INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 05

Modelling Example with MySQL Workbench

Translating ER into Logical and Physical Model

Administration

- Workshop 14 running 15:15-17:15 on Tuesdays in PAR-Alice Hoy-101 STAYS!
  - Everyone voted for it to stay
  - But please bring your laptops, it's room without computers
- Workshop 20 running 11:00-13:00 on Wednesdays in PAR-Alice Hoy-211?
  - Interested?
- Feedback from student representatives
  - Thank you!
- We would love more feedback from you
  - Please bring a sheet of paper tomorrow and share your thoughts
  - I like/don't like, It would be good to do, Could you…

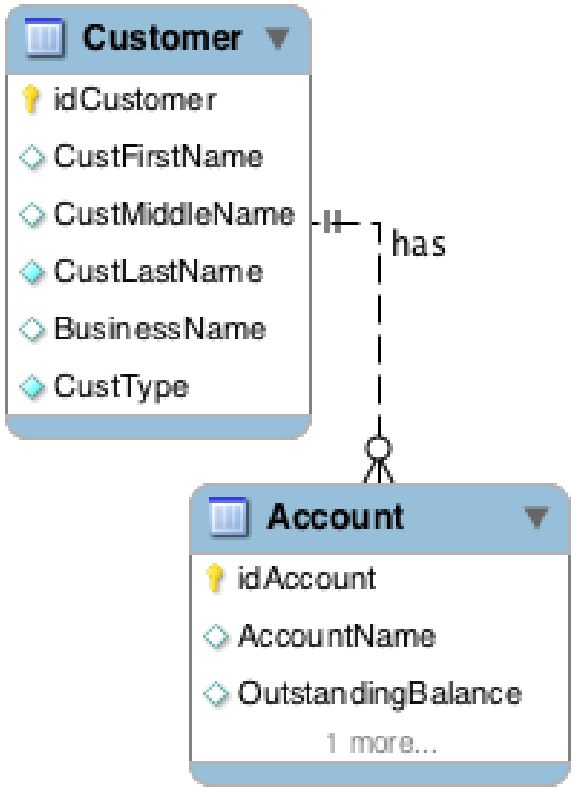
Coverage

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

Small Example Conceptual Model + Tables

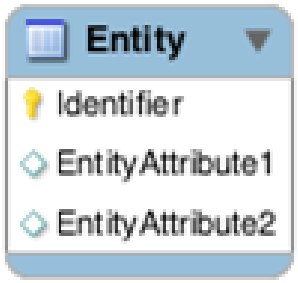
CustID	CustomerFirst Name	CustMiddle Name	CustLastName	BusinessName	CustType
029112	Peter		Smith		Personal
002301	James		Jones	JJ Enterprises	Company

Account ID	AccountName	Outstanding Balance
01	Peter Smith	245.25
05	JJ Ent.	552.39
06	JJ Ent. Mgr	10.25

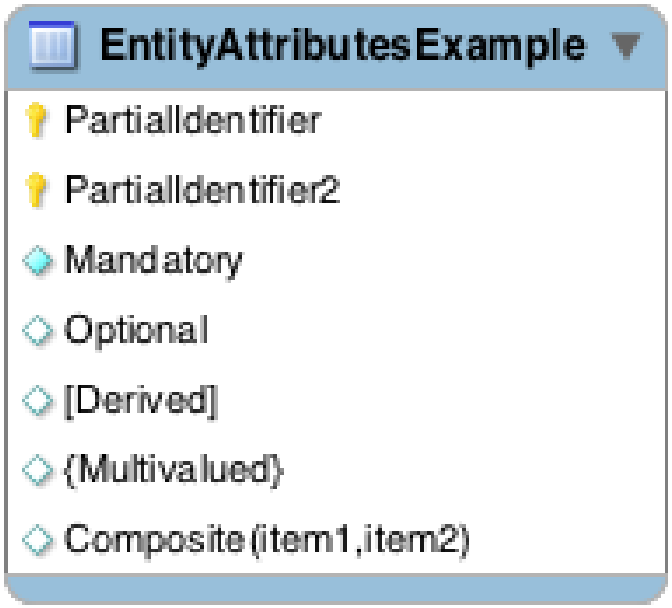


Conventions of ER Modelling (Workbench)

- Entity



- Attributes

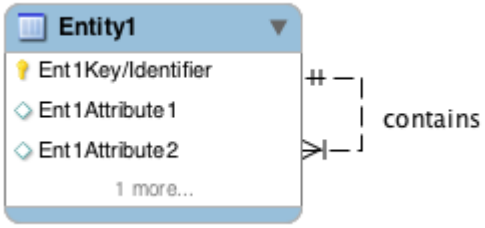


- Identifier or key
  - Fully identifies an instance
  - Partial Identifier
    - Identifies an instance in conjunction with one or more partial identifiers
- Attributes
  - Mandatory
  - Optional
  - Derived
    - [YearsEmployed]
  - Multivalued
    - {Skill}
  - Composite
    - Name (First, Middle, Last)

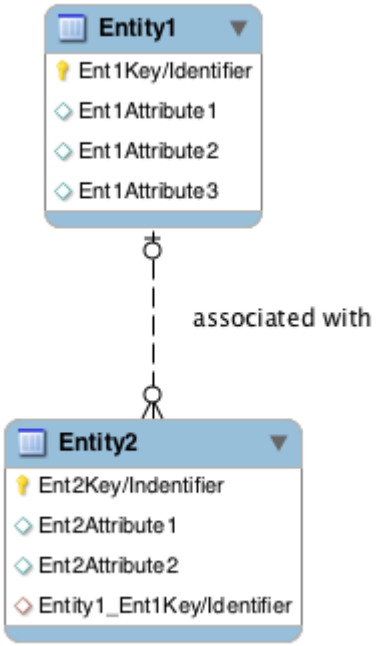
Conventions of ER Modelling (Workbench)

- Relationship Degrees

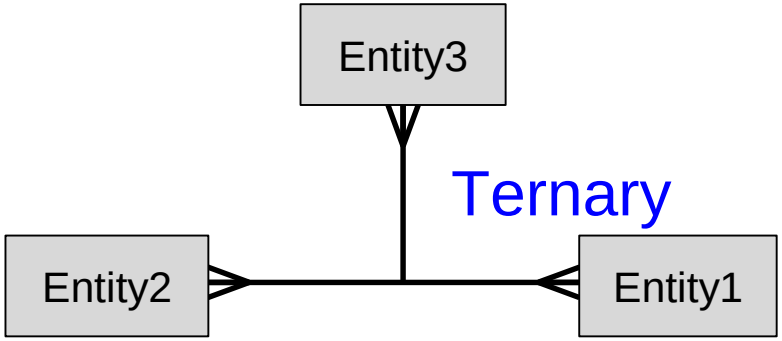
Unary

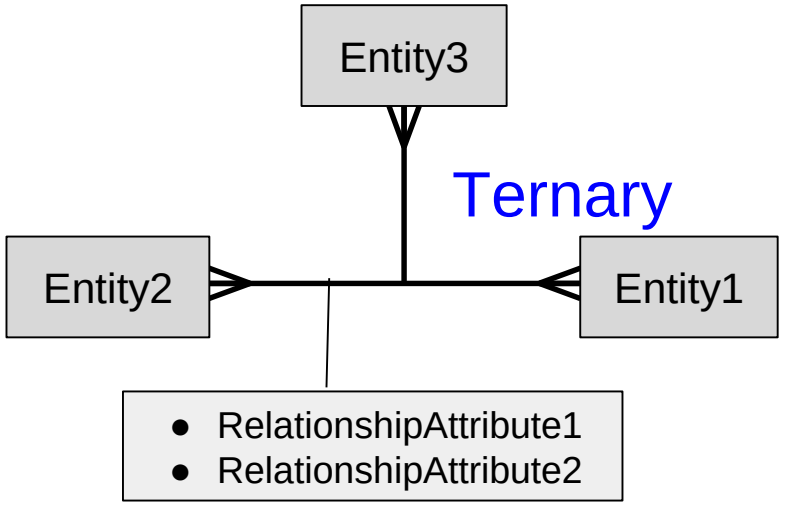


Binary



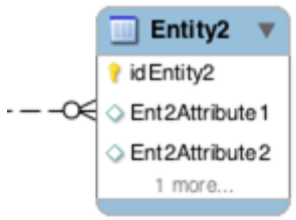
Ternary





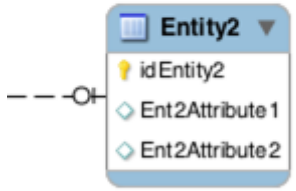
Conventions of ER Modelling (Workbench)

- Cardinality Constraints

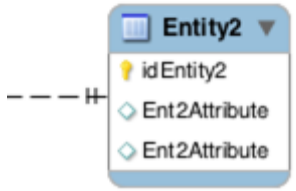


Optional Many
Partial participation
Without key constraint

Mandatory Many
Total participation
Without key constraint



Optional One
Partial participation
Key constraint



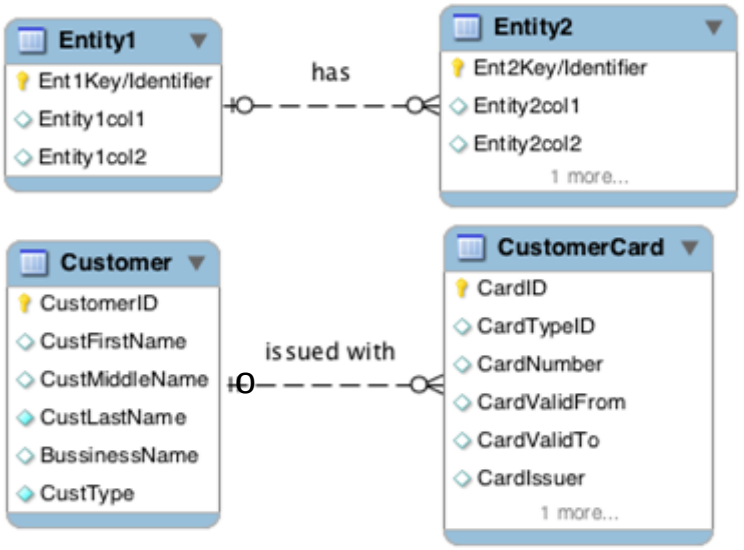
Mandatory One
Total participation
Key constraint

- 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)

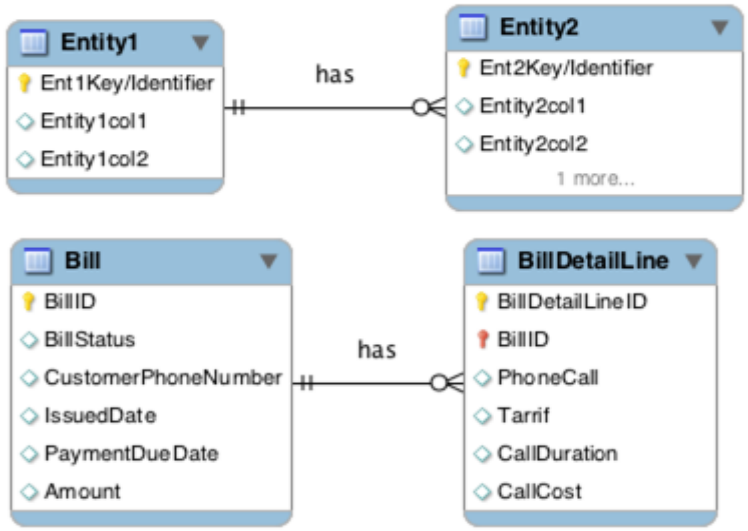
Strong Entity

- Entity 2
  - Can exist by itself

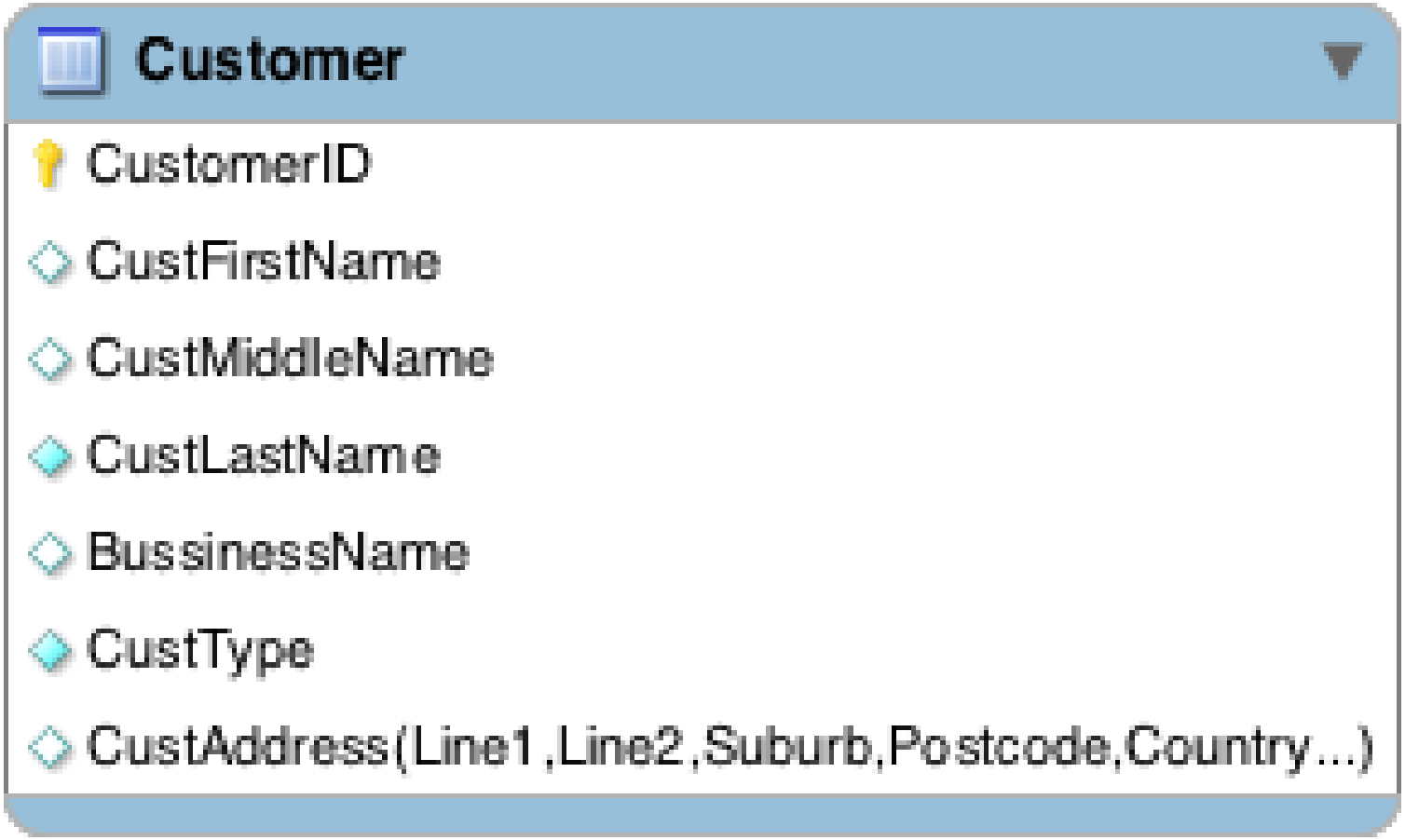


Weak Entity

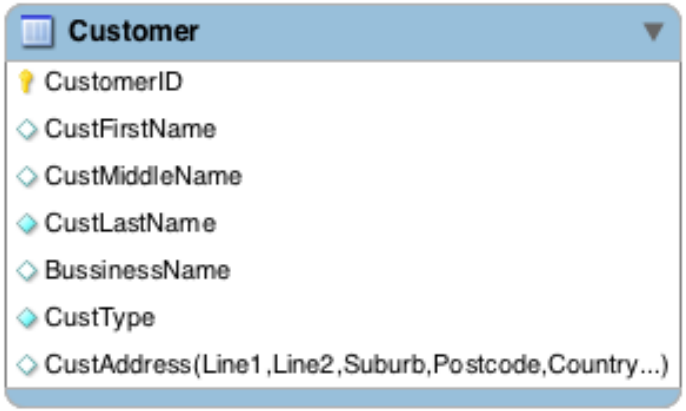
- Entity 2
  - Can't exist without Entity 1
  - Needs the FK as part of its composite PK



Single Entity (Conceptual)



Logical design – Single Entity
How to convert from conceptual design

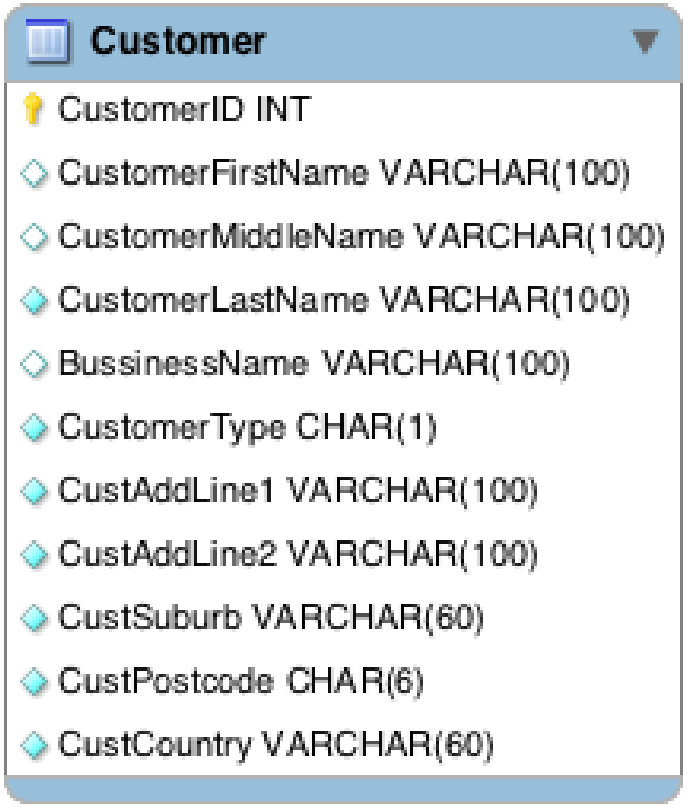


- Convert the ER into a logical model
  - Customer=(CustomerID, CustFirstName, CustMiddleName, CustLastName, BusinessName, CustType, CustAddLine1, CustAddLine2, CustSuburb, CustPostcode, CustCountry)

- Main issues
  - Convert composite and multi-valued attributes
    - Multi-Attribute values can become another table
    - Shown later
  - Resolve many-many relationships
    - Shown later
  - Add foreign keys at crows foot end of relationships
    - Shown later

Convert from Logical to Physical Design

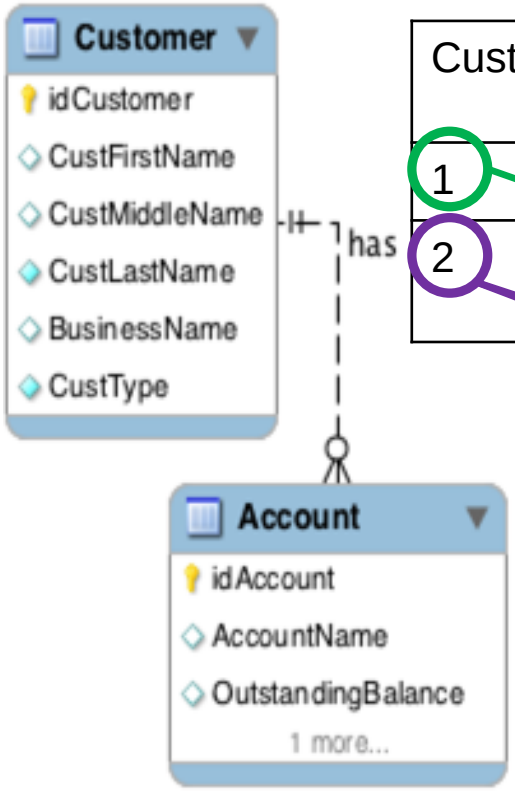
- Generate attribute data types




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

More than One Entity

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

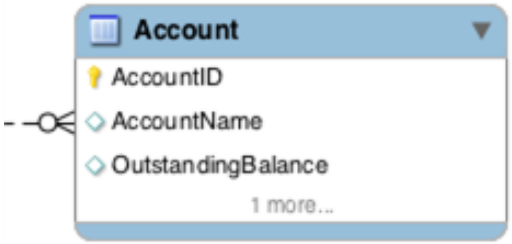


CustID	CustomerFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company

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

Logical Design - Account

Conceptual Design

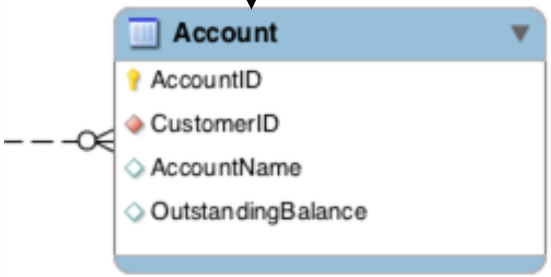


- Main issues
  - Convert composite and multi-valued attributes
    - None to worry about
  - Resolve many-many relationships
  - Add foreign keys at crows feet end of relationships
    - See FK1 – CustomerID
    - This is the link to the customer table
    - Every row in the account table must have a CustomerID from Customer
      - Referential integrity

Logical Design

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

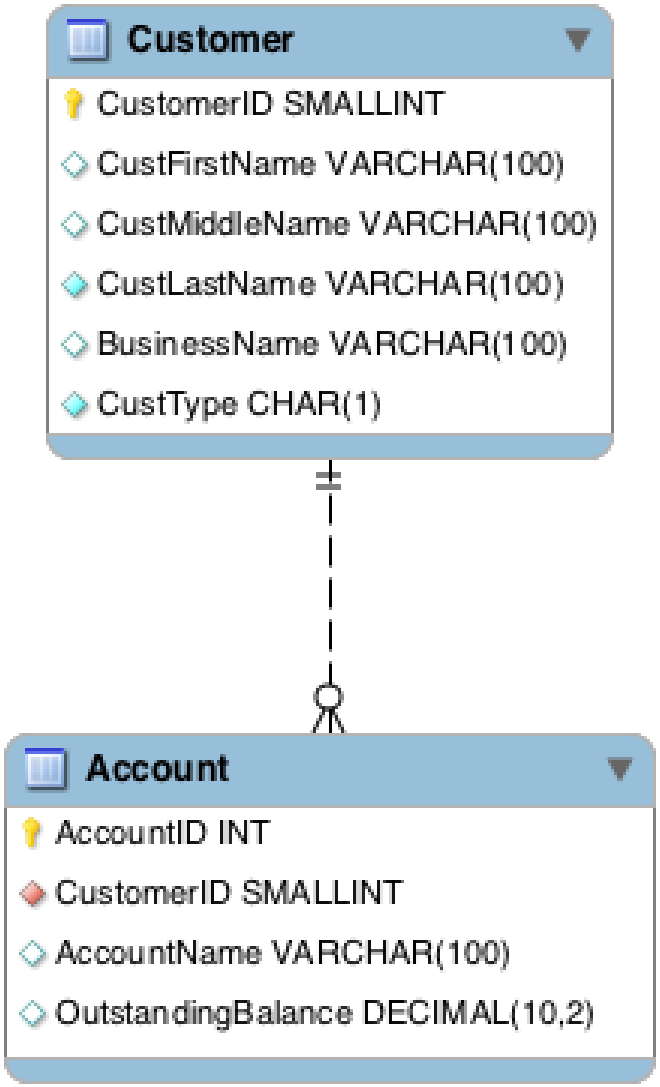
Note: Underline = PK, italic and underline = FK, underline and bold = PFK



Physical Design - Account

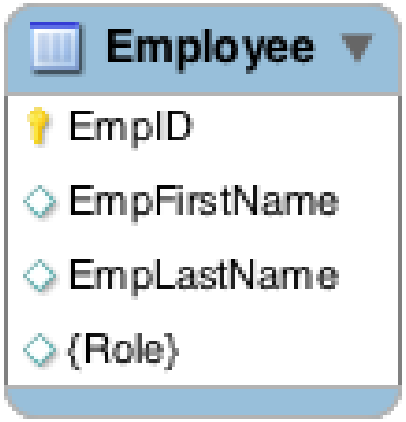
- Attribute data types

```
CREATE TABLE Account (
   AccountID              smallint         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;
```

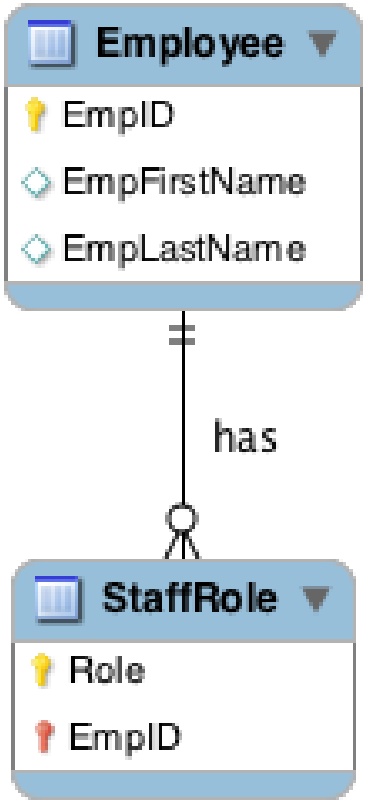


Dealing with Multi-Valued Attributes

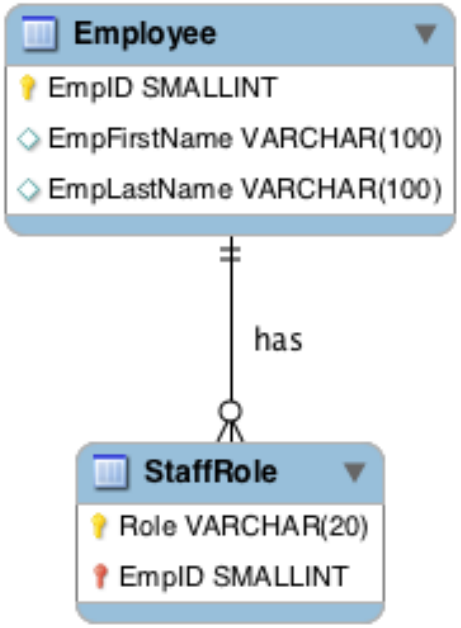
Conceptual Design



Logical Design



Physical Design

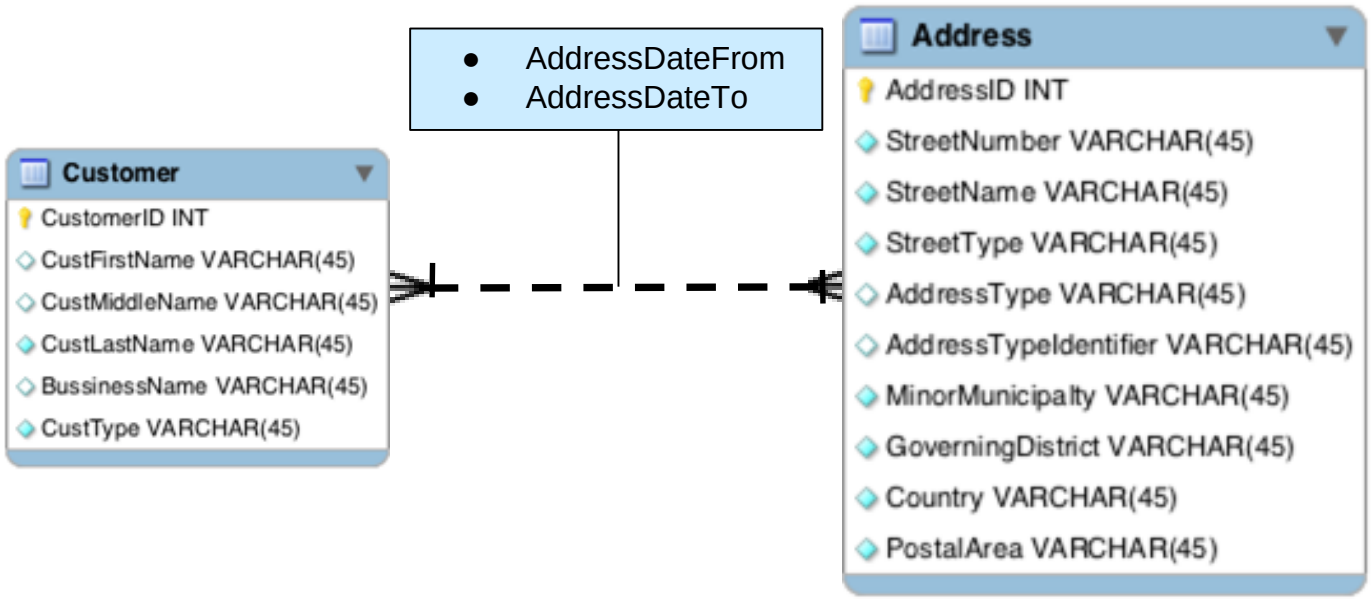


- StaffRole is an example of a weak entity
  - We show this with a solid line in Workbench

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

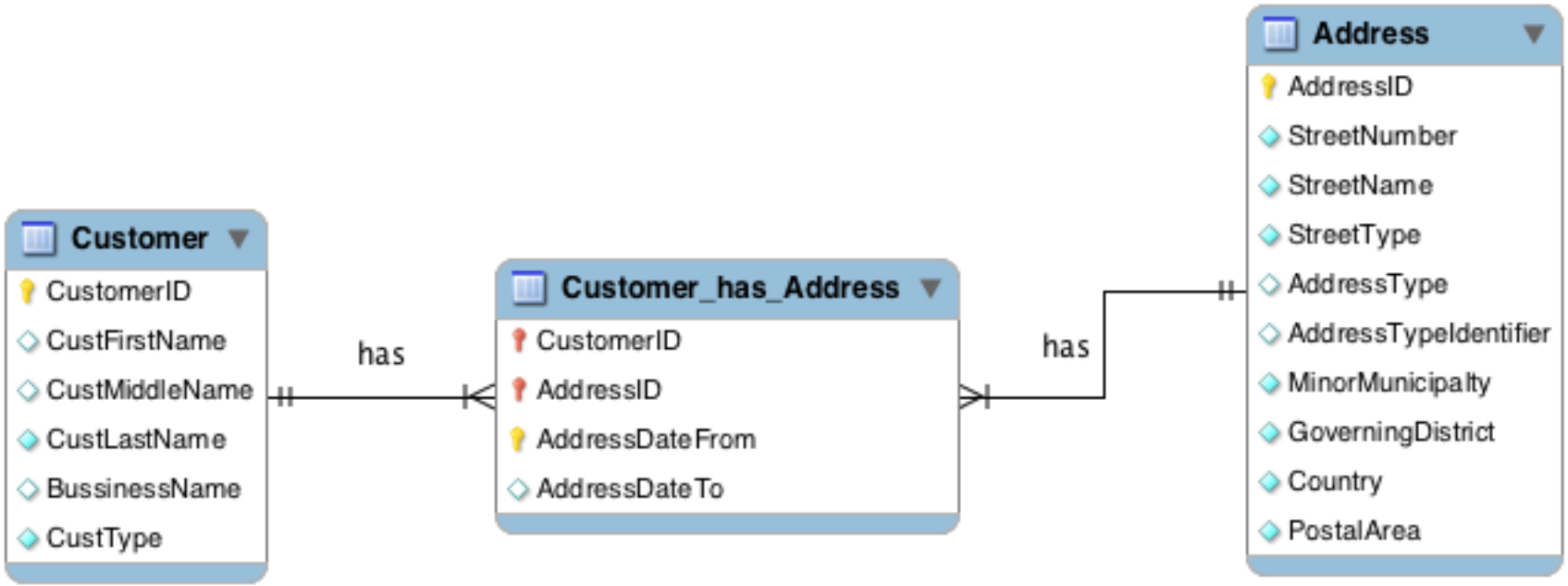
Many to Many Relationship

- How to deal with customer addresses…
  - The fact is that customers change addresses
    - AND we probably 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 the logical diagram we create an **Associative Entity** between the other 2 entities

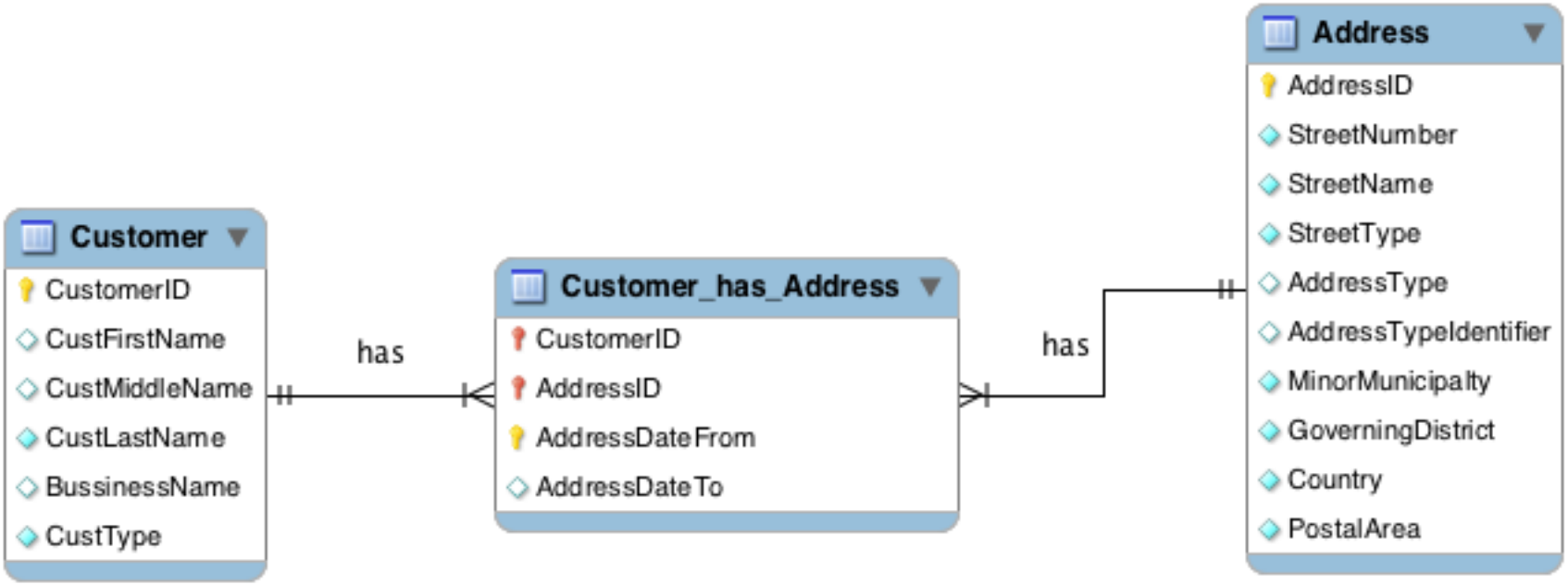


  - AddressDateTo will be NULL if an address is the current address.

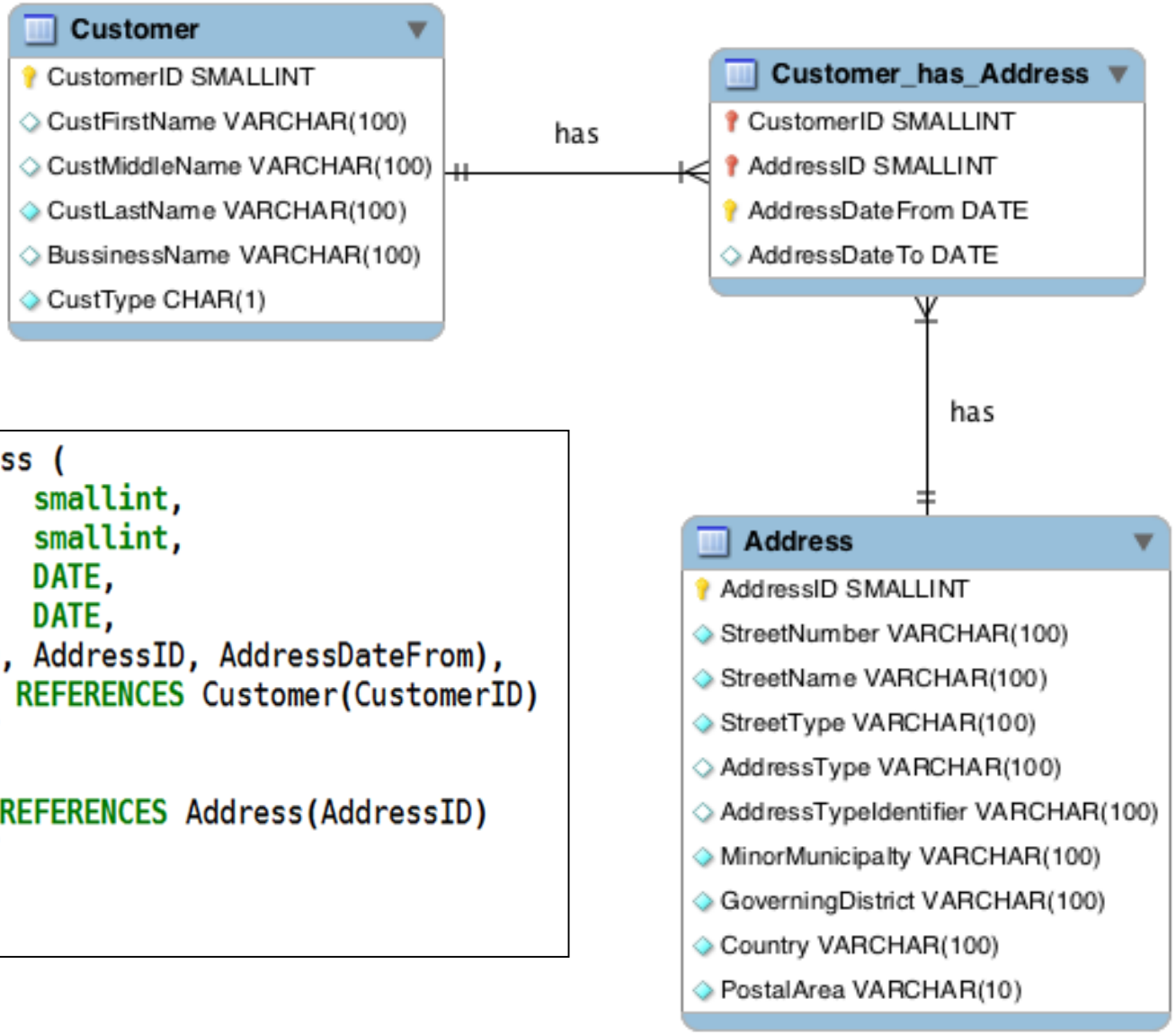
Many to Many - Logical Model

- 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



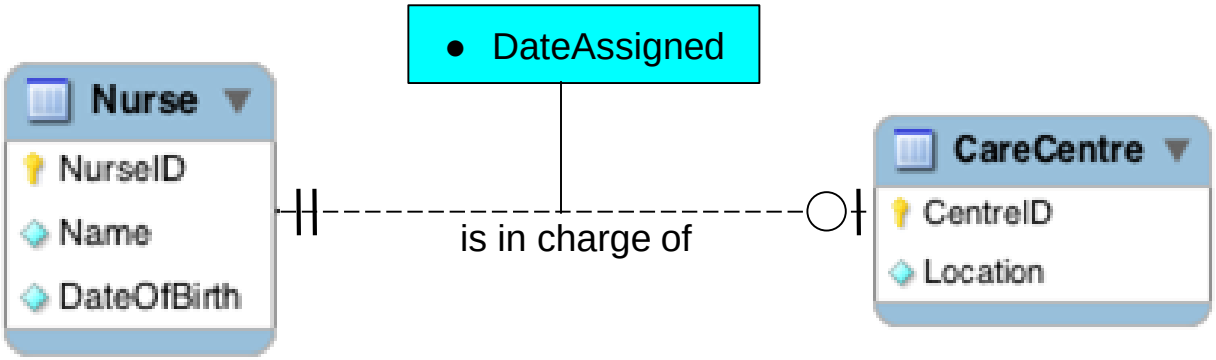
Many to Many - Physical Model




```
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

- Given this example… How do we implement it…

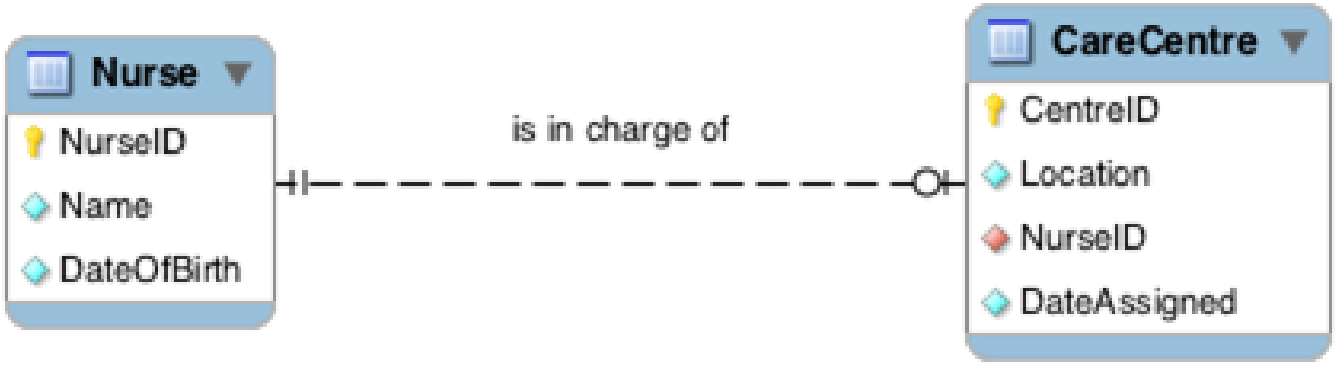


- Note: Date_assigned is a descriptive attribute of the relationship
  - They go into the associative entity for M-M

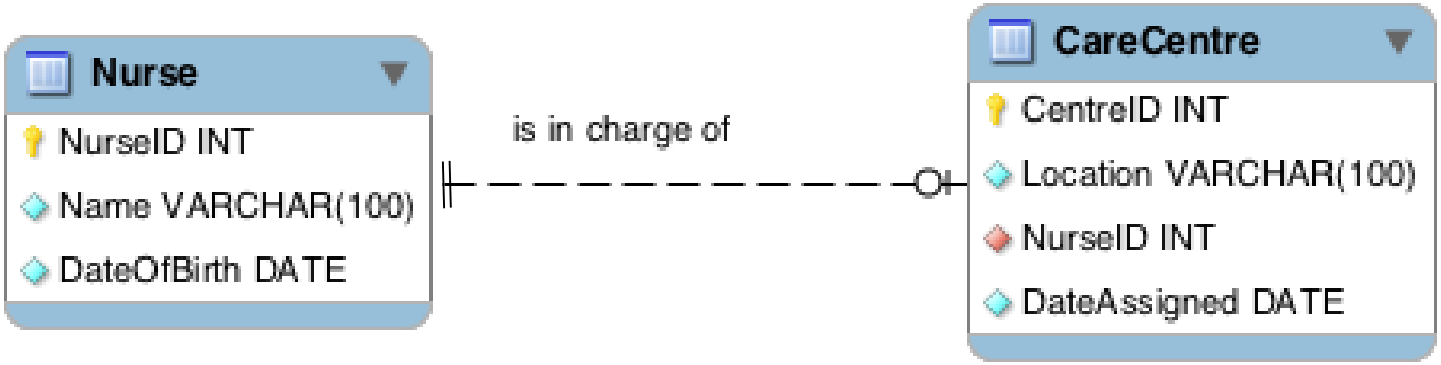
- Need to decide whether to put the foreign key inside Nurse or CareCentre (in which case you would have the Date_Assigned in the same location)
  - Where would the least NULL values be?
  - The rule is the OPTIONAL side of the relationship gets the foreign key

Binary One-One Relationship – Logical and Physical Design

- **Logical**
  - Nurse = (NurseID, Name, DateOfBirth)
  - CareCentre = (CentreID, Loction, *NurseID*, DateAssigned)



- **Physical**

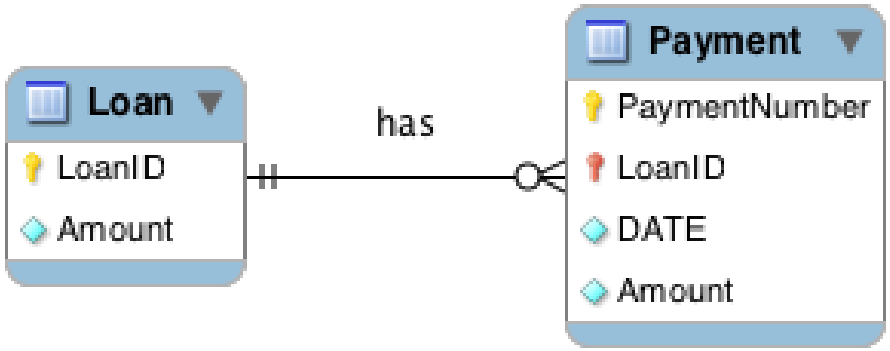


Summary of Binary Relationships

- One-to-Many
  - Primary key on the one side becomes a foreign key on the many side
- Many-to-Many
  - Create an Associative Entity (a new relation) with the primary keys of the two entities it relates as the combined primary key
  - Then treat it like a One-to-Many (between Assoc. E. and original)
- 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

Strong and Weak Entity- Identifying Relationship

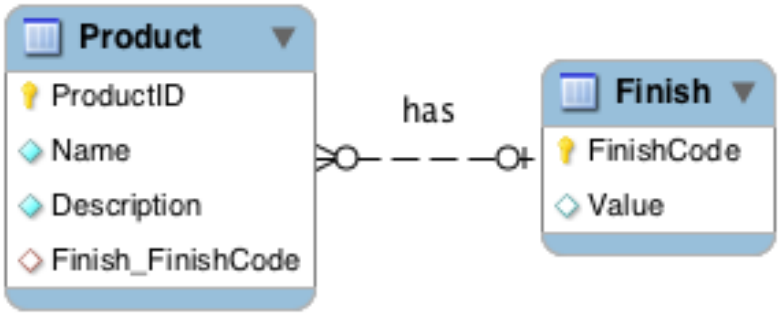
- How to map an Identifying relationship
  - Map it the same way – Foreign Key goes into the relationship at the crow's foot end.
  - Only Difference is… The Foreign Key becomes **part of the Primary Key**



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

"Code" Lookup

- Consider the following logical design



ProductID	Name	…	Finish
1	Chair		A
2	Desk		C
3	Table		B
4	Book		A

Code	Value
A	Birch
B	Maple
C	Oak

- Physical design decision
  - Implement as 2 tables or one. Remember the trade-offs (space vs speed)?

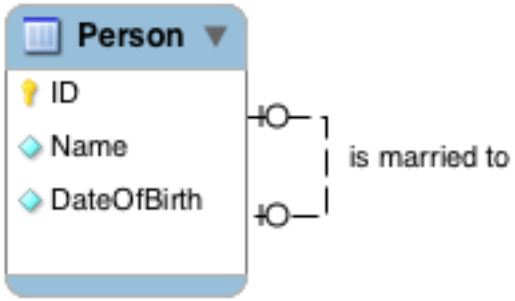
ProductID	Name	…	Finish
1	Chair		Birch
2	Desk		Oak
3	Table		Maple
4	Bookcase		Birch

Unary Relationships

- Operate in the same way exactly 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 different names for the Foreign keys of course
      - Both Foreign keys become the combined key of the Associative Entity

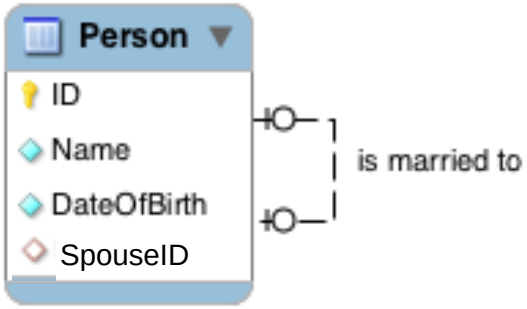
Unary – One-to-One

Conceptual Design



Logical Design

- Person = (ID, Name, DateOfBirth, *SpouseID*)



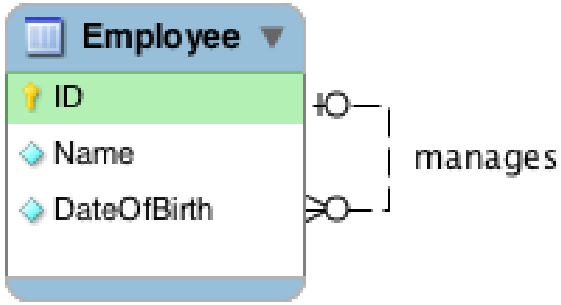
Physical Design

```
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

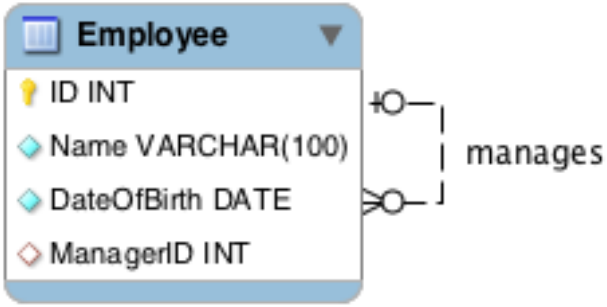
Unary – One-to-Many

Conceptual Design



Logical Design

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

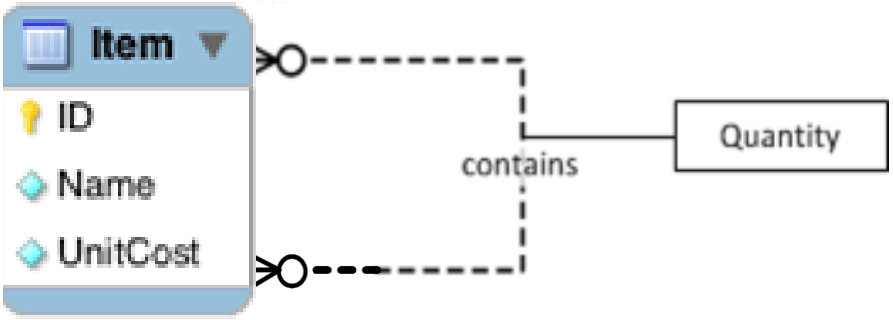


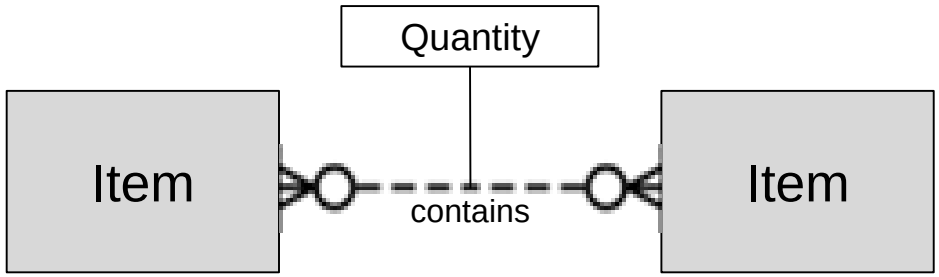
Physical Design

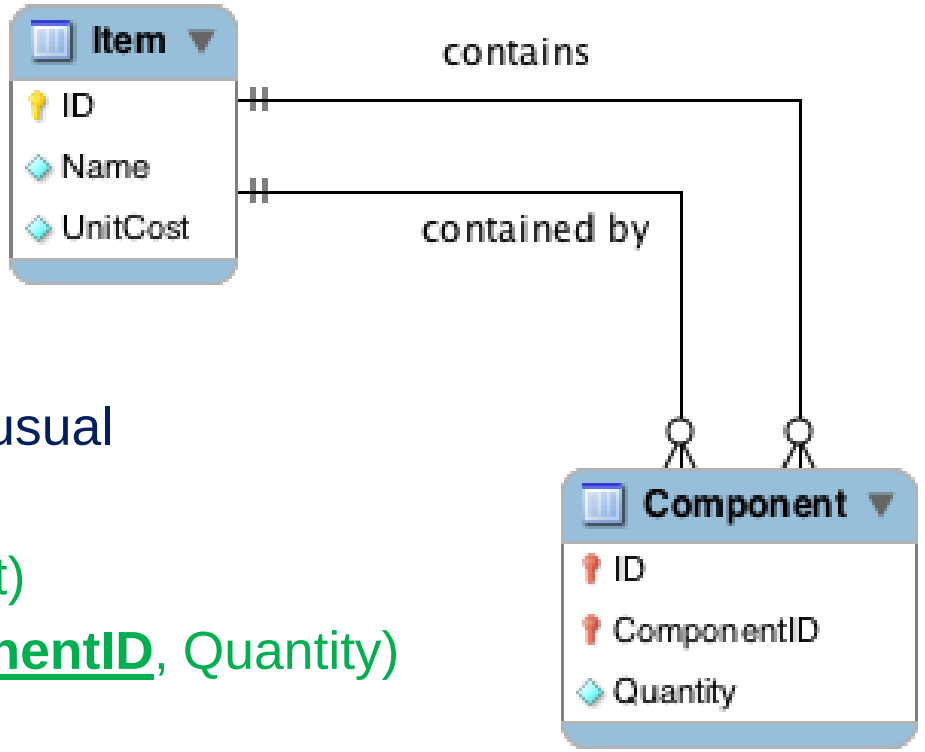
```
CREATE TABLE Employee(
  ID smallint NOT NULL,
  Name VARCHAR(100) NOT NULL,
  DateOfBirth DATE NOT NULL,
  ManagerID smallint ,
  PRIMARY KEY (ID),
  FOREIGN KEY (ManagerID )
  REFERENCES Employee(ID)
  ON DELETE RESTRICT
  ON UPDATE CASCADE);
```

ID	Name	DOB	MngrID
1	Ann	1969-06-12	
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)

Unary – Many-to-Many

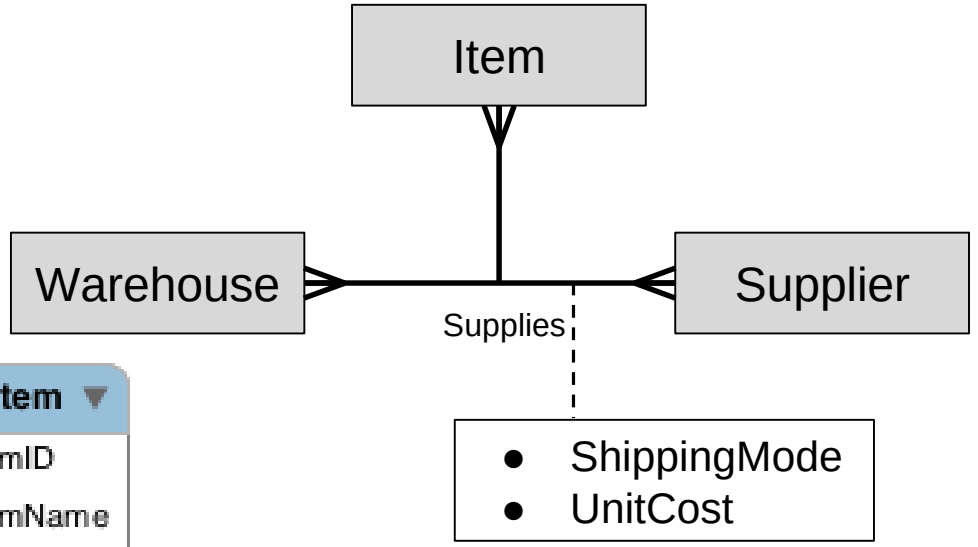
- Implementation

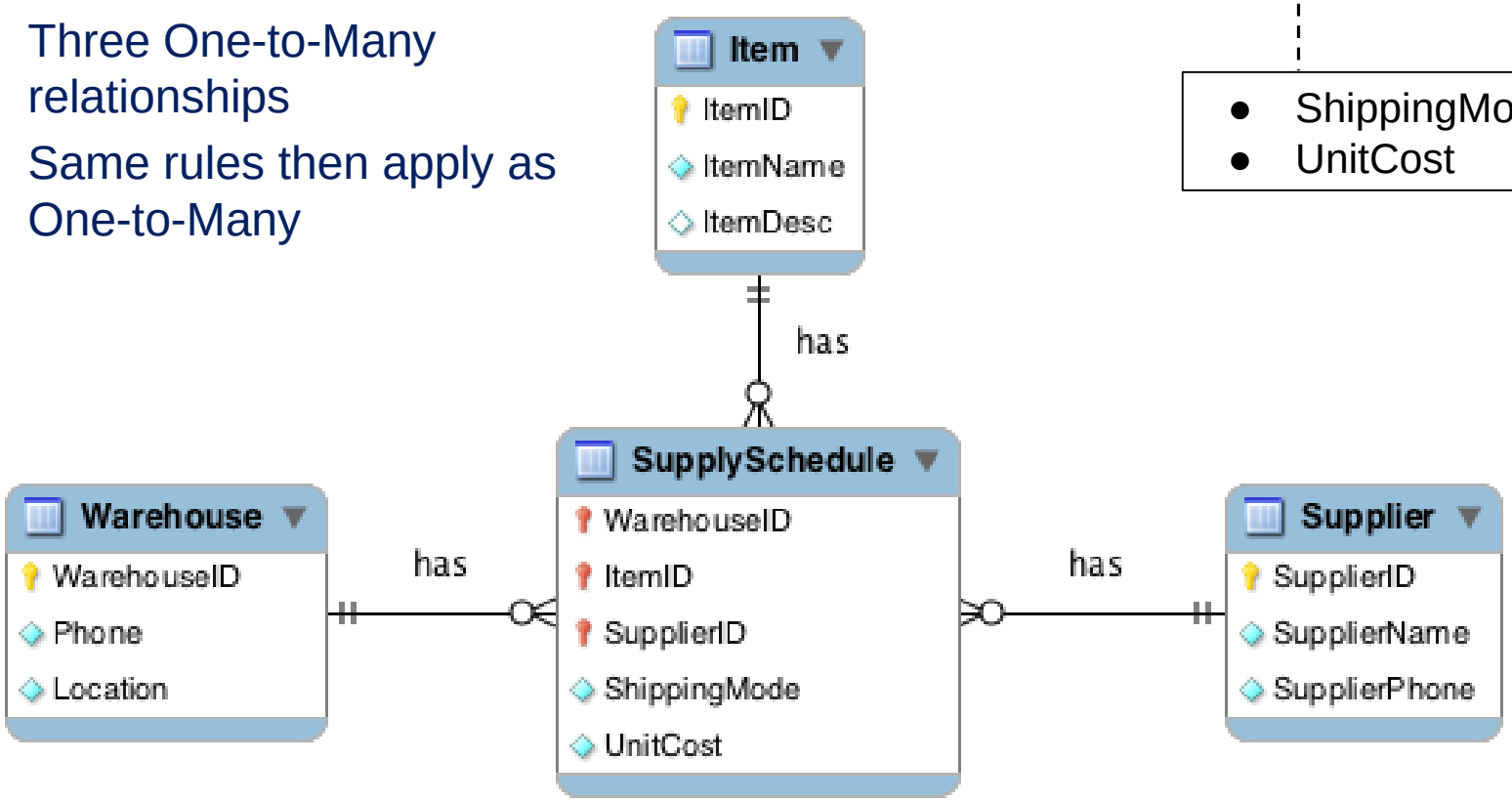
```
CREATE TABLE Part (
    ID                  smallint,
    Name                VARCHAR(100)    NOT NULL,
    UnitCost            DECIMAL(6,2)    NOT NULL,
    PRIMARY KEY  (ID)
) ENGINE=InnoDB;
```

```
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

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





What's Examinable

- Need to be able to draw conceptual, logical and physical diagrams
  - For conceptual both Chen's and Crow's foot notations are acceptable
- Create table SQL statements
- This is a good preparation for Assessment 1!

Next Lecture

- Extended ER modelling

LGBTQ+ In Engineering & IT: Diversity Discussion Panel

A discussion panel of academics and industry professionals sharing experiences about Queer inclusion in the engineering & IT industry

A great opportunity for **everyone** to join in the conversation, to help create a culture that is inclusive of all people.

WHEN: Tuesday 22nd August, 4:00 – 5:00
WHERE: Old Engineering, A1 Theatre

Followed by food and drinks

All are welcome

Find the event on Facebook at https://www.facebook.com/events/1202777319317 84

THE UNIVERSITY OF MELBOURNE

MELBOURNE SCHOOL OF ENGINEERING