DETI-MakerLab

Development of an application and database to manage an hacker space (MakerLab)

Diogo Ferreira 76425
Pedro Martins 76551

What is DETI-Maker Lab ?

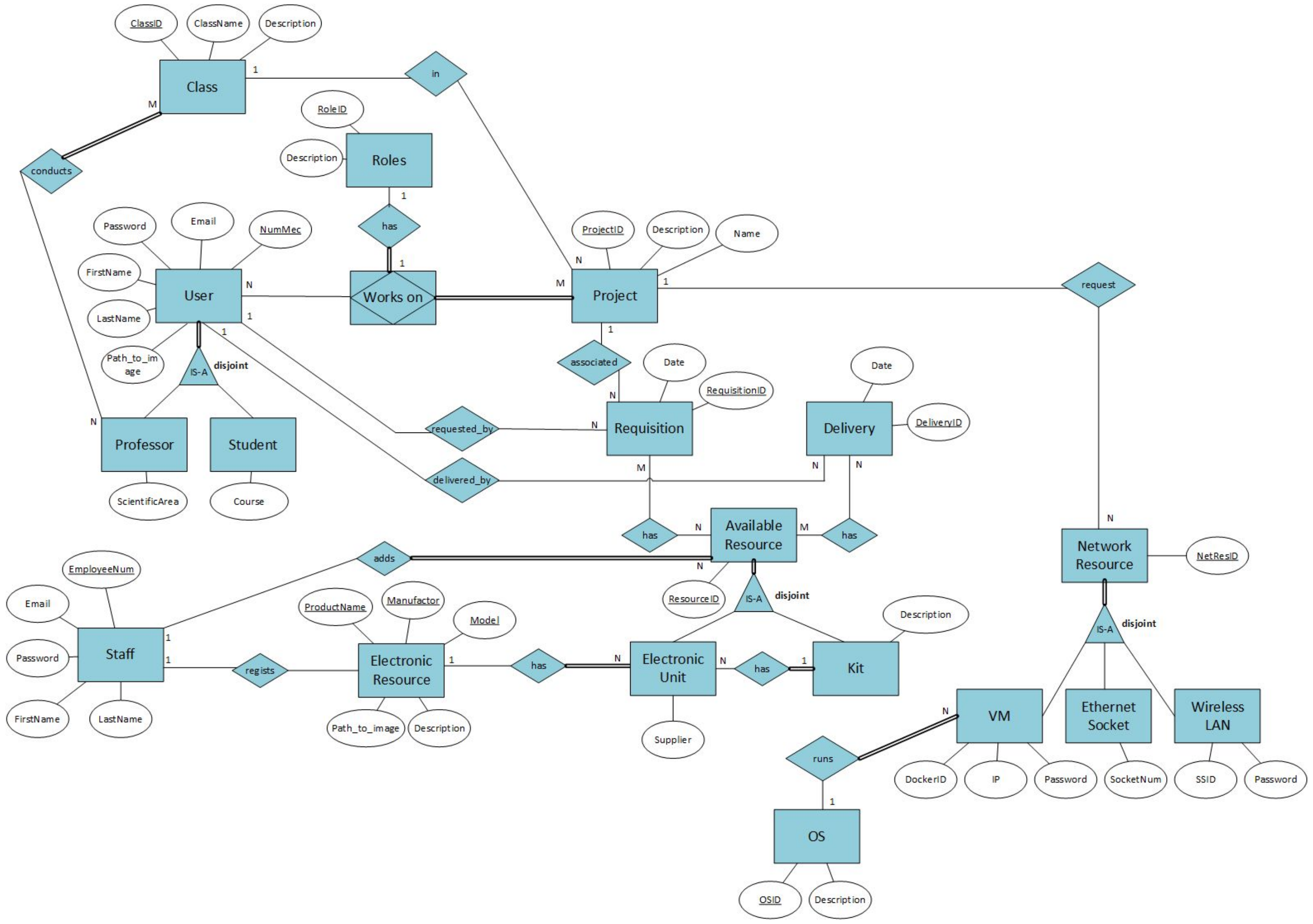
- DETI MakerLab app is a system designed to **manage a modern and innovative room**.
- This room is filled with electronic components and devices, such as Arduinos, Raspberries, 3D printers and a network closet. The space aims at being the room to carry on projects inside DETI.
- The DETI MakerLab software will hopefully address all the users needs to develop their projects at MakerLab.

What is DETI-Maker Lab ?

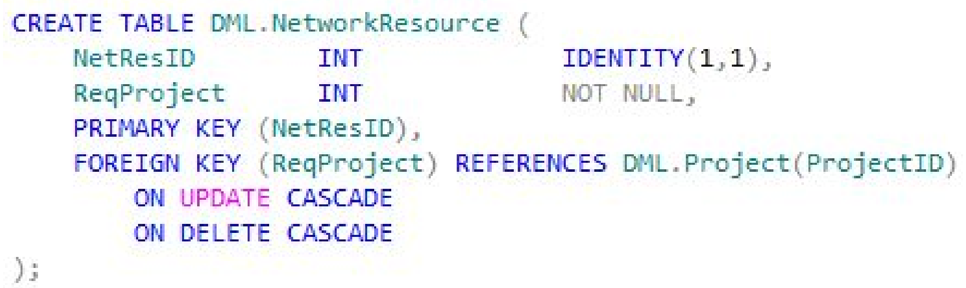
- Users (Students and Professors) & Staff
- Projects, with members (roles) and within a scope of a class (or not)
- Requisitions and deliveries of networking resources (WiFi, ethernet sockets, VMs)
- Requisitions and deliveries of electronic units and/or kits (aggregation of units)
- Electronic equipments may have multiple units

What we’ve done

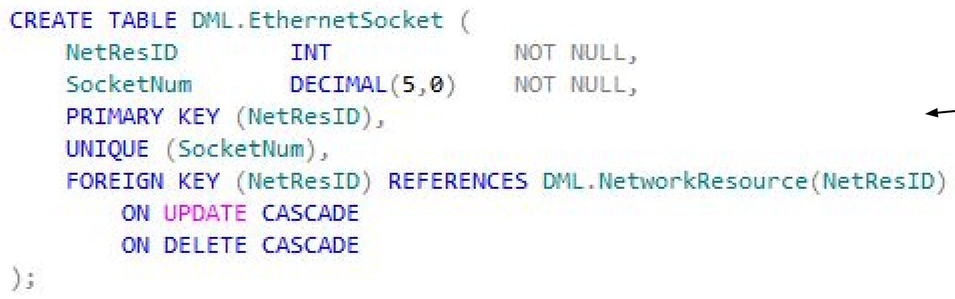
23 Tables
2 Types


ClassID
ClassName
Description
Class
in
RoleID
Description
Roles
has
conducts
Password
Email
NumMec
FirstName
User
Works on
Project
ProjectID
Description
Name
request
LastName
Path_to_image
IS-A
disjoint
associated
Date
RequisitionID
Date
DeliveryID
Delivery
Requisition
requested_by
delivered_by
Professor
Student
ScientificArea
Course
has
Available Resource
has
Network Resource
NetResID
adds
EmployeeNum
Email
Password
Staff
FirstName
LastName
regists
ProductName
Manufactor
Model
Electronic Resource
Path_to_image
Description
has
Electronic Unit
Supplier
has
Kit
Description
ResourceID
IS-A
disjoint
IS-A
disjoint
VM
Ethernet Socket
Wireless LAN
DockerID
IP
Password
SocketNum
SSID
Password
runs
OS
OSID
Description
1
M
N
M
1
1
1
N
1
1
N
N
M
N
N
N
N
M
N
1
1
1
N
N
1
N
N
1

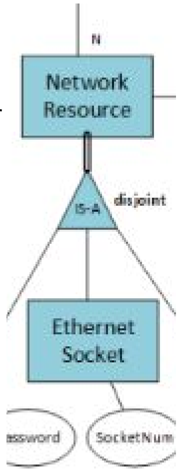
From general to particular (hierarchy)



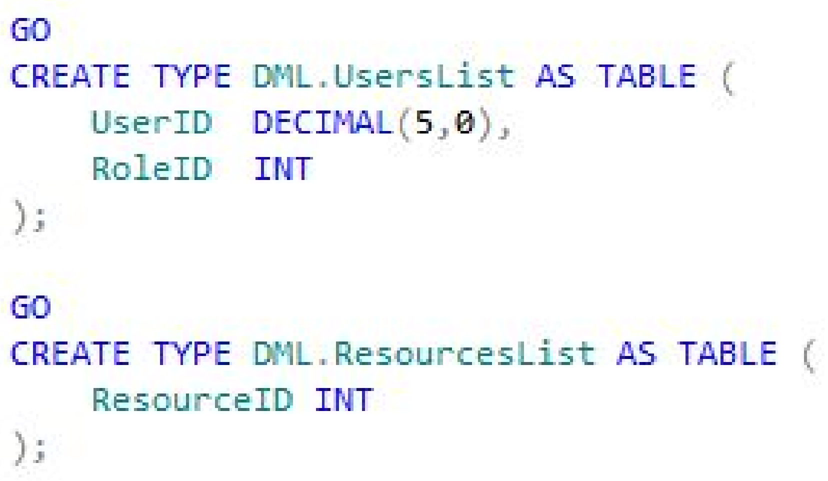
```
CREATE TABLE DML.NetworkResource (
    NetResID        INT             IDENTITY(1,1),
    ReqProject      INT             NOT NULL,
    PRIMARY KEY (NetResID),
    FOREIGN KEY (ReqProject) REFERENCES DML.Project(ProjectID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```



```
CREATE TABLE DML.EthernetSocket (
    NetResID        INT             NOT NULL,
    SocketNum       DECIMAL(5,0)    NOT NULL,
    PRIMARY KEY (NetResID),
    UNIQUE (SocketNum),
    FOREIGN KEY (NetResID) REFERENCES DML.NetworkResource(NetResID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```



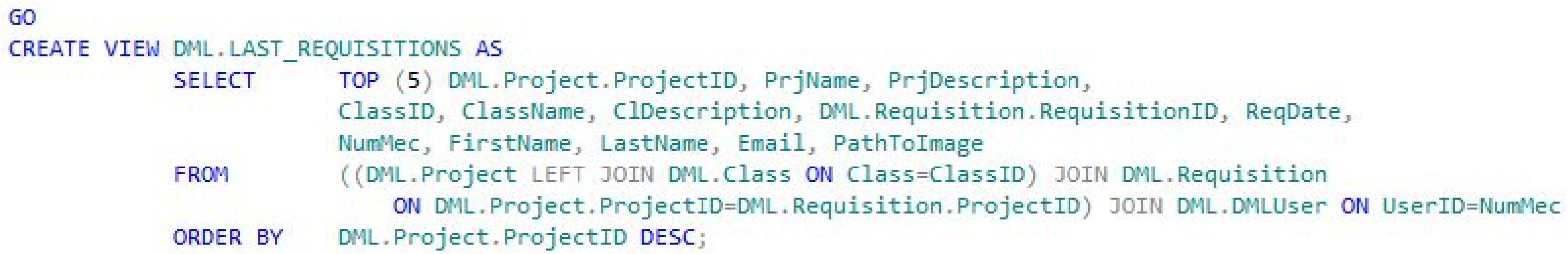
Type creation



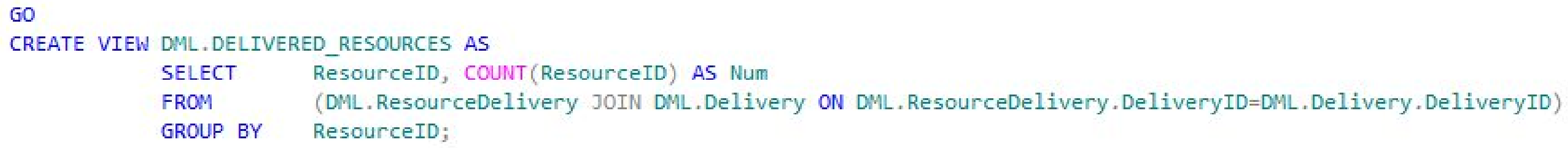
```
GO
CREATE TYPE DML.UsersList AS TABLE (
    UserID  DECIMAL(5,0),
    RoleID  INT
);

GO
CREATE TYPE DML.ResourcesList AS TABLE (
    ResourceID INT
);
```

9 Views

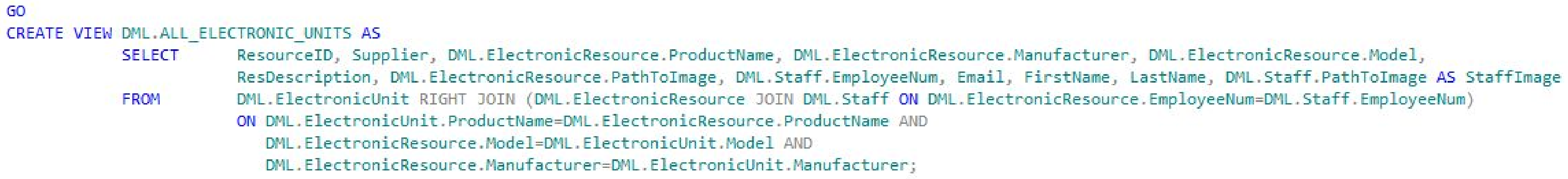
To help us getting some information



```
GO
CREATE VIEW DML.LAST_REQUISITIONS AS
            SELECT      TOP (5) DML.Project.ProjectID, PrjName, PrjDescription,
                        ClassID, ClassName, ClDescription, DML.Requisition.RequisitionID, ReqDate,
                        NumMec, FirstName, LastName, Email, PathToImage
            FROM        ((DML.Project LEFT JOIN DML.Class ON Class=ClassID) JOIN DML.Requisition
                            ON DML.Project.ProjectID=DML.Requisition.ProjectID) JOIN DML.DMLUser ON UserID=NumMec
            ORDER BY    DML.Project.ProjectID DESC;
```



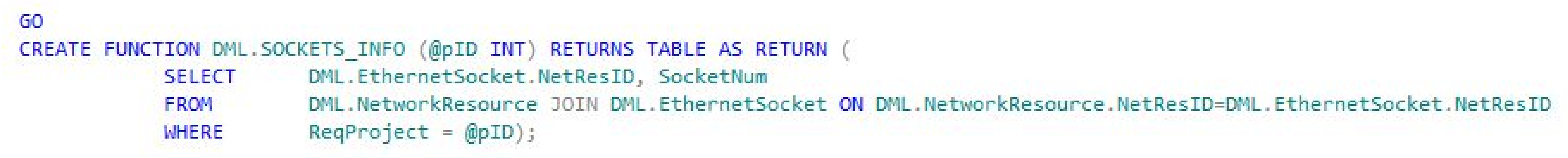
```
GO
CREATE VIEW DML.DELIVERED_RESOURCES AS
            SELECT      ResourceID, COUNT(ResourceID) AS Num
            FROM        (DML.ResourceDelivery JOIN DML.Delivery ON DML.ResourceDelivery.DeliveryID=DML.Delivery.DeliveryID)
            GROUP BY    ResourceID;
```



```
GO
CREATE VIEW DML.ALL_ELECTRONIC_UNITS AS
            SELECT      ResourceID, Supplier, DML.ElectronicResource.ProductName, DML.ElectronicResource.Manufacturer, DML.ElectronicResource.Model,
                        ResDescription, DML.ElectronicResource.PathToImage, DML.Staff.EmployeeNum, Email, FirstName, LastName, DML.Staff.PathToImage AS StaffImage
            FROM        DML.ElectronicUnit RIGHT JOIN (DML.ElectronicResource JOIN DML.Staff ON DML.ElectronicResource.EmployeeNum=DML.Staff.EmployeeNum)
                        ON DML.ElectronicUnit.ProductName=DML.ElectronicResource.ProductName AND
                           DML.ElectronicResource.Model=DML.ElectronicUnit.Model AND
                           DML.ElectronicResource.Manufacturer=DML.ElectronicUnit.Manufacturer;
```

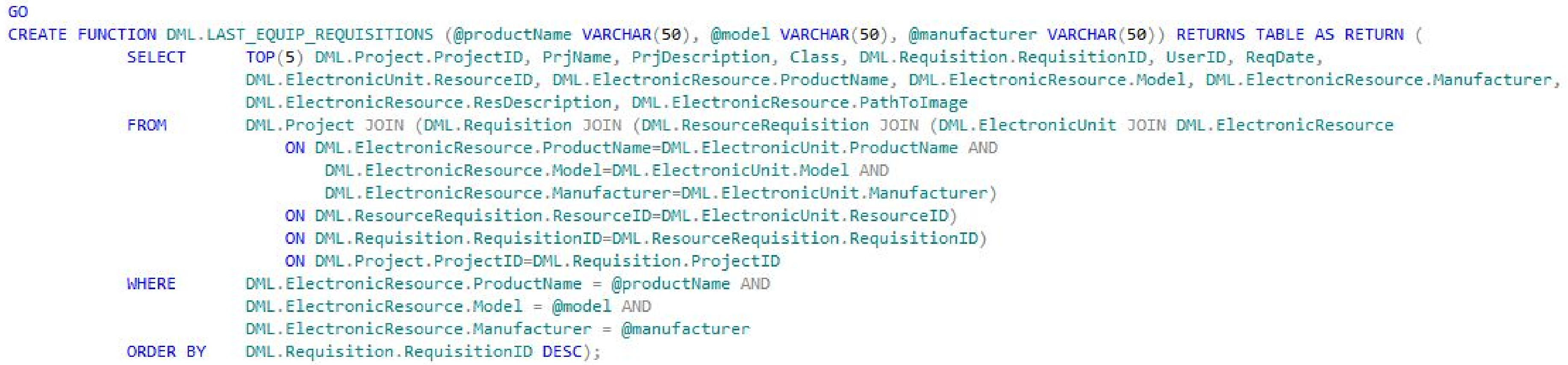
13 User Defined Functions

From simple ones



```
GO
CREATE FUNCTION DML.SOCKETS_INFO (@pID INT) RETURNS TABLE AS RETURN (
            SELECT      DML.EthernetSocket.NetResID, SocketNum
            FROM        DML.NetworkResource JOIN DML.EthernetSocket ON DML.NetworkResource.NetResID=DML.EthernetSocket.NetResID
            WHERE       ReqProject = @pID);
```

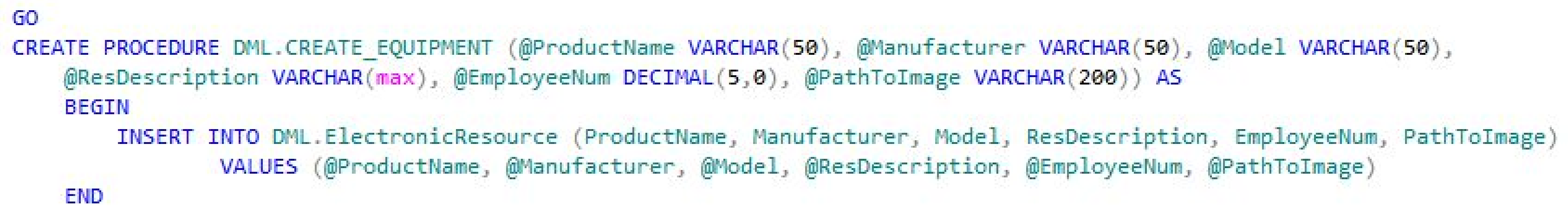
To the biggest and more complex



```
GO
CREATE FUNCTION DML.LAST_EQUIP_REQUISITIONS (@productName VARCHAR(50), @model VARCHAR(50), @manufacturer VARCHAR(50)) RETURNS TABLE AS RETURN (
            SELECT      TOP(5) DML.Project.ProjectID, PrjName, PrjDescription, Class, DML.Requisition.RequisitionID, UserID, ReqDate,
                        DML.ElectronicUnit.ResourceID, DML.ElectronicResource.ProductName, DML.ElectronicResource.Model, DML.ElectronicResource.Manufacturer,
                        DML.ElectronicResource.ResDescription, DML.ElectronicResource.PathToImage
            FROM        DML.Project JOIN (DML.Requisition JOIN (DML.ResourceRequisition JOIN (DML.ElectronicUnit JOIN DML.ElectronicResource
                            ON DML.ElectronicResource.ProductName=DML.ElectronicUnit.ProductName AND
                                DML.ElectronicResource.Model=DML.ElectronicUnit.Model AND
                                DML.ElectronicResource.Manufacturer=DML.ElectronicUnit.Manufacturer)
                            ON DML.ResourceRequisition.ResourceID=DML.ElectronicUnit.ResourceID)
                            ON DML.Requisition.RequisitionID=DML.ResourceRequisition.RequisitionID)
                            ON DML.Project.ProjectID=DML.Requisition.ProjectID
            WHERE       DML.ElectronicResource.ProductName = @productName AND
                        DML.ElectronicResource.Model = @model AND
                        DML.ElectronicResource.Manufacturer = @manufacturer
            ORDER BY    DML.Requisition.RequisitionID DESC);
```

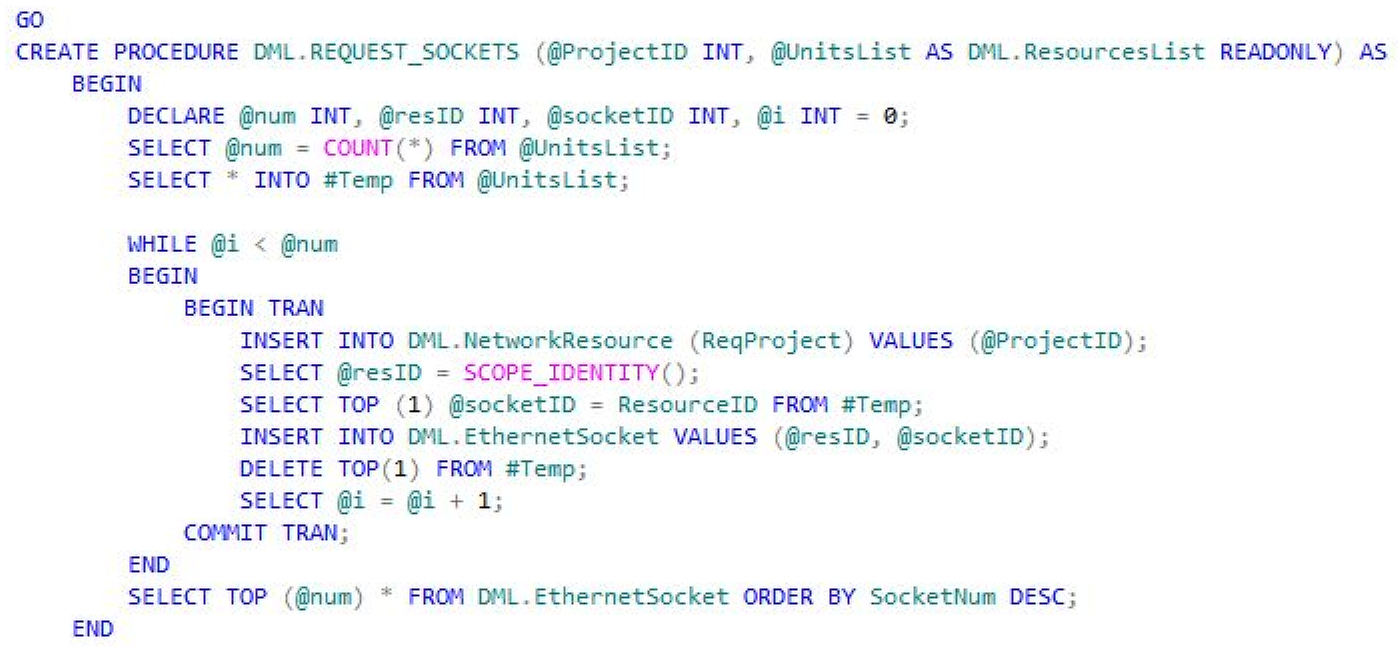
24 Procedures

From the simple ones



```
GO
CREATE PROCEDURE DML.CREATE_EQUIPMENT (@ProductName VARCHAR(50), @Manufacturer VARCHAR(50), @Model VARCHAR(50),
    @ResDescription VARCHAR(max), @EmployeeNum DECIMAL(5,0), @PathToImage VARCHAR(200)) AS
    BEGIN
        INSERT INTO DML.ElectronicResource (ProductName, Manufacturer, Model, ResDescription, EmployeeNum, PathToImage)
                VALUES (@ProductName, @Manufacturer, @Model, @ResDescription, @EmployeeNum, @PathToImage)
    END
```

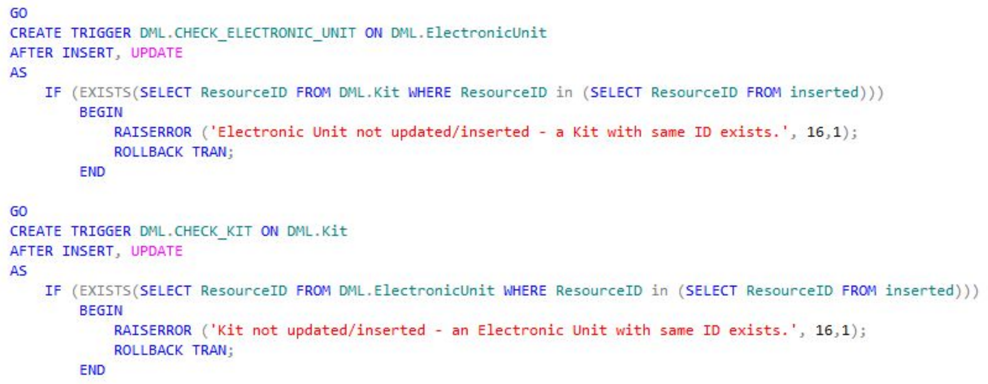
To the biggest and more complex



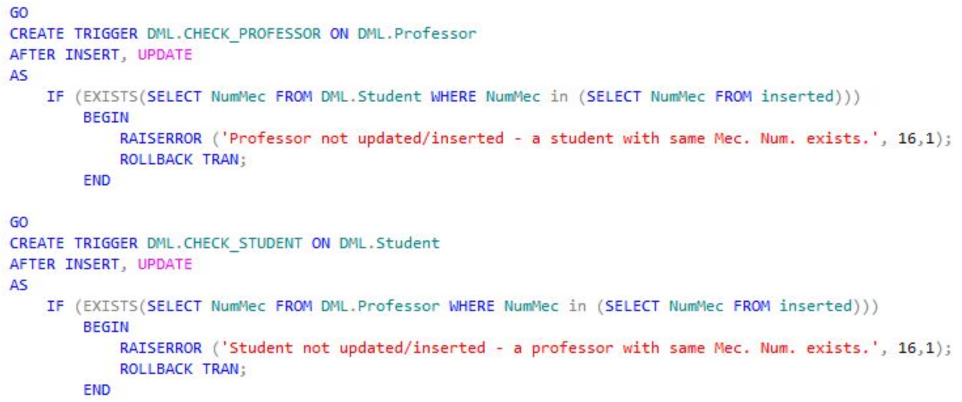
```
GO
CREATE PROCEDURE DML.REQUEST_SOCKETS (@ProjectID INT, @UnitsList AS DML.ResourcesList READONLY) AS
    BEGIN
        DECLARE @num INT, @resID INT, @socketID INT, @i INT = 0;
        SELECT @num = COUNT(*) FROM @UnitsList;
        SELECT * INTO #Temp FROM @UnitsList;

        WHILE @i < @num
        BEGIN
            BEGIN TRAN
                INSERT INTO DML.NetworkResource (ReqProject) VALUES (@ProjectID);
                SELECT @resID = SCOPE_IDENTITY();
                SELECT TOP (1) @socketID = ResourceID FROM #Temp;
                INSERT INTO DML.EthernetSocket VALUES (@resID, @socketID);
                DELETE TOP(1) FROM #Temp;
                SELECT @i = @i + 1;
            COMMIT TRAN;
        END
        SELECT TOP (@num) * FROM DML.EthernetSocket ORDER BY SocketNum DESC;
    END
```

4 Triggers
2 index

To check correct inserts and updates

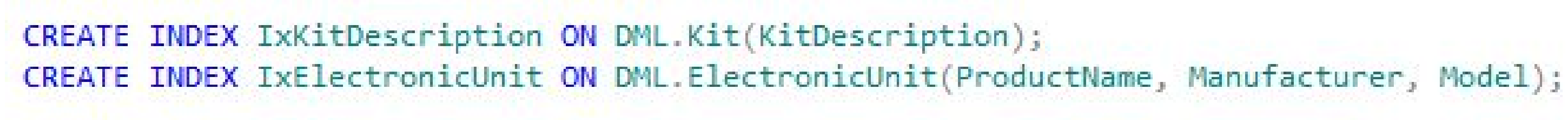


```
GO
CREATE TRIGGER DML.CHECK_ELECTRONIC_UNIT ON DML.ElectronicUnit
AFTER INSERT, UPDATE
AS
    IF (EXISTS(SELECT ResourceID FROM DML.Kit WHERE ResourceID in (SELECT ResourceID FROM inserted)))
        BEGIN
            RAISERROR ('Electronic Unit not updated/inserted - a Kit with same ID exists.', 16,1);
            ROLLBACK TRAN;
        END

GO
CREATE TRIGGER DML.CHECK_KIT ON DML.Kit
AFTER INSERT, UPDATE
AS
    IF (EXISTS(SELECT ResourceID FROM DML.ElectronicUnit WHERE ResourceID in (SELECT ResourceID FROM inserted)))
        BEGIN
            RAISERROR ('Kit not updated/inserted - an Electronic Unit with same ID exists.', 16,1);
            ROLLBACK TRAN;
        END
```



```
GO
CREATE TRIGGER DML.CHECK_PROFESSOR ON DML.Professor
AFTER INSERT, UPDATE
AS
    IF (EXISTS(SELECT NumMec FROM DML.Student WHERE NumMec in (SELECT NumMec FROM inserted)))
        BEGIN
            RAISERROR ('Professor not updated/inserted - a student with same Mec. Num. exists.', 16,1);
            ROLLBACK TRAN;
        END

GO
CREATE TRIGGER DML.CHECK_STUDENT ON DML.Student
AFTER INSERT, UPDATE
AS
    IF (EXISTS(SELECT NumMec FROM DML.Professor WHERE NumMec in (SELECT NumMec FROM inserted)))
        BEGIN
            RAISERROR ('Student not updated/inserted - a professor with same Mec. Num. exists.', 16,1);
            ROLLBACK TRAN;
        END
```

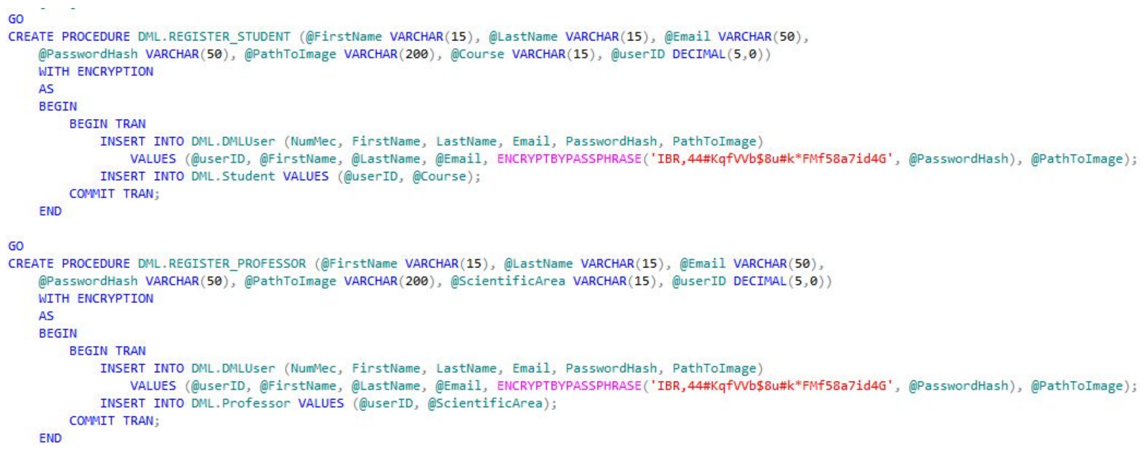
To improve database speed



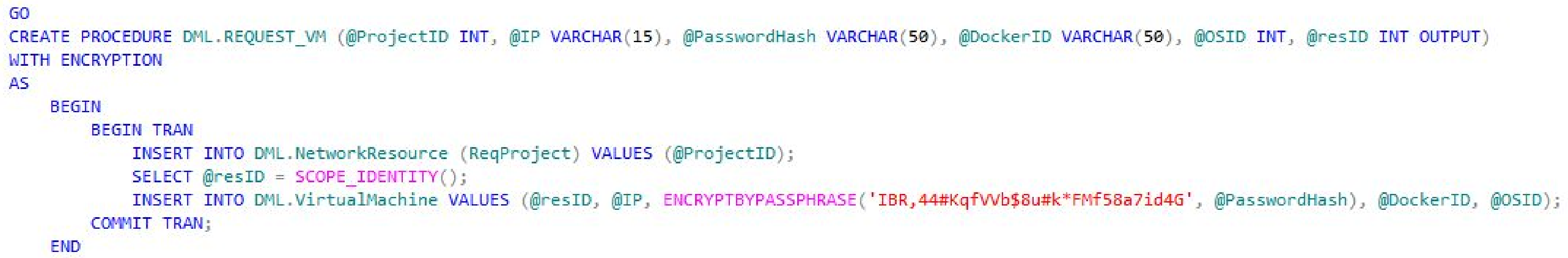
```sql
CREATE INDEX IxKitDescription ON DML.Kit(KitDescription);
CREATE INDEX IxElectronicUnit ON DML.ElectronicUnit(ProductName, Manufacturer, Model);
```

Security & Robustness

Password hashing, encryption and transactions



```sql
GO
CREATE PROCEDURE DML.REGISTER_STUDENT (@FirstName VARCHAR(15), @LastName VARCHAR(15), @Email VARCHAR(50),
    @PasswordHash VARCHAR(50), @PathToImage VARCHAR(200), @Course VARCHAR(15), @userID DECIMAL(5,0))
    WITH ENCRYPTION
    AS
    BEGIN
        BEGIN TRAN
            INSERT INTO DML.DMLUser (NumMec, FirstName, LastName, Email, PasswordHash, PathToImage)
                VALUES (@userID, @FirstName, @LastName, @Email, ENCRYPTBYPASSPHRASE('IBR,44#KqfVVb$8u#k*FMf58a7id4G', @PasswordHash), @PathToImage);
            INSERT INTO DML.Student VALUES (@userID, @Course);
        COMMIT TRAN;
    END

GO
CREATE PROCEDURE DML.REGISTER_PROFESSOR (@FirstName VARCHAR(15), @LastName VARCHAR(15), @Email VARCHAR(50),
    @PasswordHash VARCHAR(50), @PathToImage VARCHAR(200), @ScientificArea VARCHAR(15), @userID DECIMAL(5,0))
    WITH ENCRYPTION
    AS
    BEGIN
        BEGIN TRAN
            INSERT INTO DML.DMLUser (NumMec, FirstName, LastName, Email, PasswordHash, PathToImage)
                VALUES (@userID, @FirstName, @LastName, @Email, ENCRYPTBYPASSPHRASE('IBR,44#KqfVVb$8u#k*FMf58a7id4G', @PasswordHash), @PathToImage);
            INSERT INTO DML.Professor VALUES (@userID, @ScientificArea);
        COMMIT TRAN;
    END
```



```sql
GO
CREATE PROCEDURE DML.REQUEST_VM (@ProjectID INT, @IP VARCHAR(15), @PasswordHash VARCHAR(50), @DockerID VARCHAR(50), @OSID INT, @resID INT OUTPUT)
WITH ENCRYPTION
AS
    BEGIN
        BEGIN TRAN
            INSERT INTO DML.NetworkResource (ReqProject) VALUES (@ProjectID);
            SELECT @resID = SCOPE_IDENTITY();
            INSERT INTO DML.VirtualMachine VALUES (@resID, @IP, ENCRYPTBYPASSPHRASE('IBR,44#KqfVVb$8u#k*FMf58a7id4G', @PasswordHash), @DockerID, @OSID);
        COMMIT TRAN;
    END
```

DETI-MakerLab Application

Characteristics

- Created with **WPF and C#**
- **Intuitive** and **simple**
- Access to information related to the **projects**, **resources**, **requisitions** and **user profiles**
- **Request or deliver** any resource (electronic or network) at any time
- **FAQ** section and **Tooltips**
- Robust to **SQL-Injection**

Questions?