МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДРАСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет  
информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Проект

по учебной дисциплине: проектирование баз данных

Проектирование базы данных “Чат”

Выполнил студент группы №M3209  
Головко Артем Антонович

Проверил:  
Повышев Владислав Вячеславович

Санкт Петербург

2017

**Предметная область:**

Чат – виртуальное место встречи в Интернете, где люди по средствам сети Интернет могут общаться друг с другом. Чат будет создан для всех желающих, кто имеет доступ к Интернет и желанием, не встречаясь лично передавать информацию – для этого нужно только зарегистрироваться и использовать свой аккаунт для связи с другими. Информация может быть разного характера – текст, фото, видео, документы и прочее. Также пользователь сможет устраивать конференции, для общения с группой пользователей.

**Возможности:**

1. Регистрировать аккаунт
2. Входить в свой аккаунт
3. Настройка личной информации пользователя:
   1. имени
   2. изменить фото
   3. контактная информация (жесткий перечень)
      1. скайп
      2. гит
      3. номер телефона
      4. о себе
      5. свой сайт
   4. изменение пароля
4. Прямое сообщение пользователю
5. Создание конференций. Либо по ссылке конференции и пользователь присоединяется, либо по приглашению – тогда надо ввести в настройки, кто может приглашать тебя в группы, во избежание спама
6. Возможность отправки сообщений, видео, документов, фото
7. Поиск по сообщениям, медиа файлам, чатам

Дополнительно:

1. Добавлять пользователей в черный список
2. Добавление пользователей в контакты по его личной ссылке
3. В настройках указывается, кто может отправлять сообщения
4. Позаботиться о безопасности шифрования. Возможно шифровать на стороне клиента

**Сущности**

1. *Медиа*. Сущность, которая содержит в себе пути до медиа файлов
2. *Пользователь*. Аккаунт пользователя. Содержит всю информацию о нем: логин, пароль, имя, фото и личную информацию
3. *Сообщение*. Содержит отправителя, текст и может содержать медиа файл
4. *Диалог*. Сущность, которая может быть одной из двух сущностей: Прямой чат или Конференция-чат. Принадлежит конкретному пользователю

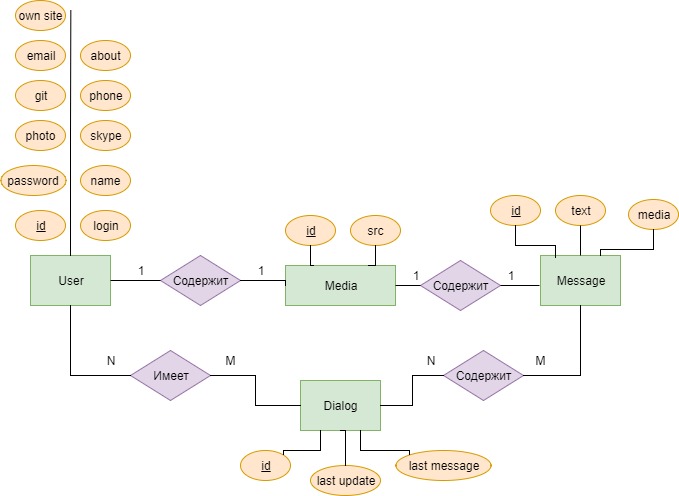
5) *Прямой чат*. Чат между пользователем и пользователем. Содержит в себе сообщения. Принадлежит конкретному пользователю

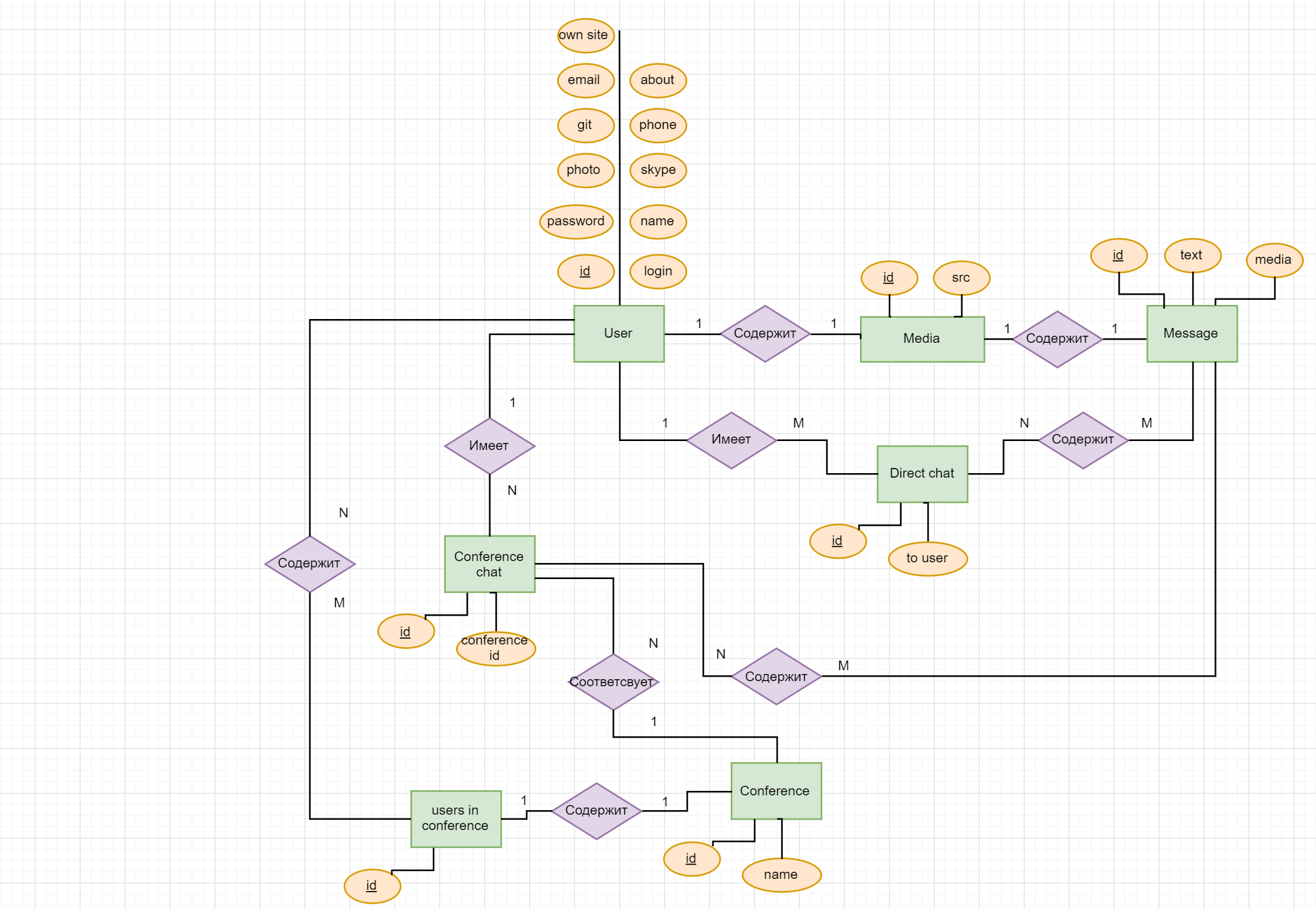
6) *Конференция чат*. Чат между группой пользователей. Содержит в себе сообщения. Принадлежит конкретному пользователю

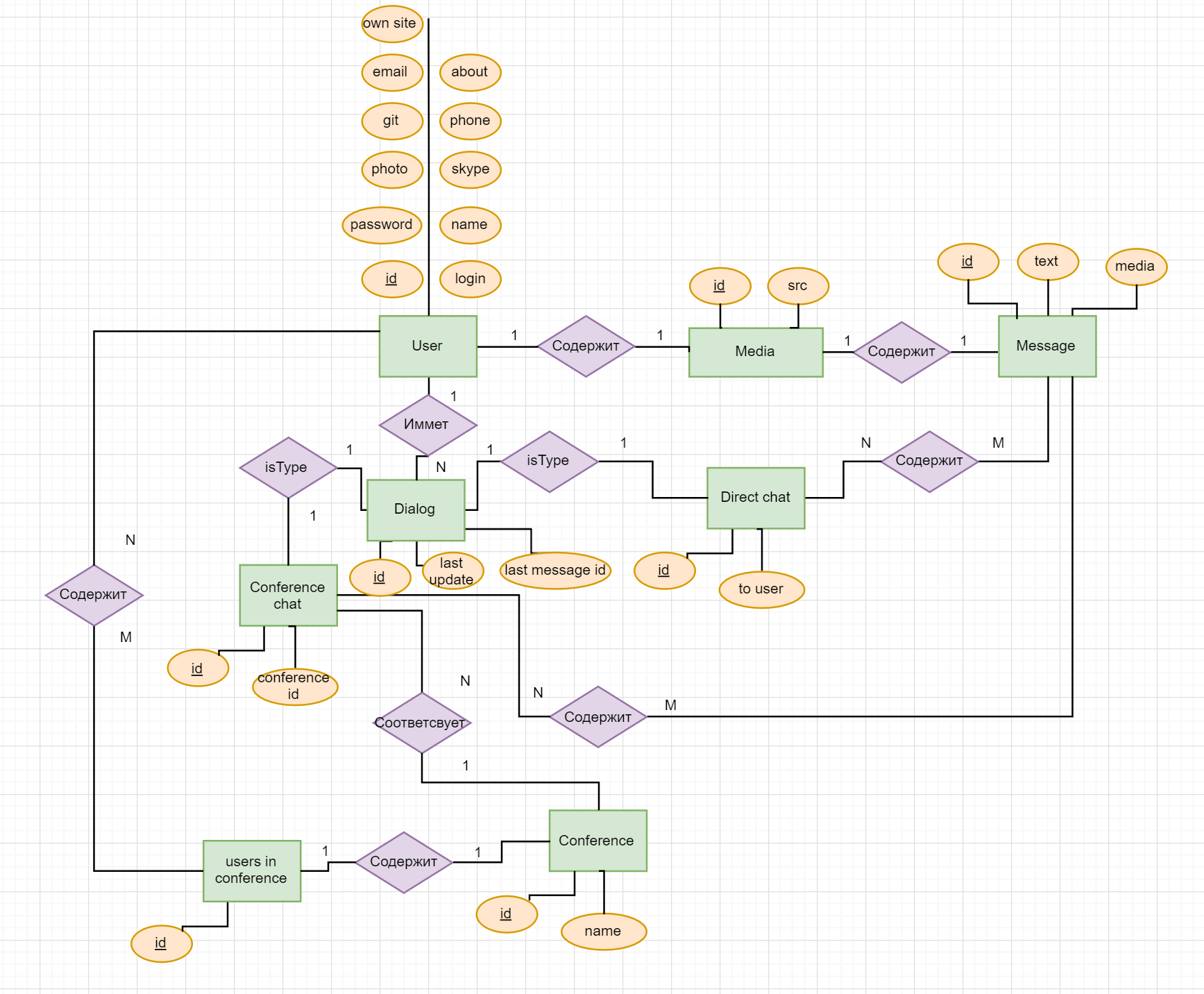
7) *Конференция*. Содержит в себе группу пользователей и имя конференции

8) *Группа пользователей*. Содержит в себе пользователей для данной конференции

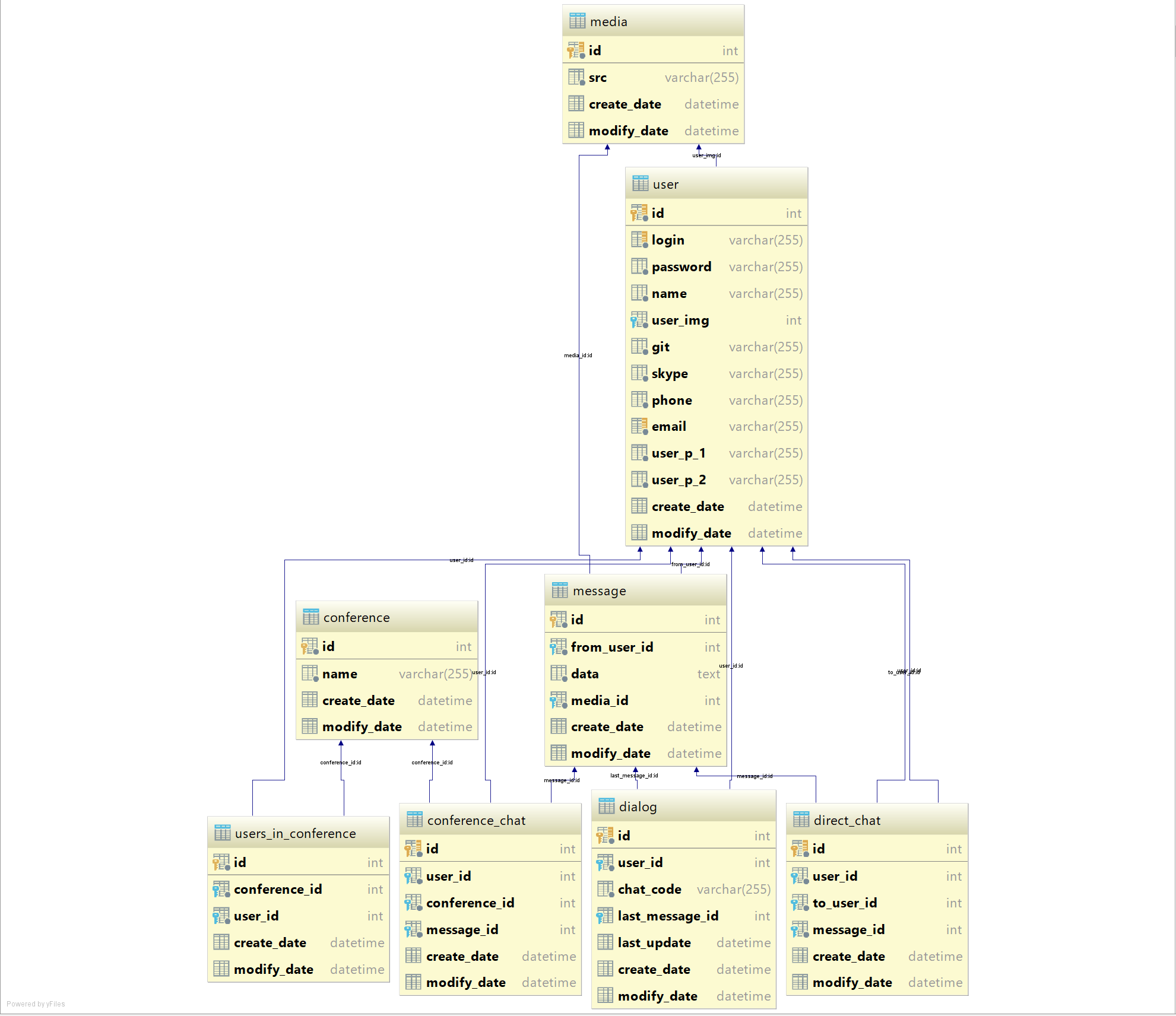
**Нотация Чена**







**Реляционная модель**



**API**

**-- Media**

-- *deleteMediaById(media\_id)* – удалить запись о медиа файлы по id из таблицы

-- *addMedia(src)* – добавить запись о медиа файле в таблицу. Возвращает id добавленного медиа файла в таблице

**-- User**

-- *addUser(login, password, name, email)* – добавить нового пользователя. Возвращает id пользователя

-- *deleteUserById(user\_id)* – удаляет пользователя из таблицы по id

-- *changeUserById(user\_id, login, password, name, user\_img,*

*git, skype, phone, email, user\_p\_1, user\_p\_2)* – изменяет пользовательские данные по id

**-- Dialog**

-- *getDialogsPage(user\_id, page\_number, dialogs\_per\_page)* – возвращает перечень диалогов для пользователя для данной страницы

-- *getDialogsCount(user\_id)* – возвращает количество диалогов пользователя

**-- Direct**

-- *getDirectChatPage(user\_id, to\_user\_id, page\_number, messages\_per\_page)* – возвращает страницу с сообщениями в диалоге между пользователями

-- *getCountDirectChatMessages(user\_id, to\_user\_id)* – возвращает количество сообщений в диалоге между пользователями

-- *deleteMessageFromDirectChat(user\_id, to\_user\_id, message\_id)* – удаляет выбранное сообщение из диалога между пользователями

-- *deleteAllMessagesFromDirectChat(user\_id, to\_user\_id)* – удаляет все сообщения из диалога между пользователями

**-- Conference**

-- *getConferenceChatPage(user\_id, conference\_id, page\_number, messages\_per\_page)* – возвращает страницу с сообщениями из конференции

-- *getCountConferenceMessages(user\_id, conference\_id)* – возвращает количество сообщений из конференции

-- *deleteMessageFromConferenceChat(user\_id, conference\_id, message\_id)* – удаляет выбранное сообщение из конференции

-- *deleteAllMessagesFromConferenceChat(user\_id, conference\_id)* – удаляет все сообщения из конференции

-- *joinToConference(user\_id, conference\_id)* – присоединяет пользователя к выбранной конференции

-- *leaveFromConference(user\_id, conference\_id)* – пользователь покидает выбранную конференцию

-- *createNewConference(user\_id, name)* – создать новую конференцию. Возвращается id конференции

-- *inviteUserToConference(user\_id, inviting\_user\_id, conference\_id)* – пригласить пользователя в конференцию

**-- Message**

-- sentMessageToUser(user\_id, to\_user\_id, data, media\_id) – отправить сообщение пользователю

-- sentMessageToConference(user\_id, conference\_id, data, media\_id) – отправить сообщение в конференцию

**-- View**

-- *getMessages* – возвращает все сообщения из базы данных

-- *getMedia* – возвращает все медиа из базы данных

-- *getUsers* – возвращает всех пользователей из базы данных

**Triggers**

1. Обновление modify\_date у всех таблиц
2. Контроль, что при удалении сообщения, если сообщений больше нет, то диалог удаляется и не показывается в списке диалогов

**Create Data Base with Tables**

IF EXISTS (SELECT name

FROM master.dbo.sysdatabases

WHERE name = 'messenger')

BEGIN

    DROP DATABASE messenger;

END

GO

CREATE DATABASE messenger;

GO

USE messenger

GO

CREATE TABLE messenger.dbo.[media] (

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [src] varchar(255) NOT NULL,

        [create\_date] datetime DEFAULT GETDATE(),

        [modify\_date] datetime DEFAULT GETDATE()

        )

GO

SET IDENTITY\_INSERT dbo.[media] ON

INSERT INTO [media] ([id], [src]) VALUES(0, '')

SET IDENTITY\_INSERT dbo.[media] OFF

CREATE TABLE messenger.dbo.[user] (

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [login] varchar(255) NOT NULL UNIQUE,

        [password] varchar(255) NOT NULL,

        [name] varchar(255) NOT NULL,

        [user\_img] int NOT NULL FOREIGN KEY REFERENCES [media](id) DEFAULT 0,

        [git] varchar(255) NOT NULL DEFAULT '',

        [skype] varchar(255) NOT NULL DEFAULT '',

        [phone] varchar(255) NOT NULL DEFAULT '',

        [email] varchar(255) NOT NULL UNIQUE,

        [user\_p\_1] varchar(255) NOT NULL DEFAULT '',

        [user\_p\_2] varchar(255) NOT NULL DEFAULT '',

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

SET IDENTITY\_INSERT dbo.[user] ON

INSERT INTO [user] ([id], [login], [password], [name], [email])

VALUES(0, 'notification', 'notification', 'notification', 'notification')

SET IDENTITY\_INSERT dbo.[user] OFF

CREATE TABLE messenger.dbo.[message](

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [from\_user\_id] int FOREIGN KEY REFERENCES [user](id) DEFAULT 0,

        [data] text NOT NULL,

        [media\_id] int NOT NULL FOREIGN KEY REFERENCES [media](id) DEFAULT 0,

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

CREATE TABLE messenger.dbo.[dialog](

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [user\_id] int NOT NULL FOREIGN KEY REFERENCES [user](id),

        [chat\_code] varchar(255) NOT NULL,

        [last\_message\_id] int FOREIGN KEY REFERENCES [message](id),

        [last\_update] datetime,

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

CREATE TABLE messenger.dbo.[conference](

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [name] varchar(255) NOT NULL,

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

CREATE TABLE messenger.dbo.[users\_in\_conference](

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [conference\_id] INT NOT NULL FOREIGN KEY REFERENCES [conference](id),

        [user\_id] INT NOT NULL FOREIGN KEY REFERENCES [user](id),

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

CREATE TABLE messenger.dbo.[direct\_chat](

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [user\_id] INT NOT NULL FOREIGN KEY REFERENCES [user](id),

        [to\_user\_id] INT NOT NULL FOREIGN KEY REFERENCES [user](id),

        [message\_id] INT NOT NULL FOREIGN KEY REFERENCES [message](id),

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

CREATE TABLE messenger.dbo.[conference\_chat](

        [id] INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [user\_id] INT NOT NULL FOREIGN KEY REFERENCES [user](id),

        [conference\_id] INT NOT NULL FOREIGN KEY REFERENCES [conference](id),

        [message\_id] INT NOT NULL FOREIGN KEY REFERENCES [message](id),

        [create\_date] datetime DEFAULT GETDATE(),

[modify\_date] datetime DEFAULT GETDATE()

        )

GO

SELECT \* FROM messenger.dbo.[user]

GO

SELECT \* FROM messenger.dbo.[media]

GO

SELECT \* FROM messenger.dbo.[dialog]

GO

SELECT \* FROM messenger.dbo.[conference]

GO

SELECT \* FROM messenger.dbo.[users\_in\_conference]

GO

SELECT \* FROM messenger.dbo.[message]

GO

SELECT \* FROM messenger.dbo.[direct\_chat]

GO

SELECT \* FROM messenger.dbo.[conference\_chat]

GO

**Procedure**

USE messenger

GO

-- Media

IF OBJECT\_ID('deleteMediaById', 'P' ) IS NOT NULL

DROP PROCEDURE deleteMediaById

GO

CREATE PROCEDURE dbo.deleteMediaById

    @id int

AS

BEGIN

    DELETE FROM [media] WHERE id = @id

END

GO

IF OBJECT\_ID('addMedia', 'P' ) IS NOT NULL

DROP PROCEDURE addMedia

GO

CREATE PROCEDURE dbo.addMedia

    @src varchar(255)

AS

BEGIN

    INSERT INTO [media]

    ([src])

    VALUES

    (@src)

    RETURN IDENT\_CURRENT('media')

END

GO

-- User

IF OBJECT\_ID('addUser', 'P' ) IS NOT NULL

DROP PROCEDURE addUser

GO

CREATE PROCEDURE dbo.addUser

    @login varchar(255),

@password varchar(255),

    @name varchar(255),

    @email varchar(255)

AS

BEGIN

    INSERT INTO [user]

    ([login], [password], [name], [email])

    VALUES

    (@login, @password, @name, @email)

    RETURN IDENT\_CURRENT('user')

END

GO

IF OBJECT\_ID('deleteUserById', 'P' ) IS NOT NULL

DROP PROCEDURE deleteUserById

GO

CREATE PROCEDURE dbo.deleteUserById

    @id int

AS

BEGIN

    DELETE FROM [user] WHERE id = @id

END

GO

IF OBJECT\_ID('changeUserById', 'P' ) IS NOT NULL

DROP PROCEDURE changeUserById

GO

CREATE PROCEDURE dbo.changeUserById

    @id int,

    @login varchar(255),

    @password varchar(255),

    @name varchar(255),

    @user\_img int,

    @git varchar(255),

    @skype varchar(255),

    @phone varchar(255),

    @email varchar(255),

    @user\_p\_1 varchar(255),

    @user\_p\_2 varchar(255)

AS

BEGIN

    UPDATE [user]

    SET [login] = @login,

        [password] = @password,

        [name] = @name,

        [user\_img] = @user\_img,

        [git] = @git,

        [skype] = @skype,

        [phone] = @phone,

        [email] = @email,

        [user\_p\_1] = @user\_p\_1,

        [user\_p\_2] = @user\_p\_2

    WHERE id = @id

END

GO

-- Direct

IF OBJECT\_ID('deleteMessageFromDirectChat', 'P' ) IS NOT NULL

DROP PROCEDURE deleteMessageFromDirectChat

GO

CREATE PROCEDURE dbo.deleteMessageFromDirectChat

    @user\_id int,

@to\_user\_id int,

    @message\_id int

AS

BEGIN

    DELETE FROM dbo.[direct\_chat]

    WHERE [user\_id] = @user\_id AND [to\_user\_id] = @to\_user\_id AND [message\_id] = @message\_id

END

GO

IF OBJECT\_ID('deleteAllMessagesFromDirectChat', 'P' ) IS NOT NULL

DROP PROCEDURE deleteAllMessagesFromDirectChat

GO

CREATE PROCEDURE dbo.deleteAllMessagesFromDirectChat

    @user\_id int,

@to\_user\_id int

AS

BEGIN

    DELETE FROM dbo.[direct\_chat]

    WHERE [user\_id] = @user\_id AND [to\_user\_id] = @to\_user\_id

END

GO

-- Conference

IF OBJECT\_ID('deleteMessageFromConferenceChat', 'P' ) IS NOT NULL

DROP PROCEDURE deleteMessageFromConferenceChat

GO

CREATE PROCEDURE dbo.deleteMessageFromConferenceChat

    @user\_id int,

@conference\_id int,

    @message\_id int

AS

BEGIN

    DELETE FROM dbo.[conference\_chat]

    WHERE [user\_id] = @user\_id AND [conference\_id] = @conference\_id AND [message\_id] = @message\_id

END

GO

IF OBJECT\_ID('deleteAllMessagesFromConferenceChat', 'P' ) IS NOT NULL

DROP PROCEDURE deleteAllMessagesFromConferenceChat

GO

CREATE PROCEDURE dbo.deleteAllMessagesFromConferenceChat

    @user\_id int,

@conference\_id int

AS

BEGIN

    DELETE FROM dbo.[conference\_chat]

    WHERE [user\_id] = @user\_id AND [conference\_id] = @conference\_id

END

GO

IF OBJECT\_ID('joinToConference', 'P' ) IS NOT NULL

DROP PROCEDURE joinToConference

GO

CREATE PROCEDURE dbo.joinToConference

    @user\_id int,

@conference\_id int

AS

BEGIN

    DECLARE @msg varchar(512)

    DECLARE @name varchar(255)

    SET @name = (SELECT [name] FROM dbo.[user] WHERE [id] = @user\_id)

    SET @msg = CONCAT(@name, ' JOIN TO CONFERENCE')

    INSERT INTO dbo.[users\_in\_conference]

    ([conference\_id], [user\_id])

    VALUES(@conference\_id, @user\_id)

    EXEC dbo.sentMessageToConference 0, @conference\_id, @msg

END

GO

IF OBJECT\_ID('leaveFromConference', 'P') IS NOT NULL

DROP PROCEDURE leaveFromConference

GO

CREATE PROCEDURE dbo.leaveFromConference

    @user\_id int,

@conference\_id int

AS

BEGIN

    DECLARE @msg varchar(512)

    DECLARE @name varchar(255)

    SET @name = (SELECT [name] FROM dbo.[user] WHERE [id] = @user\_id)

    SET @msg = CONCAT(@name, ' LEFT CONFERENCE')

    EXEC dbo.sentMessageToConference 0, @conference\_id, @msg

    DELETE FROM dbo.[users\_in\_conference]

    WHERE [conference\_id] = @conference\_id AND [user\_id] = @user\_id

END

GO

IF OBJECT\_ID('createNewConference', 'P' ) IS NOT NULL

DROP PROCEDURE createNewConference

GO

CREATE PROCEDURE dbo.createNewConference

    @user\_id int,

@conference\_name varchar(255)

AS

BEGIN

    DECLARE @msg varchar(512)

    DECLARE @name varchar(255)

    SET @name = (SELECT [name] FROM dbo.[user] WHERE [id] = @user\_id)

    SET @msg = CONCAT(@name, ' CREATED CONFERENCE')

    DECLARE @conference\_id int

    INSERT INTO dbo.[conference]

    ([name])

    VALUES

    (@conference\_name)

    SET @conference\_id = IDENT\_CURRENT('conference')

    INSERT INTO dbo.[users\_in\_conference]

    ([conference\_id], [user\_id])

    VALUES(@conference\_id, @user\_id)

    EXEC dbo.sentMessageToConference 0, @conference\_id, @msg

    RETURN @conference\_id

END

GO

IF OBJECT\_ID('inviteUserToConference', 'P' ) IS NOT NULL

DROP PROCEDURE inviteUserToConference

GO

CREATE PROCEDURE dbo.inviteUserToConference

    @user\_id int,

    @inviting\_user\_id int,

@conference\_id int

AS

BEGIN

    DECLARE @msg varchar(512)

    DECLARE @name varchar(255)

    DECLARE @invited\_name varchar(255)

    SET @name = (SELECT [name] FROM dbo.[user] WHERE [id] = @user\_id)

    SET @invited\_name = (SELECT [name] FROM dbo.[user] WHERE [id] = @inviting\_user\_id)

    SET @msg = CONCAT(@name, ' INVITED TO CONFERENCE ', @invited\_name)

    INSERT INTO dbo.[users\_in\_conference]

    ([conference\_id], [user\_id])

    VALUES(@conference\_id, @inviting\_user\_id)

    EXEC dbo.sentMessageToConference 0, @conference\_id, @msg

END

GO

-- Sent Message

IF OBJECT\_ID('sentMessageToUser', 'P' ) IS NOT NULL

DROP PROCEDURE sentMessageToUser

GO

CREATE PROCEDURE dbo.sentMessageToUser

    @user\_id int,

@to\_user\_id int,

    @data text,

    @media\_id int = 0

AS

BEGIN

    DECLARE @message\_id int;

    INSERT INTO [message]

    ([from\_user\_id], [data], [media\_id])

    VALUES

    (@user\_id, @data, @media\_id)

    SET @message\_id = IDENT\_CURRENT('message')

    INSERT INTO [direct\_chat]

    ([user\_id], [to\_user\_id], [message\_id])

    VALUES

    (@user\_id, @to\_user\_id, @message\_id)

    INSERT INTO [direct\_chat]

    ([user\_id], [to\_user\_id], [message\_id])

    VALUES

    (@to\_user\_id, @user\_id, @message\_id)

END

GO

IF OBJECT\_ID('sentMessageToConference', 'P' ) IS NOT NULL

DROP PROCEDURE sentMessageToConference

GO

CREATE PROCEDURE dbo.sentMessageToConference

    @user\_id int,

@conference\_id int,

    @data text,

    @media\_id int = 0

AS

BEGIN

    DECLARE @message\_id int;

    INSERT INTO [message]

    ([from\_user\_id], [data], [media\_id])

    VALUES

    (@user\_id, @data, @media\_id)

    SET @message\_id = IDENT\_CURRENT('message')

    DECLARE @user\_in\_conf\_id int

    DECLARE users\_in\_conf CURSOR

    LOCAL STATIC READ\_ONLY FORWARD\_ONLY

    FOR

    SELECT [user\_id] FROM dbo.[users\_in\_conference]

    WHERE [conference\_id] = @conference\_id

    OPEN users\_in\_conf

    FETCH NEXT FROM users\_in\_conf INTO @user\_in\_conf\_id

    WHILE @@FETCH\_STATUS = 0

    BEGIN

        INSERT INTO [conference\_chat]

        ([user\_id], [conference\_id], [message\_id])

        VALUES

        (@user\_in\_conf\_id, @conference\_id, @message\_id)

        -- next user

        FETCH NEXT FROM users\_in\_conf INTO @user\_in\_conf\_id

    END

    CLOSE users\_in\_conf

    DEALLOCATE users\_in\_conf

END

GO

**Functions**

USE messenger

GO

-- Dialog

IF OBJECT\_ID('getDialogsPage', N'IF' ) IS NOT NULL

DROP FUNCTION getDialogsPage

GO

CREATE FUNCTION dbo.getDialogsPage(@user\_id int, @page\_number int, @dialogs\_per\_page int)

RETURNS TABLE

AS

RETURN (

SELECT \* FROM [dialog] WHERE [user\_id] = @user\_id

ORDER BY [last\_update] DESC

OFFSET (@page\_number - 1) \* @dialogs\_per\_page ROWS

FETCH NEXT @dialogs\_per\_page ROWS ONLY

);

GO

IF OBJECT\_ID('getDialogsCount', N'FN' ) IS NOT NULL

DROP FUNCTION getDialogsCount

GO

CREATE FUNCTION dbo.getDialogsCount(@user\_id int)

RETURNS INT

AS

BEGIN

DECLARE @dialogsCount INT;

SET @dialogsCount = (SELECT COUNT(\*)

FROM dbo.[dialog]

WHERE [user\_id] = @user\_id)

RETURN @dialogsCount;

END

GO

-- Direct

IF OBJECT\_ID('getDirectChatPage', N'IF' ) IS NOT NULL

DROP FUNCTION getDirectChatPage

GO

CREATE FUNCTION dbo.getDirectChatPage(@user\_id int, @to\_user\_id int, @page\_number int, @dialogs\_per\_page int)

RETURNS TABLE

AS

RETURN (

SELECT dbo.[message].[id], dbo.[message].[from\_user\_id], dbo.[message].[data], dbo.[message].[media\_id],

dbo.[message].[create\_date], dbo.[message].[modify\_date]

FROM (dbo.[direct\_chat] JOIN dbo.[message] ON dbo.[direct\_chat].[message\_id] = dbo.[message].[id])

WHERE dbo.[direct\_chat].[user\_id] = @user\_id and dbo.[direct\_chat].[to\_user\_id] = @to\_user\_id

ORDER BY dbo.[message].[modify\_date] DESC

OFFSET (@page\_number - 1) \* @dialogs\_per\_page ROWS

FETCH NEXT @dialogs\_per\_page ROWS ONLY

);

GO

IF OBJECT\_ID('getCountDirectChatMessages', N'FN' ) IS NOT NULL

DROP FUNCTION getCountDirectChatMessages

GO

CREATE FUNCTION dbo.getCountDirectChatMessages(@user\_id int, @to\_user\_id int)

RETURNS INT

AS

BEGIN

DECLARE @messagesCount INT;

SET @messagesCount = (SELECT COUNT(\*)

FROM dbo.[direct\_chat]

WHERE [user\_id] = @user\_id and [to\_user\_id] = @to\_user\_id)

RETURN @messagesCount;

END

GO

-- Conference

IF OBJECT\_ID('getConferenceChatPage', N'IF' ) IS NOT NULL

DROP FUNCTION getConferenceChatPage

GO

CREATE FUNCTION dbo.getConferenceChatPage(@user\_id int, @conference\_id int, @page\_number int, @dialogs\_per\_page int)

RETURNS TABLE

AS

RETURN (

SELECT dbo.[message].[id], dbo.[message].[from\_user\_id], dbo.[message].[data], dbo.[message].[media\_id],

dbo.[message].[create\_date], dbo.[message].[modify\_date]

FROM (dbo.[conference\_chat] JOIN dbo.[message] ON dbo.[conference\_chat].[message\_id] = dbo.[message].[id])

WHERE dbo.[conference\_chat].[user\_id] = @user\_id and dbo.[conference\_chat].[conference\_id] = @conference\_id

ORDER BY dbo.[message].[modify\_date] DESC

OFFSET (@page\_number - 1) \* @dialogs\_per\_page ROWS

FETCH NEXT @dialogs\_per\_page ROWS ONLY

);

GO

IF OBJECT\_ID('getCountConferenceMessages', N'FN' ) IS NOT NULL

DROP FUNCTION getCountConferenceMessages

GO

CREATE FUNCTION dbo.getCountConferenceMessages(@user\_id int, @conference\_id int)

RETURNS INT

AS

BEGIN

DECLARE @messagesCount INT;

SET @messagesCount = (SELECT COUNT(\*)

FROM dbo.[conference\_chat]

WHERE [user\_id] = @user\_id and [conference\_id] = @conference\_id)

RETURN @messagesCount;

END

GO

**Views**

USE messenger

GO

CREATE OR ALTER VIEW getMessages

AS

SELECT \* FROM dbo.[message]

GO

CREATE OR ALTER VIEW getMedia

AS

SELECT \* FROM dbo.[media]

GO

CREATE OR ALTER VIEW getUsers

AS

SELECT \* FROM dbo.[user]

GO

**Triggers**

USE messenger

GO

-- Update Modify Date

CREATE TRIGGER UpdateModifyDateOnMedia

ON dbo.[media]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[media]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnUser

ON dbo.[user]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[user]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnConference

ON dbo.[conference]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[conference]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnDialog

ON dbo.[dialog]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[dialog]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnMessage

ON dbo.[message]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[message]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnUsersInConference

ON dbo.[users\_in\_conference]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[users\_in\_conference]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnConferenceChat

ON dbo.[conference\_chat]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[conference\_chat]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

CREATE TRIGGER UpdateModifyDateOnDirectChat

ON dbo.[direct\_chat]

AFTER UPDATE

AS

BEGIN

UPDATE dbo.[direct\_chat]

SET [modify\_date] = GETDATE()

WHERE [id] IN (SELECT DISTINCT id FROM Inserted)

END

GO

-- Direct Chat triggers

CREATE TRIGGER TriggerForDeleteMessageFromDirectChat

ON dbo.[direct\_chat]

AFTER DELETE

AS

BEGIN

DECLARE @user\_id int

DECLARE @to\_user\_id int

DECLARE @chat\_code varchar(255)

SET @user\_id = (SELECT DISTINCT [user\_id] FROM deleted)

SET @to\_user\_id = (SELECT DISTINCT [to\_user\_id] FROM deleted)

SET @chat\_code = CONCAT('u', @to\_user\_id)

IF NOT EXISTS (

SELECT \* FROM [direct\_chat]

WHERE [user\_id] = @user\_id AND [to\_user\_id] = @to\_user\_id)

BEGIN

DELETE FROM [dialog]

WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code

END ELSE

BEGIN

DECLARE @last\_update datetime

DECLARE @message\_id int

SELECT @last\_update = [modify\_date], @message\_id = [id]

FROM dbo.getDirectChatPage(@user\_id, @to\_user\_id, 1, 1)

UPDATE [dialog]

SET

[last\_update] = @last\_update,

[last\_message\_id] = @message\_id

     WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code

END

END

GO

CREATE TRIGGER TriggerForAddMessageInDirectChat

ON dbo.[direct\_chat]

AFTER INSERT

AS

BEGIN

DECLARE @user\_id int

DECLARE @to\_user\_id int

DECLARE @chat\_code varchar(255)

DECLARE @message\_id int

DECLARE @last\_update datetime

SET @user\_id = (SELECT DISTINCT [user\_id] FROM Inserted)

SET @to\_user\_id = (SELECT DISTINCT [to\_user\_id] FROM Inserted)

SET @chat\_code = CONCAT('u', @to\_user\_id)

SET @message\_id = (SELECT DISTINCT [message\_id] FROM Inserted)

SET @last\_update = (SELECT [modify\_date] FROM dbo.[message] WHERE [id] = @message\_id)

IF NOT EXISTS (

SELECT \* FROM [dialog]

WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code)

BEGIN

INSERT INTO [dialog]

([user\_id], [chat\_code], [last\_update], [last\_message\_id])

VALUES

(@user\_id, @chat\_code, @last\_update, @message\_id)

END

UPDATE [dialog]

SET

[last\_update] = @last\_update,

[last\_message\_id] = @message\_id

    WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code

END

GO

-- Conference Chat triggers

CREATE TRIGGER TriggerForDeleteMessageFromConferenceChat

ON dbo.[conference\_chat]

AFTER DELETE

AS

BEGIN

DECLARE @user\_id int

DECLARE @conference\_id int

DECLARE @chat\_code varchar(255)

SET @user\_id = (SELECT DISTINCT [user\_id] FROM deleted)

SET @conference\_id = (SELECT DISTINCT [conference\_id] FROM deleted)

SET @chat\_code = CONCAT('c', @conference\_id)

IF NOT EXISTS (

SELECT \* FROM [conference\_chat]

WHERE [user\_id] = @user\_id AND [conference\_id] = @conference\_id)

BEGIN

DELETE FROM [dialog]

WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code

END ELSE

BEGIN

DECLARE @last\_update datetime

DECLARE @message\_id int

SELECT @last\_update = [modify\_date], @message\_id = [id]

FROM dbo.getConferenceChatPage(@user\_id, @conference\_id, 1, 1)

UPDATE [dialog]

SET

[last\_update] = @last\_update,

[last\_message\_id] = @message\_id

     WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code

END

END

GO

CREATE TRIGGER TriggerForAddMessageInConferenceChat

ON dbo.[conference\_chat]

AFTER INSERT

AS

BEGIN

DECLARE @user\_id int

DECLARE @conference\_id int

DECLARE @chat\_code varchar(255)

DECLARE @message\_id int

DECLARE @last\_update datetime

SET @user\_id = (SELECT DISTINCT [user\_id] FROM Inserted)

SET @conference\_id = (SELECT DISTINCT [conference\_id] FROM Inserted)

SET @chat\_code = CONCAT('c', @conference\_id)

SET @message\_id = (SELECT DISTINCT [message\_id] FROM Inserted)

SET @last\_update = (SELECT [modify\_date] FROM dbo.[message] WHERE [id] = @message\_id)

IF NOT EXISTS (

SELECT \* FROM [dialog]

WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code)

BEGIN

INSERT INTO [dialog]

([user\_id], [chat\_code], [last\_update], [last\_message\_id])

VALUES

(@user\_id, @chat\_code, @last\_update, @message\_id)

END

UPDATE [dialog]

SET

[last\_update] = @last\_update,

[last\_message\_id] = @message\_id

    WHERE [user\_id] = @user\_id AND [chat\_code] = @chat\_code

END

GO