[1. Brief Explanation of Facebook and Linked In 2](#_Toc407583118)

[1.1. Linked In 2](#_Toc407583119)

[1.2. Facebook 3](#_Toc407583120)

[2. Analysis report for each of the database model/application 4](#_Toc407583121)

[2.1. Aim of applications 4](#_Toc407583122)

[2.2. Main entities 4](#_Toc407583123)

[2.3. Characteristics of each entity model 4](#_Toc407583124)

[2.3.1. Characteristics of entities for Facebook database model 4](#_Toc407583125)

[2.3.2. Characteristics of entities for LinkedIn database model 6](#_Toc407583126)

[2.4. Relationships exists among the entities 7](#_Toc407583127)

[2.4.1. Relationships among the entities for Facebook database model 7](#_Toc407583128)

[2.4.2. Relationships among the entities for Linked IN database model 9](#_Toc407583129)

[2.5. Constraints related to entities 11](#_Toc407583130)

[2.5.1. Facebook Constraints 11](#_Toc407583131)

[2.5.1.1. Referential Integrity Constraints 11](#_Toc407583132)

[2.5.1.2. Key Constraints 12](#_Toc407583133)

[2.5.2. Linked IN Constraints 13](#_Toc407583134)

[2.5.2.1. Referential Integrity Constraints 13](#_Toc407583135)

[2.5.2.2. Key Constraints 14](#_Toc407583136)

[3. DDL Statements 15](#_Toc407583137)

[4. Populate the database 22](#_Toc407583138)

[5. TRIGGERS 26](#_Toc407583139)

[6. Check Constraints and Assertions 33](#_Toc407583140)

[6.1. Check Constraints 33](#_Toc407583141)

[6.2. Assertions 34](#_Toc407583142)

[7. SQL Statements 36](#_Toc407583143)

[7.1. Sample INSERT, DELETE and UPDATE Statements 36](#_Toc407583144)

[7.1.1. INSERTs 36](#_Toc407583145)

[7.1.2. UPDATEs 36](#_Toc407583146)

[7.1.3. DELETEs 37](#_Toc407583147)

[7.2. SELECT Statements 37](#_Toc407583148)

[7.2.1. Using one table 37](#_Toc407583149)

[7.2.2. Using minimum 2 tables 37](#_Toc407583150)

[7.2.3. Using minimum 3 tables 38](#_Toc407583151)

# Brief Explanation of Facebook and Linked In

## Linked In

Linked In is a social network based on the business sector. It provides variety informations that are entered by users like personal informations, abilities and job experiences. It also provides to contact with users in same tribes. It contains want ads and users can find job. On the other hand, Linked In contains many different numbers of news sources according to users’ interests.

Linked In is a user-oriented application. Linked In can not exist without the user. Users enter own informations. These informations are works that company, address, schools that study or graduate, job experiences, abilities and marital status.

Users contact with each other. This connection is mutual and now this people can communicate and follow with each other via Linked In. Also, you can follow someone without connection. Users can recommend someone to other users.

There are groups and thanks to groups, you can come together with people that are in same tribe. There are sharings like events, seminars, job adverts in groups. You can make this sharings and also you can comment on these shares.

There are pages that you can follow them and these pages may belong to the companies. Various news and informations posted on these pages.

## Facebook

Facebook is an application to find user’s friend and to show user’s social life to user’s friend. We can call that social application.

People need an e-mail to create their profile in Facebook and every user login application via e-mail and the password that is created by user. Users can write their personal information like relationship status , phone , interests , education , family relations and every kind of hobbies that users have . Also , people make friendlist too. They can have chance to share their information with their friends via facebook. Users can upload their photos to show their life in facebook and it helps people to follow their lifes.

Also, users may write their opinion and share these in their profile . Creating blogs is a way of posting ideas , lyrics etc. to user’s profile too. With Facebook application , users can join different kind of organization like pages , groups and networks ( i.e family , school , city , country) .

Events that are created in facebook helps people to follow real life activities and it’s kind of remanding system for user’s. Because of the social requirements and to show people’s real life reflect in application , Facebook application let user like , make and share the friend’s posts.

Nofication Service is the other service which help user’s to get knowledge about their relation between posts and friends. Facebook consider user’s security too and user can change their privacy in order to hide their information ( i.e photos , status ) from the other user that is not being wanted to see posts.

Chating and messaging help people to communicate in facebook and user’s texts in messaging and chating are not visible for the other users that are not in the conversation and facebook provides video chatting too.

Facebook provides different kind of application like games too.It makes facebook more enjoyable and attractive. Facebook supports different languages and because of that people use facebook all around the world.

# Analysis report for each of the database model/application

* 1. Aim of applications

- *Aim of Facebook* application is to store the datum of people’s social life  ( hobbies , profiles , life events , people’s character ) on database and according to user’s privacy which is choosen by users, facebook provides to access this informations via online.  
  
- *Aim of Linkedin* application is to store the datum of people’s business life like CV , experiences , knowledge and organization’s informations like address , job offers etc. Linkedin gives people to change to show their [professional](http://www2.zargan.com/tr/page/search?Text=professional) abilities in social platform.

## Main entities

* Main entities are User, Country, Bookmark\_category, Feed\_category and Privacy\_type\* for *Facebook Database model.*
* Main entities are Members, Organizations, Ref\_marital\_status, Addresses, Ref\_CV\_Sections and Ref\_Profile\_Sections for *Linked IN Database model.*

## Characteristics of each entity model

### Characteristics of entities for Facebook database model

* User : It stores the main information of facebook user. Each user has own user\_id and it is primary key of User entity. Other attributes of User entity are token, username, password, Name\_first, Name\_middle, Name\_last, email\_id, Picture, active, online, created\_at.
* Profile: It stores the personal characteristics of each user. Each user profile has own profile\_id and it is primary key of Profile entity. It uses User\_id as a foreign key. Other attributes of Profile entity are privacy, rating, dob, about\_me, relationship, looking\_for, phone, interests, education, hobbies, fav\_movies, fav\_artists, fav\_books, fav\_animals, religion, everything\_else and created\_at.
* Status: It stores the post that is created by user. Each status has its own status\_id as Primary Key. User\_id is a foreign key in this entity. Other attributes of Status entity are message, created\_at, thumbs\_up, thumbs\_down, privacy, is\_reply, to\_fb, to\_twitter.
* Privacy: It stores the privacy settings of each user. Privacy\_id is used for Primary Key in this entity.Also , user\_id is a foreign key . Other attributes of Privacy entity are profile , address , status, bookmark , feed , activity , friend , friend\_list ,nickname.
* Country: It stores every county name and their id.Country\_id is Primary Key of this entity.
* City : It stores the name of every city in each country. It uses city\_id as Primary Key and country\_id as a foreign key.
* Address: It stores address information. Address\_id is primary key of this entity.Foreign keys of this entity are profile\_id and city\_id. Other attributes of this entity are privacy and address.
* Lang : It stores language of the application that every user uses.Language\_id is primary key and user\_id is a foreing key in this entity.Other attribute is lang.
* Nickname: It stores nickname for each user. Nickname\_id is primary key and User\_id is a foreign key in this entity. Other attributes of Nickname entity are Nickname and Privacy.
* Friend\_list : It stores the list of friendship for users. Each friendship has Friend\_List\_Id as primary key. Friend\_id and User\_id are foreign key of this entity. Other attributes of Friend\_List entity are Name and Privacy.
* Friend : It stores the informations of each friendship. Friend\_id is primary key. User\_id and Friend\_List\_Id are foreign key in this entity. Other attributes of Friend entity are Friend\_User\_Id, Is\_Subscriber, Privacy and Created\_at.
* Thumb\_Up\_Down : It stores the thumps of status. It has Thumb\_up\_down\_id as primary key. Foreign keys are Status\_Id and Friend\_Id of this entity. Other attributes are Flag and Created\_at
* Notification : It stores the notifications for serving to user. Notification\_id is primary key and User\_id is a foreign key in this entity. Other attributes are Msg, Type, Privacy and Created\_at.
* Comment : It stores comments that are make by users to other users’ situations. It has Comment\_id as primary key. Status\_id and Friend\_id are foreign key in this entity. Other attributes are Message and Created\_At.
* Chat : It stores chatting informations between users. Chat\_id is primary key and User\_id is a foreign key in this entity. Other attributes are To\*\*\*, Msg and Created\_at.
* Bookmark: It stores the bookmarks. It has Bookmark\_id as primary key. Foreign key attributes of this entity are Bookmark\_Category\_id and Bookmark\_Sub\_Category\_id. Other attributes are Url, Rating, Privacy and Created\_at.
* Bookmark\_Category : It stores Bookmark\_Category\_id as primary key. It also has unique attribute that is Name.
* Bookmark\_Sub\_Category : It stores Bookmark\_Sıub\_Category as primary key. It also has unique attribute that is Name and foreign key attribute that is Bookmark\_Category\_id.
* Bookmark\_Info : It stores informations of each bookmark. Boomark\_Info\_id is primary key. Bookmark\_id and User\_id are foreign key attributes in this entity. Other attributes are Favorite, Clicks and Privacy.
* Feed : It stores the news feed. Feed\_id is primary key. Feed\_Category\_id and Feed\_Sub\_Category\_id are foreign keys in this entity. Other attributes are Feed\_Url, Rating, Privacy and Created\_at.
* Feed\_Category : It stores the Feed\_Category\_id as primary key. Name is unique attribute in this entity.
* Feed\_Sub\_Category : It stores the Feed\_Sub\_Category\_id as primary key. Feed\_Category\_id is foreign key in this entity. It has unique attributes that is Name.
* Feed\_Info : It stores information of each feed. Feed\_Info\_id is primary key. Feed\_id and User\_id are foreign keys in this entity. Other attributes are Favorite, Clicks and Privacy.
* Message : It stores messages between users. Message\_id is primary key and User\_id is foreign key in this entity. Other attributes are Message, Created\_at, Is\_read, Is\_spam, To and Is\_reply.
* Blog : It stores the blog that is created by users. Blog\_id is primary key and User\_id is foreign key in this entity. Other attributes are Message, Author and Created\_at.
* Privacy\_type : It stores Privacy\_type\_id as primary key and Name attribute as unique.

### Characteristics of entities for LinkedIn database model

* Orginazations: It stores the name of companys , their description and details.Organization\_ID is used as Primary key in Organizations Entity.Other attributes of this entity are Organization\_Name and Organization\_Description.
* Connections : It stores the friends of members and Connection\_ID is Primary key for this entity. Connection\_Member\_ID and Member\_ID are foreign keys in this entity.Data \_connection\_made (?)
* People\_Being\_Followed:It stores the information of who follows who and when s/he start to follow other people. Date\_Started\_Following is a primary key in People\_Being\_Followed entity. Member\_ID and Member\_Being\_Followed\_ID are Composite Key in People\_Being\_Followed entity.Other attribute is Date\_Stopped\_Following.
* Recommendations:It stores and who is suggested by who. Member\_Recommending\_ID and Member\_Being\_Recommended\_ID are composite key in Recommendations.It means that they can use as Primary Key or Foreign Key.The other attributes of this entity is Date\_of\_Recommendation and Other\_Details.
* Members : It stores the member’s information.Member\_ID is Primary key for this entity and Adress\_ID , Current\_Organization\_ID and Marital\_Status\_Code are used as foreing keys.Other attributes of this entity are Date\_Joined , Date\_of\_Birth , Email\_Address , Email\_Password , First\_Name,Middle\_Name,Last\_Name and Gender.
* CVs : It stores the uploading information of CV.CV\_ID is used as Primary key and Member\_ID is used as Foreing Key in CVs entity.Other attributes are Date\_Created and Date\_Updated.
* CV\_Sections : It stores the sections of CV.CV\_Section\_ID is primary key for this entity. CV\_ID and CV\_Section\_Code are used as Foreign Keys.Other attributes are Date\_Created , Date\_Updated and CV\_Section\_Text.
* Ref\_CV\_Sections: It stores CV’S personal details. CV\_Section\_Code is Primary key for this entity and other attributes are CV\_Section\_Description , Jobs and Qualifications.
* Members\_Profiles : It stores creating information of Members.Profile\_ID is Primary key and Member\_ID is foreign key in this entity.Also , other attributes are Date\_Created and Date\_Last\_Updated.
* Ref\_Marital\_Status : It stores the relationship status of members.It uses Marital\_Status\_code as Primary key.Other attribute is Marital\_Status\_Description.
* Addresses: It stores the address informations of members. Address\_ID is primary key of this entity.Line\_1,Line\_2,Line\_3,City,State\_County\_Province , Zip\_or\_Postcode,Country , Other\_Details are the other attributes of this entity.
* Groups: It stores the group informations. Group\_ID is primary key for this entity and Created\_by\_Member\_ID is a foreign key for this entity.Group\_Name , Group\_Description, Group\_Date\_Started , Group\_Date\_Ended , Group\_Date\_Last\_Activity are the other attributes.
* Members\_Groups : It stores the relations between members and groups.Member\_ID and Group\_ID are Composit Key in this entity . Date\_Joined and Date\_Left are the other attributes.
* Profile\_Sections : It stores creating information of profiles. Profile\_Section\_ID is primary key and Profile\_Section\_Code is a foreing key for this entity.Other attributes are Date\_Created , Date\_Updated and Profile\_Section\_Text.
* Ref\_Profile\_Sections : It stores the details of profile information. Profile\_Section\_Code is Primary key of this entity and Profile\_Section\_Description is an attribute too.

## Relationships exists among the entities

### Relationships among the entities for Facebook database model

* COUNTRY and CITY entities have a relationship. Every city has only one country. Country\_id is a foreign key in City entity.
* Country – City : 1 to Many
* CITY and ADDRESS entities have a relationship. Every address belongs only one city. City\_id is a foreign key in Address entity.
* City – Address : 1 to Many
* PROFILE and ADDRESS entities have a relationship. Every address belongs at least one profile. Profile\_id is a foreign key in ADDRESS entity.
* Profile – Address : 1 to 1
* LANG and USER entities have a relationship. Every user control their profile in a language that they choose. User\_id is a foreign key in Lang entity.
* Lang – User : 1 to 1
* NICKNAME and USER have a relationship. Every user has a nickname. User\_id is foreign key in NICKNAME entity.
* Nickname – User : 1 to 1
* FRIEND\_LIST has relationship with FRIEND entity. Every friend belongs to one friend list. Friend\_id is foreign key in FRIEND\_LIST entity.
* Friend\_List – Friend : 1 to Many
* FRIEND has relationship with USER entity. User\_id and Friend\_user\_id are foreing keys in FRIEND entity. This relationship is many to many. Because two records in User can reference same record of Friend.
* Friend – User : Many to Many
* THUMB\_UP\_DOWN has relationship with STATUS. Every thumb belongs to one status. Status\_id is foreign key in THUMB\_UP\_DOWN entity.
* Status – Thumb\_Up\_Down : 1 to Many

* THUMB\_UP\_DOWN has relationship with FRIEND. Every thumb are make by user. Friend\_id is foreign key in THUMB\_UP\_DOWN entity.
* Friend – Thumb\_Up\_Down : 1 to Many
* PROFILE has relationship with USER. Each profile must belong to only one user. User\_id is foreign key in PROFILE entity.
* Profile – User : 1 to 1
* COMMENT has relationship with STATUS. Every comment belongs to one status. Status\_id is foreign key in COMMENT entity.
* Status – Comment : 1 to Many
* COMMENT has relationship with FRIEND. Every comment are make by only one user. Friend\_id is foreign key in COMMENT entity.
* Friend – Comment : 1 to Many
* STATUS has relationship with USER. Each status is make by user. User\_id is a foreign key in STATUS entity.
* User – Status : 1 to Many
* NOTIFICATION has relationship with USER. Each notification is serve to user by automatically. User\_id is foreign key in NOTIFICATION entity.
* User – Notification : 1 - Many

* CHAT has relationship with USER. Chats are between users. User\_id and To\_User\_id are foreign keys in CHAT entity.
* Chat – User : Many - Many
* PRIVACY has relationship with USER. Each user has own primary setting. User\_id is a foreign key in PRIVACY entity.
* Privacy – User : 1 - 1
* MESSAGE has relationship with USER. Messages are between users. User\_id and To\_User\_id are foreign keys in MESSAGE entity.
* Message – User : Many - Many

* BOOKMARK has relationship with BOOKMARK\_CATEGORY. Each bookmark has own category. Bookmark\_Category\_id is foreign key in BOOKMARK entity.
* Bookmark\_Category – Bookmark : 1 - Many
* BOOKMARK has relationship with BOOKMARK\_SUB\_CATEGORY. Each bookmark has own sub category. Bookmark\_Sub\_Category\_id are foreign keys in BOOKMARK entity.
* Bookmark\_Sub\_Category – Bookmark : 1 - Many
* BOOKMARK\_SUB\_CATEGORY has relationship with BOOKMARK\_CATEGORY. Each bookmark sub category belongs to bookmark category. Bookmark\_Category\_id is foreign key in BOOKMARK\_SUB\_CATEGORY.
* Bookmark\_Category – Bookmark\_Sub\_Category : 1 - Many
* BOOKMARK\_INFO has relationship with USER. Each bookmark info has user. User\_id is foreign keys in BOOKMARK\_INFO entity.
* User – Bookmark\_Info : 1 - Many
* BOOKMARK\_INFO has relationship with BOOKMARK. Each bookmark has informations that can be more than one. Bookmark\_id is foreign key in BOOKMARK\_INFO entity.
* Bookmark – Bookmark\_Info : 1 - Many
* FEED has relationships with FEED\_CATEGORY. Each feed has own category. Feed\_Category\_id is foreign key in FEED entity.
* Feed\_Category – Feed : 1 - Many
* FEED has relationships with FEED\_SUB\_CATEGORY. Each feed has own sub category. Feed\_Sub\_Category\_id is foreign key in FEED entity.
* Feed\_Sub\_Category – Feed : 1 – Many
* FEED\_SUB\_CATEGORY has relationship with FEED\_CATEGORY. Each feed sub category belongs to feed category. Feed\_Category\_id is foreign key in FEED\_SUB\_CATEGORY entity.
* Feed\_Category – Feed\_Sub\_Category : 1 - Many
* FEED\_INFO has relationship with USER. Each feed info has user. User\_id is foreign key in FEED\_INFO entity.
* User – Feed\_Info : 1 - Many
* FEED\_INFO has relationships with FEED. Each feed has informations that can be more than one. Feed\_id is foreign key in FEED\_INFO entity.
* Feed – Feed\_Info : 1 - Many

### Relationships among the entities for Linked IN database model

* Members have relationships with Organizations,Ref\_Marital\_Status , Addresses. Members have current organization,address info and marital status infos . Current\_Organization\_ID , Address\_ID , Martial\_Status\_Code are foreign keys in Members entity.
* Members - Organizations : 1 to 1
* Members - Addresses : 1 to 1
* Members - Ref\_Marital\_Status : 1 to 1
* Connections has a relationship with Members.To make connection needs two people and Connection\_Member\_ID and Member\_ID are foreign keys in Connections entity.
* Connections – Members : Many to Many
* People\_Being\_Followed has a relationship with Members.To follow people needs Member\_ID. Member\_ID and Member\_Being\_Followed\_ID are foreign keys.
* People\_Being\_Followed – Members : Many to Many
* Recommmendatations has a relationship with Members.To recommend somebody to somebody , Member\_IDs are needed. Member\_Recommending\_ID and Member\_Being\_Recommended\_ID are Foreign keys.
* Recommendations – Members : Many to Many
* CVs entity has a relationship with Members.Every CV belongs to one member and Member\_ID a foreign key in CVs entity.
* CV – Members : 1 to 1
* CV\_Sections has relationships with CVs and Ref\_CV\_Sections. Every CV has sections and some sections. CV\_ID and CV\_Section\_Code are foreign keys in CV\_Sections entity.
* CV\_Sections - CVs : Many to 1
* CV\_Sections - Ref\_CV\_Sections : 1 to 1
* Members\_Profiles has a relationship with Members.Every member has profile and Member\_ID is a foreign key in Members\_Profiles entity.
* Members\_Profiles – Members : 1 to 1
* Profile\_Sections has a relationship with Ref\_Profile\_Sections.Every profile is referenced by different sections. Profile\_Section\_Code is a foreign key in Profile\_Sections.
* Profile\_Sections - Ref\_Profile\_Sections : many to many
* Groups entity has a relationship with Members\_Groups.Every groups is created by one member. Created\_By\_Member\_ID is a foreign key in Groups entity.
* Groups – Members : many to many
* Members\_Groups has relationships with Groups , Member entities.Members join groups with Member\_ID so Member\_ID and Group\_ID are foreign keys in Member\_Groups entities.
* Members\_Groups – Members : many to many
* Members\_Groups – Groups : 1 to 1

## Constraints related to entities

### Facebook Constraints

#### Referential Integrity Constraints

* Country\_id is a foreign key in City entity.

• CITY.Country\_id -> COUNTRY.Country\_id

* City\_id is a foreign key in Address entity.

• ADDRESS.City\_id -> CITY.City\_id

* Profile\_id is a foreign key in ADDRESS entity.

• ADDRESS.Profile\_id -> PROFILE.Profile\_id

* User\_id is a foreign key in Lang entity.

• LANG.User\_id -> USER.User\_id

* User\_id is foreign key in NICKNAME entity.

• NICKNAME.User\_id -> USER.User\_id

* Friend\_id and User\_id are foreign keys in FRIEND\_LIST entity.

• FRIEND\_LIST.Friend\_id -> FRIEND.Friend\_id

• FRIEND\_LIST.User\_id -> USER.User\_id

* Friend\_list\_id, Friend\_user\_id and User\_id are foreign keys in FRIEND entity.

• FRIEND.User\_id -> USER.User\_id

• FRIEND.Friend\_user\_id -> USER.User\_id

• FRIEND.Friend\_list\_id -> FRIEND\_LIST.Friend\_list\_id

* Status\_id and Friend\_id are foreign keys in THUMB\_UP\_DOWN entity.

• THUMB\_UP\_DOWN.Friend\_id -> FRIEND.Friend\_id

• THUMB\_UP\_DOWN.Status\_id -> STATUS.Status\_id

* User\_id is foreign key in PROFILE entity.

• PROFILE.User\_id -> USER.User\_id

* Status\_id and Friend\_id are foreign keys in COMMENT entity.

• COMMENT.Status\_id -> STATUS.Status\_id

• COMMENT.Friend\_id -> FRIEND.Friend\_id

* User\_id is a foreign key in STATUS entity.

• STATUS.User\_id -> USER.User\_id

* User\_id is foreign key in NOTIFICATION entity.

• NOTIFICATION.User\_id -> USER.User\_id

* User\_id and To\_User\_id are foreign keys in CHAT entity.

• CHAT.User\_id -> USER.User\_id

• CHAT.To\_User\_id -> USER.User\_id

* User\_id is a foreign key in PRIVACY entity.

• PRIVACY.User\_id -> USER.User\_id

* User\_id and To\_User\_id are foreign keys in MESSAGE entity.

• MESSAGE.User\_id -> USER.User\_id

• MESSAGE.To\_User\_id -> USER.User\_id

* Bookmark\_Category\_id and Bookmark\_Sub\_Category\_id are foreign keys in BOOKMARK entity.

• BOOKMARK. Bookmark\_Category\_id -> BOOKMARK\_CATEGORY. Bookmark\_Category\_id

• BOOKMARK. Bookmark\_Sub\_Category\_id -> BOOKMARK\_SUB\_CATEGORY. Bookmark\_Sub\_Category\_id

* Bookmark\_Category\_id is foreign key in BOOKMARK\_SUB\_CATEGORY.

• BOOKMARK\_SUB\_CATEGORY. Bookmark\_Category\_id -> BOOKMARK\_CATEGORY. Bookmark\_Category\_id

* Bookmark\_id and User\_id are foreign keys in BOOKMARK\_INFO entity.

• BOOKMARK\_INFO.Bookmark\_id -> BOOKMARK.Bookmark\_id

• BOOKMARK.User\_id -> USER.User\_id

* Feed\_Category\_id and Feed\_Sub\_Category\_id are foreign keys in FEED entity.

• FEED. Feed\_Category\_id -> FEED\_CATEGORY. Feed\_Category\_id

• FEED. Feed\_Sub\_Category\_id -> FEED\_SUB\_CATEGORY. Feed\_Sub\_Category\_id

* Feed\_Category\_id is foreign key in FEED\_SUB\_CATEGORY entity.

• FEED\_SUB\_CATEGORY.Feed\_Category\_id -> FEED\_CATEGORY.Feed\_Category\_id

* Feed\_id and User\_id are foreign keys in FEED\_INFO entity.

• FEED\_INFO.Feed\_id -> FEED.Feed\_id

• FEED\_INFO.User\_id -> USER.User\_id

#### Key Constraints

* USER> Primary Key: User\_id
* PROFILE> Primary Key(Profile\_id), Unique(User\_id)
* STATUS> Primary Key(Status\_id)
* PRIVACY> Primary Key(Privacy\_id), Unique(User\_id)
* COUNTRY> Primary Key(Country\_id)
* CITY> Primary Key(City\_id)
* ADDRESS> Primary Key(Address\_id), Unique(Profile\_id)
* LANG> Primary Key(Lang\_id), Unique(User\_id)
* NICKNAME> Primary Key(Nickname\_id), Unique(User\_id)
* FRIEND\_LIST> Primary Key(Friend\_list\_id)
* FRIEND> Primary Key(Friend\_id)
* THUMB\_UP\_DOWN> Primary Key(Thumb\_up\_down\_id)
* COMMENT> Primary Key(Comment\_id)
* NOTIFICATION> Primary Key(Notification\_id)
* CHAT> Primary Key(Chat\_id)
* MESSAGE> Primary Key(Message\_id)
* BOOKMARK> Primary Key(Bookmark\_id)
* BOOKMARK\_CATEGORY> Primary Key(Bookmark\_category\_id), Unique(Name)
* BOOKMARK\_SUB\_CATEGORY> Primary Key(Bookmark\_sub\_category\_id), Unique(Name)
* BOOKMARK\_INFO> Primary Key(Bookmark\_info\_id)
* FEED> Primary Key(Feed\_id)
* FEED\_CATEGORY> Primary Key(Feed\_category\_id), Unique(Name)
* FEED\_SUB\_CATEGORY> Primary Key(Feed\_sub\_category\_id), Unique(Name)
* FEED\_INFO> Primary Key(Feed\_info\_id)
* PRIVACY\_TYPE> Primary Key(Privacy\_type\_id), Unique(Name)

### Linked IN Constraints

#### Referential Integrity Constraints

* Members.Current\_Organization\_ID 🡪 Organizations.Organization\_ID
* Members.Address\_ID 🡪 Addresses.Address\_ID
* Members.Marital\_Status\_Code 🡪 Ref\_Marital\_Status.Marital\_Status\_Code
* Connections.Connection\_Member\_ID 🡪 Members.Member\_ID
* Connections.Member\_ID 🡪 Members.Member\_ID
* People\_Being\_Followed.Member\_ID 🡪 Members.Member\_ID
* People\_Being\_Followed.Member\_Being\_Followed\_ID 🡪 Members.Member\_ID
* Recommendations.Member\_Recommending\_ID 🡪 Members.Member\_ID
* REcommendations.Member\_Being\_Recommended\_ID 🡪 Members.Member\_ID
* CVs.Member\_ID 🡪 Members.Member\_ID
* CV\_Sections.CV\_ID 🡪 CVs.CV\_ID
* CV\_Sections.CV\_Section\_Code 🡪 Ref\_CV\_Sections.CV\_Section\_Code
* Members\_Profiles.Member\_ID 🡪 Members.Member\_ID
* Profile\_Sections.Profile\_Section\_Code 🡪 Ref\_Profile\_Sections 🡪 Profile\_Section\_Code
* Groups. Created\_By\_Member\_ID 🡪 Members.Member\_ID
* Members\_Groups.Member\_ID 🡪 Members.Member\_ID
* Members\_Groups.Group\_ID 🡪 Groups.Group\_ID

#### Key Constraints

* Organizations > Primary Key(Organization\_ID)
* Connections > Primary Key(Connection\_ID)
* People\_Being\_Followed > Primary Key(Date\_Started\_Following)
* Recommendations> Primary Key(Member\_Recommending\_ID , Member\_Being\_Recommended\_ID)
* CV> Primary Key(CV\_ID)
* CV\_Sections> Primary Key(CV\_Section\_ID)
* Ref\_CV\_Sections> Primary Key (CV\_Section\_Code)
* Members\_Profile> Primary Key(Profile\_ID)
* Profile\_Sections> Primary Key(Profile\_Section\_ID)
* Ref\_Profile\_Sections> Primary Key(Profile\_Section\_Code)
* Members\_Groups> Primary Key(Member\_ID,Group\_ID)
* Groups> Primary Key(Group\_ID)
* Address> Primary Key(Address\_ID)
* Ref\_Marital\_Status> Primary Key(Marital\_Status\_Code)
* Members> Primary Key(Member\_ID)

# DDL Statements

**>>**  We used Microsoft SQL Server 2012 for our project.

Use master

Go

**CREATE DATABASE FaceLinked**

Go

Use FaceLinked

Go

**CREATE TABLE MEMBER**

**(**

Member\_id int NOT NULL IDENTITY (1,1),

Fname nvarchar(20) NOT NULL,

Lname nvarchar(25) NOT NULL,

Email nvarchar(50) NOT NULL,

Password nvarchar(20) NOT NULL,

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Member\_id),

Unique (Email),

CONSTRAINT Password\_Const CHECK(len(Password) >= 6)

**);**

**CREATE TABLE ORGANIZATION**

(

Organization\_id int NOT NULL IDENTITY(1,1),

Name nvarchar(40) NOT NULL,

City nvarchar(167) NOT NULL,

PRIMARY KEY(Organization\_id)

**);**

**CREATE TABLE JOB\_OFFER**

**(**

Job\_offer\_id int NOT NULL IDENTITY(1,1),

Organization\_id int NOT NULL,

Offer\_date date NOT NULL DEFAULT GETDATE(),

Description nvarchar(500),

PRIMARY KEY(Job\_offer\_id),

FOREIGN KEY(Organization\_id) REFERENCES ORGANIZATION(Organization\_id)

ON UPDATE CASCADE ON DELETE CASCADE

**);**

**CREATE TABLE APPLICATION**

**(**

Application\_id int NOT NULL IDENTITY(1,1),

Job\_offer\_id int NOT NULL,

Member\_id int NOT NULL,

App\_date date NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Application\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Job\_offer\_id) REFERENCES JOB\_OFFER(Job\_offer\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

UNIQUE(Job\_offer\_id,Member\_id)

**);**

**CREATE TABLE PRIVACY**

**(**

Privacy\_id int NOT NULL IDENTITY(1,1),

Privacy\_status nvarchar(25)

PRIMARY KEY(Privacy\_id)

**);**

**CREATE TABLE PROFILE**

**(**

Profile\_id int NOT NULL IDENTITY(1,1),

Member\_id int NOT NULL,

Num\_of\_friends int NOT NULL DEFAULT(0),

Birth\_date date NOT NULL,

Sex nvarchar(10) NOT NULL CHECK (Sex='Male' OR SEX='Female'),

Marital\_status nvarchar(10) NOT NULL CHECK (Marital\_status='Single' OR Marital\_status='Married'),

Organization\_id int,

Phone nvarchar(15),

Photo nvarchar(100),

Privacy int NOT NULL DEFAULT(2),

Religion nvarchar(30),

Fav\_animal nvarchar(100),

Fav\_artist nvarchar(100),

Fav\_book nvarchar(100),

Fav\_movie nvarchar(100),

PRIMARY KEY(Profile\_id),

UNIQUE(Member\_id),

FOREIGN KEY(Privacy) REFERENCES PRIVACY(Privacy\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Organization\_id) REFERENCES ORGANIZATION(Organization\_id)

ON UPDATE CASCADE ON DELETE SET NULL

**);**

**CREATE TABLE HOBBIE**

**(**

Hobbie\_id int NOT NULL IDENTITY(1,1),

Profile\_id int NOT NULL,

Hobbie nvarchar(50),

PRIMARY KEY(Hobbie\_id),

FOREIGN KEY(Profile\_id) REFERENCES PROFILE(Profile\_id)

ON UPDATE CASCADE ON DELETE CASCADE

**);**

**CREATE TABLE ADDRESS**

**(**

Address\_id int NOT NULL IDENTITY(1,1),

Profile\_id int NOT NULL,

Address nvarchar(75) NOT NULL,

City nvarchar(167) NOT NULL,

Country nvarchar(45) NOT NULL,

Zip nvarchar(15) NOT NULL,

Privacy int NOT NULL DEFAULT(1),

PRIMARY KEY(Address\_id),

FOREIGN KEY(Profile\_id) REFERENCES PROFILE(Profile\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Privacy) REFERENCES PRIVACY(Privacy\_id),

UNIQUE(Profile\_id)

**);**

**CREATE TABLE FRIEND**

**(**

Friend\_id int NOT NULL IDENTITY(1,1),

Member\_id int NOT NULL,

Friend\_member\_id int NOT NULL,

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Friend\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Friend\_member\_id) REFERENCES MEMBER(Member\_id),

UNIQUE(Member\_id,Friend\_member\_id),

CONSTRAINT Friend\_const CHECK (Member\_id != Friend\_member\_id)

**);**

**CREATE TABLE STATUS**

**(**

Status\_id int NOT NULL IDENTITY(1,1),

Member\_id int NOT NULL,

Thumbs\_up int NOT NULL DEFAULT(0),

Thumbs\_down int NOT NULL DEFAULT(0),

Message nvarchar(255) NOT NULL,

To\_twitter bit NOT NULL,

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

Privacy int NOT NULL DEFAULT(2),

PRIMARY KEY(Status\_id),

FOREIGN KEY(Privacy) REFERENCES PRIVACY(Privacy\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

**);**

**CREATE TABLE THUMB\_UP\_DOWN**

**(**

Thumb\_id int NOT NULL IDENTITY(1,1),

Status\_id int NOT NULL,

Member\_id int NOT NULL,

Flag bit NOT NULL,

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Thumb\_id),

FOREIGN KEY(Status\_id) REFERENCES STATUS(Status\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

**);**

**CREATE TABLE COMMENT**

**(**

Comment\_id int NOT NULL IDENTITY(1,1),

Status\_id int NOT NULL,

Member\_id int NOT NULL,

Message nvarchar(100),

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Comment\_id),

FOREIGN KEY(Status\_id) REFERENCES STATUS(Status\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

**);**

**CREATE TABLE FOLLOW**

**(**

Member\_id int NOT NULL,

Following\_id int NOT NULL,

Following\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Member\_id,Following\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Following\_id) REFERENCES MEMBER(Member\_id),

CONSTRAINT Following\_const CHECK (Member\_id!=Following\_id)

**);**

**CREATE TABLE RECOMMEND**

**(**

Member\_id int NOT NULL,

Recommender\_id int NOT NULL,

Being\_rec\_id int NOT NULL,

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Member\_id,Recommender\_id,Being\_rec\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Being\_rec\_id) REFERENCES MEMBER(Member\_id),

FOREIGN KEY(Recommender\_id) REFERENCES MEMBER(Member\_id),

CONSTRAINT Recommend\_1 CHECK (Member\_id!=Being\_rec\_id),

CONSTRAINT Recommend\_2 CHECK (Member\_id!=Recommender\_id),

CONSTRAINT Recommend\_3 CHECK (Being\_rec\_id != Recommender\_id)

**);**

**CREATE TABLE MESSAGE**

**(**

Message\_id int NOT NULL IDENTITY(1,1),

Member\_id int NOT NULL,

To\_user int NOT NULL,

Message nvarchar(500),

is\_read bit NOT NULL default(0),

Created\_at smalldatetime NOT NULL DEFAULT CAST(GETDATE() as smalldatetime),

is\_spam bit,

PRIMARY KEY(Message\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(To\_user) REFERENCES MEMBER(Member\_id),

**);**

**CREATE TABLE NOTIFICATION**

**(**

Notification\_id int NOT NULL IDENTITY(1,1),

Member\_id int NOT NULL,

Comment\_id int DEFAULT(NULL),

Thumb\_id int DEFAULT(NULL),

Msg nvarchar(100) NOT NULL,

Created\_at smalldatetime NOT NULL DEFAULT GETDATE(),

PRIMARY KEY(Notification\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id),

FOREIGN KEY(Comment\_id) REFERENCES COMMENT(Comment\_id),

FOREIGN KEY(Thumb\_id) REFERENCES THUMB\_UP\_DOWN(Thumb\_id)

ON UPDATE CASCADE ON DELETE SET DEFAULT,

**);**

**CREATE TABLE CV**

**(**

Cv\_id int NOT NULL IDENTITY(1,1),

Member\_id int NOT NULL,

Cv\_title nvarchar(20) NOT NULL,

PRIMARY KEY(Cv\_id),

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

**);**

**CREATE TABLE LANGUAGE\_LEVEL**

**(**

Level\_id int NOT NULL IDENTITY(1,1),

Level\_name nvarchar(30) NOT NULL,

PRIMARY KEY(Level\_id)

**);**

**CREATE TABLE LANGUAGE**

**(**

Language\_id int NOT NULL IDENTITY(1,1),

Cv\_id int NOT NULL,

Language nvarchar(30) NOT NULL,

Level\_id int NOT NULL,

PRIMARY KEY(Language\_id),

FOREIGN KEY(Cv\_id) REFERENCES CV(Cv\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Level\_id) REFERENCES LANGUAGE\_LEVEL(Level\_id)

**);**

**CREATE TABLE EDUCATION**

**(**

Education\_id int NOT NULL IDENTITY(1,1),

Cv\_id int NOT NULL,

School\_name nvarchar(100),

Start\_date date NOT NULL,

Ending\_date date NOT NULL,

PRIMARY KEY(Education\_id),

FOREIGN KEY(Cv\_id) REFERENCES CV(Cv\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT Date\_const CHECK ( Start\_date < Ending\_date)

**);**

**CREATE TABLE WORK\_EXPERIENCE**

**(**

Work\_exp\_id int NOT NULL IDENTITY(1,1),

Cv\_id int NOT NULL,

Company\_name nvarchar(100) NOT NULL,

Info\_about\_work nvarchar(40) NOT NULL,

Start\_date date NOT NULL,

Leaving\_date date,

PRIMARY KEY(Work\_exp\_id),

FOREIGN KEY(Cv\_id) REFERENCES CV(Cv\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT Date\_work\_const CHECK (Start\_date < Leaving\_date)

**);**

**CREATE TABLE SKILL**

**(**

Skill\_id int NOT NULL IDENTITY(1,1),

Cv\_id int NOT NULL,

Skill nvarchar(40) NOT NULL,

PRIMARY KEY(Skill\_id),

FOREIGN KEY(Cv\_id) REFERENCES CV(Cv\_id)

ON UPDATE CASCADE ON DELETE CASCADE

**);**

**CREATE TABLE BOOKMARK\_CATEGORY**

**(**

Bookmark\_category\_id int NOT NULL IDENTITY(1,1),

Name nvarchar(30),

PRIMARY KEY(Bookmark\_category\_id),

UNIQUE(Name)

**);**

**CREATE TABLE BOOKMARK**

**(**

Bookmark\_id int NOT NULL IDENTITY(1,1),

Bookmark\_category\_id int NOT NULL,

Title nvarchar(100) NOT NULL,

Definition nvarchar(100) NOT NULL,

Creater\_id int NOT NULL,

Rating int NOT NULL DEFAULT(0),

PRIMARY KEY(Bookmark\_id),

FOREIGN KEY(Bookmark\_category\_id) REFERENCES BOOKMARK\_CATEGORY(Bookmark\_Category\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Creater\_id) REFERENCES MEMBER(Member\_id)

**);**

**CREATE TABLE BOOKMARK\_INFO**

**(**

Bookmark\_info\_id int NOT NULL IDENTITY(1,1),

Bookmark\_id int NOT NULL,

Member\_id int NOT NULL,

Favorite bit NOT NULL DEFAULT(0),

PRIMARY KEY(Bookmark\_info\_id),

FOREIGN KEY(Bookmark\_id) REFERENCES BOOKMARK(Bookmark\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

UNIQUE(Bookmark\_id, Member\_id)

**);**

**CREATE TABLE FEED\_CATEGORY**

**(**

Feed\_category\_id int NOT NULL IDENTITY(1,1),

Name nvarchar(30),

PRIMARY KEY(Feed\_category\_id),

UNIQUE(Name)

**);**

**CREATE TABLE FEED**

**(**

Feed\_id int NOT NULL IDENTITY(1,1),

Feed\_category\_id int NOT NULL,

Creater\_id int NOT NULL,

PRIMARY KEY(Feed\_id),

FOREIGN KEY(Feed\_category\_id) REFERENCES FEED\_CATEGORY(Feed\_category\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Creater\_id) REFERENCES MEMBER(Member\_id),

**);**

**CREATE TABLE FEED\_INFO**

**(**

Feed\_info\_id int NOT NULL IDENTITY(1,1),

Feed\_id int NOT NULL,

Member\_id int NOT NULL,

PRIMARY KEY(Feed\_info\_id),

FOREIGN KEY(Feed\_id) REFERENCES FEED(Feed\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(Member\_id) REFERENCES MEMBER(Member\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

UNIQUE(Feed\_id,Member\_id)

**);**

# Populate the database

**INSERT INTO LANGUAGE\_LEVEL**

**VALUES**('Beginner'),

('Elementary'),

('Pre-Intermediate'),

('Intermediate'),

('Upper Intermediate'),

('Advanced')**;**

**INSERT INTO BOOKMARK\_CATEGORY**

**VALUES**('Group'),

('Page'),

('Event')**;**

**INSERT INTO PRIVACY**

**VALUES**('Just me'),

('Just friends'),

('Everybody')**;**

**INSERT INTO FEED\_CATEGORY**

**VALUES**('THUMB'),

('COMMENT'),

('STATUS')**;**

**INSERT INTO ORGANIZATION**

**VALUES**('Osmanlıspor FK','Ankara'),

('Petkim PetroKimya Holding A.Ş.','İzmir'),

('Garanti Bankası','İstanbul')**;**

**INSERT INTO MEMBER**

**VALUES**('Onurhan','Çelik','celikonurhan@gmail.com','co4444','2014-09-10 20:50'),

('Ahmet','Gül','ahmet.gul9393@gmail.com','ag0202','2014-09-10 21:22'),

('Yılmaz','Vural','yvural@mynet.com','123456','2014-10-10 13:15'),

('Aylin','Bener','abener@hotmail.com','aa11bb22cc','2014-12-10 11:41'),

('Anıl','Öztürk','anil.ozturk35@gmail.com','buca3535','2014-12-10 09:38'),

('Deniz','Sarı','d.sari@hotmail.com','king55','2014-12-22 21:21')**;**

**INSERT INTO PROFILE**

**VALUES**(1,0,'1993-04-22','Male','Single',null,'5353707916','myphoto.jpeg',DEFAULT,'Muslim','Canary','Natalie Portman','Samarkand','Léon'),

(2,0,'1993-10-05','Male','Single',null,'5546525492','photo.png',DEFAULT,'Muslim','Fish','Matt Damon','Komiklig','The Departed'),

(3,0,'1953-01-01','Male','Married',1,'5553332211',null,DEFAULT,'Muslim',null,null,null,null),

(4,0,'1987-12-02','Female','Single',null,'5442213355','ben.jpeg',DEFAULT,'Muslim','Cat','Türkan Şoray','Başucumda Müzik','Unutursam Fısılda'),

(5,0,'1993-01-21','Male','Single',null,'5375007115',null,DEFAULT,'Muslim','Cat','Jim Carrey',null,'Eternal Sunshine of the Spotless Mind'),

(6,0,'1979-11-24','Male','Married',null,'5079729102','foto.png',DEFAULT,'Muslim','Giraffe','Jude Law','Improbable','Enemy of the Gates')**;**

**INSERT INTO HOBBIE**

**VALUES**(1,'Dart'),

(1,'Bisiklet'),

(4,'Kitap okumak'),

(3,'At binmek'),

(5,'Film izlemek'),

(6,'Tiyatroya gitmek'),

(2,'Yüzme')

(1,'Bilgisayar oyunları')**;**

**INSERT INTO ADDRESS**

**VALUES**(1,'Öğrenci Köyü Bornova','İzmir','Türkiye','35100',DEFAULT),

(2,'Karataş','İzmir','Türkiye','35260',DEFAULT),

(3,'Gazi Mah.','Ankara','Türkiye','06560',DEFAULT),

(4,'Kadıköy','İstanbul','Türkiye','34710',DEFAULT),

(5,'Buca','İzmir','Türkiye','35390',DEFAULT),

(6,'Yeşilyurt','İzmir','Türkiye','35160',DEFAULT)**;**

**INSERT INTO CV**

**VALUES**(1,'CV'),

(2,'CV'),

(3,'CV'),

(4,'CV'),

(5,'CV'),

(6,'CV')**;**

**INSERT INTO LANGUAGE**

**VALUES**(1,'English',4),

(2,'English',5),

(5,'English',4),

(3,'English',3),

(4,'English',4),

(4,'German',2),

(3,'German',4)**;**

**INSERT INTO EDUCATION**

**VALUES**(1,'Ömer Seyfettin Lisesi','2007','2011'),

(1,'Ege Üniversitesi - Bilgisayar Mühendisliği','2011','2016'),

(2,'İzmir Atatürk Lisesi','2010','2011'),

(2,'Ege Üniversitesi - Bigisayar Mühendisliği','2011','2016'),

(5,'Ege Üniversitesi - Bigisayar Mühendisliği','2011','2016'),

(3,'Adapazarı Lisesi','1970','1974'),

(3,'19 Mayıs Gençlik ve Spor Akademisi','1975','1979'),

(4,'ODTÜ - Kimya Mühendisliği','2005','2010'),

(6,'İstanbul Üniversitesi - Hukuk', '1998','2002')**;**

**INSERT INTO WORK\_EXPERIENCE**

**VALUES**(3,'Kasımpaşa SK','Teknik Direktör','2009','2012'),

(3,'Antalyaspor SK','Teknik Direktör','2005','2007'),

(6,'Adalet Hukuk Bürosu','Avukat','2005','2010'),

(4,'Çimentaş Çimento Fabrikası','Stajyer','2009-07-10','2009-08-20'),

(4,'Acar Boya Sanayi','Üretim Mühendisi','2006','2009')**;**

**INSERT INTO SKILL**

**VALUES**(1,'C'),

(1,'C#'),

(2,'C'),

(2,'C#'),

(5,'C'),

(5,'C#'),

(4,'Microsoft Excel'),

(4,'Autocad'),

(3,'Coaching'),

(3,'Motivation'),

(1,'Microsoft Office'),

(1,'Windows'),

(2,'Technology'),

(4,'Problem Solving'),

(4,'Industrial Experience'),

(6,'Critical Thinking'),

(6,'Investigative'),

(1,'Microsoft SQL Server')**;**

**INSERT INTO FRIEND VALUES**(1,2,'2014-09-10 22:15')**;**

**INSERT INTO FRIEND VALUES**(1,5,'2014-12-15 21:00')**;**

**INSERT INTO FRIEND VALUES**(2,5,DEFAULT)**;**

**INSERT INTO FRIEND VALUES**(1,4,DEFAULT)**;**

**INSERT INTO FRIEND VALUES**(4,6,DEFAULT)**;**

**INSERT INTO FRIEND VALUES**(3,6,DEFAULT)**;**

**INSERT INTO BOOKMARK VALUES**(1,'Sinema Aşıkları','Film önerme vs.',4,DEFAULT)**;**

**INSERT INTO BOOKMARK VALUES(**2,'Fenerbahçe','Fan page',1,DEFAULT**);**

**INSERT INTO BOOKMARK VALUES(**3,'Bil-Müh Party','20 Şubatta Eğlenmeye hazır mıyız?',2,DEFAULT)**;**

**INSERT INTO BOOKMARK\_INFO VALUES**(1,1,1)**;**

**INSERT INTO BOOKMARK\_INFO VALUES**(3,1,DEFAULT)**;**

**INSERT INTO BOOKMARK\_INFO VALUES**(3,5,1)**;**

**INSERT INTO BOOKMARK\_INFO VALUES**(2,3,1)**;**

**INSERT INTO BOOKMARK\_INFO VALUES**(1,6,DEFAULT)**;**

**INSERT INTO BOOKMARK\_INFO VALUES**(2,6,DEFAULT)**;**

**INSERT INTO FOLLOW VALUES**(5,3,DEFAULT)**;**

**INSERT INTO FOLLOW VALUES**(3,4,DEFAULT)**;**

**INSERT INTO RECOMMEND VALUES**(1,4,6,DEFAULT)**;**

**INSERT INTO RECOMMEND VALUES**(5,6,3,DEFAULT)**;**

**INSERT INTO JOB\_OFFER VALUES** (2,'2014-10-01','Deneyimli Kimya Mühendisi')**;**

**INSERT INTO JOB\_OFFER** VALUES (3,'2014-11-05','Gelişmeye açık yeni mezun Bilgisayar Mühendisi')**;**

**INSERT INTO APPLICATION VALUES**(1,4,'2014-10-06')**;**

**INSERT INTO APPLICATION VALUES**(2,1,'2014-11-06')**;**

**INSERT INTO APPLICATION VALUES**(2,2,'2014-11-07')**;**

**INSERT INTO APPLICATION VALUES**(2,5,DEFAULT);

**INSERT INTO MESSAGE VALUES**(1,5,'Partiye gidiyoz mu??',DEFAULT,'2014-12-15 20:42',null);

**INSERT INTO MESSAGE VALUES**(5,1,'İstanbulda olabilirim o tarihte duruma göre bakarız.',DEFAULT,'2014-12-15 20:45',null);

**INSERT INTO MESSAGE VALUES**(1,5,'Tamamdır..',DEFAULT,'2014-12-15 20:46',null);

**INSERT INTO MESSAGE VALUES**(5,3,'İyi maçtı hocam.',DEFAULT,DEFAULT,null);

**INSERT INTO MESSAGE VALUES**(3,6,'Bizim dava ne oldu avukat bey',DEFAULT,'2014-11-09 14:51',null);

**INSERT INTO MESSAGE VALUES**(6,3,'Her şey iyi görünüyor hocam',DEFAULT,'2014-11-09 15:12',null);

**INSERT INTO STATUS VALUES**(1,DEFAULT,DEFAULT,'Herkesin yeni yılı kutlu olsun.',1,'2014-12-28 23:21',DEFAULT);

**INSERT INTO STATUS VALUES**(4,DEFAULT,DEFAULT,'Kordonda balık sefası.. foto.jpg',0,'2014-11-30 20:50',DEFAULT)**;**

**INSERT INTO STATUS VALUES**(3,DEFAULT,DEFAULT,'Umarım sonraki maç daha iyi oynarız.',1,'2014-12-05 21:02',3)**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(1,2,1,'2014-12-28 23:26')**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(1,4,1,'2014-12-29 08:22')**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(1,5,1,'2014-12-29 14:32')**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(2,1,1,'2014-11-30 21:38')**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(2,6,1,DEFAULT)**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(3,1,0,DEFAULT)**;**

**INSERT INTO THUMB\_UP\_DOWN VALUES**(3,5,1,DEFAULT)**;**

**INSERT INTO COMMENT VALUES**(1,2,'Senin de Onurhan..','2014-12-28 23:27')**;**

**INSERT INTO COMMENT VALUES**(2,1,'oo afiyet olsun..','2014-11-30 21:39')**;**

**INSERT INTO COMMENT VALUES**(2,4,'gel beraber olsun.','2014-11-30 21:45')**;**

**INSERT INTO COMMENT VALUES**(3,5,'iddaa yattı hocam ya','2014-12-05 22:02')**;**

# TRIGGERS

* TRG\_Friend adlı trigger, FRIEND tablosuna kayıt girildikten sonra çalışmaktadır. Kişilerin PROFILE tablosundaki arkadaş sayıları sahasını artırır. Ayrıca arkadaş olan kişilerin herhangi biri bir diğerini takip ediyorsa takip etme FOLLOW sahasından silinir.

|  |
| --- |
| Use FaceLinked  GO  **CREATE TRIGGER TRG\_Friend**  **ON FRIEND**  **AFTER INSERT**  **AS BEGIN**  UPDATE PROFILE SET PROFILE.Num\_of\_friends = PROFILE.Num\_of\_friends + 1 FROM PROFILE,inserted WHERE PROFILE.Member\_id = inserted.Member\_id  UPDATE PROFILE SET PROFILE.Num\_of\_friends = PROFILE.Num\_of\_friends + 1 FROM PROFILE,inserted WHERE PROFILE.Member\_id = inserted.Friend\_member\_id  DECLARE @Member int  DECLARE @Following int  INSERT INTO FRIEND(Member\_id,Friend\_member\_id)  SELECT Friend\_member\_id,Member\_id  FROM inserted    IF EXISTS(SELECT \*  FROM FOLLOW,inserted  WHERE inserted.Member\_id=FOLLOW.Member\_id AND inserted.Friend\_member\_id=FOLLOW.Following\_id)  BEGIN  DELETE FROM FOLLOW  WHERE FOLLOW.Member\_id IN(SELECT inserted.Member\_id  FROM inserted)  AND FOLLOW.Following\_id IN(SELECT inserted.Friend\_member\_id  FROM inserted)  END  **END;** |

* TRG\_Inc\_Rating adlı trigger BOOKMARK\_INFO’ya kayıt eklendikten sonra ilgili bookmarkın Rating sahasınıartırır.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Inc\_Rating**  **ON BOOKMARK\_INFO**  **AFTER INSERT**  **AS BEGIN**  UPDATE BOOKMARK SET BOOKMARK.Rating = BOOKMARK.Rating + 1  FROM BOOKMARK,inserted  WHERE BOOKMARK.Bookmark\_id IN (inserted.Bookmark\_id)  **END;** |

* TRG\_Notice\_Comment adlı trigger COMMENT tablosuna kayıt eklendikten sonra ilgili mesajı ayarlayıp Notification tablosuna ve Feed tablosuna otomatik kayıt eklenmesini sağlıyor.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Notice\_Comment**  **ON COMMENT**  **AFTER INSERT**  **AS BEGIN**  DECLARE @Message nvarchar(100)  SELECT @Message=(MEMBER.Fname + ' ' + MEMBER.Lname + ' senin paylaşımına yorum yaptı.')  FROM inserted,MEMBER  WHERE inserted.Member\_id=MEMBER.Member\_id  INSERT INTO NOTIFICATION(Member\_id,Comment\_id,Thumb\_id,Msg)  SELECT STATUS.Member\_id,inserted.Comment\_id,NULL,@Message  FROM inserted, STATUS  WHERE inserted.Status\_id=STATUS.Status\_id    INSERT INTO FEED(Feed\_category\_id,Creater\_id)  SELECT 2,Member\_id  FROM inserted  **END;** |

* TRG\_Notice\_Thumb adlı trigger THUMB\_UP\_DOWN tablosuna kayıt eklendikten sonra beğenip beğenmeme durumuna göre ilgili mesajı ayarlayıp Notification ve Feed tablosuna kayıt eklenmesini sağlıyor.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Notice\_Thumb**  **ON THUMB\_UP\_DOWN**  **AFTER INSERT**  **AS BEGIN**  DECLARE @Message nvarchar(100)  SELECT @Message=(MEMBER.Fname + ' ' + MEMBER.Lname)  FROM inserted,MEMBER  WHERE inserted.Member\_id = MEMBER.Member\_id  IF(SELECT Flag FROM inserted)=1  BEGIN  SELECT @Message = (@Message + ' ' + 'senin paylaşımını beğendi.')  UPDATE STATUS SET STATUS.Thumbs\_up=STATUS.Thumbs\_up+1  FROM STATUS,inserted WHERE STATUS.Status\_id IN(inserted.Status\_id)  END  ELSE  BEGIN  SELECT @Message = (@Message + ' ' + 'senin paylaşımını beğenmedi.')  UPDATE STATUS SET STATUS.Thumbs\_down=STATUS.Thumbs\_down+1  FROM STATUS,inserted WHERE STATUS.Status\_id IN(inserted.Status\_id)  END    INSERT INTO NOTIFICATION(Member\_id,Comment\_id,Thumb\_id,Msg)  SELECT STATUS.Member\_id,NULL,inserted.Thumb\_id,@Message  FROM inserted,STATUS |
| WHERE STATUS.Status\_id=inserted.Status\_id    INSERT INTO FEED(Feed\_Category\_id,Creater\_id)  SELECT 1,Member\_id  FROM inserted  **END;** | | |

* TRG\_Feed triggerı STATUS’e kayıt eklendikten sonra Feed tablosuna otomatik kayıt ekliyor.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Feed**  **ON STATUS**  **AFTER INSERT**  **AS BEGIN**  INSERT INTO FEED(Feed\_category\_id,Creater\_id)  SELECT 3,MEMBER.Member\_id  FROM inserted,MEMBER  WHERE inserted.Member\_id = MEMBER.Member\_id  **END;** |

* TRG\_Bookmark\_Creater adlı trigger BOOKMARK tablosuna kayıt eklendikten sonra otomatik olarak bookmark’ı kuran kişiyi ilgili bookmark’a Bookmark\_Info tablosunda kaydediyor.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Bookmark\_Creater**  **ON BOOKMARK**  **AFTER INSERT**  **AS BEGIN**  INSERT INTO BOOKMARK\_INFO(Bookmark\_id,Member\_id,Favorite)  SELECT inserted.Bookmark\_id,inserted.Creater\_id,1  FROM inserted  **END;** |

* TRG\_Feed\_Info adlı trigger FEED tablosuna kayıt girildikten sonra otomatik olarak Feed\_Info tablosuna, ilgili feed’in oluşturucusunun arkadaşlarını ve onu takip edenleri ekliyor.

|  |
| --- |
| **CREATE TRIGGER TRG\_Feed\_Info**  **ON FEED**  **AFTER INSERT**  **AS BEGIN**  INSERT INTO FEED\_INFO(Feed\_id,Member\_id)  SELECT Feed\_id,FRIEND.Friend\_member\_id  FROM inserted,FRIEND  WHERE inserted.Creater\_id = FRIEND.Member\_id    INSERT INTO FEED\_INFO(Feed\_id,Member\_id)  SELECT Feed\_id, FOLLOW.Member\_id  FROM inserted,FOLLOW  WHERE inserted.Creater\_id = FOLLOW.Following\_id  **END;** |

* TRG\_Message\_isspam adlı trigger MESSAGE tablosuna kayıt girildikten sonra o girilen kaydı update ediyor. Eğer mesaj atan kişi attığı kişiyle arkadaş değilse ‘is\_spam’ değişkenini True değeri olarak değiştiriyor, arkadaş iseler False olarak değiştiriyor.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Message\_isspam**  **ON MESSAGE**  **AFTER INSERT**  **AS BEGIN**  IF EXISTS ( SELECT \*  FROM FRIEND,inserted  WHERE FRIEND.Member\_id = inserted.Member\_id  AND FRIEND.Friend\_member\_id=inserted.To\_user)  BEGIN  UPDATE MESSAGE SET MESSAGE.is\_spam = 0  FROM MESSAGE,inserted  WHERE MESSAGE.Message\_id = inserted.Message\_id  END  ELSE  BEGIN  UPDATE MESSAGE SET MESSAGE.is\_spam = 1  FROM MESSAGE,inserted  WHERE MESSAGE.Message\_id = inserted.Message\_id  END  **END;** |

* TRG\_Set\_Rating\_ForDelete adlı triggerda BOOKMARK\_INFO entitysinden bir kayıt silindikten sonra yapılması gereken işlemleri yapıyoruz. Silinen kayıttan Bookmark\_ID’ye ulaşıyoruz ve bu bookmark\_ID’nin PK olduğu bookmark’taki rating’I 1 azaltıyoruz.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Set\_Rating\_ForDelete**  **ON BOOKMARK\_INFO**  **AFTER DELETE**  **AS BEGIN**  UPDATE BOOKMARK SET BOOKMARK.Rating = BOOKMARK.Rating-1  FROM deleted,BOOKMARK  WHERE deleted.Bookmark\_id = BOOKMARK.Bookmark\_id  **END;** |

* TRG\_After\_Delete\_Set adlı triggerda MEMBER entitysinden bir kayıt silindikten sonra yapılması gereken işlemleri yapıyoruz.Öncelikle silinecek member’ın beğenmiş olduğu Bookmark’lardaki toplam beğenme sayısını bir azaltıyoruz ve daha sonra silinecek olan bu kullanıcının arkadaşlarının toplam friend sayısını bir azaltıyoruz.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_After\_Delete\_Set**  **ON MEMBER**  **AFTER DELETE**  **AS BEGIN**  UPDATE BOOKMARK SET BOOKMARK.Rating = BOOKMARK.Rating-1  FROM deleted,BOOKMARK,BOOKMARK\_INFO  WHERE deleted.Member\_id = BOOKMARK\_INFO.Member\_id AND BOOKMARK.Bookmark\_id = BOOKMARK\_INFO.Bookmark\_id  UPDATE PROFILE SET PROFILE.Num\_of\_friends = PROFILE.Num\_of\_friends-1  FROM deleted,PROFILE,FRIEND  WHERE deleted.Member\_id = FRIEND.Member\_id AND FRIEND.Friend\_member\_id = PROFILE.Member\_id  **END;** |

* TRG\_After\_Delete\_Set\_Thumb adlı triggerda Thumb\_up\_Down’dan bir satırı sildikten sonra yapılması gereken işlemleri yapıyoruz. Öncelikle silinecek satırdaki flag attribute’dan silinecek satırın bir beğeni mi yoksa bir beğenmeme mi olduğunu anlıyoruz ve eğer bir beğenme ise o Status’teki beğenme sayısını bir azaltıyoruz.Eğer bir beğenmeme durumu ise o Status’teki beğenmeme sayısını bir azaltıyoruz .

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_After\_Delete\_Set\_Thumb**  **ON THUMB\_UP\_DOWN**  **AFTER DELETE**  **AS BEGIN**  DECLARE @Flag bit  SELECT @Flag = deleted.flag FROM deleted    IF ( @Flag = 1 )  BEGIN  UPDATE STATUS SET STATUS.Thumbs\_up = STATUS.Thumbs\_up -1  FROM STATUS,deleted  WHERE STATUS.Status\_id = deleted.Status\_id  END    ELSE  BEGIN  UPDATE STATUS SET STATUS.Thumbs\_down = STATUS.Thumbs\_down -1  FROM STATUS,deleted  WHERE STATUS.Status\_id = deleted.Status\_id  END  **END;** |

* TRG\_Member\_Delete adlı trigger MEMBER sahasından bir kayıt silinince çalışır. Silinen kaydın ilişkili olduğu FRIEND, THUMB\_UP\_DOWN, COMMENT, FOLLOW, RECOMMEND, MESSAGE, NOTIFICATION, BOOKMARK, FEED ve STATUS tablolarındaki kayıtları da siler.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Member\_Delete**  **ON MEMBER**  **INSTEAD OF DELETE**  **AS BEGIN**  DELETE FROM FRIEND  WHERE FRIEND.Friend\_member\_id  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM THUMB\_UP\_DOWN  WHERE THUMB\_UP\_DOWN.Member\_id  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM COMMENT  WHERE COMMENT.Member\_id  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM FOLLOW  WHERE FOLLOW.Following\_id  IN ( SELECT Member\_id  FROM deleted)  DELETE FROM RECOMMEND  WHERE RECOMMEND.Being\_Rec\_id  IN ( SELECT Member\_id  FROM deleted)  DELETE FROM RECOMMEND  WHERE RECOMMEND.Recommender\_id  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM MESSAGE  WHERE MESSAGE.To\_user  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM NOTIFICATION  WHERE NOTIFICATION.Member\_id  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM BOOKMARK  WHERE BOOKMARK.Creater\_id  IN ( SELECT Member\_id  FROM deleted)    DELETE FROM FEED  WHERE FEED.Creater\_id  IN ( SELECT Member\_id  FROM deleted)  DELETE FROM STATUS  WHERE STATUS.Member\_id  IN ( SELECT Member\_id |
| FROM deleted)  DELETE FROM MEMBER WHERE Member\_id IN (SELECT Member\_id FROM deleted)  **END;** | | |

* TRG\_Delete\_Comment adlı trigger COMMENT tablosundan kayıt silinince çalışır. Silinen kaydın ilişkili olduğu NOTIFICATION tablosundaki kaydı update yapar.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Delete\_Comment**  **ON COMMENT**  **INSTEAD OF DELETE**  **AS BEGIN**  UPDATE NOTIFICATION SET NOTIFICATION.Comment\_id=NULL  WHERE NOTIFICATION.Comment\_id  IN ( SELECT Comment\_id  FROM deleted  )  DELETE FROM COMMENT WHERE Comment\_id IN (SELECT Comment\_id FROM deleted)  **END;** |

* TRG\_Delete\_Friend adlı triggerda Friend tablosundan bir satırın silinmesi durumunu düzelttik.Öncelikli olarak arkadaş olunduğunda tabloya karşılıklı ekleme yapıldığı için FRIEND tablosundan silinecek arkadaşlıktaki Member\_id ve Friend\_member\_id ‘ye gore sildik ve silinen arkadaşlık sonucunda arkadaşlık sayılarını güncellemek sureti ile 1’er azalttık.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Delete\_Friend**  **ON FRIEND**  **FOR DELETE**  **AS BEGIN**  DELETE FROM FRIEND  FROM FRIEND,deleted  WHERE FRIEND.Member\_id=deleted.Friend\_member\_id AND FRIEND.Friend\_member\_id=deleted.Member\_id  UPDATE PROFILE SET PROFILE.Num\_of\_friends = PROFILE.Num\_of\_friends - 1 FROM PROFILE,deleted WHERE PROFILE.Member\_id = deleted.Member\_id  UPDATE PROFILE SET PROFILE.Num\_of\_friends = PROFILE.Num\_of\_friends - 1 FROM PROFILE,deleted WHERE PROFILE.Member\_id = deleted.Friend\_member\_id  **END;** |

* TRG\_Delete\_Status adlı triggerda Status’un silinmesi durumuna müdahale ettik .Status silinirken Statusun foreign key olarak gittiği entity olan Comment’ın içinden o Status ile ilgili satırları once sildik , daha sonra statusun kendisini siliyoruz.

|  |
| --- |
| GO  **CREATE TRIGGER TRG\_Delete\_Status**  **ON STATUS**  **INSTEAD OF DELETE**  **AS BEGIN**  DELETE FROM COMMENT WHERE Status\_id IN (SELECT Status\_id FROM deleted)  DELETE FROM STATUS WHERE Status\_id IN (SELECT Status\_id FROM deleted)  **END;** |

# Check Constraints and Assertions

## Check Constraints

* MEMBER tablosunda bulunan Password\_Const check constraint’i kullanıcıların şifresinin en az 6 karakterden oluşması gerektiğini belirtiyor.
* FRIEND tablosunda bulunan Friend\_const adlı check constraint kişinin kendisiyle arkadaş olmasını engelliyor.
* FOLLOW tablosunda bulunan Following\_const adlı check constraint kişinin kendisini takip etmesini engelliyor.
* RECOMMEND tablosunda bulunan 3 farklı Recommend\_1/2/3 check constraintleri kişinin kendisini önermesini, bir kişinin o kişiye o kişiyi önermesini engelliyor.
* EDUCATION tablosunda bulunan Date\_const adlı check constraint bir kişinin ilgili öğrenime başlama tarihinin bitirme tarihinden önce olmasını sağlıyor.
* WORK\_EXPERIENCE tablosunda bulunan Date\_work\_const adlı check constraint bir kişinin ilgili işe başlama tarihinin ayrılma tarihinden önce olmasını sağlıyor.

## Assertions

* Assertion\_Check\_Friendship adlı triggerı assertion gereksinimi olarak yaptık. RECOMMEND sahasına veri girilmesi durumunda kontrol yapıyoruz. Öneren kişinin öneriyi sunduğu kişiyle arkadaş olması gerektiğini ve önerilen kişiyle önerinin sunulduğu kişinin arkadaş olmaması gerektiğini sağlar bu assertion.

|  |
| --- |
| GO  **CREATE TRIGGER Assertion\_Check\_Friendship**  **ON RECOMMEND**  **FOR INSERT**  **AS BEGIN**  DECLARE @Mem\_id int  DECLARE @Rec\_id int  DECLARE @Being\_id int  SELECT @Mem\_id=Member\_id,@Rec\_id=Recommender\_id,@Being\_id=Being\_rec\_id  FROM inserted  IF EXISTS ( SELECT \*  FROM FRIEND  WHERE @Mem\_id=FRIEND.Member\_id AND @Being\_id=FRIEND.Friend\_member\_id)  BEGIN  print 'Önerinin sunulduğu kişiyle önerilen kişi arkadaş olmamalı..!'  rollback transaction  END  ELSE IF NOT EXISTS ( SELECT \*  FROM FRIEND  WHERE @Mem\_id=FRIEND.Member\_id AND @Rec\_id=FRIEND.Friend\_member\_id)  BEGIN  print 'Önerinin sunulduğu kişiyle öneri yapan arkadaş olmalı..!'  rollback transaction  END  **END;** |

* Assertion\_Check\_Job\_Dates adlı triggerı assertion varsayımı olarak kullandık. APPLICATION tablosuna kayıt girilmesi durumunda kontrol sağlıyoruz. Bir kişinin başvurduğu iş ilanının ilan tarihi başvuru tarihinden önce olması gereksinimi sağladık.

|  |
| --- |
| **GO CREATE TRIGGER Assertion\_Check\_Job\_Dates**  **ON APPLICATION**  **FOR INSERT**  **AS BEGIN**  DECLARE @O\_date date  DECLARE @A\_date date  SELECT @O\_date=JOB\_OFFER.Offer\_date, @A\_date=App\_date  FROM JOB\_OFFER,inserted  WHERE inserted.Job\_offer\_id=JOB\_OFFER.Job\_offer\_id  IF(@A\_date<@O\_date) BEGIN  print 'İşe başvurma tarihi iş ilanının tarihinden sonra olmalı..!'  rollback transaction  END  **END;** |

* Assertion\_Privacy\_Control\_Comment ve Assertion\_Privacy\_Control\_Thumb adlı triggerları assertion olarak kullandık. COMMENT veya THUMB\_UP\_DOWN tablosuna kayıt girilmesi durumunda kontrol sağlıyoruz. Bir kişi bir duruma (STATUS) yorum veya beğeni yapacaksa eğer durumun Privacy sahasına göre ve yorum/beğeni yapan kişiyle durumu paylaşan kişinin arkadaş olup olmamaları durumlarına göre yorumun atılmasını engelliyoruz. İlgili kaydın ilişkili olduğu Status tablosundaki kaydın Privacy sahası ‘1’ değeri yani ‘Sadece ben’ ise o duruma sadece durum sahibi yorum/beğeni yapabilir. ‘2’ değeri yani ‘Sadece arkadaşlar’ ise o duruma sadece durum sahibiyle arkadaş olanlar yorum/beğeni yapabilir. ‘3’ değeri yani ‘Herkes’ ise o duruma herkes yorum/beğeni yapabilir.

|  |
| --- |
| GO  **CREATE TRIGGER Assertion\_Privacy\_Control\_Comment**  **ON COMMENT**  **FOR INSERT**  **AS BEGIN**  DECLARE @Privacy int  DECLARE @who int  SELECT @Privacy = (STATUS.Privacy), @who=(STATUS.Member\_id)  FROM STATUS,inserted  WHERE Status.Status\_id = inserted.Status\_id  IF(@Privacy = 1 AND @who != (SELECT Member\_id FROM inserted)) /\*Just me\*/  BEGIN  print 'Error - Privacy Control..'  rollback transaction  END  ELSE IF(@Privacy = 2 AND @who != (SELECT Member\_id FROM inserted)) /\*Just friends\*/  BEGIN  IF NOT EXISTS (SELECT \*  FROM inserted,STATUS,FRIEND  WHERE inserted.Status\_id=STATUS.Status\_id AND STATUS.Member\_id=FRIEND.Member\_id AND  inserted.Member\_id=FRIEND.Friend\_member\_id)  BEGIN  print 'Error - Privacy Control..'  rollback transaction  END  END  **END;** |

|  |
| --- |
| GO  **CREATE TRIGGER Assertion\_Privacy\_Control\_Thumb**  **ON THUMB\_UP\_DOWN**  **FOR INSERT**  **AS BEGIN**  DECLARE @Privacy int  DECLARE @who int  SELECT @Privacy = (STATUS.Privacy), @who=(STATUS.Member\_id)  FROM STATUS , inserted  WHERE Status.Status\_id = inserted.Status\_id    IF(@Privacy = 1 AND @who != (SELECT Member\_id FROM inserted))  BEGIN  print 'Error - Privacy Control..'  rollback transaction |

*TRIGGER Assertion\_Privacy\_Control\_Thumb - Devamı*

|  |
| --- |
| ELSE IF (@Privacy = 2 AND @who != (SELECT Member\_id FROM inserted))  BEGIN  IF NOT EXISTS (Select \*  FROM inserted,STATUS,FRIEND  WHERE inserted.Status\_id = Status.Status\_id *AND*   STATUS.Member\_id = FRIEND.Member\_id *AND*  inserted.Member\_id = FRIEND.Friend\_member\_id)  BEGIN  print 'Error - Privacy Control..'  rollback transaction  END  END  **END;** |

# SQL Statements

## Sample INSERT, DELETE and UPDATE Statements

### INSERTs

|  |
| --- |
| **INSERT INTO MEMBER**(Fname,Lname,Email,Password)  **VALUES**('Cengiz','Bursalıoğlu','cengiz\_b@gmail.com','2583691')**;**  **INSERT INTO PROFILE**(Birth\_date,Fav\_animal,Fav\_artist,Fav\_book,Fav\_movie,  Marital\_status,Member\_id,Sex,Privacy)  **VALUES**('2014-02-09','Dog','Keremcem','Beyazıt','Av Mevsimi',  'Single','7','Male',DEFAULT)**;**  **INSERT INTO ADDRESS**(Address,City,Country,Privacy,Profile\_id,Zip)  **VALUES**('Niksar','TOKAT','Türkiye',DEFAULT,'7','846845')**;**  **INSERT INTO FRIEND VALUES**(7,1,DEFAULT);  **INSERT INTO THUMB\_UP\_DOWN VALUES**(1,7,1,DEFAULT); |

### UPDATEs

|  |
| --- |
| UPDATE ADDRESS SET City='İzmir' , Address = 'Bornova' WHERE ADDRESS.Profile\_id  IN ( SELECT PROFILE.Profile\_id  FROM PROFILE,MEMBER  WHERE PROFILE.Profile\_id = Member.Member\_id AND MEMBER.Fname = 'Cengiz' );  UPDATE PROFILE SET Phone = '05055555555' WHERE PROFILE.Member\_id = 7; |

### DELETEs

|  |
| --- |
| DELETE FROM ADDRESS WHERE ADDRESS.Address = 'Bornova';  DELETE FROM MEMBER WHERE Member\_id=7; |

## SELECT Statements

### Using one table

* Member\_id=1 olan kullanıcının favoriye aldığı bookmark sayısını veren sorgu.

**SELECT** COUNT(Favorite)

**FROM** BOOKMARK\_INFO

**WHERE** BOOKMARK\_INFO.Member\_id=1 *AND* BOOKMARK\_INFO.Favorite=1

* Kullanıcıların Member\_id'lerini ve her kullanıcının attıkları toplam mesaj sayısını veren sorgu.

**SELECT** Member\_id,COUNT(Member\_id) *as* Number\_of\_Message

**FROM** MESSAGE

**GROUP BY** Member\_id

* Mail adresinin uzantısı ‘gmail.com’ olan kişilerin ad soyadlarını veren sorguyu yazınız.

**SELECT** Fname,Lname

**FROM** MEMBER

**WHERE** MEMBER.Email *LIKE* '%gmail.com%'

### Using minimum 2 tables

* İzmir'deki firmaların açtığı iş ilanlarının ilan numarasını ve ilan tanımını veren sorgu.

**SELECT** JOB\_OFFER.Job\_offer\_id,JOB\_OFFER.Description

**FROM** JOB\_OFFER , ORGANIZATION

**WHERE** JOB\_OFFER.Organization\_id = ORGANIZATION.Organization\_id *AND* ORGANIZATION.City ='İzmir';

* Tüm kayıtlı kişileri arkadaş sayıları azalan şekilde ad,soyad ve arkadaş sayılarını veren sorguyu yazınız.

**SELECT** Fname,Lname,Num\_of\_friends

**FROM** MEMBER,PROFILE

**WHERE** MEMBER.Member\_id = PROFILE.Member\_id

**ORDER** BY Num\_of\_friends DESC

* Member\_id=6 olan kişinin arkadaşlarını ad,soyad şeklinde listeleyiniz.

**SELECT** MEMBER.Fname,MEMBER.Lname

**FROM** MEMBER

**WHERE** MEMBER.Member\_id *IN* (**SELECT** FRIEND.Friend\_member\_id

**FROM** MEMBER,FRIEND

**WHERE** MEMBER.Member\_id=6 *AND* MEMBER.Member\_id=FRIEND.Member\_id)

* Member\_id=5 olan kişiye gelen mesajları gönderen kişinin ad,soyadını ve mesajı veren sorguyu yazınız.

**SELECT** MEMBER.Fname,MEMBER.Lname,MESSAGE.Message

**FROM** MESSAGE,MEMBER

**WHERE** MESSAGE.Member\_id=MEMBER.Member\_id AND MESSAGE.Message\_id

*IN* (**SELECT** MESSAGE.Message\_id

**FROM** MESSAGE

**WHERE** MESSAGE.To\_user=5)

### Using minimum 3 tables

* Adres şehri İzmir olanları ad-soyad'a göre alfabetik şekilde sıralayan Member\_id, ad, soyad çıktısını veren sorguyu yazınız.

**SELECT** MEMBER.Member\_id,MEMBER.Fname, MEMBER.Lname

**FROM** ADDRESS,PROFILE,MEMBER

**WHERE** ADDRESS.City = 'İzmir' *AND* ADDRESS.Profile\_id = PROFILE.Profile\_id *AND* PROFILE.Member\_id=MEMBER.Member\_id

**ORDER BY** Fname,Lname

* Job\_offer\_id=2 olan iş ilanına başvuru yapan adayların İngilizce dilini bilenlerin ad,soyad ve İngilizce seviyelerini veren sorguyu yazınız. Sıralamayı dil seviyelerini azalacak şekilde yapınız.

**SELECT** MEMBER.Fname, MEMBER.Lname, LANGUAGE.Level\_id

**FROM** APPLICATION,MEMBER,CV,LANGUAGE

**WHERE** APPLICATION.Job\_offer\_id=2 *AND* APPLICATION.Member\_id=MEMBER.Member\_id AND MEMBER.Member\_id=CV.Member\_id

*AND* CV.Cv\_id=LANGUAGE.Cv\_id *AND* LANGUAGE.Language *LIKE* 'English'

**ORDER BY** LANGUAGE.Level\_id *DESC*

* Member\_no=1 olan kişiye gelen bildirimlerin yaratıcılarının Member\_id'sini ve bildirim mesajlarını veren sorgu.

(**SELECT** COMMENT.Member\_id, NOTIFICATION.Msg

**FROM** NOTIFICATION,COMMENT

**WHERE** NOTIFICATION.Member\_id=1 *AND* NOTIFICATION.Comment\_id=COMMENT.Comment\_id)

*UNION*

(**SELECT** THUMB\_UP\_DOWN.Member\_id, NOTIFICATION.Msg

**FROM** NOTIFICATION,THUMB\_UP\_DOWN

**WHERE** NOTIFICATION.Member\_id=1 *AND* NOTIFICATION.Thumb\_id=THUMB\_UP\_DOWN.Thumb\_id)