# Comp 2001 Set Exercises Report

## Entity Relationship Diagram

## Normalisation

## SQL Commands for creating tables

CREATE SCHEMA CW1

CREATE TABLE CW1.[User] (

user\_id INTEGER PRIMARY KEY IDENTITY(1,1),

username VARCHAR(50),

email VARCHAR(80),

password VARCHAR(40),

account\_type VARCHAR(15)

);

CREATE TABLE CW1.[Favourite\_Activities] (

activity\_id INTEGER,

user\_id INTEGER,

FOREIGN KEY (activity\_id) REFERENCES CW1.[Activity](activity\_id),

FOREIGN KEY (user\_id) REFERENCES CW1.[User](user\_id)

);

CREATE TABLE CW1.[Activity] (

activity\_id INTEGER PRIMARY KEY IDENTITY(1,1),

activity\_type VARCHAR(30)

);

CREATE TABLE CW1.[Follow\_List] (

user\_id INTEGER,

FOREIGN KEY (user\_id) REFERENCES CW1.[User](user\_id),

follow\_id INTEGER,

FOREIGN KEY (follow\_id) REFERENCES CW1.[User](user\_id),

);

CREATE TABLE CW1.[Archive\_User] (

user\_id INTEGER PRIMARY KEY,

username VARCHAR(50),

email VARCHAR(80),

password VARCHAR(40),

account\_type VARCHAR(15)

);

CREATE TABLE CW1.[Archive\_Favourite\_Activities] (

activity\_id INTEGER,

user\_id INTEGER,

FOREIGN KEY (activity\_id) REFERENCES CW1.[Activity](activity\_id),

FOREIGN KEY (user\_id) REFERENCES CW1.[Archive\_User](user\_id)

);

CREATE TABLE CW1.[Archive\_Follow\_List] (

user\_id INTEGER,

follow\_id INTEGER,

x);

Result:

## Creating Views

CREATE VIEW CW1.[Main\_View] AS

SELECT u.username AS "Username", u.email AS "Email", u.account\_type AS "Account Type",

(SELECT COUNT(\*) FROM CW1.[Follow\_List] f WHERE u.user\_id = f.user\_id) Following,

(SELECT COUNT(\*) FROM CW1.[Follow\_List] f WHERE u.user\_id = f.follow\_id) Followers,

STRING\_AGG(a.activity\_type,', ') AS "Favourite Activities"

FROM CW1.[User] u

LEFT JOIN CW1.[Favourite\_Activities] f ON u.user\_id = f.user\_id

LEFT JOIN CW1.[Activity] a ON f.activity\_id = a.activity\_id

GROUP BY u.user\_id, u.account\_type, u.username, u.email

SELECT \* FROM CW1.[Main\_View]

CREATE VIEW CW1.[Archive\_View] AS

SELECT u.username AS "Archive Username", u.email AS "Archive Email", u.account\_type AS "Archive Account Type",

(SELECT COUNT(\*) FROM CW1.[Archive\_Follow\_List] f WHERE u.user\_id = f.user\_id) Following,

(SELECT COUNT(\*) FROM CW1.[Archive\_Follow\_List] f WHERE u.user\_id = f.follow\_id) Followers,

STRING\_AGG(a.activity\_type,', ') AS "Favourite Activities"

FROM CW1.[Archive\_User] u

LEFT JOIN CW1.[Archive\_Favourite\_Activities] f ON u.user\_id = f.user\_id

LEFT JOIN CW1.[Activity] a ON f.activity\_id = a.activity\_id

GROUP BY u.user\_id, u.account\_type, u.username, u.email

SELECT \* FROM CW1.[Archive\_View]

Result:

## Stored procedures

CREATE PROCEDURE CW1.[Add\_User]

@username VARCHAR(50),

@email VARCHAR(80),

@password VARCHAR(40),

@account\_type VARCHAR(15)

AS

BEGIN

INSERT INTO CW1.[User] (username,email,password,account\_type)

VALUES

(@username, @email, @password, @account\_type);

END

EXEC CW1.[Add\_User] 'Veraint', 'Veraint@students.plymouth.ac.uk', 'testpassword', 'admin'

CREATE PROCEDURE CW1.[Edit\_Username]

@user\_id INTEGER,

@new\_username VARCHAR(50)

AS

BEGIN

UPDATE CW1.[User]

SET username = @new\_username

WHERE user\_id = @user\_id;

END

EXEC CW1.[Edit\_Username] 4, "Pat102"

CREATE PROCEDURE CW1.[Edit\_Email]

@user\_id INTEGER,

@new\_email VARCHAR(80)

AS

BEGIN

UPDATE CW1.[User]

SET email = @new\_email

WHERE user\_id = @user\_id;

END

EXEC CW1.[Edit\_Email] 4, "Pat102@gmail.com"

CREATE PROCEDURE CW1.[Edit\_Password]

@user\_id INTEGER,

@new\_password VARCHAR(40)

AS

BEGIN

UPDATE CW1.[User]

SET password = @new\_password

WHERE user\_id = @user\_id;

END

EXEC CW1.[Edit\_Password] 4, "Pats password"

CREATE PROCEDURE CW1.[Edit\_Account\_Type]

@user\_id INTEGER,

@new\_account\_type VARCHAR(15)

AS

BEGIN

UPDATE CW1.[User]

SET account\_type = @new\_account\_type

WHERE user\_id = @user\_id;

END

EXEC CW1.[Edit\_Account\_Type] 4, "user"

CREATE PROCEDURE CW1.[Archive\_User\_Procedure]

@user\_id INTEGER

AS

BEGIN

INSERT INTO CW1.[Archive\_User] (user\_id, username, email, password, account\_type)

SELECT \*

FROM CW1.[User]

WHERE user\_id = @user\_id;

INSERT INTO CW1.[Archive\_Favourite\_Activities] (activity\_id, user\_id)

SELECT \*

FROM CW1.[Favourite\_Activities]

WHERE user\_id = @user\_id;

INSERT INTO CW1.[Archive\_Follow\_List] (user\_id, follow\_id)

SELECT \*

FROM CW1.[Follow\_List]

WHERE user\_id = @user\_id;

INSERT INTO CW1.[Archive\_Follow\_List] (user\_id, follow\_id)

SELECT \*

FROM CW1.[Follow\_List]

WHERE follow\_id = @user\_id;

DELETE FROM CW1.[Favourite\_Activities]

WHERE user\_id = @user\_id;

DELETE FROM CW1.[Follow\_List]

WHERE user\_id = @user\_id;

DELETE FROM CW1.[Follow\_List]

WHERE follow\_id = @user\_id;

DELETE FROM CW1.[User]

WHERE user\_id = @user\_id;

END

EXEC CW1.[Archive\_User\_Procedure] 2

CREATE PROCEDURE CW1.[Follow\_User]

@user\_id INTEGER,

@follow\_id INTEGER

AS

BEGIN

INSERT INTO CW1.[Follow\_List] (user\_id, follow\_id)

VALUES

(@user\_id, @follow\_id);

END

EXEC CW1.[Follow\_User] 1 , 2

CREATE PROCEDURE CW1.[Unfollow\_User]

@user\_id INTEGER,

@follow\_id INTEGER

AS

BEGIN

DELETE FROM CW1.[Follow\_List]

WHERE @user\_id = user\_id AND @follow\_id = follow\_id

END

EXEC CW1.[Unfollow\_User] 1 , 2

CREATE PROCEDURE CW1.[Add\_Activity]

@activity\_type VARCHAR(30)

AS

BEGIN

INSERT INTO CW1.[Activity] (activity\_type)

VALUES

(@activity\_type);

END

EXEC CW1.[Add\_Activity] "Running"

CREATE PROCEDURE CW1.[Favourite\_Activity]

@user\_id INTEGER,

@activity\_id INTEGER

AS

BEGIN

INSERT INTO CW1.[Favourite\_Activities] (activity\_id, user\_id)

VALUES

(@user\_id, @activity\_id);

END

EXEC CW1.[Favourite\_Activity] 3, 4

CREATE PROCEDURE CW1.[Un\_Favourite\_Activity]

@user\_id INTEGER,

@activity\_id INTEGER

AS

BEGIN

DELETE FROM CW1.[Favourite\_Activities]

WHERE @user\_id = user\_id AND @activity\_id = activity\_id

END

EXEC CW1.[Un\_Favourite\_Activity] 3, 4

## Triggers

CREATE TRIGGER CW1.[Email\_Trigger]

ON CW1.[User]

FOR INSERT, UPDATE

AS

DECLARE @email VARCHAR(80)

SELECT email FROM INSERTED

WHERE NOT inserted.email LIKE '%@%.%'

BEGIN

RAISERROR ('Email is invalid', 16, 1);

ROLLBACK TRANSACTION;

END