# Systems Design and Databases (CIS1018-N)

## Week 4

Entity Relationship Diagrams (ERD)

A Data Modeling Case Tool for Relational Databases

Introduction to SQL Server

# Teaching Team

**Module Leader & Lecturer:** Dr Yar Muhammad
**Email:** Yar.Muhammad@tees.ac.uk
**Office:** G0.39 (Greig Building)

**Tutor:**

- Dr Mengda He
- Mr Mansha Nawaz
- Mr Vishalkumar Thakor

**My Academic Hub Time Slots:**
Monday 10:00 - 11:00 in Room IT1.13 (Europa Building)
Tuesday 13:00 - 14:00 in Room IT1.13 (Europa Building)

- See Blackboard Ultra for online materials: https://bb.tees.ac.uk/

# Lectures & IT Labs

| Lectures – Dr Yar Muhammad | Tuesdays @ 2-3 pm | Thursdays @ 1-2 pm |
|---|---|---|
| Week 1 – Week 12 | CL1.87 | |

| Tutor – Thursday | IT Lab Session Room #: IT2.42 |
|---|---|
| Mr Mansha Nawaz<br>M.Nawaz@tees.ac.uk | Time: 3 – 5 pm |

| Tutor – Friday | IT Lab Session Room #: OL3 |
|---|---|
| Dr Yar Muhammad<br>Yar.Muhammad@tees.ac.uk | Time: 9 – 11 am &<br>11 am – 1 pm |
| Dr Mengda He<br>M.He@tees.ac.uk | Time: 9 – 11 am |
| Mr Vishalkumar Thakor<br>V.Thakor@tees.ac.uk | Time: 11 am – 1 pm &<br>1 – 3 pm |
| Mr Mansha Nawaz<br>M.Nawaz@tees.ac.uk | Time: 1 – 3 pm |
| | |

# Schedule 1/3

**Systems Design and Databases CIS1018-N**
**Weekly Plan for the Activities**

| Week | Lecturer | Lecture Demo | Lab Exercises & Solutions | ICA Tasks: |
|---|---|---|---|---|
| 01 | • Module Introduction,<br>• System Design,<br>• Introduction Databases (DDL, DML, DCL, TCL) | • Requirement List &<br>• MoSCoW Wireframe Design & Templates,<br>• User Stories | • Team Setup,<br>• Hands-on to collect/pick the Requirements from MoSCoW and write Writing User stories on each<br>• **Tutorial 1** | Requirements List & MosCOW, User stories |
| 02 | • UML and UML Tool, | • Use Case Diagrams from Requirements List and Wireframe | • Hands-on Use Case Diagrams Activities<br>• **Tutorial 2** | Each Wireframe has associated Use Case Activity<br><br>Deadline for Team Setup is Week # 2, by Friday 07/10/2022 before 4pm |
| 03 | • Sequence Diagrams | • Class Diagrams | • Hands-on Sequence & Class Diagrams Activities<br>• **Tutorial 3** | Each Wireframe has associated Sequence and Class Diagrams |
| 04 | • Entity Relationship Diagrams (ERD) A Data Modelling Case Tool for Relational Databases | • Introduction to SQL Server<br>• Walk-through: SQL Quick Guide 1 - How to use SSMS to build Databases | • **Tutorial 4**<br>• Lab Resources: SQL Quick Guide 1 | Each Wireframe has associated Class Diagram |

Systems Design - UML

Analysis

Design

# Schedule 2/3

Databases - TSQL

| Week | Lecturer | Lecture Demo | Lab Exercises & Solutions | ICA Tasks: |
|---|---|---|---|---|
| **05** | • Querying with Select | **Demo A** – Writing Simple SELECT Statements<br>**Demo B/C** – Eliminating Duplicates with DISTINCT<br>**Demo D** - Writing Simple CASE | • TSQL-Mod03 Lab-Exercise 1-4<br>• **Tutorial 5** | SQL Task A: TSQL03 Querying with Select<br>• Writing Simple SELECT Statements<br>• Eliminating Duplicates with DISTINCT<br>• Using Column and Table Aliases<br>• Writing Simple CASE Expressions |
| **06** | • Querying with Multiple Tables | **Demo B** – Relating 2 or more tables – Joins & Joining multiple tables – inner, outer and cross. | • TSQL-Mod04 Exercise 1-5<br>• **Tutorial 6** | SQL Task B: TSQL04 – Querying with Multiple Tables<br>• Relating 2 or more tables – Joins<br>• Joining multiple tables – inner, outer and cross. |
| **07** | • Sorting and Filtering Data | **Demo A** – Sort with ORDER BY<br>**Demo B** – Filter with WHERE Clause<br>**Demo C** – Filtering with Top OffsetFetch<br>**Demo D** – Handling NULL | • TSQL-Mod05 Exercise 1 – 4<br>• **Tutorial 7** | SQL Task C: TSQL05 – Sort and Filtering Data<br>• Sort with Order By<br>• Filter with Where By<br>• Filter with top offsetfetch<br>• Handling Nulls |
| **Submission ICA 1 (Group Submission) -> Deadline is Wednesday 16/11/2022 before 4pm** | | | | |
| **08** | • Working with SQL Server Data | **Demo A** - Conversion in a Query<br>**Demo B** - collation in a query<br>**Demo C** - date and time functions | • TSQL-Mod06 Exercise 1 – 4<br>• **Tutorial 8** | SQL Task D: TSQL06 – Working with SQL Server Data<br>• Conversion in a Query<br>• collation in a query<br>• date and time functions |

# Schedule 3/3

Databases - TSQL

| | | | | |
|---|---|---|---|---|
| **09** | • Using DML to modify Data | **Demo A** - Adding Data to Tables<br>**Demo B** - Modifying and Removing Data<br>**Demo C** - Generating Automatic Column Values | • TSQL-Mod07 Exercise 1 – 2<br>• **Tutorial 9** | SQL Task E: TSQL07– Using DML to Modify Data<br>• Adding Data to Tables<br>• Modifying and Removing Data<br>• Generating Automatic Column Values |
| **10** | • Using built in Functions | **Demo A** – Scalar Functions<br>**Demo B** – Cast Functions<br>Demo C – If Functions<br>**Demo D** – IsNull Functions | • TSQL-Mod08 Exercise 1 – 3<br>• **Tutorial 10** | SQL Task F: TSQL08– Using Built-In Functions<br>• Writing Queries with Built-In Functions<br>• Using Conversion Functions<br>• Using Logical Functions<br>• Using Functions to Work with NULL |
| **11** | • Walk through SQL Quick Guide 2 - Create a Tables and Relationships via SSMS GUI | • Walk through:<br>• SQL Quick Guide 3 - Create Query, View through Designer | Hands-on:<br>• SQL Server Quick Guide 2 | SQL Server – Introduction to SQL Server and SSMS |
| **12** | Support | Support | Hands-on:<br>• SQL Server Quick Guide 3 | SQL Server – Introduction to SQL Server and SSMS |
| **Submission ICA 2 (Individual Submission) -> Deadline is Wednesday 11/01/2023 before 4pm** | | | | |

# What are Entities?

Design **Entities** are **classes whose design specifications** have been completed to such a degree that they can be drafted for implemented in a Development Toolset**.**

- The **ERD** helps the developer to progress onto the process of actual implementing the tables in a databse

**Why are the entities (class objects) included in the UML?**

- The **Entities (class objects)** that are included in the UML are what represent the **building blocks of the same objects or things**.
  - Customer things? User things?
  - Order things?  Sales things?
  - Basket things?
  - Despatch things?

- This is the reason why the **ERD** are also referred to as the **building blocks** of a Database.
- When it comes to the varying elements in **the ERDs**, there are a few we needs to be highlight  and considered ….

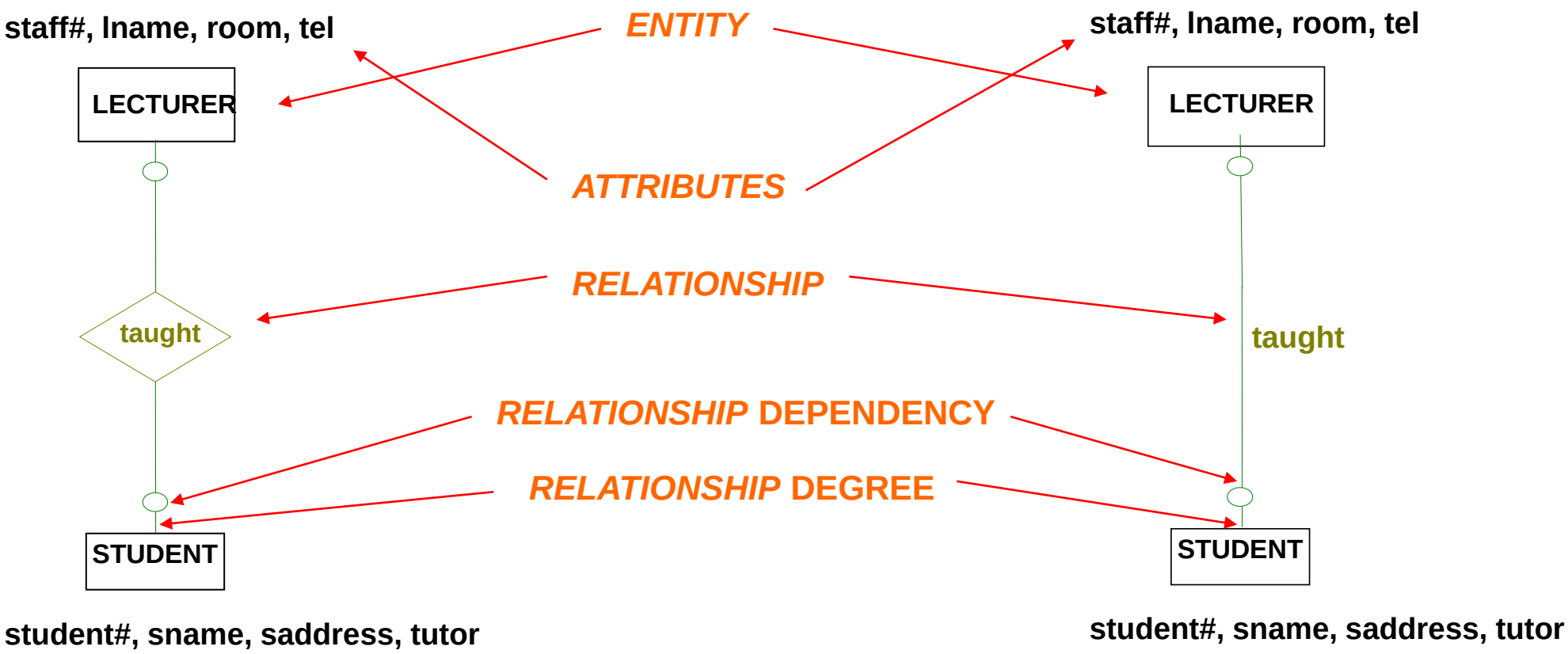
# Entity-Relationship Diagrams (ERD).

- System Data Model is perceived in terms of an Entity Relationship Diagram (ERD).
- The ERD can be translated into Conventional DBMS Table Structures
    - **Hierarchical (IMS)**
    - **Codasyl**
    - **Relational (MS SQL Server or MS Access)**
- ERD consists of
    - **ENTITIES**
    - **ATTRIBUTES**
    - **RELATIONSHIPS (Degree & Dependency)**

- Start by identifying the entities and their attributes
- Identify associations/relationships between entities
- Draw up an ER Diagram (ERD) using appropriate notation
- Applying *rules*, translate ERD into a set of well-normalised tables

# ERD Notations

## ER Diagrams (ERD).
***Howe Notation***




## ER Diagrams (ERD).
***Crows Feet Notation***

***The creative bit is the identification of appropriate Entities, attributes and their relationships***
***The rest is learning how to adhere to and apply appropriate rules of ER Modelling correctly***

# Identifying ENTITIES Entity is an object being modelled

**ENTITIES**

- If the scenario you are working with is in the form of narrative it is likely that most of the entities of interest will appear in the narrative as nouns (words used as labels for things) e.g. Customer, Delivery Note, Bill of Materials, Prescription, Drug, Payment etc…
- In a real systems analysis and design the entities will be derived from the DFD
- They can be also appearing as paperwork, physical items, software components, and similar.
- Sometimes they are a little more abstract e.g. assignment, delivery, load, journey…..

**NOUN**

- Table type of related data.
- One Row Per Entity Occurrence.
- independent existence
- uniquely identifiable by their *properties*
- something you need to keep a list of!
- Nouns / thing / object
- eg employee, order, machine, property
- Look for object name or thing.
- eg INVOICE, ORDER, STUDENT, LECTURER
- Look for things requiring a list of associated records or tabular list of data.
- Each existence of an entity value requires a record.
- Look for things with record keys.
- eg stock code -> STOCK reg. no. -> CAR

# ENTITIES

**Example A**


**Example B**


**Example C**


We can identify entitles as they tend to have a **unique key** to identify it.  Also known as a **Primary Key.**

# ATTRIBUTES are the related data items for an entity

**ATTRIBUTE**

**Entities have sets of attributes.**

- **If you cannot find more than one attribute for the entity you have chosen, or if the attributes seem very basic (or trivial) it should cross your mind that this entity may be an attribute of something rather than an entity.**
- **This does not mean that entities with only one attribute must be incorrect- but they must be judged carefully.**

Choosing Attributes

**DATA LABEL**

- Attribute
  - property of an entity
- Also refereed to as data or data items.
- Look for data associated to an entity type.
  - INVOICE has invoice#, invoice date, invoice total, etc.
  - ORDER has order#, order date, order total, etc.
  - STUDENT has student#, student name, etc.
  - LECTURER has staff#, lecturer name, room, tel, etc.

- eg
  - **an EMPLOYEE attributes might be:**
  - **national insurance number, salary, date of birth**

  - **an ORDER attributes might be:**
  - **date of order, order number, customer ID**

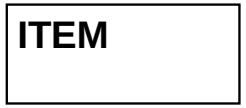
  - **a MACHINE attributes might be:**
  - **model number, capacity, description**

# ATTRIBUTES

**Example A**

**invoice#, invoice date, invoice total**


**item#, item name, item desc**




**Example B**

**order#, order date, order total**


**part#, part name, part desc**

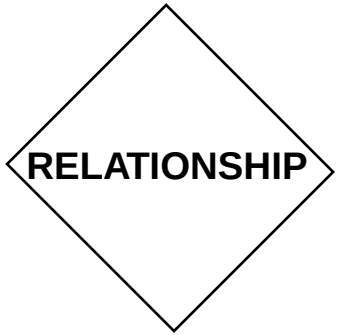

**Example C**

**staff#, lecturer name, room, tel, etc.**

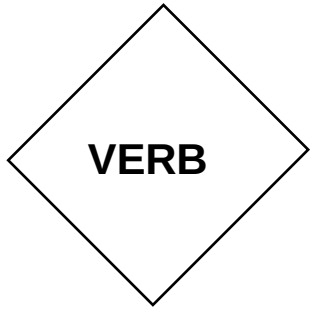

**student#, student name, etc.**

# Identifying Relationships

Relationship is an association between two or more entities.




- **If the scenario you are working with is in the form of narrative it is likely that most of the relationships of interest will appear in the narrative as verbs or verb phrases (words used to denote an activity or association) e.g. purchases, delivers, picks_up_at, starts_at etc..**
- **Relationships are associations between forms, documents or activities carried out with documents e.g. deliver_to, picked_up_at.**
- **However, it is highly unlikely that each document will correspond to a single entity. Each document will be a combination of many entities**
- **Entities, attributes and relationships can be much more abstract than this.**
- **However, the basic points regarding entities, attributes and relationships should be understood before tackling more abstract data objects.**




- Relationship used to share attributes (data) between entities.
- Identified by a Verb linking entities together.
  - INVOICE invoiced-item ITEM
  - ORDER ordered-part PART
  - LECTURER teaches STUDENT
  - LECTURER councils STUDENT
- Eg

employee and machine

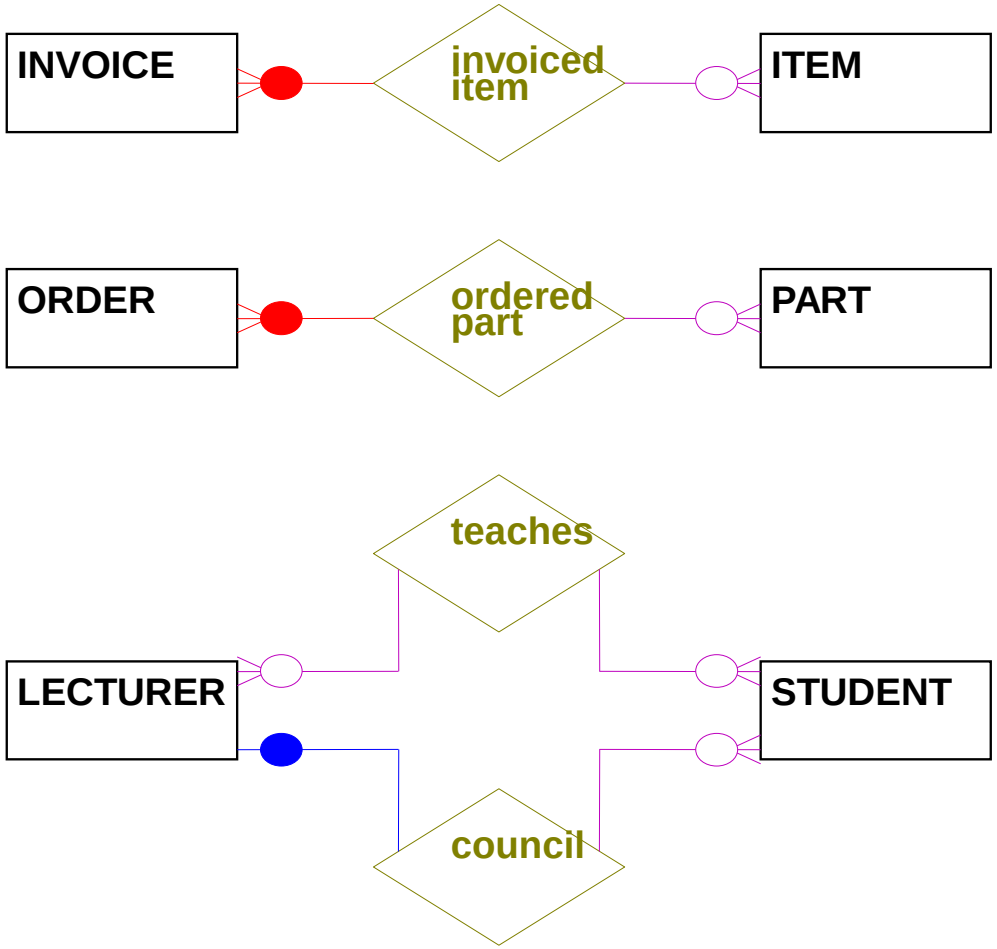
  - **relationship might be *works on***
  - **: *An employee works on exactly one machine***

order and stock

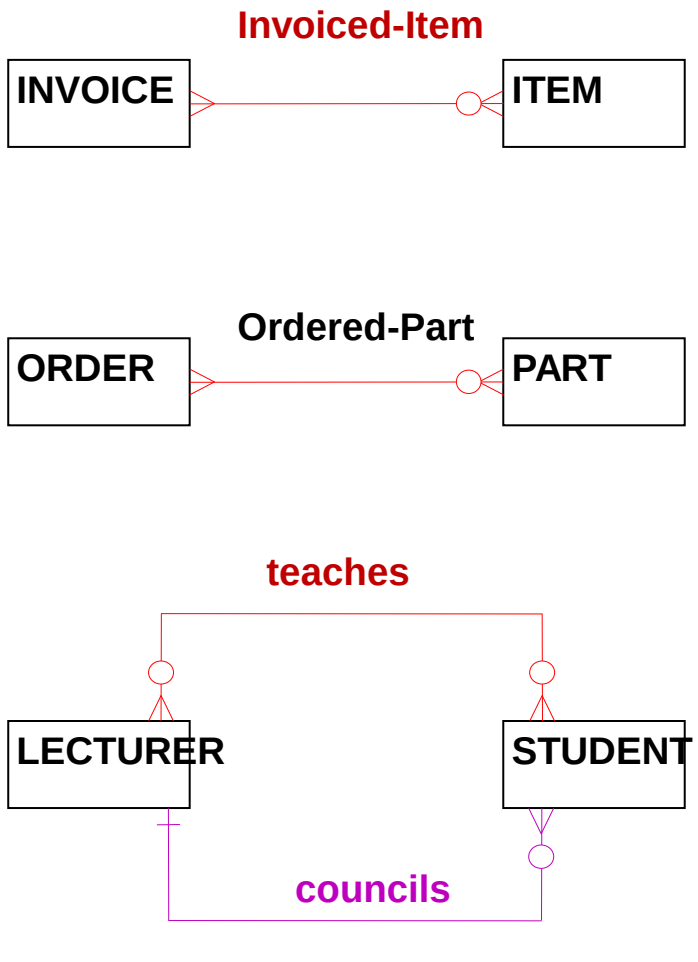
  - **relationship might be *relates to***
  - **: *An order relates to many stock items***

# A relationship degree & dependency determines the rules as to how entities are linked together

**RELATIONSHIP**
**Howe Notation**




**RELATIONSHIP**
**Crows Feet Notation**



# RELATIONSHIP DEGREE.

- These provide use the means to build the DATA MODEL
- Used to indicate how entities are linked
- The relationship allows use to share data via the link.
- Used by the DBMS to link across tables for multiple views.
- Three possible degrees
  - 1:1 one to one


  - 1:m one to many


  - m:n many to many

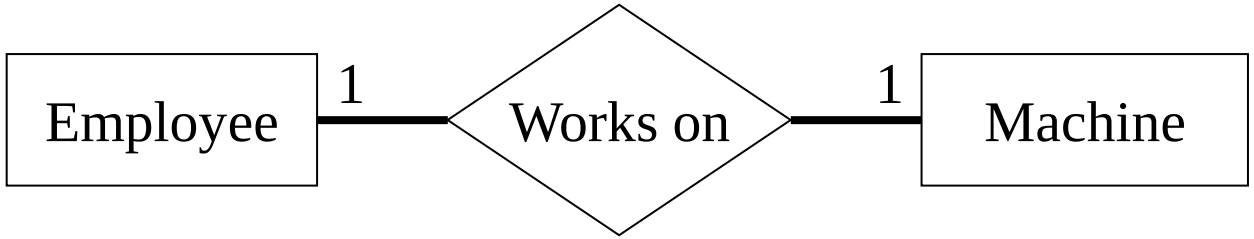

- Each represent a unique view to sharing data across entities.

# 1:1 Relationship (one to one)

**Enterprise Rules Represented:**

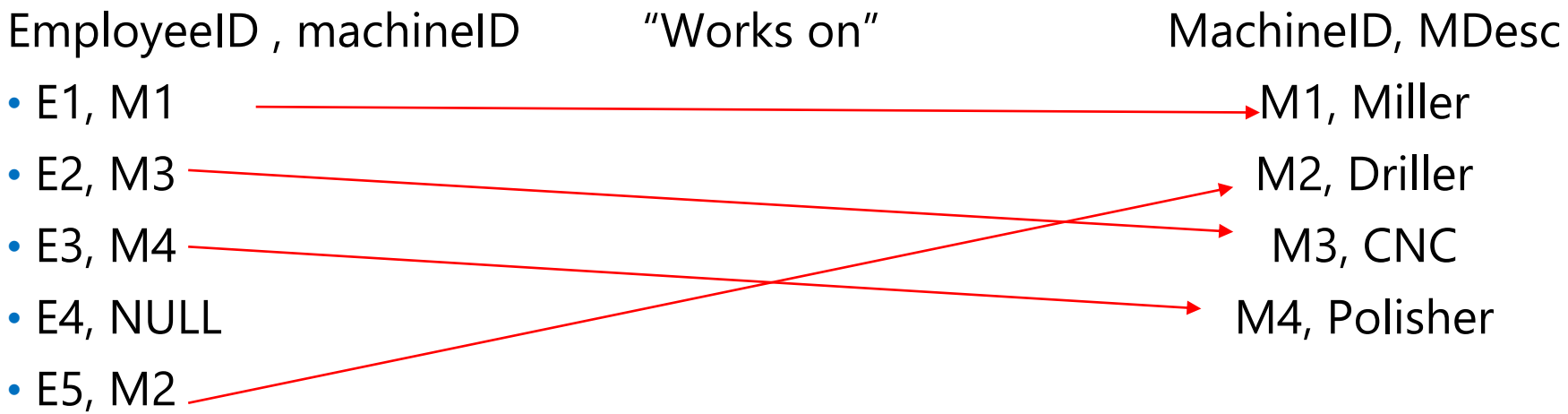
**"An employee *works on* at most one machine"**

**"A machine *is worked on* by at most one employee"**

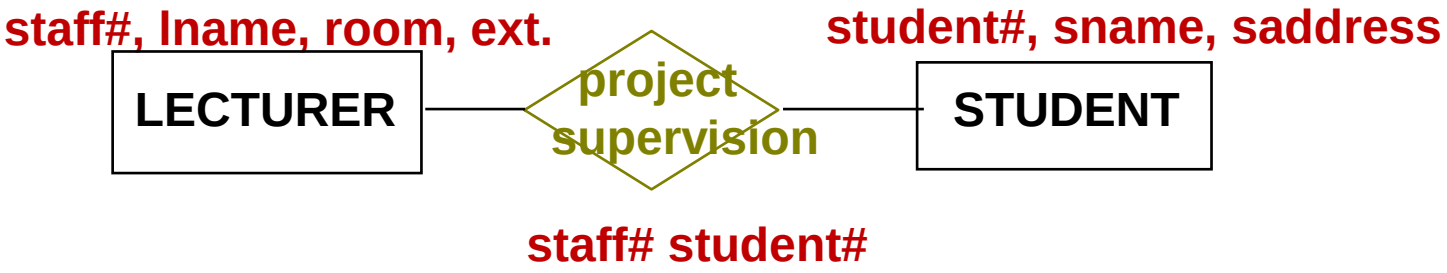





## Entity Relationship Occurrence Diagram 1:1



# 1:1 Relationship (one to one)




- ***Enterprise Rule :***

  **A LECTURER provides PROJ-SUP for a STUDENT**

  **A STUDENT has PROJ-SUP with a LECTURER**

- ***Skeleton Tables***

  **LECTURER(staff#, lname, room, ext, student#)**

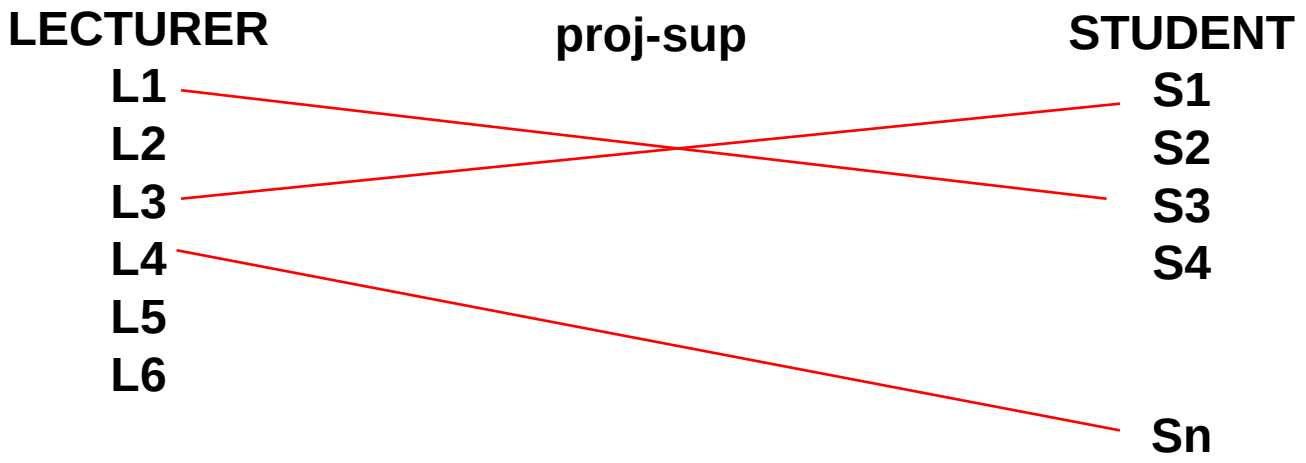
  **STUDENT(student#, sname, saddress)**

*** You may import the staff# as a foreign key in the STUDENT table.**

*** Only one import of foreign key required.**

*** Do NOT import keys into both adjoining tables**

# Occurrence Diagram




- ***Populated Tables*** *from sample occurrence diagram*

**LECTURER**

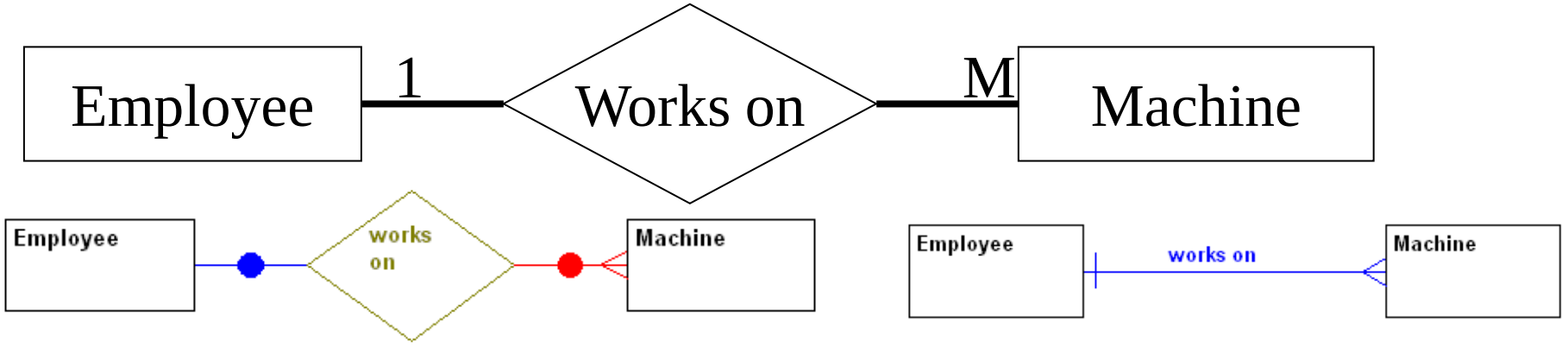
| staff# | lname | room | ext | student# |
|---|---|---|---|---|
| L1 | ALBERT | A4 | 12 | S3 |
| L2 | BLOGGS | B2 | 18 | |
| L3 | COATES | C1 | 22 | S1 |
| L4 | GREEN | A3 | 11 | |
| L5 | JONES | D1 | 53 | Sn |
| L6 | SMITH | A5 | 18 | |

**STUDENT**

| student# | sname | saddress |
|---|---|---|
| S1 | bloggs.. | |
| S2 | jones.. | |
| S3 | green.. | |
| S4 | smith.. | |
| Sn | zues.. | |

# 1:M Relationship (one to many) name relationships from M to 1

- **Enterprise Rules Represented:** **"An employee *may work on* one or many machines"**
  **"A machin\e *is worked on* by at most one employee"**



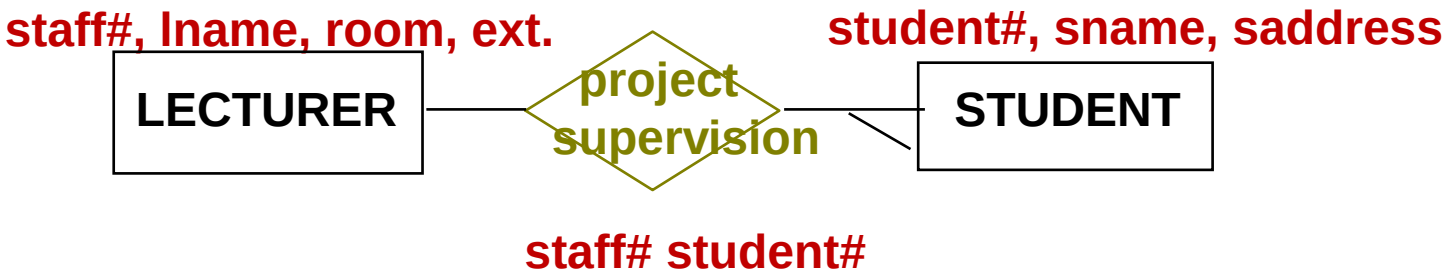

## Entity Relationship Occurrence Diagram 1:M

Employee "Works on" Machine

- E1 → M1, M2
- E2 → M2
- E3 → M3
- E4 → M4
- E5

- **What would a M:1 look like?**

# 1:M Relationship (one to many)




- ***Enterprise Rule :***
  - **'A LECTURER COUNSELS MANY STUDENT**
  - **'MANY STUDENTS COUNCELLED by A LECTURER**

- ***Skeleton Tables :***

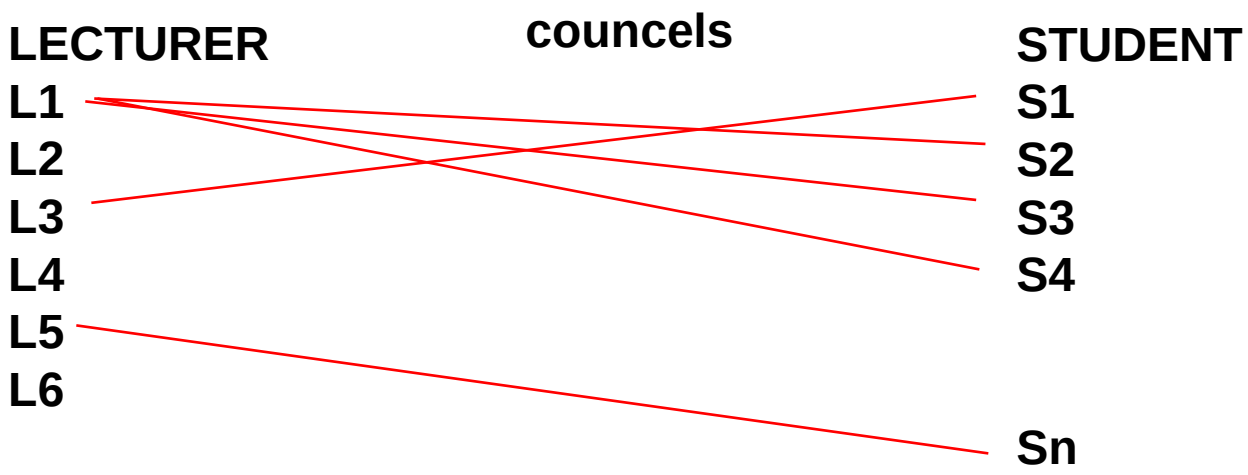
  **LECTURER(staff#, lname, room, ext)**

  **STUDENT(student#, sname, saddress, councilorstaff#)**

*** Foreign key in always imported from the *one* entity the *many* entity.**

*** student.councelorstaff# would link to lecturer.staff#**

# Occurrence Diagram




- ***Populated Tables** from sample occurrence diagram*

**LECTURER**

| staff# | lname | room | ext |
|---|---|---|---|
| L1 | ALBERT | A4 | 12 |
| L2 | BLOGGS | B2 | 18 |
| L3 | COATES | C1 | 22 |
| L4 | GREEN | A3 | 11 |
| L5 | JONES | D1 | 53 |
| L6 | SMITH | A5 | 18 |

**STUDENT**

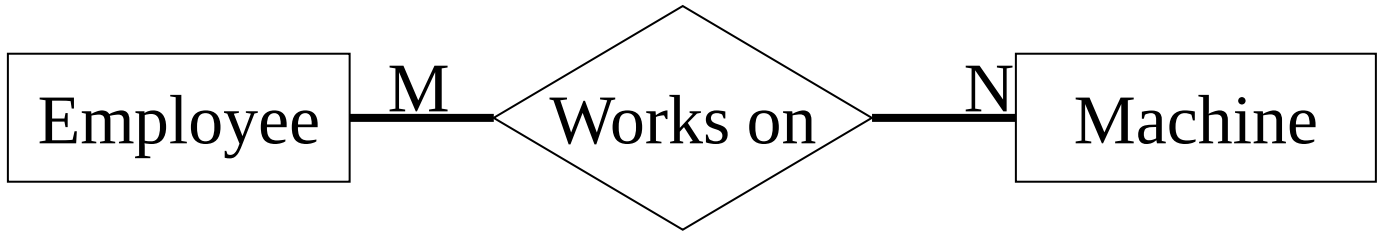
| student# | sname | staff# |
|---|---|---|
| S1 | bloggs | L3 |
| S2 | jones | L1 |
| S3 | green | L1 |
| S4 | smith | L1 |
| Sn | zues | L5 |

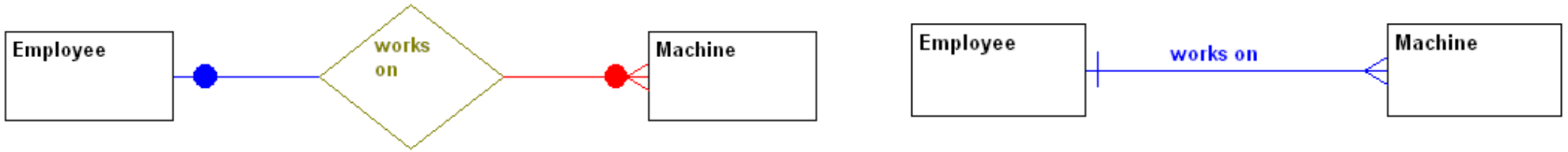
# M:N Relationship (many to many) M:M

**Enterprise Rules Represented:**

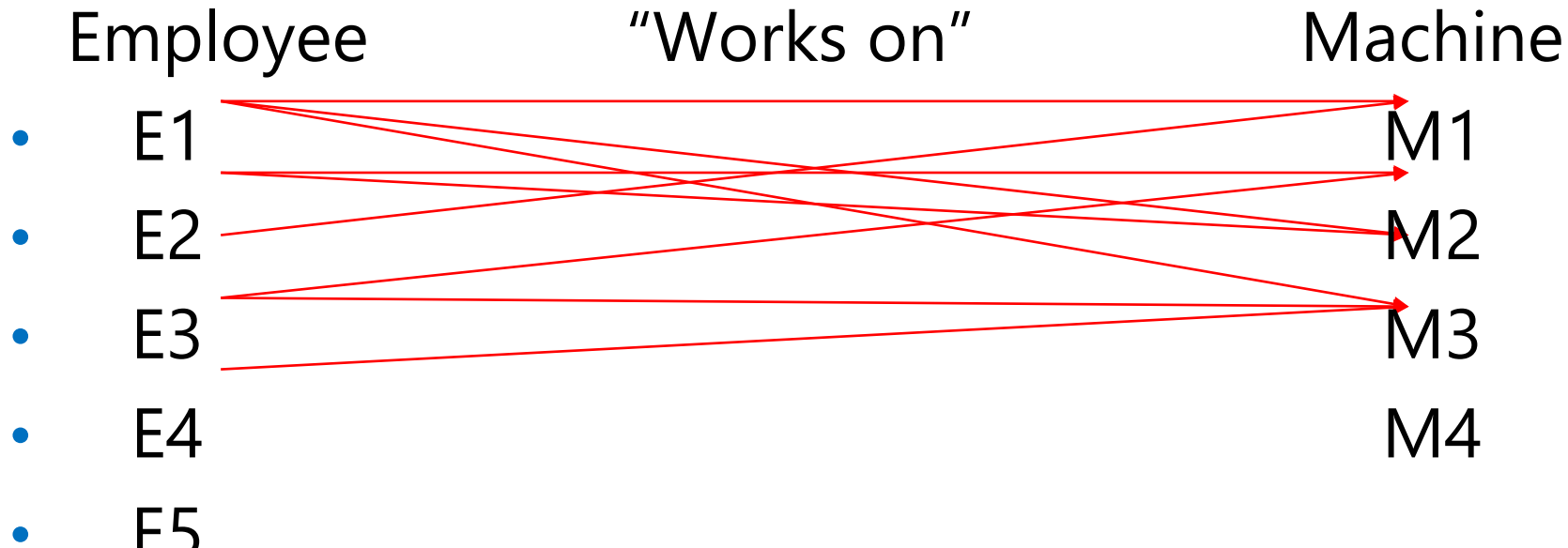
**"An employee *may work on* one or many machines"**

**"A machine *is worked on* by one or many employees"**

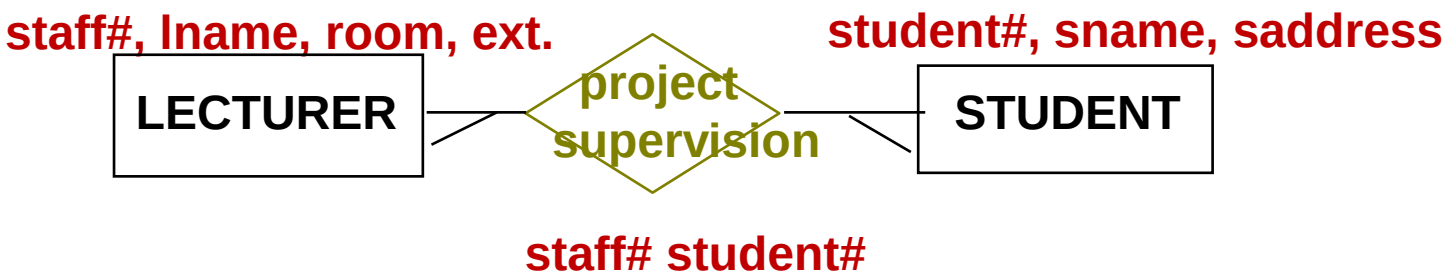



## Entity Relationship Occurrence Diagram M:M

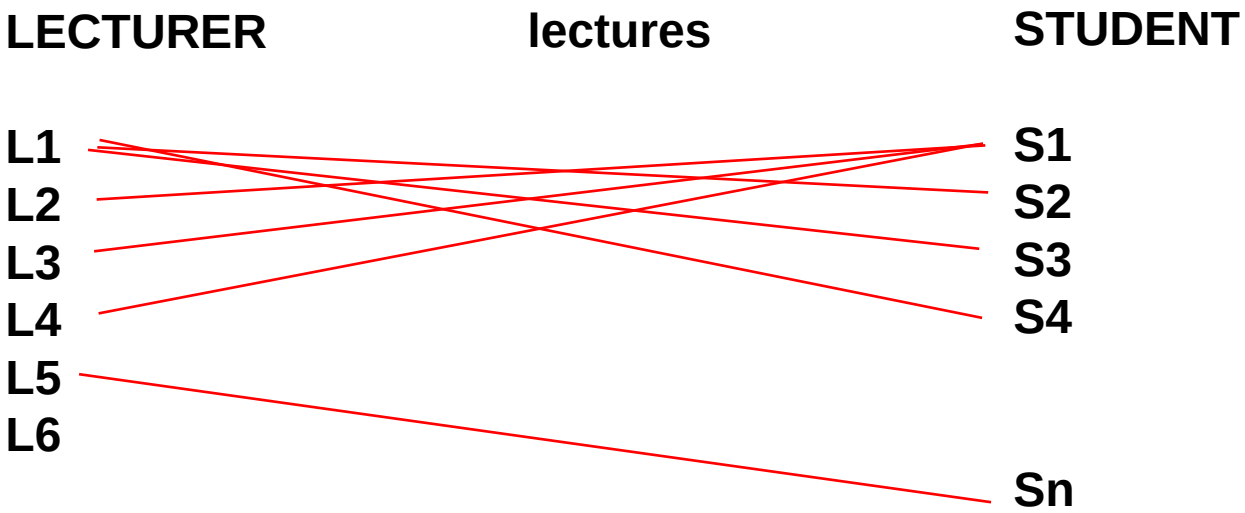


# m:n MANY-to-MANY Relationship




– ***Enterprise Rule :***
  - **‘MANY LECTURERS LECTURES MANY STUDENTS**
  - **‘MANY STUDENTS LECTURED by MANY LECTURER**

– ***Occurrence Diagram :***



# eLearning Platform – student - modules

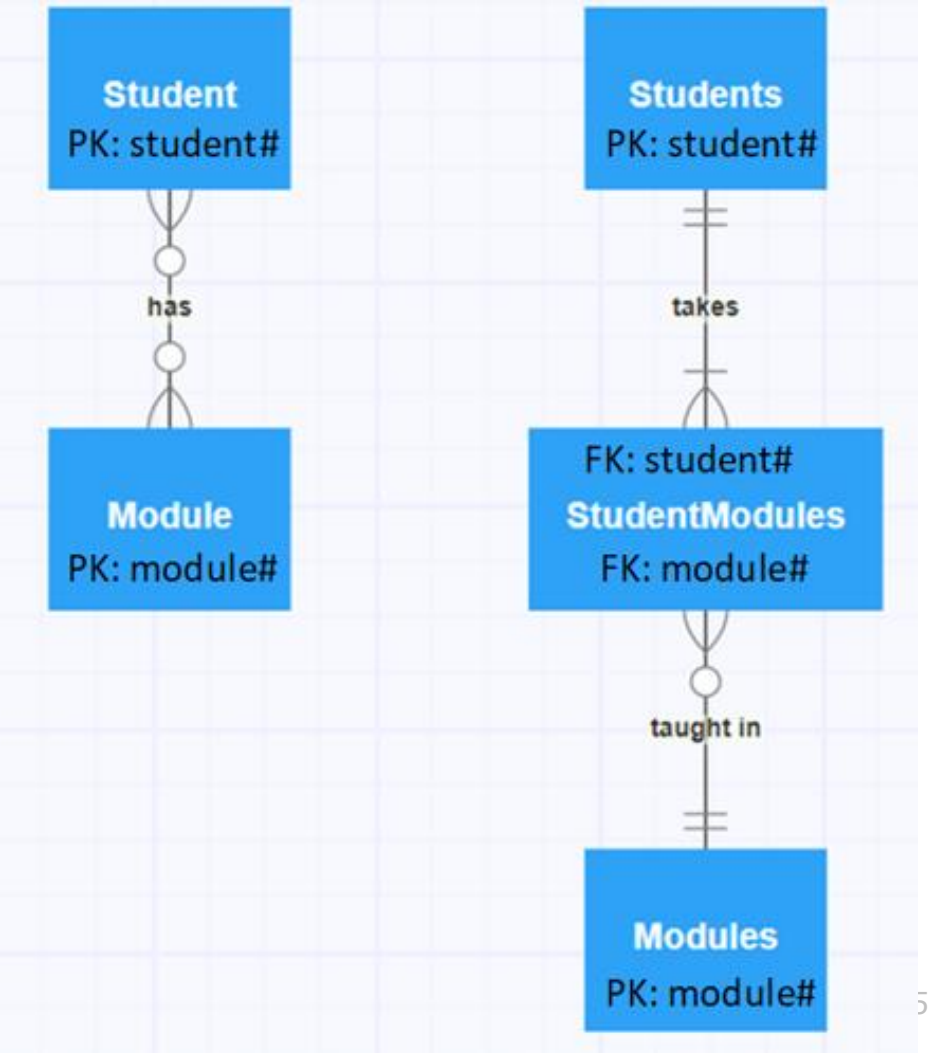
**Student {0..n}-has-{0..n} Module**
**Students {1}-takes-{1..n} StudentModules {0..n}-taught in-{1} Modules**

**many to many: always decompose to**
**1:m and m:1**

You merge the 2 entity (class) names and create a new entity type!

**This is what the process of Normalisation and Entity Relationship modelling is all about.**

**A primary key PK from the one table transposes into the many table as a foreign key FK**



# Normalization

- A step by step process to produce more efficient and accurate database design
- Purpose is to produce an anomaly free design

# Anomalies

- An inconsistent, incomplete or incorrect state of database
- Four types of anomalies are of concern here; Redundancy, insertion, deletion and updation

# Normalization

- A strongly recommended step
- Normalized design makes the maintenance of database easier
- Normalization applied on each table of a database design

# Normalization

- Performed after the logical database design
- Informally also performed during conceptual database design

# Normalization Process

- Different forms or levels of normalization
- Called first, second, third and so on forms
- Each form has got certain conditions
- If a table fulfils the condition(s) for a normal form, then the table is in that normal form

# Normalized Database Design

- Process is applied on each table
- The minimum form in which all tables are called the normal form of the entire database
- Objective is to place the database in highest form of normalization

# Normal Forms

- Why Normalization ?
  - To produced smarter, more efficient, more accurate tables
- Normalization has different forms/levels, each form/level has different requirements.
- We will check data with each normalize form.

# Normalization Forms (cont..)

- Normalization is basically; a process of efficiently organizing data in a database.
- There are two goals of the normalization process
  - Eliminate redundant data (for example sorting the data more than one table)
  - Ensure data dependencies make sense (only storing related data in a table)
- Both are worthy goals as they reduce the amount of space a data consumes and ensure that data is logically stored

# Normalization Forms (cont..)

- Normalization is the process of structuring relational database schema such that most ambiguity is removed.
- Most database designers do not attempt to implement anything higher than **Third Normal Form or Boyce-Codd Normal Form (BCNF)**

# First Normal Form (1NF)

**A relation is in first normal form. if only if, every attribute in every tuple contains an atomic value**

- Every attribute is **single valued for each tuple**
  - This mean that each attribute in each row, or each cell of the table, contains only one value.
- No repeating fields or groups are allowed.
- Domains of attributes of a relation are atomic, that is they consist of single units that cannot be broken down further
- There is no multivalued (repeating group) in the relation multiple values create problem in performing operations like select or join

# First Normal From Example

| stdID | stdName | stdAddress | progName | bkID |
|---|---|---|---|---|
| S1020 | Sohail | Clifton | BS-CS | B00129 |
| S1038 | Shahid | Phase 1 | BS-IT | B00327 |
| S1015 | Fawad | Phase 5 | BS-SE | B08945, B06352 |
| S1020 | Jawad | Clifton | M.Sc | B08474 |

- Now in this table there is no unique value for every tupe, like S1015, there are two values for bkID
- SELECT and JOIN condition will not work properly

# First Normal From Example (Cont..)

| stdID | stdName | stdAddress | progName | bkID |
|---|---|---|---|---|
| S1020 | Sohail | Clifton | BS-CS | B00129 |
| S1038 | Shahid | Phase 1 | BS-IT | B00327 |
| S1015 | Fawad | Phase 5 | BS-SE | B08945 |
| S1015 | Fawad | Phase 5 | BS-SE | B06352 |
| S1020 | Jawad | Clifton | M.Sc | B08474 |

Now this table is in first normal form and for **every tuple there is unique value**.

# Second Normal Form (2NF)

- Full functional dependency:
  - An attribute B is fully functionally dependent on A if the B can be determined by whole of A not by any proper subset of A

# Consider the relation

CLASS (crID, stdID, stdName, fID, room, grade)

stdID ⟶ stdName *(Partially depend on stdID)*

crID ⟶ fID, room *(Partially depend on crID)*

**crID, stdID ⟶ stdName, fID, room, grade (Fully)**

# Second Normal Form (2NF)

- A relation is in 2nd normal form if it is in the first normal form and all non key attributes are fully functionally dependent on key, that is, there is no partial dependency

- CLASS (crID, stdID, stdName, fID, room, grade)
- crID, stdID  stdName, fID, room, grade
- stdID  stdName
- crID  fID, room

If not fully functional on key so it should not be 2nd normalized form

# Anomalies

- Redundancy
- Insertion Anomaly
- Deletion Anomaly
- Updation Anomaly

# Anomalies

| crID | stdID | stdName | fID | room | grade |
|---|---|---|---|---|---|
| C3456 | S1020 | Sohail | F2345 | 104 | B |
| C5678 | S1020 | Sohail | F4567 | 106 | |
| C3456 | S1038 | Fawad | F2345 | 104 | A |
| C5678 | S1015 | Jawad | F4567 | 106 | B |
| C237 | S1070 | Paul | F4567 | 321 | |

# How to do it in 2NF

- Relation is decomposed based on Full Dependency (FDs)
- Create multiple relations

***CLASS*** *(crID, stdID, stdName, fID, room, grade)*

| | |
|---|---|
| *crID, stdID* | *stdName, fID, room, grade* |
| *stdID* | *stdName* |
| *crID* | *fID, room* |

# Decomposed into three tables

- ***STD*** (stdID, stdName)
- ***COURSE*** (crID, fID, room)
- ***CLASS*** (crID, stdID, grade)

- Each of these tables is in second normalization form
- No anomalies available now in this form and way to say this as 2NF

# Third Normal Form (3NF)

- A table is in third normal form (3NF)  if it is in 2NF and there is not transitive dependency, that is, no non-key attribute is dependent on another non-key attribute

- All non key attributes are functionally dependent only upon the primary key.

# Transitive Dependency

Transitive: Non key not dependent on non key

***STD*** (***stdID***, stNmae, stAddr, prName, prCrdts)

***stdID*** stdName, stdAddr, prName, prCrdts

***prName*** prCrdts

# All Anomalies is there

| stdID | stdName | stAdr | prName | prCrdts |
|---|---|---|---|---|
| S1020 | Sohail | Clifton | BS CS | 64 |
| S1038 | Shahid | Phase 1 | BS SE | 132 |
| S1015 | Jawad | Phase 2 | BS IT | 140 |
| S1016 | James | Phase 3 | BS IT | 140 |
| S1018 | Khalid | Phase 5 | BBA | 134 |

***STD*** (stdID, stdName, stAdr, prName)

***PROGRAM*** (prName, prCrdts)

- **Each of the table is in 3NF**
- **Free of all anomalies**

# Boyce-Codd Normal Form (BCNF)

- A general form of 3NF
- Every relation in BCNF is in 3NF vice-versa is not always true
- 90% sure if you transform it in 3NF, it is in BCNF
- If you done BCNF 100% sure it is in 3NF
- 10% chance if it is in 3NF but not in BCNF, it is some specifications.
- 3NF is checked in steps, BCNF checked directly
- A table is in BCNF if every determinant is a candidate key

# Boyce-Codd Normal Form Contd…

- Situation when table in 3NF is not in BCNF
  - Only one candidate key which is primary key (3NF and BCNF)
  - Multiple candidate keys depend on single attribute
    - Alarm situations
  - Multiple composite candidate keys
  - Overlapping one or more attribute which is common between these candidate keys

# After BCNF

- We have Fourth (4NF), fifth (5NF) and domain-key Normal Forms exist (DKNF)
- 4NF deals with multivalued dependency
- 5NF deals with possible lossless decompositions.
- DKNF reduces further chances of any possible inconsistency.

# Demonstration: Introducing Microsoft SQL Server 2019

In this demonstration you will see how to:

- Use SSMS to connect to an on-premises instance of SQL Server 2019
- Explore databases and other objects
- Work with T-SQL scripts

# Demonstration: Introducing Microsoft SQL Server 2019

- **Demonstration Steps**

1. Start **SQL Server Management Studio** and connect to **SQL Server 2019** engine instance using Windows authentication.

2. Unzip the Lab Session Demo files in a folder **D:\Demofiles\Mod01** folder. This includes all the Demo, Exercises and Solution files with a .**SQL** extension. (All **SQL** files can also be opened in Notepad)

3. Double Click the appropriate **SQL** file and it should load automatically in the **Query Window.** Otherwise open in Notepad and copy and past into the Query Window.

4. Make sure you click on the correct **SQL Server Database** in the **Object Explorer**

## Logon Information

## SQL Server Instance is the Lab PC name [or number]
## Use [windows authentication]

# Demonstration: Introducing Microsoft SQL Server 2019

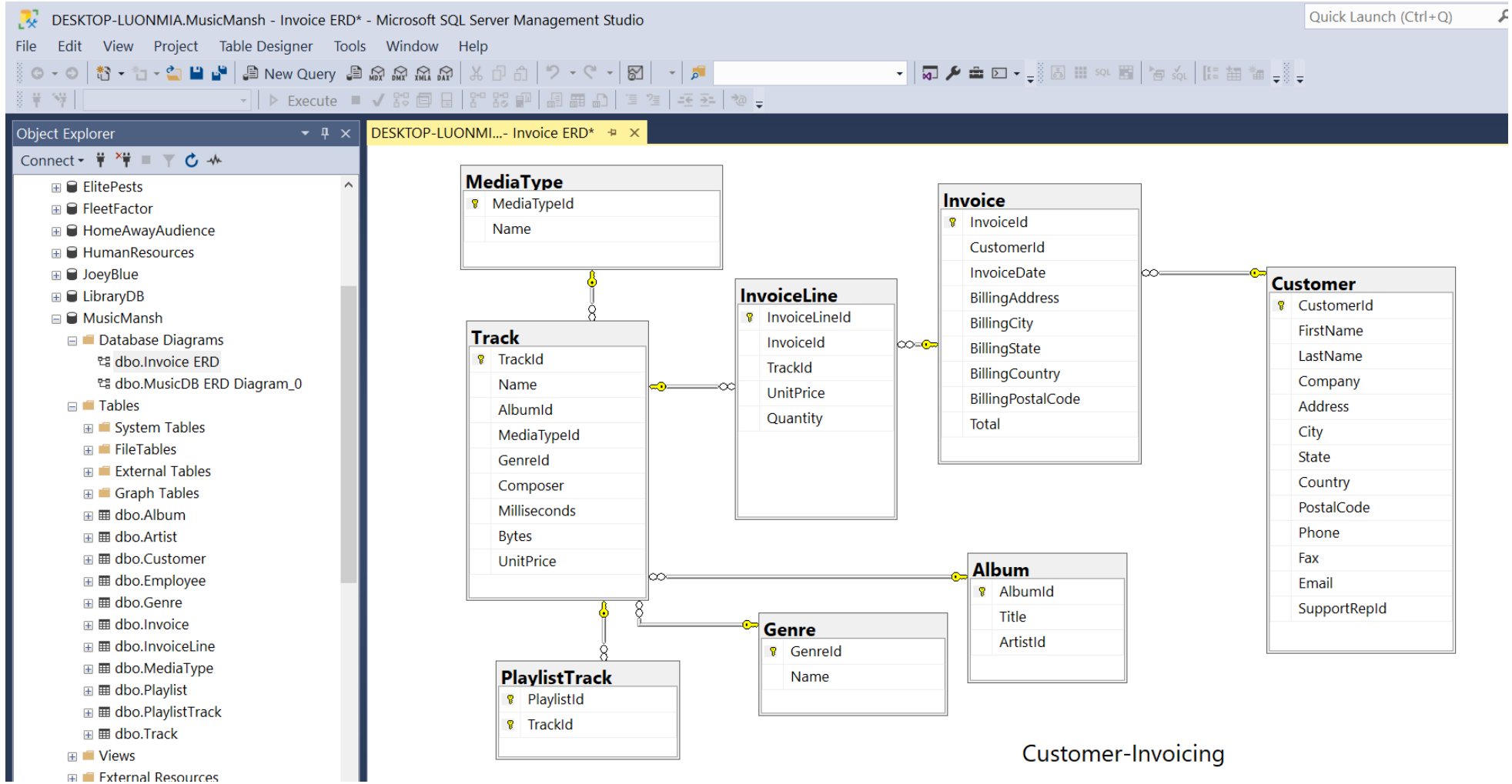
- **Explore Database and Other Objects**

1. In Object Explorer, expand the **Databases** folder to see a list of databases.
2. Expand the **TSQL** database.
3. Expand the **Tables** folder.
4. Expand the **Sales.Customers** table.
5. Expand the **Columns** folders.
6. View the list of columns, and the data type information for each column.
7. Note the data type for the **companyname** column.

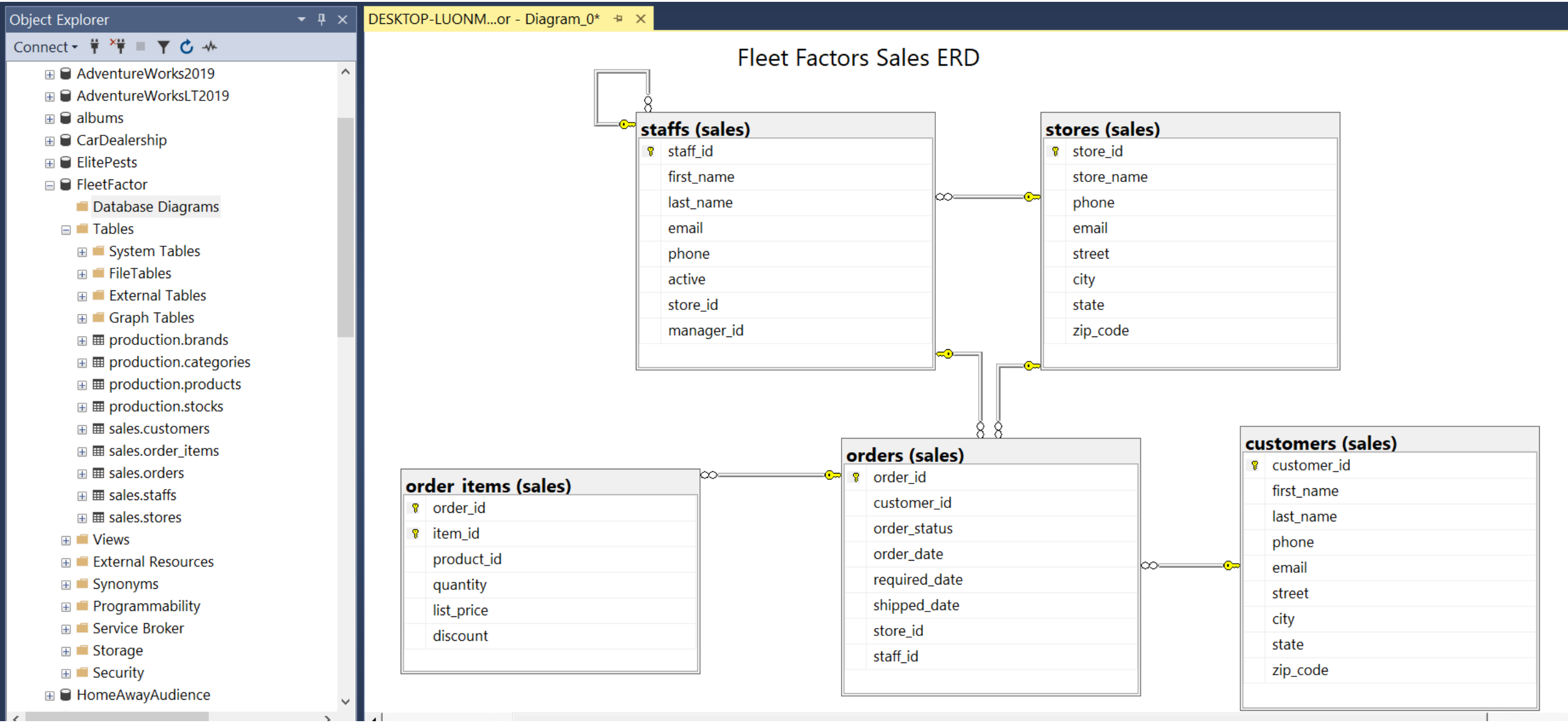
- **Work with T-SQL Scripts (or the appropriate SQL Server Database Sample)**

1. If the Solution Explorer pane is not visible, on the **View** menu, click **Solution Explorer**.
2. In Solution Explorer, notice it is empty.

# SQL Demo


DESKTOP-LUONMIA.MusicMansh - Invoice ERD* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Project Table Designer Tools Window Help
New Query
Execute
Object Explorer
Connect
ElitePests
FleetFactor
HomeAwayAudience
HumanResources
JoeyBlue
LibraryDB
MusicMansh
Database Diagrams
dbo.Invoice ERD
dbo.MusicDB ERD Diagram_0
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Album
dbo.Artist
dbo.Customer
dbo.Employee
dbo.Genre
dbo.Invoice
dbo.InvoiceLine
dbo.MediaType
dbo.Playlist
dbo.PlaylistTrack
dbo.Track
Views
External Resources
DESKTOP-LUONMI...- Invoice ERD*
MediaType
MediaTypeId
Name
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total
Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId
InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity
Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice
Album
AlbumId
Title
ArtistId
Genre
GenreId
Name
PlaylistTrack
PlaylistId
TrackId
Customer-Invoicing

# eCommerce SQL solution for Fleet Factors



# Supporting Material

- Highly recommended to walk through in your own time

# Entity Relationship Diagram (ERD) Training Video

https://youtu.be/-fQ-bRllhXc

# SQL Server Management Studio (SSMS) | Full Course


https://youtu.be/Q8gBvsUjTLw