULL CONSTRAINT fk\_SessionRoom FOREIGN KEY (RoomId) REFERENCES Room(RoomId),

BeginDate DATETIME NOT NULL,

EndDate DATETIME NOT NULL,

CONSTRAINT ck\_SessionProperDateTimeOrder CHECK (BeginDate < EndDate),

Title VARCHAR(200) NOT NULL,

Description VARCHAR(500),

);

CREATE TABLE SpecialSession

(

SpecialSessionId INT IDENTITY(1,1) CONSTRAINT pk\_Session PRIMARY KEY,

ConferenceId INT NOT NULL CONSTRAINT fk\_SessionConference FOREIGN KEY (ConferenceId) REFERENCES Conference(ConferenceId),

ChairId INT NOT NULL CONSTRAINT fk\_SessionChair FOREIGN KEY (ChairId) REFERENCES Account(AccountId),

BeginDate DATETIME NOT NULL,

EndDate DATETIME NOT NULL,

CONSTRAINT ck\_SpecialSessionProperDateTimeOrder CHECK (BeginDate < EndDate),

RoomId INT NOT NULL CONSTRAINT fk\_SpecialSessionRoom FOREIGN KEY (RoomId) REFERENCES Room(RoomId),

Title VARCHAR(200) NOT NULL,

Description VARCHAR(300)

);

CREATE TABLE Presentation

(

PresentationId INT IDENTITY(1,1) CONSTRAINT pk\_Presentation PRIMARY KEY,

PresenterId INT NOT NULL CONSTRAINT fk\_PresentationPresenter FOREIGN KEY (PresenterID) REFERENCEs Account(AccountId),

Title VARCHAR(200) NOT NULL,

Description VARCHAR(500),

ArticleId INT NOT NULL CONSTRAINT fk\_PresentationArticle FOREIGN KEY (ArticleId) REFERENCES Article(ArticleId),

SessionId INT CONSTRAINT fk\_PresentationSession FOREIGN KEY (SessionId) REFERENCES Session(SessionId),

SpecialSessionId INT CONSTRAINT fk\_PresentationSpecialSession FOREIGN KEY (SpecialSessionId) REFERENCES SpecialSession(SpecialSessionId),

CONSTRAINT ck\_BelongsToEitherSessionOrSpecialSessionButNotBoth CHECK (([SessionId] IS NULL AND [SpecialSessionId] IS NOT NULL) OR ([SessionId] IS NOT NULL AND [SpecialSessionId] IS NULL) OR

([SessionId] IS NULL AND [SpecialSessionId] IS NULL)),

Grade DECIMAL(3,2) CONSTRAINT ck\_Grade CHECK(Grade >=0 AND Grade <=10)

);

CREATE TABLE Award

(

AwardId INT IDENTITY(1,1) CONSTRAINT pk\_Award PRIMARY KEY,

PresentationId INT UNIQUE NOT NULL,

FOREIGN KEY (PresentationId) REFERENCES Presentation(PresentationId),

Date DATE NOT NULL

);

CREATE TABLE Submission

(

SubmissionId INT IDENTITY(1,1) CONSTRAINT pk\_Submission PRIMARY KEY,

ArticleId INT NOT NULL,

FOREIGN KEY (ArticleId) REFERENCES Article(ArticleId),

ArticleFile VARBINARY(MAX) NOT NULL,

SubmissionDate DATE NOT NULL DEFAULT GETDATE()

);

FUNCTIONS:

CREATE FUNCTION CheckUpper(@Name VARCHAR(MAX))

RETURNS BIT

WITH EXECUTE AS CALLER

AS

BEGIN

DECLARE @toReturn INT;

IF BINARY\_CHECKSUM(@Name) = BINARY\_CHECKSUM(UPPER(@Name))

SELECT @toReturn = 1;

ELSE

SELECT @toReturn = 0;

RETURN @toReturn

END;

GO

CREATE FUNCTION GetConferenceEndDate(@ConferenceId INT)

RETURNS DATE

WITH EXECUTE AS CALLER

AS

BEGIN

RETURN (SELECT EndDate

FROM Conference

WHERE ConferenceId=@ConferenceId)

END;

GO

CREATE FUNCTION GetConferenceBeginDate(@ConferenceId INT)

RETURNS DATE

WITH EXECUTE AS CALLER

AS

BEGIN

RETURN (SELECT BeginDate

FROM Conference

WHERE ConferenceId=@ConferenceId)

END;

ALTER:

ALTER TABLE Article

ADD FOREIGN KEY (PresentationId) REFERENCES Presentation(PresentationId);

ALTER TABLE Article

ADD FOREIGN KEY (SpecialSessionId) REFERENCES SpecialSession(SpecialSessionId);

ALTER TABLE EmergencyInfo

ADD CONSTRAINT check\_UpperInfo CHECK(dbo.CheckUpper(EmergencyInfo)=1);

ALTER TABLE Task

ADD CONSTRAINT checkTaskEnd CHECK (DATEDIFF(WEEK, dbo.GetConferenceEndDate(ConferenceId), EndDate) <= 1);

ALTER TABLE Event

ADD CONSTRAINT checkDatesWithConf CHECK (dbo.GetConferenceBeginDate(ConferenceId) <= BeginDate AND dbo.GetConferenceEndDate(ConferenceId) >= EndDate);

TRIGGERS:

CREATE TRIGGER Over4Authors ON Article

AFTER INSERT, UPDATE

AS

IF EXISTS (SELECT COUNT(ArticleId)

FROM ArticleAuthor

HAVING COUNT(ArticleId)>4)

BEGIN

RAISERROR ('One article cannot have more than 4 authors!', 16, 1);

ROLLBACK TRANSACTION;

RETURN

END;

GO

CREATE TRIGGER Over5Accommodation ON AccommodationInfo

AFTER INSERT, UPDATE

AS

IF EXISTS (SELECT ConferenceId

FROM AccommodationInfo

GROUP BY ConferenceId

HAVING COUNT(AccommodationInfoId)>5)

BEGIN

RAISERROR ('One conference cannot be assigned more than 5 pieces of accommodation info!', 16, 1);

ROLLBACK TRANSACTION;

RETURN

END;

GO

CREATE TRIGGER IsReviewer ON Review

AFTER INSERT, UPDATE

AS

IF EXISTS (SELECT r.ReviewId

FROM Review r

WHERE r.ReviewId NOT IN (SELECT r.ReviewId

FROM Review r

JOIN Article a ON r.ArticleId=a.ArticleId

LEFT JOIN ConferenceStaff cs ON (cs.ConferenceId=a.ConferenceId AND cs.AccountId=r.ReviewerId)

LEFT JOIN Role rl ON rl.RoleId=cs.RoleId

WHERE rl.Name='REVIEWER'))

BEGIN

RAISERROR ('The user needs to be a Reviewer in order to leave reviews', 16, 1);

ROLLBACK TRANSACTION;

RETURN

END;

GO

CREATE TRIGGER IsStaffMember ON Task

AFTER INSERT, UPDATE

AS

IF EXISTS (SELECT t.TaskId

FROM Task t

WHERE t.TaskId NOT IN (SELECT t.TaskId

FROM Task t

LEFT JOIN ConferenceStaff cs ON (cs.ConferenceId=t.ConferenceId AND cs.AccountId=t.EmployeeId)

LEFT JOIN Role rl ON rl.RoleId=cs.RoleId

WHERE rl.Name='CONFERENCE STAFF MEMBER'))

BEGIN

RAISERROR ('The user needs to be a Conference Staff Member in order to be assigned to task', 16, 1);

ROLLBACK TRANSACTION;

RETURN

END;

GO

CREATE TRIGGER IsStaffManager ON Task

AFTER INSERT, UPDATE

AS

IF EXISTS (SELECT t.TaskId

FROM Task t

WHERE t.TaskId NOT IN (SELECT t.TaskId

FROM Task t

LEFT JOIN ConferenceStaff cs ON (cs.ConferenceId=t.ConferenceId AND cs.AccountId=t.ManagerId)

LEFT JOIN Role rl ON rl.RoleId=cs.RoleId

WHERE rl.Name='CONFERENCE STAFF MANAGER'))

BEGIN

RAISERROR ('The user needs to be a Conference Staff Manager in order to be assigned to task as manager', 16, 1);

ROLLBACK TRANSACTION;

RETURN

END;

GO

CREATE TRIGGER CheckEventForOverlappingEvent

ON [dbo].Event

AFTER INSERT,UPDATE

AS

IF EXISTS(

Select begindate, enddate

from Event

where ConferenceId in (select conferenceid from inserted) AND

( (SELECT BeginDate FROM inserted) > BeginDate AND (SELECT BeginDate FROM inserted) < EndDate) OR

( (SELECT EndDate FROM inserted) > BeginDate AND (SELECT EndDate FROM inserted) < EndDate) or

( (SELECT BeginDate FROM inserted) = BeginDate AND (SELECT EndDate FROM inserted) = EndDate)

) OR

(select count( \* )

FROM Event

WHERE ConferenceId in (select conferenceid from inserted) AND

((SELECT BeginDate FROM inserted) = BeginDate AND (SELECT EndDate FROM inserted) = EndDate) ) > 1

BEGIN

RAISERROR('There is already an event in this time slot', 16, 1)

ROLLBACK

END

GO

/\*CREATE TRIGGER CheckEventForOverlappingPresentation

ON [dbo].Event

AFTER INSERT,UPDATE

AS

IF EXISTS(

Select begindate, enddate

from Presentation p

join Session s on s.SessionId = p.SessionId

where s.ConferenceId in (select conferenceid from inserted) AND

( (SELECT BeginDate FROM inserted) > BeginDate AND (SELECT BeginDate FROM inserted) < EndDate) OR

( (SELECT EndDate FROM inserted) > BeginDate AND (SELECT EndDate FROM inserted) < EndDate) or

( (SELECT BeginDate FROM inserted) = BeginDate AND (SELECT EndDate FROM inserted) = EndDate)

)

BEGIN

RAISERROR('There is at least one presentation during this event dedicated time.', 16, 1)

ROLLBACK

END

GO

CREATE TRIGGER CheckPresentationForOverlappingEvent

ON [dbo].[Presentation ]

AFTER INSERT,UPDATE

AS

IF EXISTS(

Select begindate, enddate

from Event

where conferenceId in (SELECT ConferenceId

FROM inserted p

join Session s on s.SessionId = p.SessionId) AND

( (SELECT BeginDate FROM inserted) > BeginDate AND (SELECT BeginDate FROM inserted) < EndDate) OR

( (SELECT EndDate FROM inserted) > BeginDate AND (SELECT EndDate FROM inserted) < EndDate) or

( (SELECT BeginDate FROM inserted) = BeginDate AND (SELECT EndDate FROM inserted) = EndDate)

)

BEGIN

RAISERROR('Presentation cannot happen in the same time as an event', 16, 1)

ROLLBACK

END

GO

CREATE TRIGGER CheckTwoPresentationsForOverlapingTimeAndLocation

ON [dbo].[Presentation ]

AFTER INSERT,UPDATE

AS

IF EXISTS(

SELECT \*

FROM Presentation

WHERE RoomId IN (SELECT RoomId FROM inserted) AND

(( (SELECT BeginDate FROM inserted) > BeginDate AND (SELECT BeginDate FROM inserted) < EndDate) OR

( (SELECT EndDate FROM inserted) > BeginDate AND (SELECT EndDate FROM inserted) < EndDate))

) OR

(select count( \* )

FROM Presentation

WHERE RoomId IN (SELECT RoomId FROM inserted) AND

((SELECT BeginDate FROM inserted) = BeginDate AND (SELECT EndDate FROM inserted) = EndDate) ) > 1

BEGIN

RAISERROR('No two presentation can take place in the same room at the same time', 16, 1)

ROLLBACK

END

GO\*/