INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 05

Modelling with MySQL Workbench

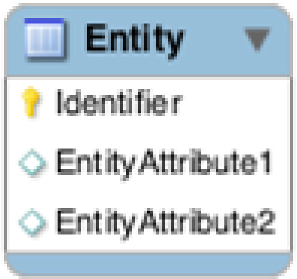
Week 3

Coverage

- Modelling with MySQL Workbench
- Recap & further design
  - Conceptual Design
  - Logical Design
  - Physical Design

Conventions of ER Modelling (Workbench)

- Entity



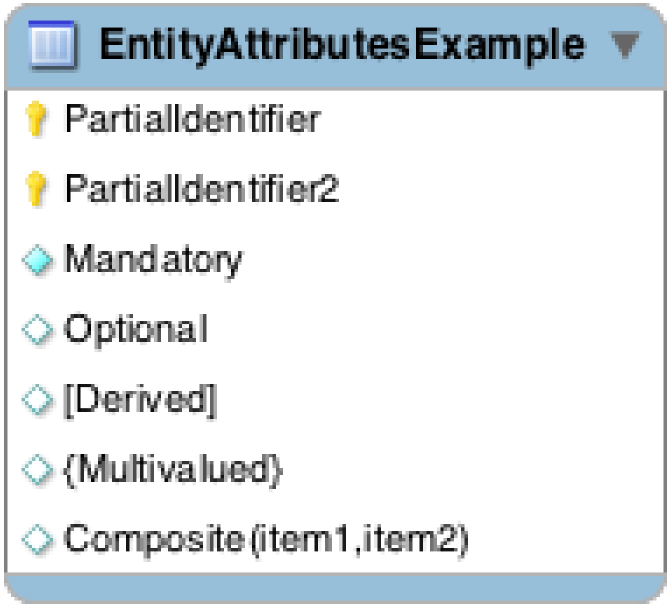
Entity

Identifier

EntityAttribute1

EntityAttribute2

- Attributes



EntityAttributesExample

PartialIdentifier

PartialIdentifier2

Mandatory

Optional

[Derived]

{Multivalued}

Composite(item1,item2)

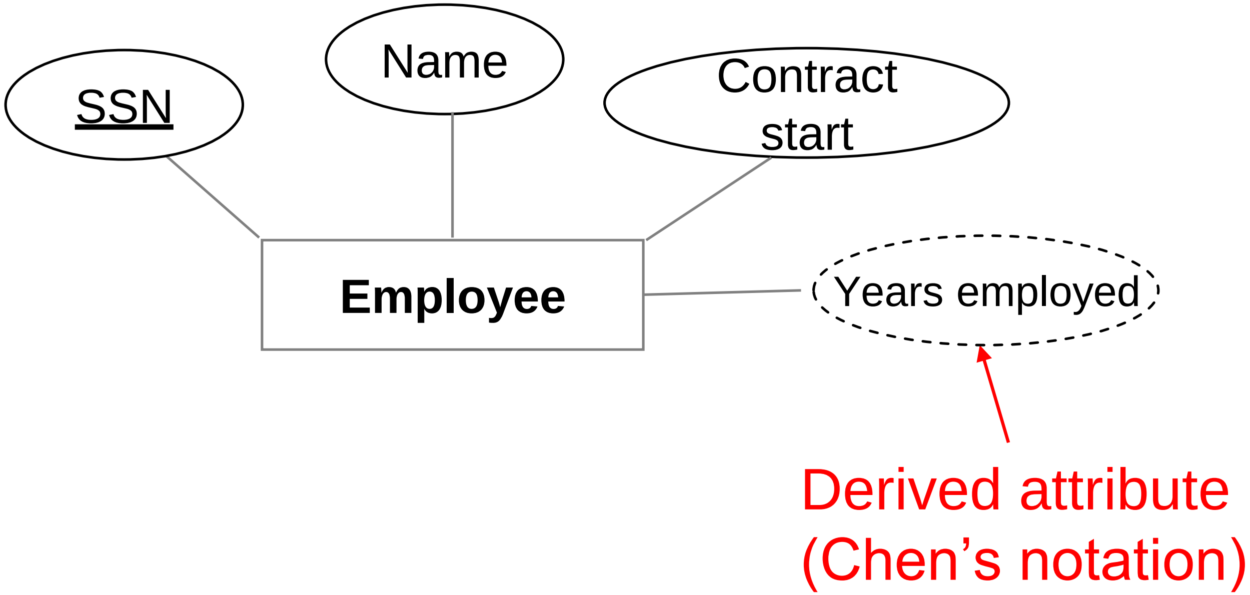
- **Identifier or key:**
  - Fully identifies an instance
- **Partial Identifier:**
  - Identifies an instance in conjunction with one or more partial identifiers
- **Attributes types:**
  - Mandatory – NOT NULL (blue diamond)
  - Optional - NULL (empty diamond)
  - Derived []
    - [YearsEmployed]
  - Multivalued {}
    - {Skill}
  - Composite ()
    - Name (First, Middle, Last)

A note on derived attributes

- Derived attributes imply that their values can be derived from some other attributes in the database. As a result, they do not need to be stored physically – they disappear at the physical design.

Example:

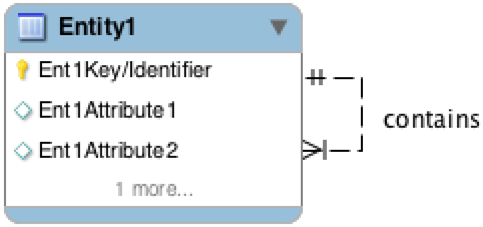
For employees we want to be able to show for how many years they have been employed.



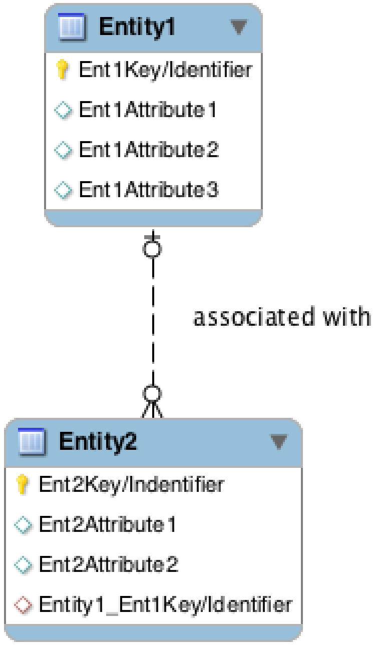
Conventions of ER Modelling (Workbench)

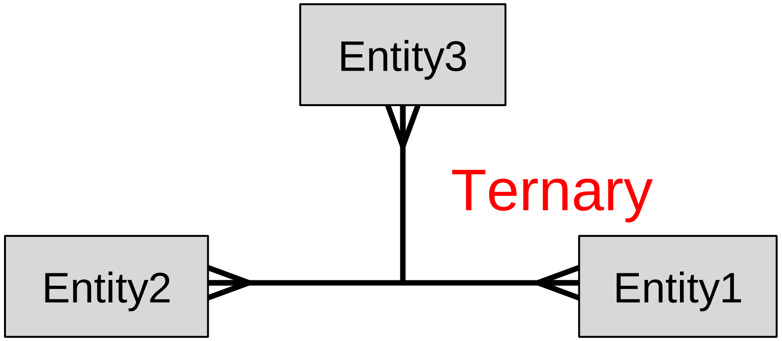
- Relationship Degrees

Unary


Entity1
Ent1Key/Identifier
Ent1Attribute1
Ent1Attribute2
1 more...
contains

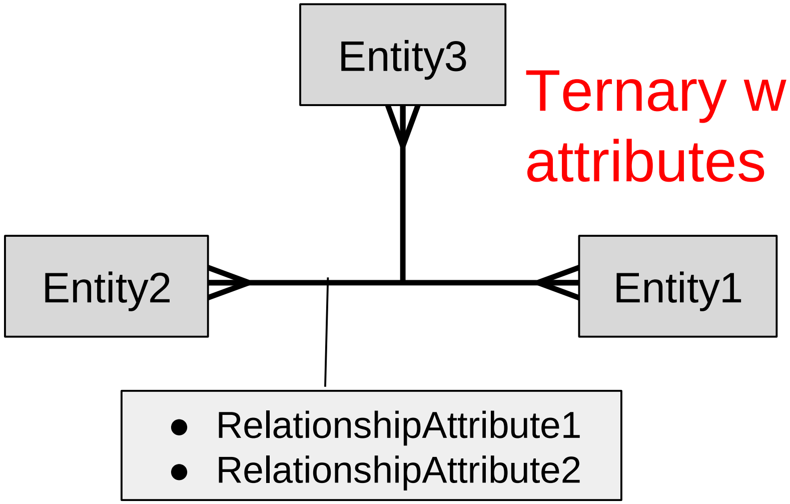
Binary


Entity1
Ent1Key/Identifier
Ent1Attribute1
Ent1Attribute2
Ent1Attribute3
associated with
Entity2
Ent2Key/Indentifier
Ent2Attribute1
Ent2Attribute2
Entity1_Ent1Key/Identifier


Entity3
Entity2
Entity1

Ternary

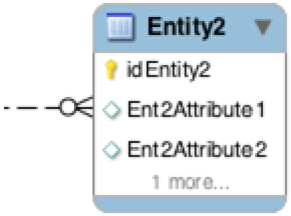
Ternary with attributes


Entity3
Entity2
Entity1

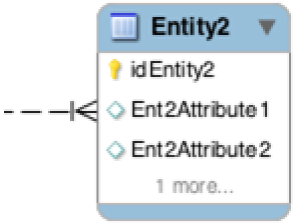
- RelationshipAttribute1
- RelationshipAttribute2

Conventions of ER Modelling (Workbench)

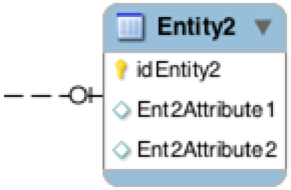
- Cardinality Constraints



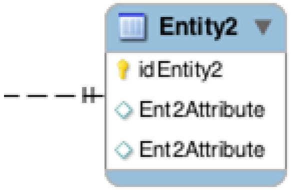
Optional Many
Partial participation
Without key constraint



Mandatory Many
Total participation
Without key constraint



Optional One
Partial participation
Key constraint



Mandatory One
Total participation
Key constraint

- Relationship Cardinality
  - One to One

    Each entity will have exactly 0 or 1 related entity
  - One to Many

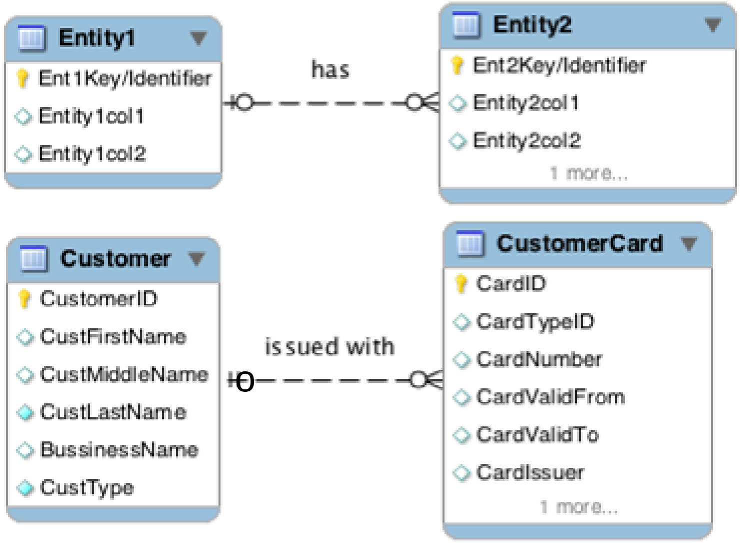
    One of the entities will have 0, 1 or *more* related entities, the other will have 0 or 1.
  - Many to Many

    Each of the entities will have 0, 1 or *more* related entities

Conventions of ER Modelling (Workbench)

Strong Entity:

- Can exist by itself
- E.g. Customer Card & Customer



Entity1
Ent1Key/Identifier
Entity1col1
Entity1col2

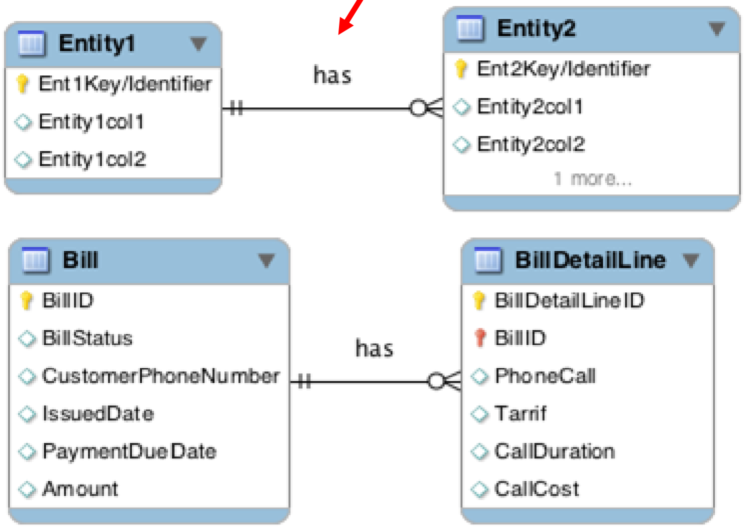
Entity2
Ent2Key/Identifier
Entity2col1
Entity2col2
1 more...

Customer
CustomerID
CustFirstName
CustMiddleName
CustLastName
BussinessName
CustType

CustomerCard
CardID
CardTypeID
CardNumber
CardValidFrom
CardValidTo
CardIssuer
1 more...

Weak Entity

- Can’t exist without the owner
- E.g. BillDetailLine



Identifying relationship

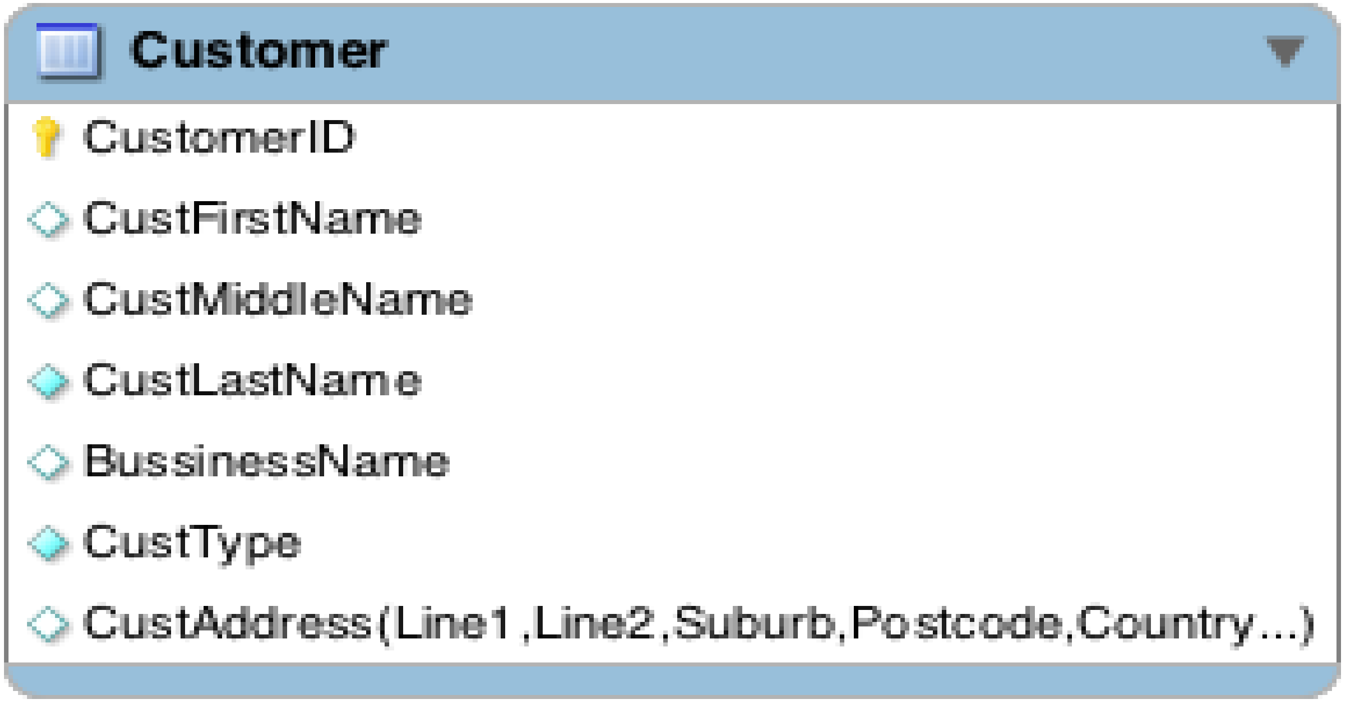
Entity1
Ent1Key/Identifier
Entity1col1
Entity1col2

Entity2
Ent2Key/Identifier
Entity2col1
Entity2col2
1 more...

Bill
BillID
BillStatus
CustomerPhoneNumber
IssuedDate
PaymentDueDate
Amount

BillDetailLine
BillDetailLineID
BillID
PhoneCall
Tarrif
CallDuration
CallCost

Single Entity (Conceptual Model)



Customer

CustomerID

CustFirstName

CustMiddleName

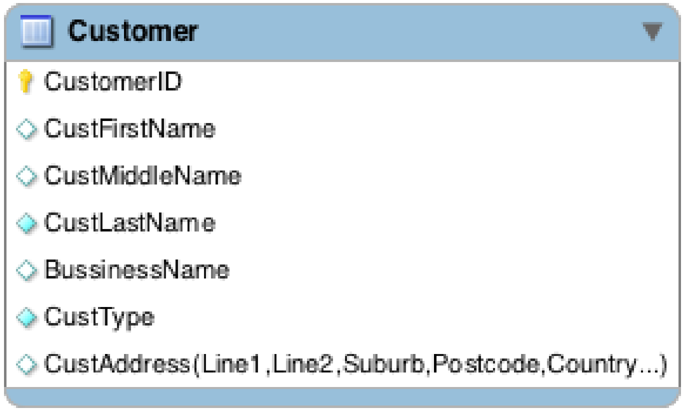
CustLastName

BussinessName

CustType

CustAddress(Line1,Line2,Suburb,Postcode,Country...)

Convert from Conceptual to Logical design



Customer

CustomerID

CustFirstName

CustMiddleName

CustLastName

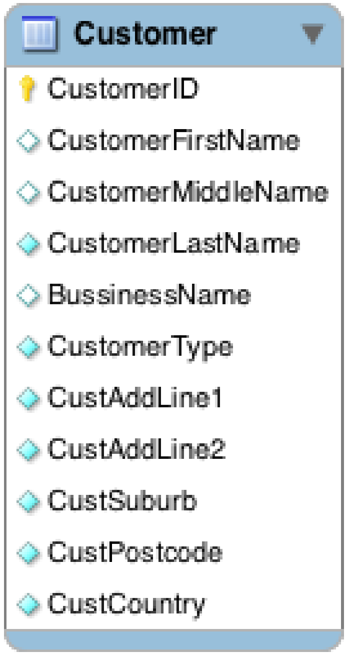
BussinessName

CustType

CustAddress(Line1,Line2,Suburb,Postcode,Country...)

- Convert the ER into a logical (rel.) model
  - Customer(CustomerID, CustFirstName, CustMiddleName, CustLastName, BusinessName, CustType, CustAddLine1, CustAddLine2, CustSuburb, CustPostcode, CustCountry)

- **Tasks checklist (from conceptual to logical):**
  1. Flatten composite and multi-valued attributes
     - Multi-value attributes can become another table
  2. Resolve many-many relationships
     - Create an associative entity
  3. Resolve one-many relationships
     - Add foreign keys at crows foot end of relationships (on the many side in the case of crows foot)



Customer

CustomerID

CustomerFirstName

CustomerMiddleName

CustomerLastName

BussinessName

CustomerType

CustAddLine1

CustAddLine2

CustSuburb

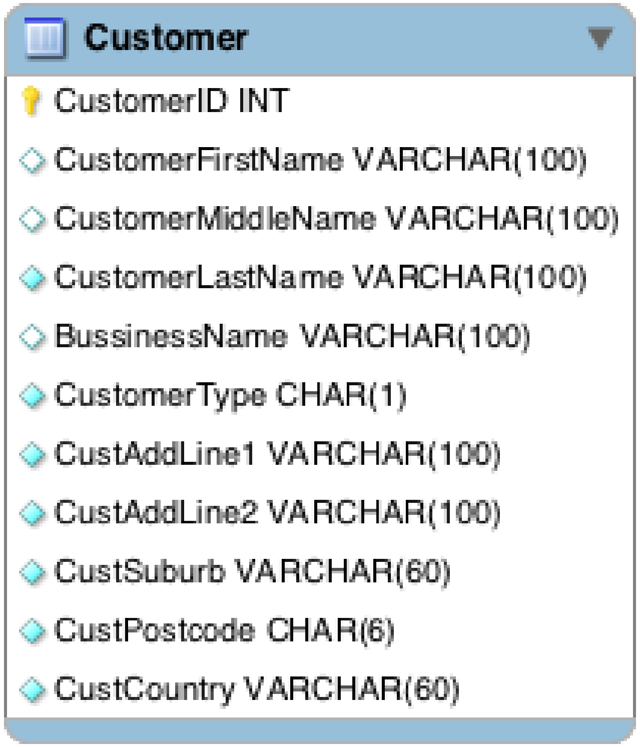
CustPostcode

CustCountry

Convert from Logical

- **Generate attribute data types (with NULL/NOT NULL)**

Physical Design:



Customer

CustomerID INT

CustomerFirstName VARCHAR(100)

CustomerMiddleName VARCHAR(100)

CustomerLastName VARCHAR(100)

BussinessName VARCHAR(100)

CustomerType CHAR(1)

CustAddLine1 VARCHAR(100)

CustAddLine2 VARCHAR(100)

CustSuburb VARCHAR(60)

CustPostcode CHAR(6)

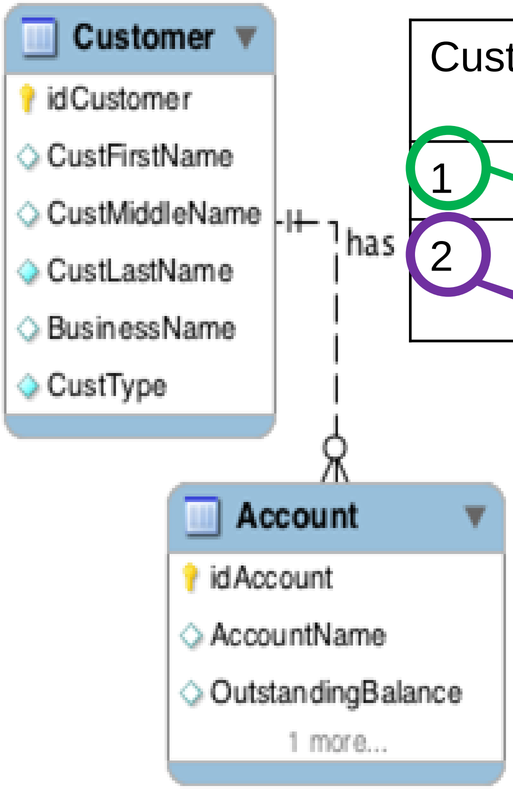
CustCountry VARCHAR(60)

Implementation:

```
CREATE TABLE Customer(
  CustomerID INT NOT NULL,
  CustFirstName VARCHAR(100),
  CustMiddleName  VARCHAR(100),
  CustLastName VARCHAR(100) NOT NULL,
  BussinessName VARCHAR(100),
  CustType VARCHAR(1) NOT NULL,
  CustAddressLine1 VARCHAR(100) NOT NULL,
  CustAddressLine2 VARCHAR(100) NOT NULL,
  CustSuburb VARCHAR(60) NOT NULL,
  CustPostcode CHAR(6) NOT NULL,
  CustCountry VARCHAR(60) NOT NULL,
 PRIMARY KEY (CustomerID));
```

More than One Entity

- A customer can have a number of Accounts
- The tables are linked through a foreign key

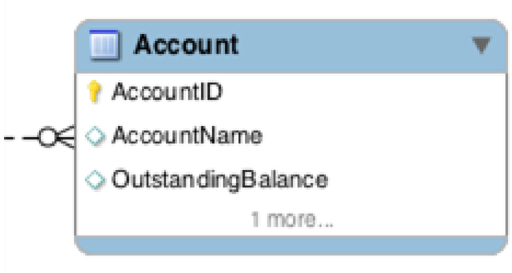


CustID	CustomerFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company

AccountID	AccountName	OutstandingBalance	CustID
01	Peter Smith	245.25	1
05	JJ Ent.	552.39	2
06	JJ Ent. Mgr	10.25	2

From Conceptual to Logical Design - Account

Conceptual Design:



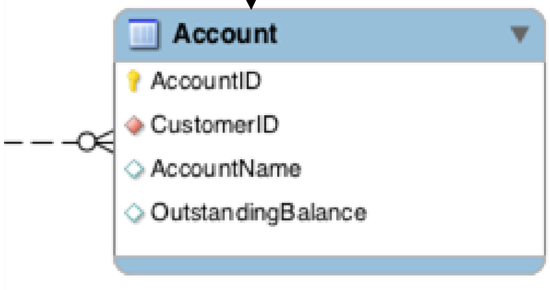
Logical Design:

Account(AccountID, AccountName, OutstandingBalance, *CustomerID*)

Note: Underline = PK, italic and underline = FK, underline and bold = PFK

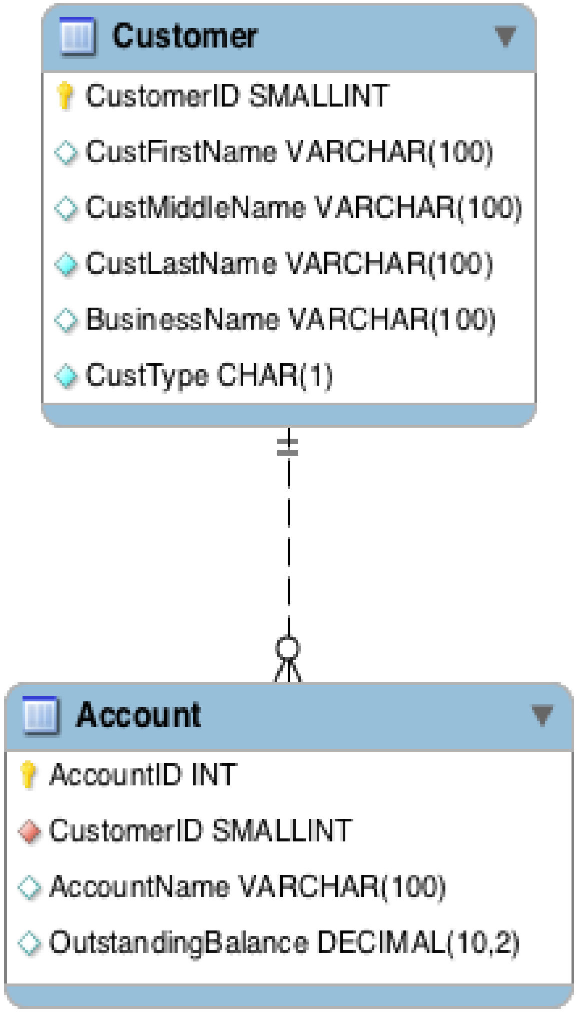
Tasks checklist:

1. Flatten composite and multi-valued attributes X
2. Resolve many-many relationships X
3. Resolve one-many relationships
   - See FK1 – CustomerID
   - Every row in the account table must have a CustomerID from Customer (referential integrity)



Physical Design & Implementation- Account

Physical design:



Customer

CustomerID SMALLINT

CustFirstName VARCHAR(100)

CustMiddleName VARCHAR(100)

CustLastName VARCHAR(100)

BusinessName VARCHAR(100)

CustType CHAR(1)

Account

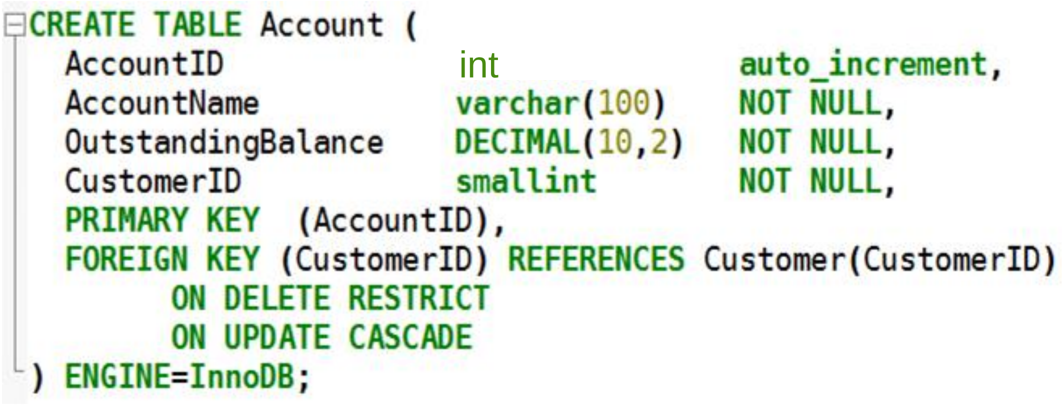
AccountID INT

CustomerID SMALLINT

AccountName VARCHAR(100)

OutstandingBalance DECIMAL(10,2)

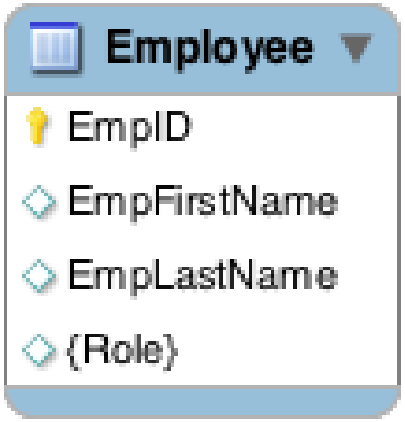
Implementation:



```
CREATE TABLE Account (
   AccountID              int           auto_increment,
   AccountName            varchar(100)  NOT NULL,
   OutstandingBalance     DECIMAL(10,2) NOT NULL,
   CustomerID             smallint      NOT NULL,
   PRIMARY KEY  (AccountID),
   FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
         ON DELETE RESTRICT
         ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Dealing with Multi-Valued Attributes: Approach 2

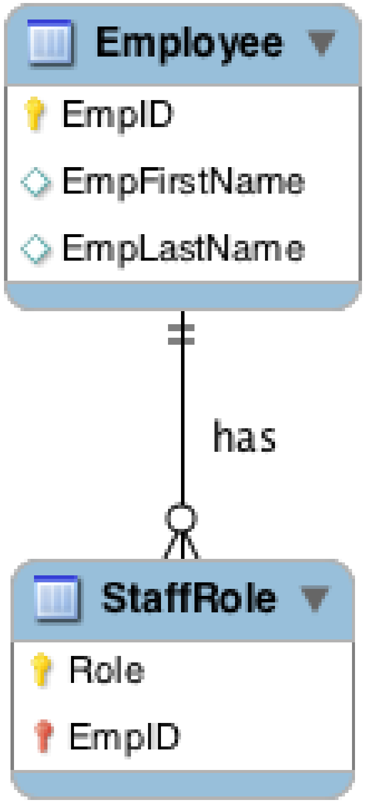
Conceptual Design:



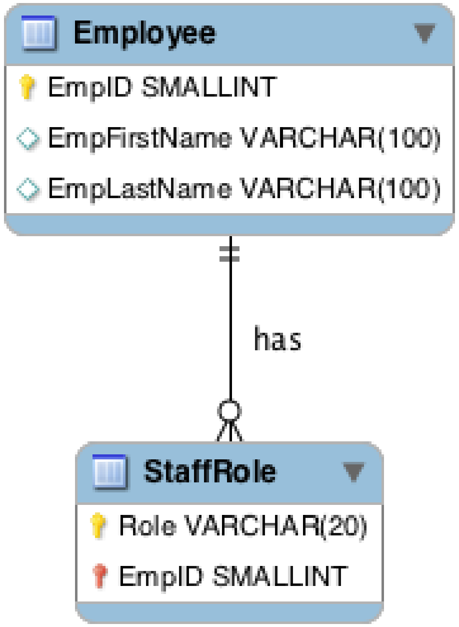
StaffRole is an example of a weak entity

- We show this with a *solid* line in Workbench

Logical Design:



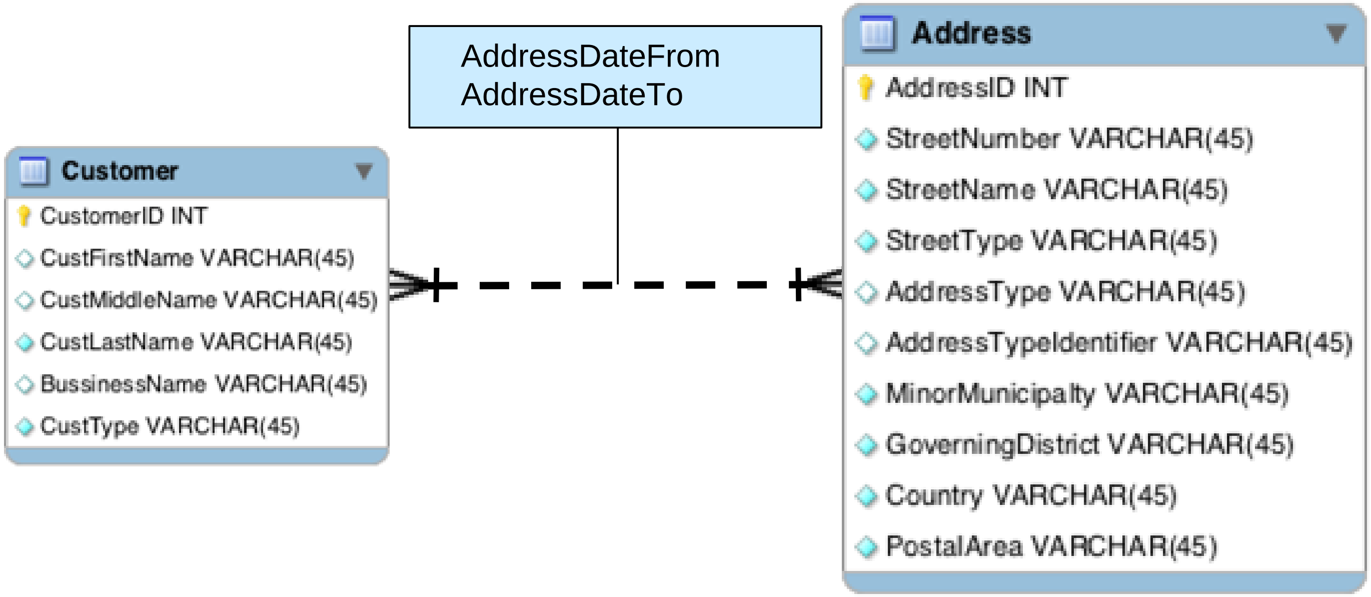
Physical Design:



If staff have only 2-3 roles you may decide to have these within the Employee table at physical design to save on “JOIN” time

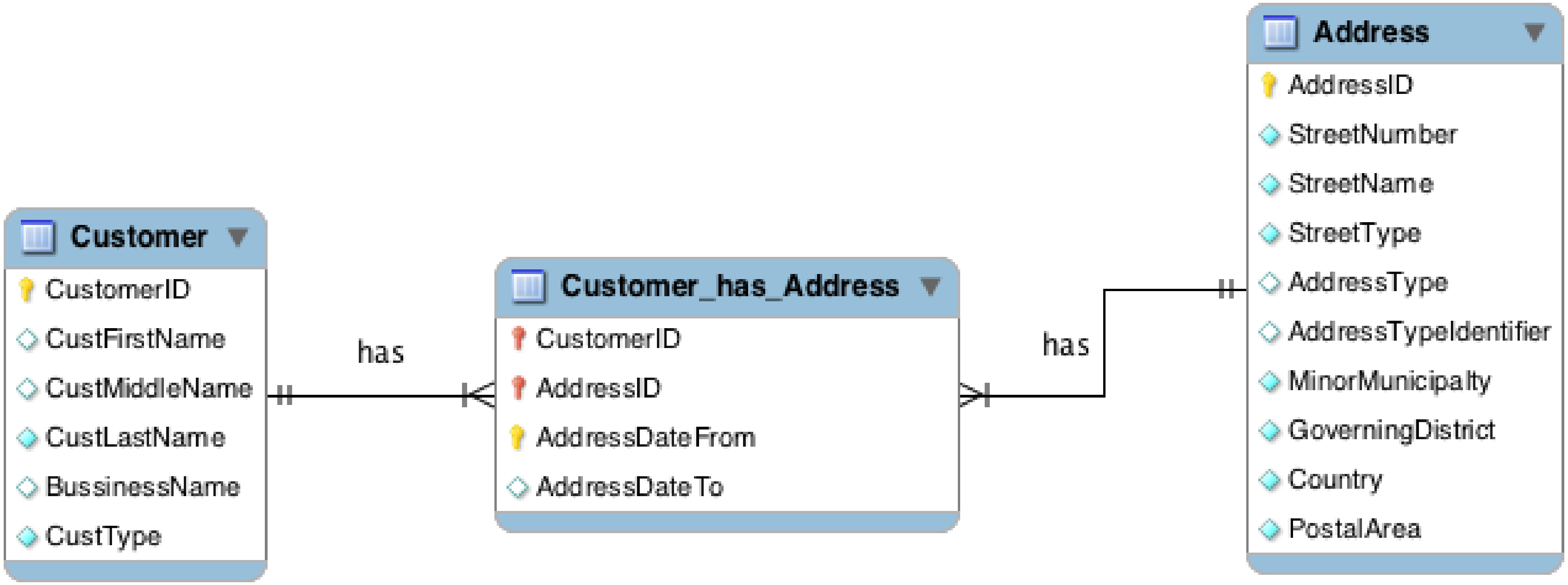
Many to Many Relationship

- How do we deal with customer addresses?
  - If customers can change addresses
    - AND imagine that we *need* to store a history of addresses for customers.
  - At the conceptual level it looks like this:



Many to Many – Logical design (Workbench)

- When converting the conceptual to logical diagram we create an **Associative Entity** between the other 2 entities



Customer
- CustomerID
- CustFirstName
- CustMiddleName
- CustLastName
- BussinessName
- CustType

Customer_has_Address
- CustomerID
- AddressID
- AddressDateFrom
- AddressDateTo

Address
- AddressID
- StreetNumber
- StreetName
- StreetType
- AddressType
- AddressTypeIdentifier
- MinorMunicipalty
- GoverningDistrict
- Country
- PostalArea

Note: AddressDateFrom/To are descriptive attributes of the relationship

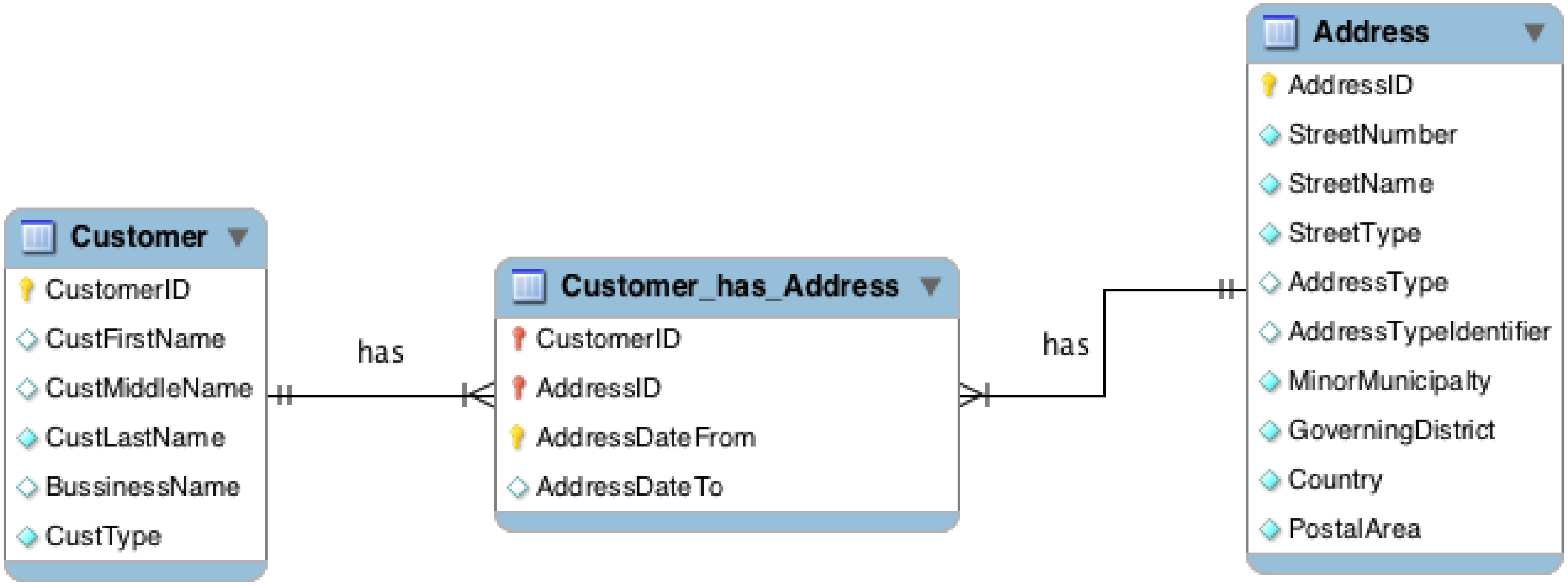
They go into the associative entity for M-M

THE UNIVERSITY OF MELBOURNE

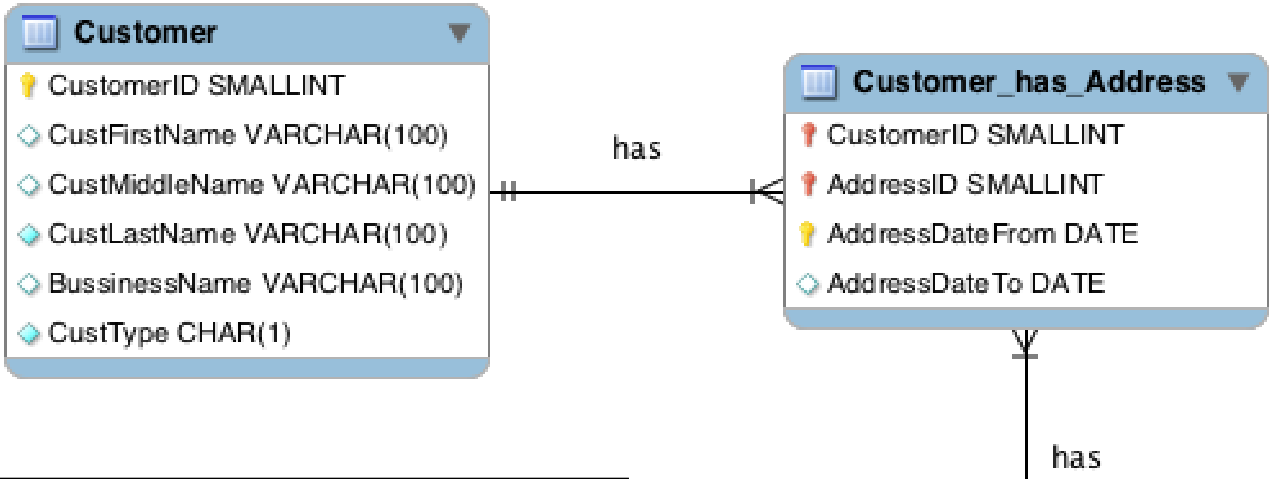
Many to Many - Logical Model

- Customer(CustomerID, CustFirstName, CustMiddleName, CustLastName, BusinessName, CustType)
- Address(AddressID, StreetNumber, StreetName, StreetType, AddressType, AddressTypeIdentifier, MinorMunicipality, MajorMunicipality, GoverningDisctrict, Country, PostalArea)
- Customer_Has_Address(**CustomerID, AddressID**, AddressDateFrom, AddressDateTo)

Note: Underline = PK, italic and underline = FK, underline and bold = PFK



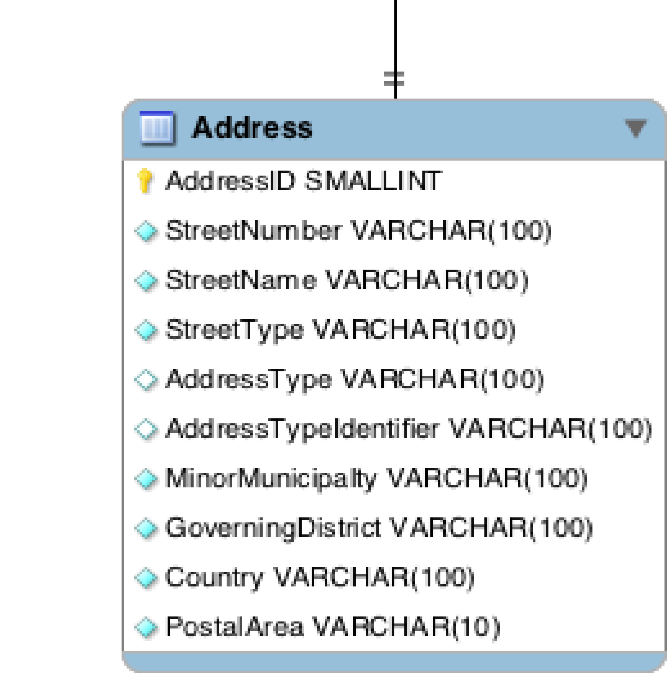
Many to Many - Physical Model & Implementation



Customer
- CustomerID SMALLINT
- CustFirstName VARCHAR(100)
- CustMiddleName VARCHAR(100)
- CustLastName VARCHAR(100)
- BussinessName VARCHAR(100)
- CustType CHAR(1)

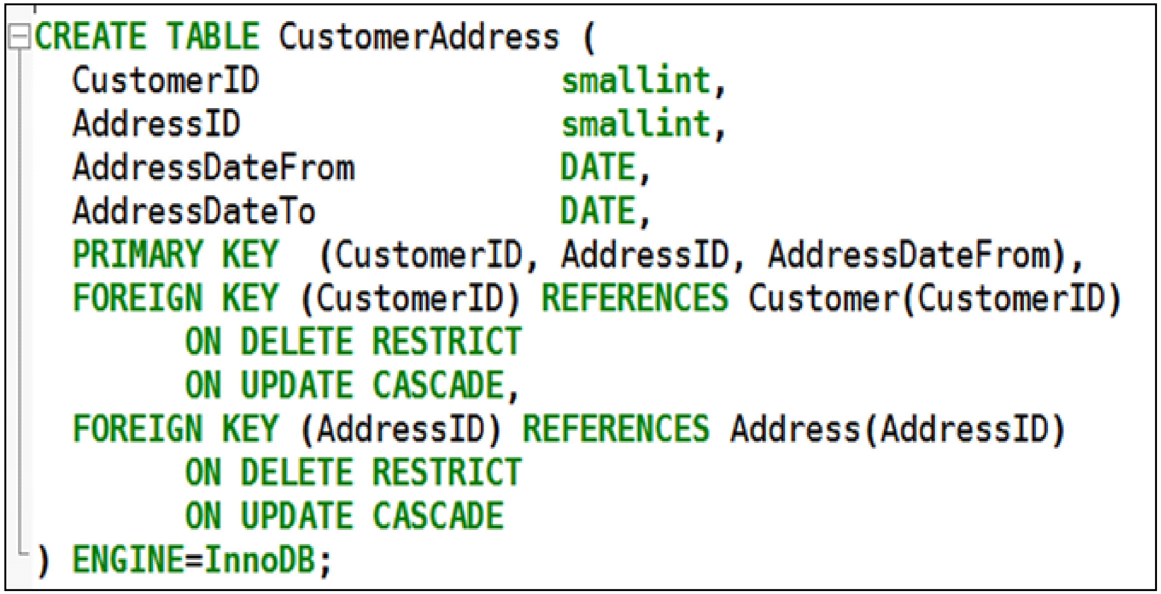
has

Customer_has_Address
- CustomerID SMALLINT
- AddressID SMALLINT
- AddressDateFrom DATE
- AddressDateTo DATE



has

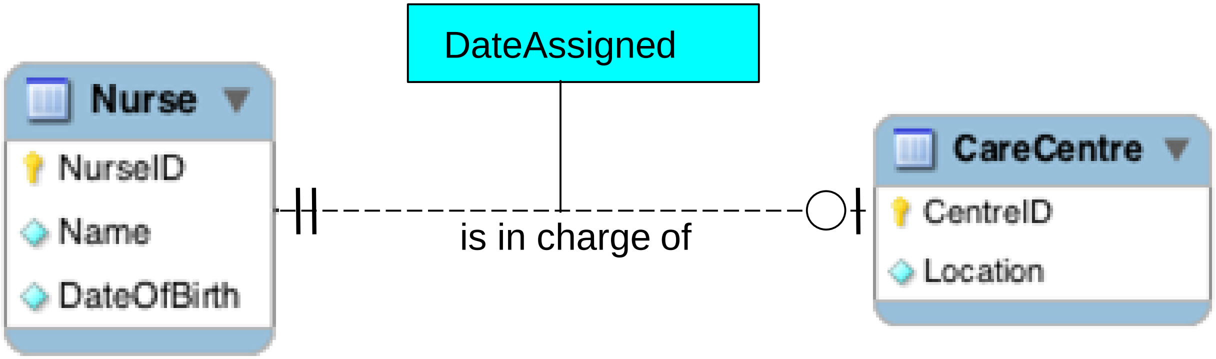
Address
- AddressID SMALLINT
- StreetNumber VARCHAR(100)
- StreetName VARCHAR(100)
- StreetType VARCHAR(100)
- AddressType VARCHAR(100)
- AddressTypeIdentifier VARCHAR(100)
- MinorMunicipalty VARCHAR(100)
- GoverningDistrict VARCHAR(100)
- Country VARCHAR(100)
- PostalArea VARCHAR(10)



```
CREATE TABLE CustomerAddress (
  CustomerID                  smallint,
  AddressID                   smallint,
  AddressDateFrom             DATE,
  AddressDateTo               DATE,
  PRIMARY KEY  (CustomerID, AddressID, AddressDateFrom),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
  FOREIGN KEY (AddressID) REFERENCES Address(AddressID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Binary One-One Relationship

- Rule: Move the key from the *one* side to the other side

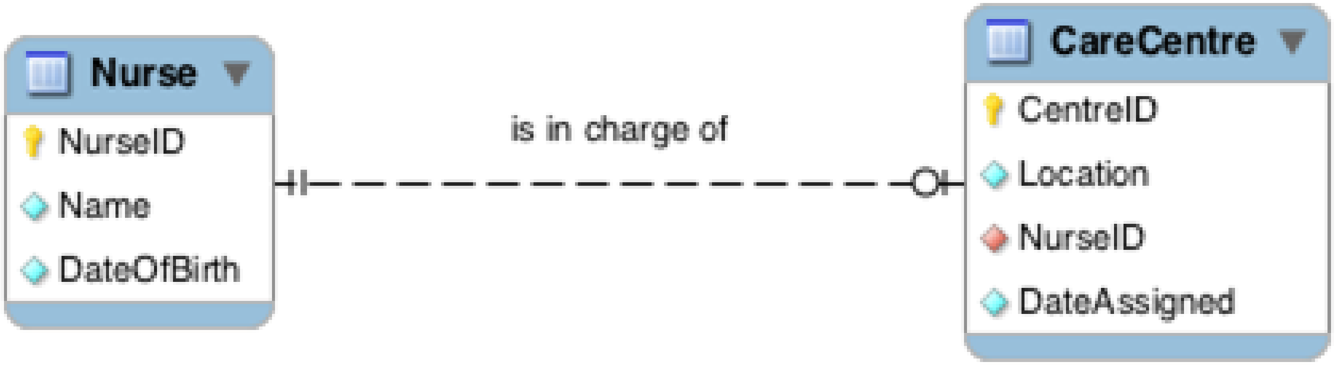


- But we have 2 “one” sides. Which one?

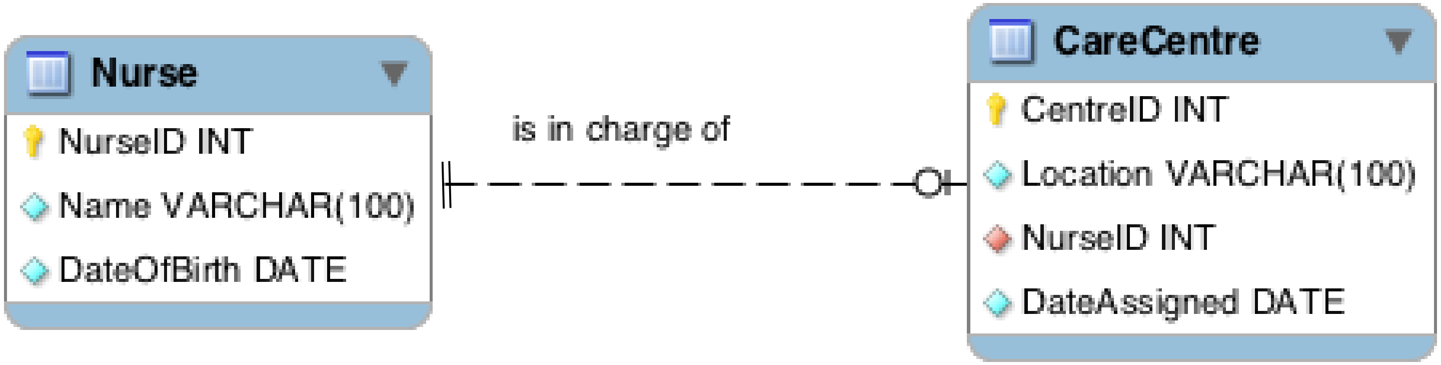
- Need to decide whether to put the foreign key inside Nurse or CareCentre (in which case you would have the Date_Assigned in the same location)
  - Where would the least NULL values be?
  - The rule is the OPTIONAL side of the relationship gets the foreign key

Binary One-One Relationship – Logical and Physical Design

- **Logical Design:**
  - Nurse(<u>NurseID</u>, Name, DateOfBirth)
  - CareCentre(<u>CentreID</u>, Location, <u>*NurseID*</u>, DateAssigned)



- **Physical Design:**



Summary of Binary Relationships From conceptual to logical

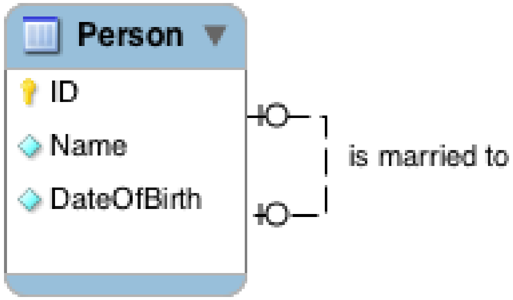
- **One-to-Many**
  - Primary key on the one side becomes a foreign key on the many side (in the case of Crow's foot)
- **Many-to-Many**
  - Create an Associative Entity (a new relation) with the primary keys of the two entities it relates to as the combined primary key
- **One-to-One**
  - Need to decide where to put the foreign key
  - The primary key on the mandatory side becomes a foreign key on the optional side
  - If two optional or two mandatory, pick one arbitrarily

Unary Relationships

- Operate in the same way as binary relationships
  - **One-to-One**
    - Put a Foreign key in the relation
  - **One-to-Many**
    - Put a Foreign key in the relation
  - **Many-to-Many**
    - Generate an Associative Entity
    - Put two Foreign keys in the Associative Entity
      - Need 2 different names for the Foreign keys
      - Both Foreign keys become the *combined* key of the Associative Entity

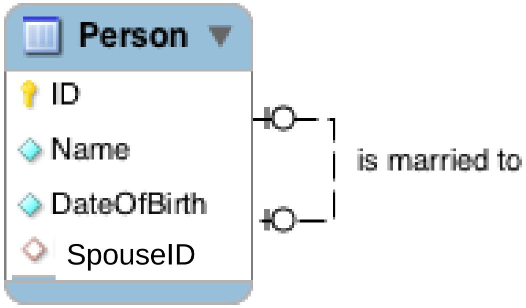
Unary: One-to-One

Conceptual Design:



Logical Design:

- Person (<u>ID</u>, Name, DateOfBirth, *<u>SpouseID</u>*)



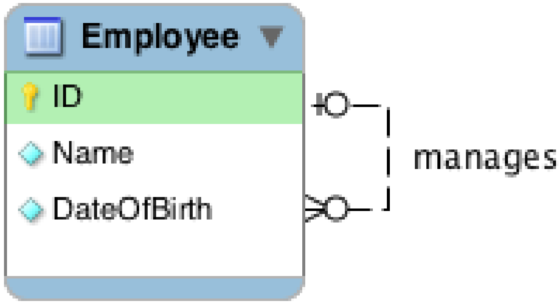
Implementation:

```
CREATE TABLE Person (
  ID INT NOT NULL,
  Name VARCHAR(100) NOT NULL,
  DateOfBirth DATE NOT NULL,
  SpouseID INT,
  PRIMARY KEY (ID),
  FOREIGN KEY (SpouseID)
  REFERENCES Person (ID)
  ON DELETE RESTRICT
  ON UPDATE CASCADE);
```

ID	Name	DOB	SpouseID
1	Ann	1969-06-12	3
2	Fred	1971-05-09	NULL
3	Chon	1982-02-10	1
4	Nancy	1991-01-01	NULL

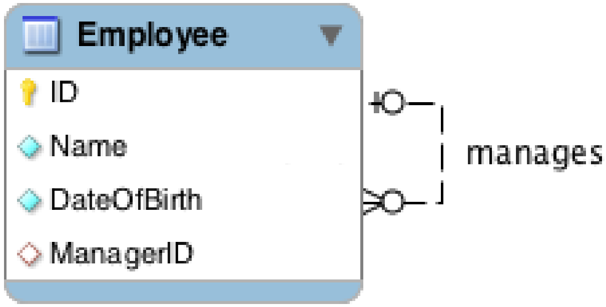
Unary: One-to-Many

Conceptual Design:



Logical Design:

- Employee (<u>ID</u>, Name, DateOfBirth, *<u>ManagerID</u>*)

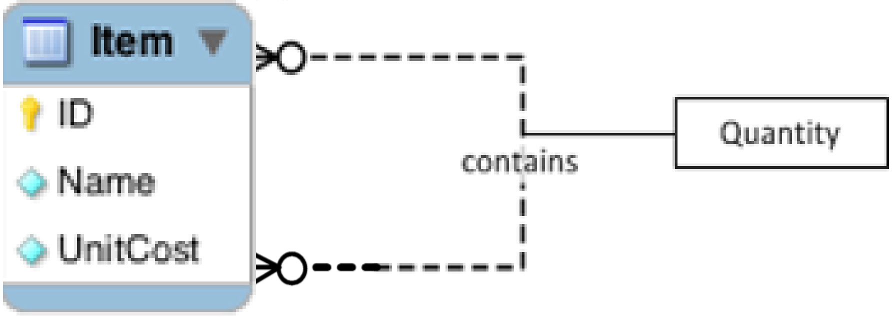


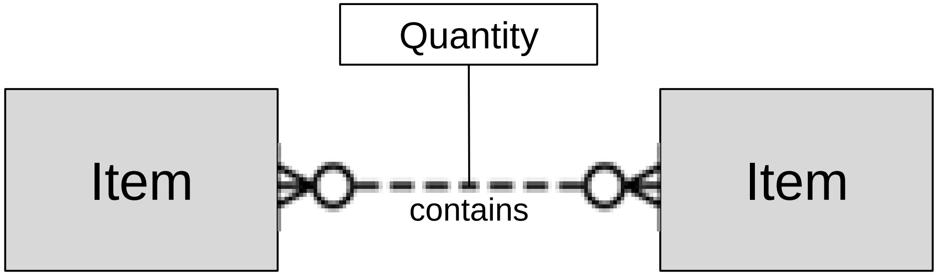
Implementation:

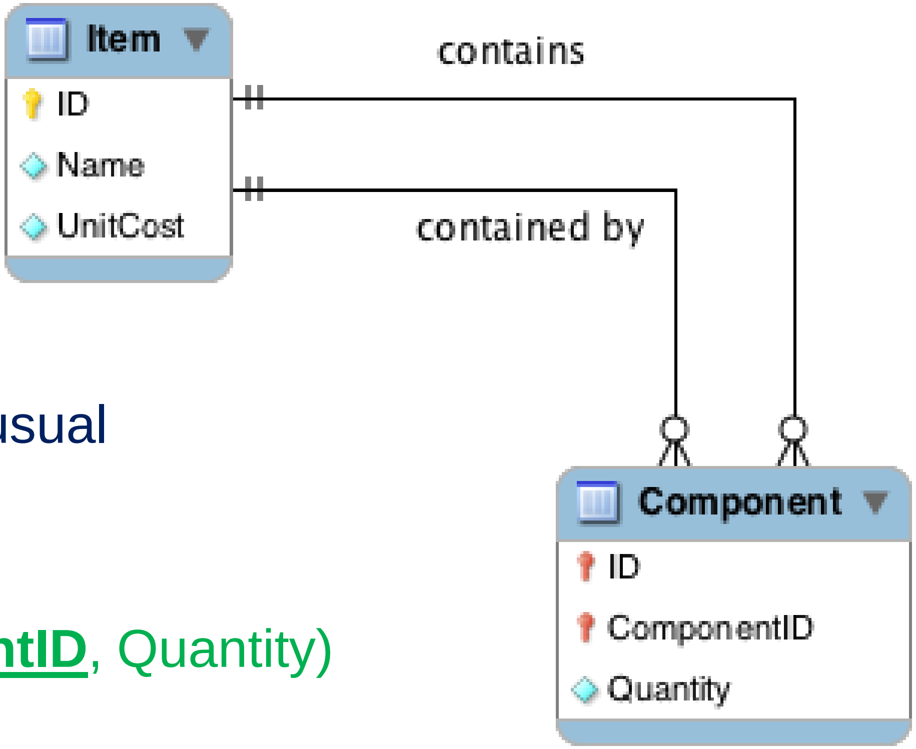
```
CREATE TABLE Employee(
  ID smallint NOT NULL,
  Name VARCHAR(100) NOT NULL,
  DateOfBirth DATE NOT NULL,
  ManagerID smallint ,
  PRIMARY KEY (ID),
  FOREIGN KEY (ManagerID)
  REFERENCES Employee(ID)
  ON DELETE RESTRICT
  ON UPDATE CASCADE);
```

ID	Name	DOB	MngrID
1	Ann	1969-06-12	NULL
2	Fred	1971-05-09	1
3	Chon	1982-02-10	1
4	Nancy	1991-01-01	1

Unary: Many-to-Many





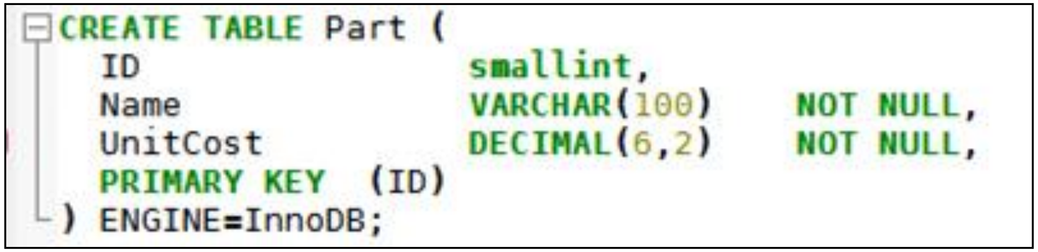


- **Logical Design:**
  - Create Associative Entity like usual
  - Generate logical model
    - Item(ID, Name, UnitCost)
    - Component(**ID, ComponentID**, Quantity)

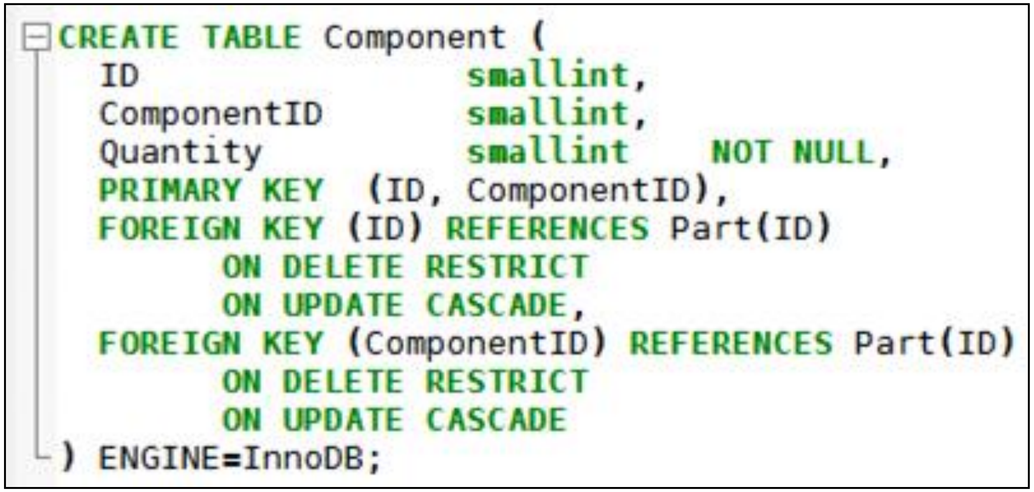
THE UNIVERSITY OF MELBOURNE

Unary: Many-to-Many Implementation

- Implementation



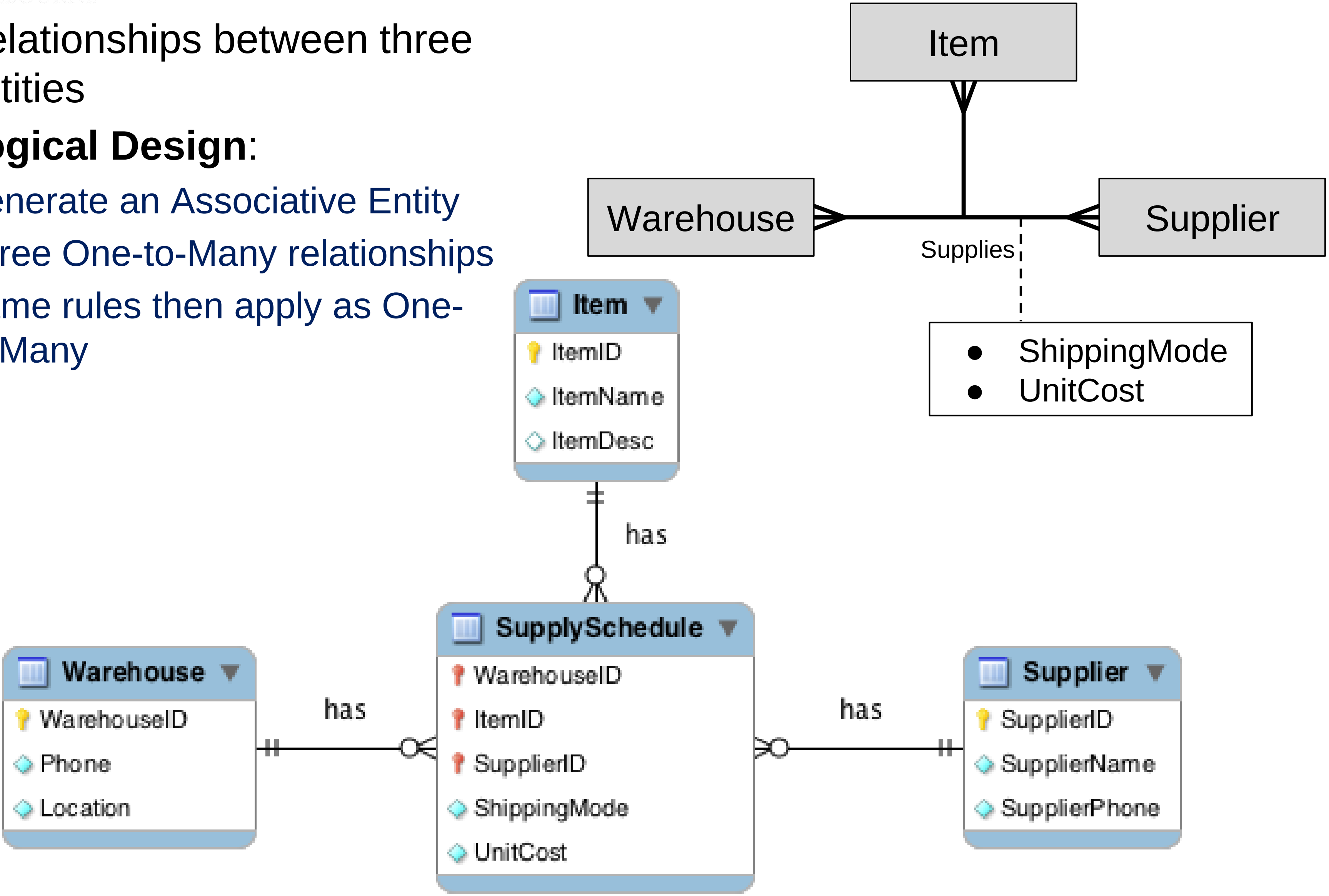
```
CREATE TABLE Part (
   ID                    smallint,
   Name                  VARCHAR(100)    NOT NULL,
   UnitCost              DECIMAL(6,2)    NOT NULL,
   PRIMARY KEY  (ID)
) ENGINE=InnoDB;
```



```
CREATE TABLE Component (
   ID                    smallint,
   ComponentID           smallint,
   Quantity              smallint    NOT NULL,
   PRIMARY KEY  (ID, ComponentID),
   FOREIGN KEY (ID) REFERENCES Part(ID)
         ON DELETE RESTRICT
         ON UPDATE CASCADE,
   FOREIGN KEY (ComponentID) REFERENCES Part(ID)
         ON DELETE RESTRICT
         ON UPDATE CASCADE
) ENGINE=InnoDB;
```

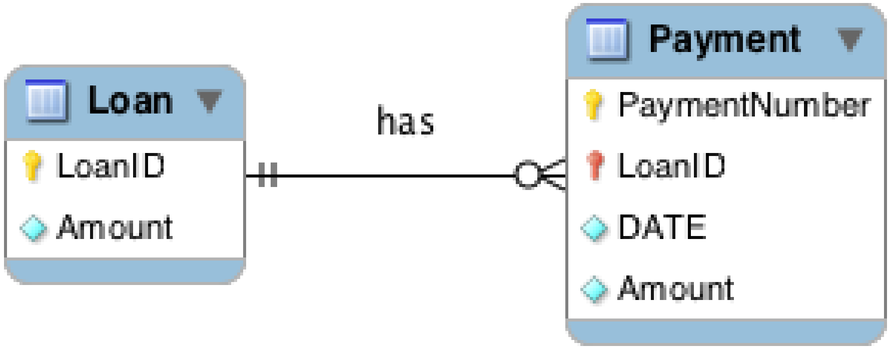
Ternary relationships: Many to Many

- Relationships between three entities
- **Logical Design**:
  - Generate an Associative Entity
  - Three One-to-Many relationships
  - Same rules then apply as One-to-Many



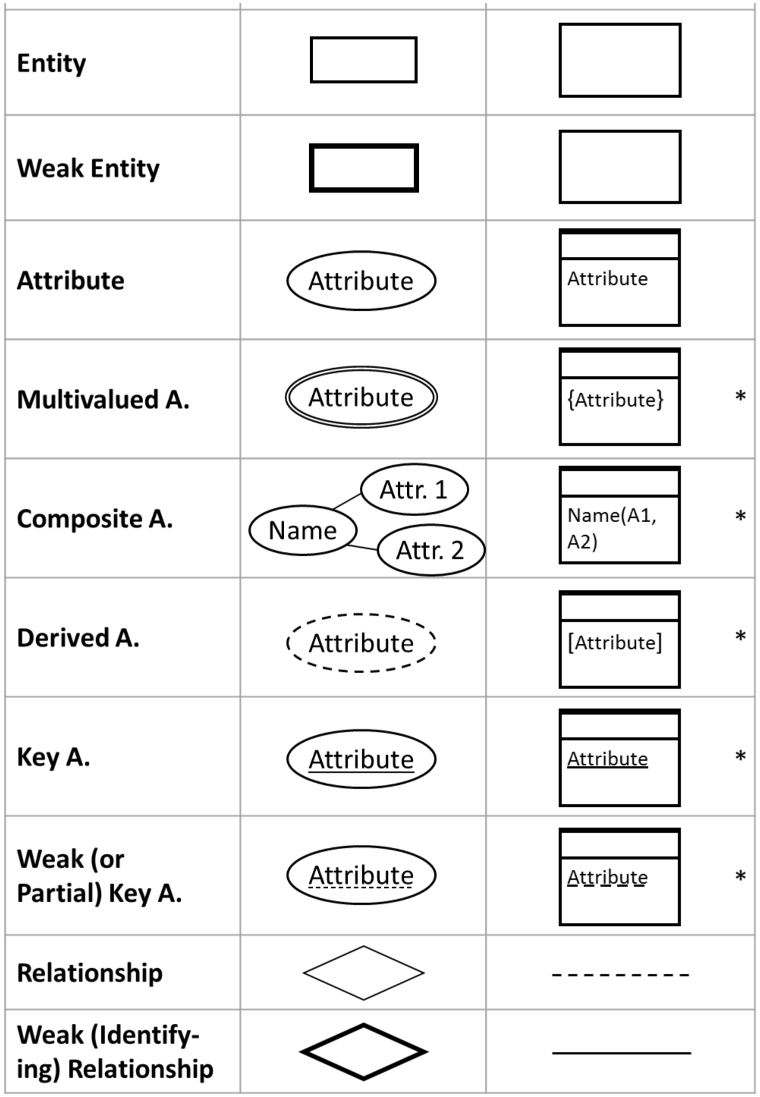
Strong and Weak Entity (Identifying Relationship)

- How to map an Identifying relationship
  - Map it the same way: Foreign Key goes into the relationship at the crow's foot end.
  - Only Difference is: The Foreign Key becomes **part of the Primary Key**



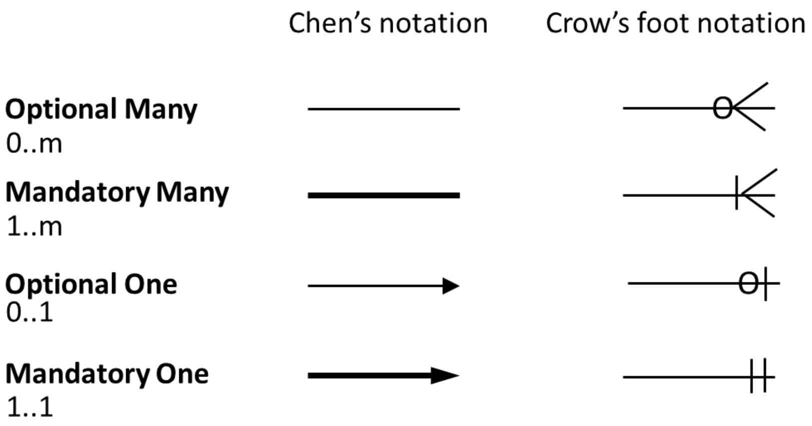
  - **Logical Design:**
    - Loan(LoanID, Amount)
    - Payment(PaymentNumber, **LoanID**, Date, Amount)
  - **Physical Design**: as per normal one-to-many

Conceptual Model Mapping (LMS)



Concept	Chen's not.	Crow's foot not.	
Entity			
Weak Entity			
Attribute	Attribute	Attribute	
Multivalued A.	Attribute	{Attribute}	*
Composite A.	Attr. 1 Name Attr. 2	Name(A1, A2)	*
Derived A.	Attribute	[Attribute]	*
Key A.	Attribute	Attribute	*
Weak (or Partial) Key A.	Attribute	Attribute	*
Relationship			
Weak (Identifying) Relationship			

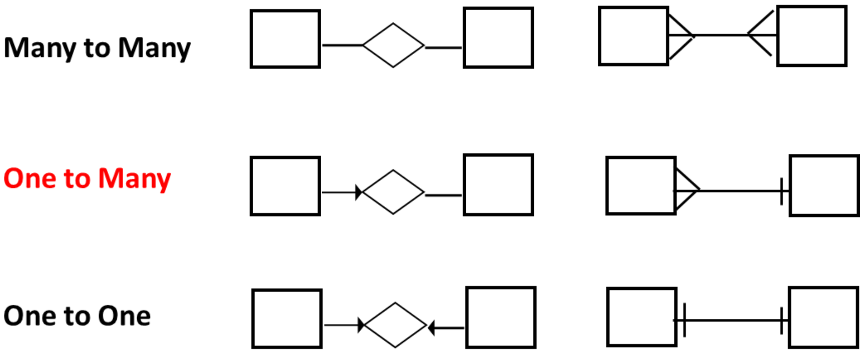
Relationship cardinalities and constraints



	Chen's notation	Crow's foot notation
Optional Many 0..m		
Mandatory Many 1..m		
Optional One 0..1		
Mandatory One 1..1		

BINARY Relationship Cardinalities

Here we just looked at cardinalities and omitted participation constraints (optional/mandatory) for clarity



Many to Many

One to Many

One to One

What's Examinable

- Need to be able to draw conceptual, logical and physical diagrams
    - Assignment 1: Conceptual Chen's pen and paper, Physical Crow's foot with MySQL Workbench
- Create table SQL statements

Next Lecture

- Hands on Modelling
- Please read the case study prior to the lecture:
  - LMS/Week 3 Medicare study