INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 06
Unary, Ternary Relationships &
Enhanced ER Modelling

Administration

- Feedback
  - –Please share what you like/dislike about the course
  - –What can be improved and how

- Assignment 1
  - –Will be online on Wednesday 16/08/17 at 10am
  - –Due date Friday 01/09/17 at 10am
  - –Submit:

  1. Conceptual design (pen and paper) – scanned/photo legible!
  2. Physical model - Workbench file (mwb)
  3. **Physical model - PDF (most important)**

Coverage

- Unary and Ternary Relationships

- Enhanced ER Modelling
  - Specialisation / Generalisation
  - Inheritance
  - Constraints on Supertype/Subtype relationships

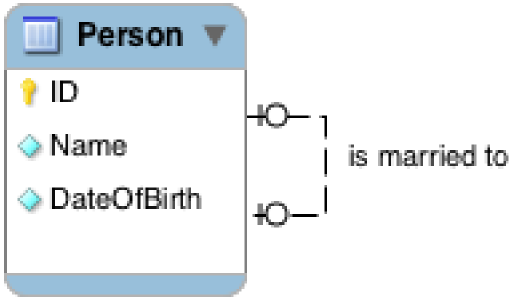
- From Conceptual Design through to Implementation

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 PK key of the Associative Entity

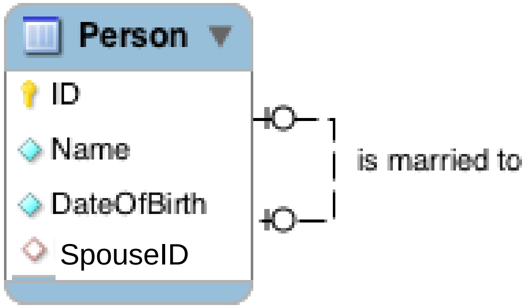
Unary – One-to-One

Conceptual Design



Logical Design

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



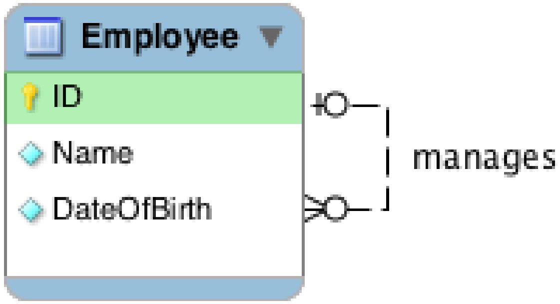
Physical Design

```
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

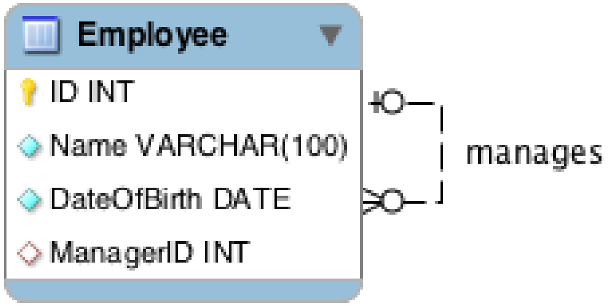
Unary – One-to-Many

Conceptual Design



Logical Design

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

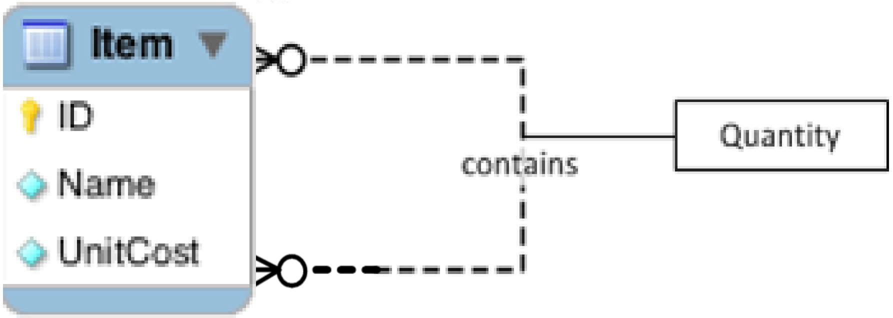


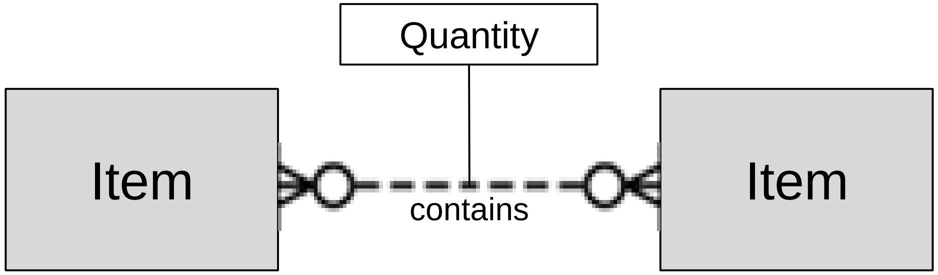
Physical Design

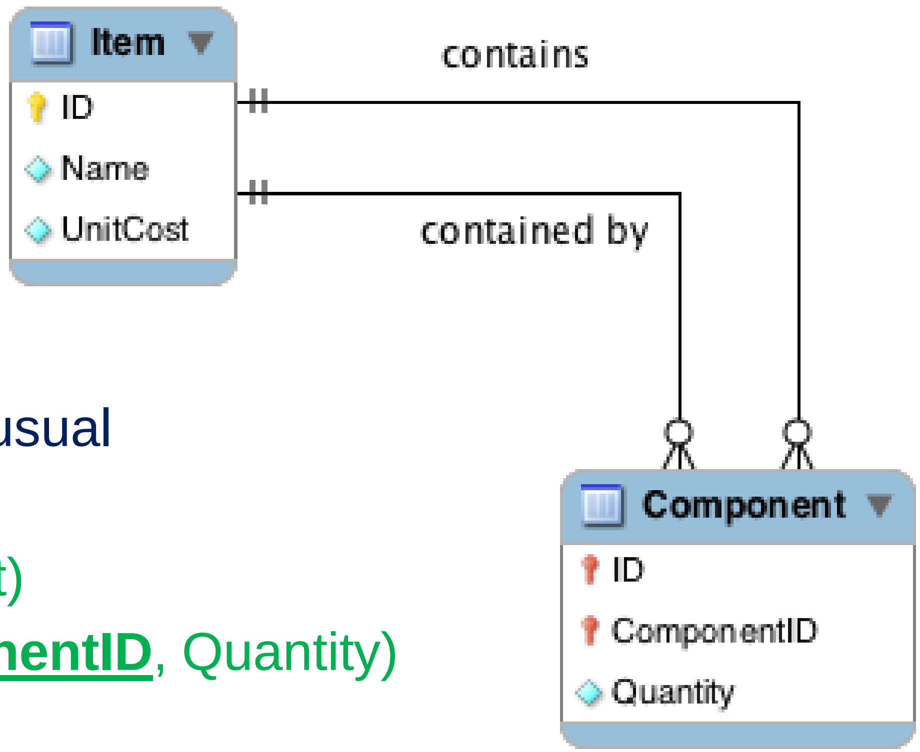
```
CREATE TABLE Employee(
  ID smallint NOT NULL,
  Name VARCHAR(100) NOT NULL,
  DateOfBirth DATE NOT NULL,
  ManagerID smallint ,
  PRIMARY KEY (ID),
  FOREIGN KEY (ManagerID )
  REFERENCES Employee(ID)
  ON DELETE RESTRICT
  ON UPDATE CASCADE);
```

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

Unary – Many-to-Many







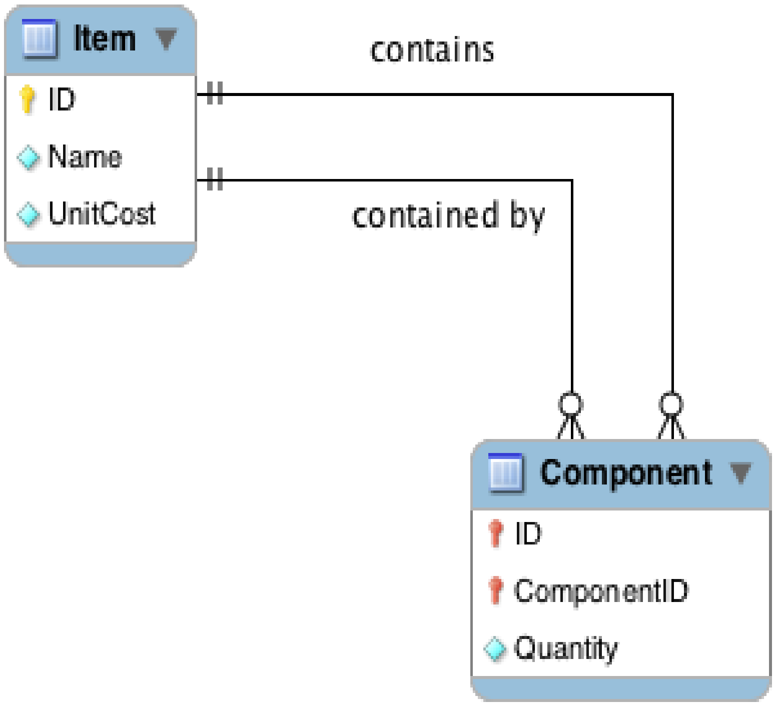
- Logical Design
  - Create Associative Entity like usual
  - Generate logical model
    - Item = (<u>ID</u>, Name, UnitCost)
    - Component = (**<u>ID, ComponentID</u>**, Quantity)

Unary – Many-to-Many

- Physical

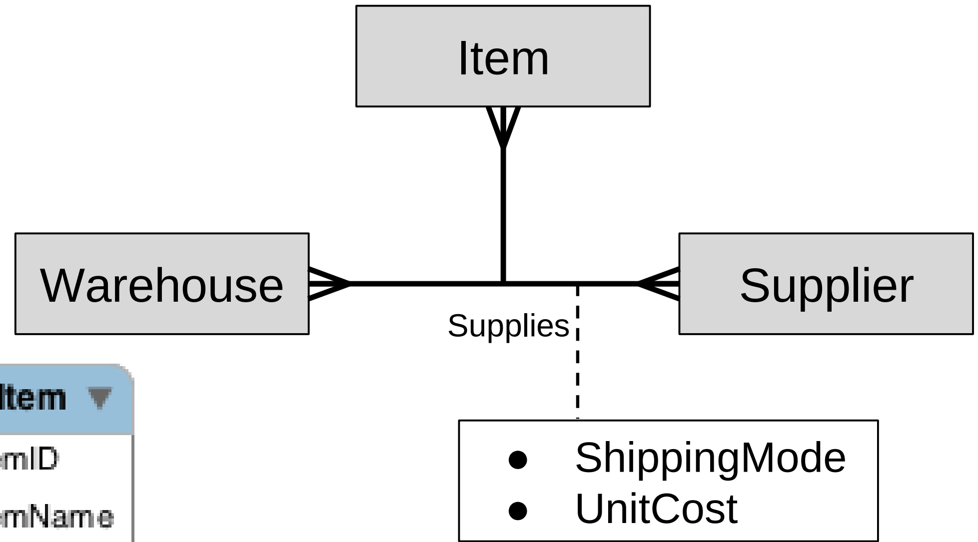
```sql
CREATE TABLE Part (
  ID                smallint,
  Name              VARCHAR(100)     NOT NULL,
  UnitCost          DECIMAL(6,2)     NOT NULL,
  PRIMARY KEY  (ID)
) ENGINE=InnoDB;
```

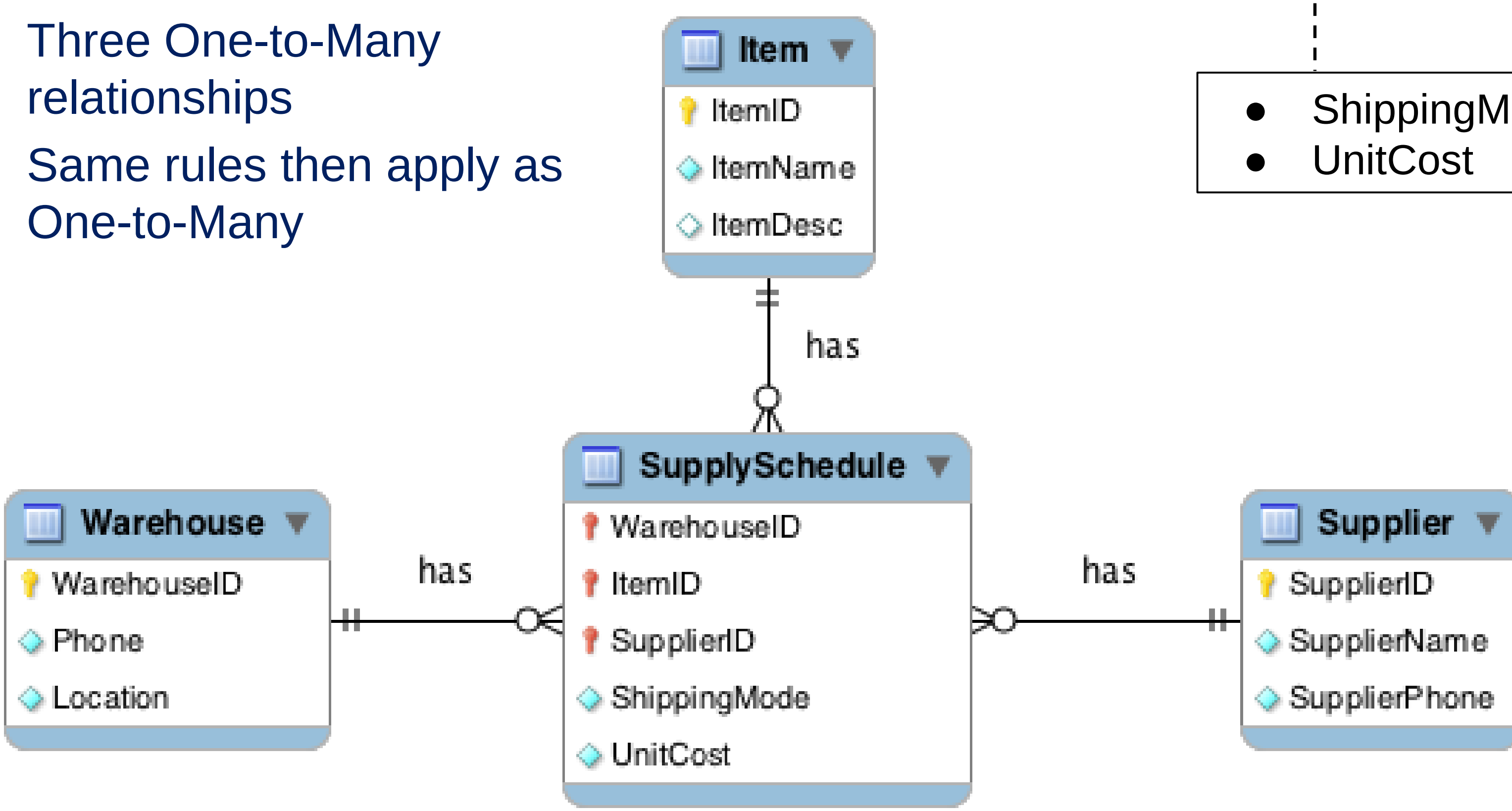
```sql
CREATE TABLE Component (
  ID                smallint,
  ComponentID       smallint,
  Quantity          smallint    NOT NULL,
  PRIMARY KEY  (ID, ComponentID),
  FOREIGN KEY (ID) REFERENCES Part(ID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
  FOREIGN KEY (ComponentID) REFERENCES Part(ID)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





Why EER?

- ER can not adequately capture complex business models
- Enhanced ER, extends functionality of ER models
  - In particular
    - Can capture supertype / subtype relationships
      - Discussed in the lecture today
    - Allows aggregation of entities
    - Allows capture of business rules that control behaviour

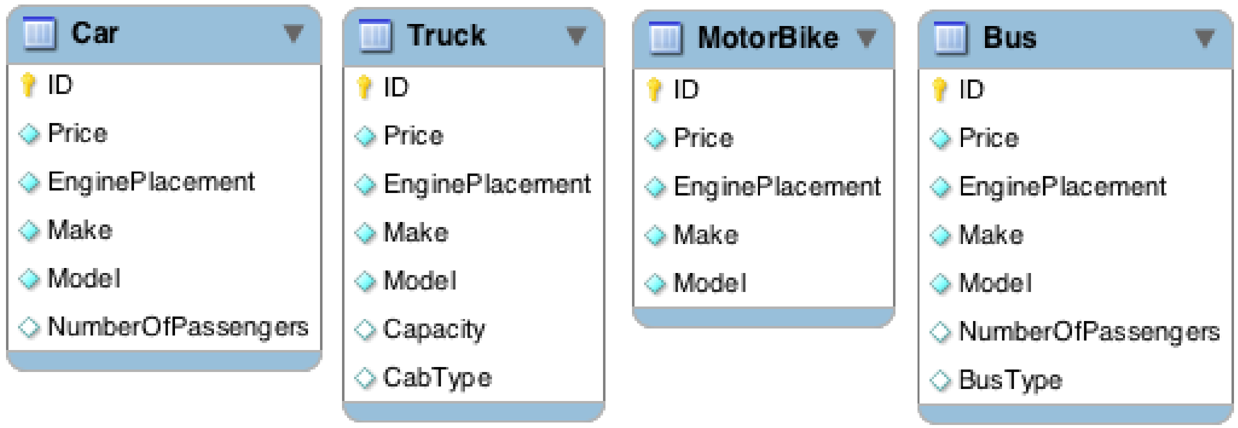
Consider the Following Scenario

- A vehicle selling organisation sells vehicles. When selling cars the organisation must record the price, engine displacement, car make, car model, and number of passengers. When selling trucks the organisation must record the price, engine displacement, truck make, truck model, capacity and cab_type. When selling motorbikes the price. engine displacement, bike make and bike model. When selling busses they must know the price, engine displacement, bus make, bus model, bus type and number of passengers.

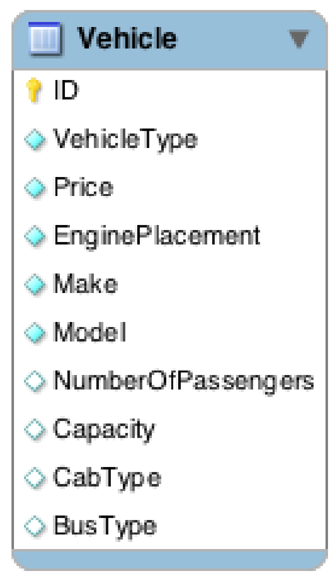
THE UNIVERSITY OF MELBOURNE

Possible Entities

- Solution 1:



- Solution 2:



An Alternate Solution - EER

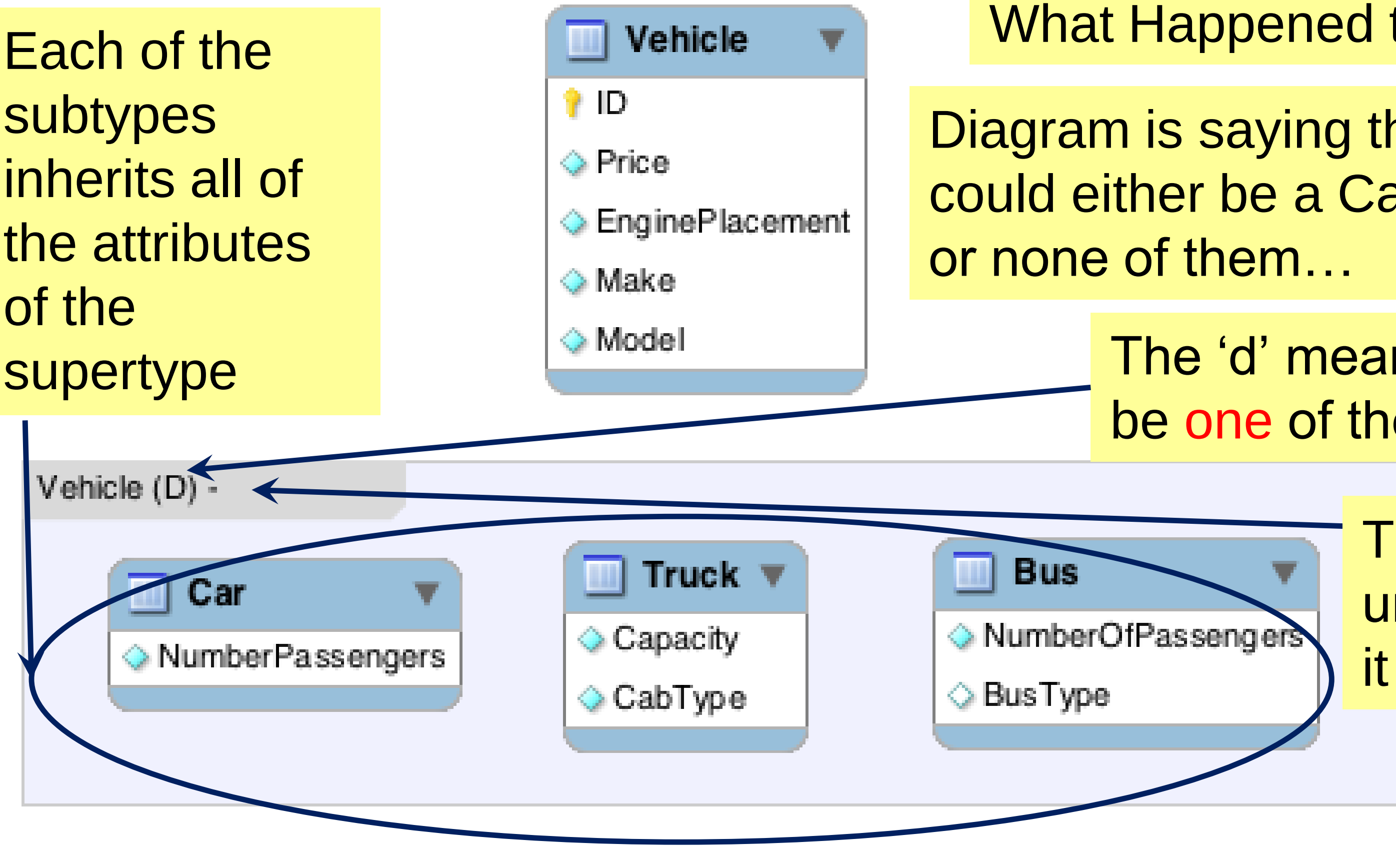
Each of the subtypes inherits all of the attributes of the supertype

What Happened to the Motorbike?

Diagram is saying that a Vehicle could either be a Car, a Truck, a Bus, or none of them…

The 'd' means disjoint (ie can be one of these…)

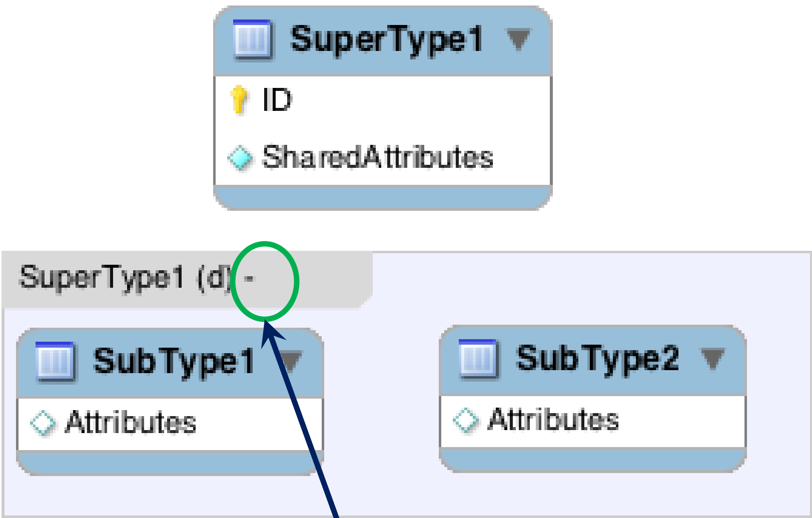
The single line under the 'd' says it can be none

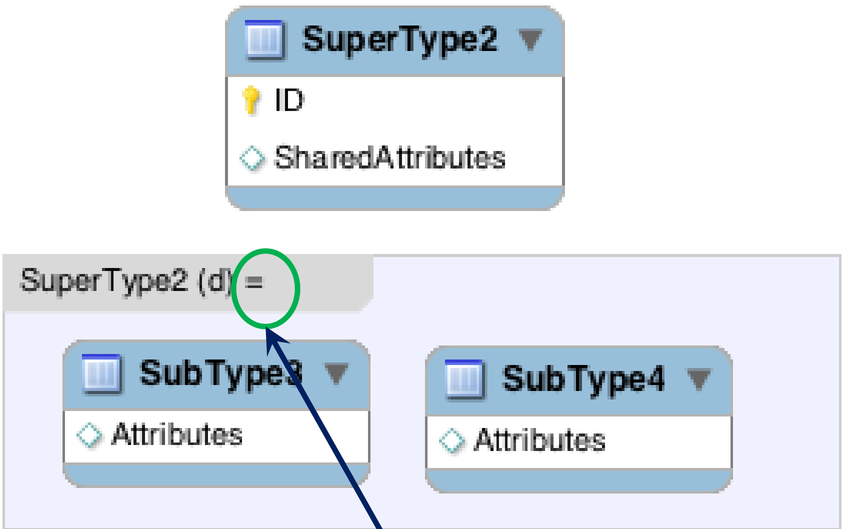


- This is known as a supertype/subtype hierarchy or generalisation/specialisation hierarchy
- Each subtype has properties that are distinct from the others
- Each subtype inherits the properties of the supertype

Constraints in Super/Subtypes (Workbench)

- Completeness Constraints
  - Specifies whether an instance of a supertype must also be an instance of at least one subtype



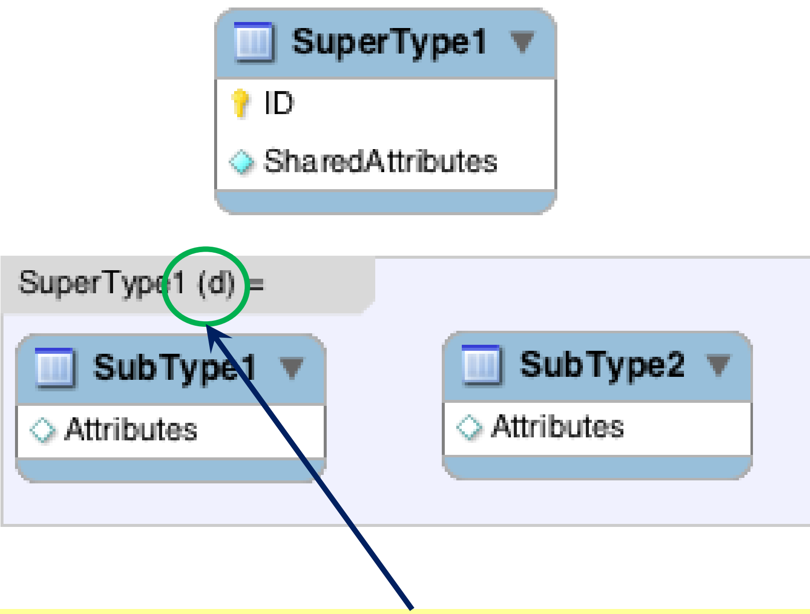


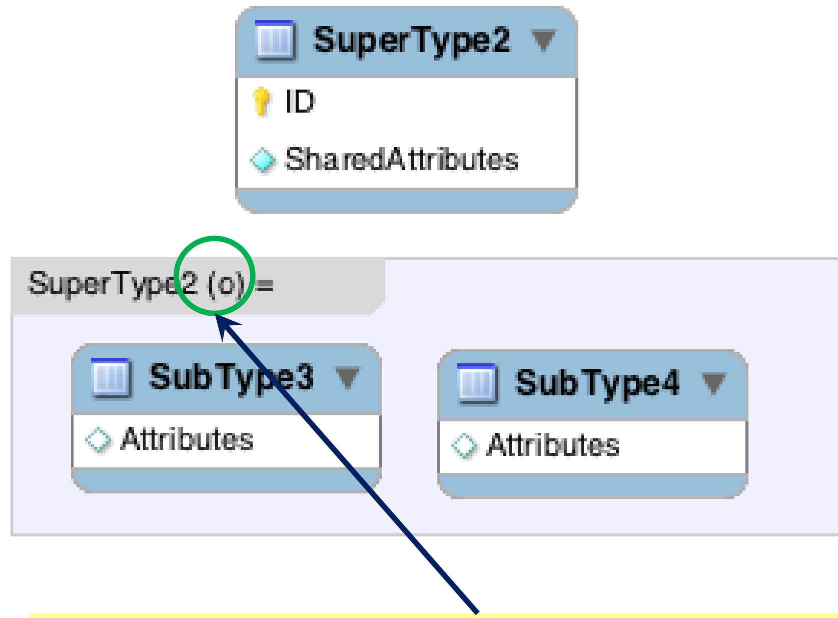
Single Line: the entity of type Supertype1 can be either Subtype1 or Subtype2 or neither

Double line: the entity of type Supertype2 MUST be either Subtype3 or Subtype4

Constraints in Super/Subtypes (Workbench)

- Disjointness Constraints
  - Specifies whether an instance of a supertype may simultaneously be a member of two (or more) subtypes



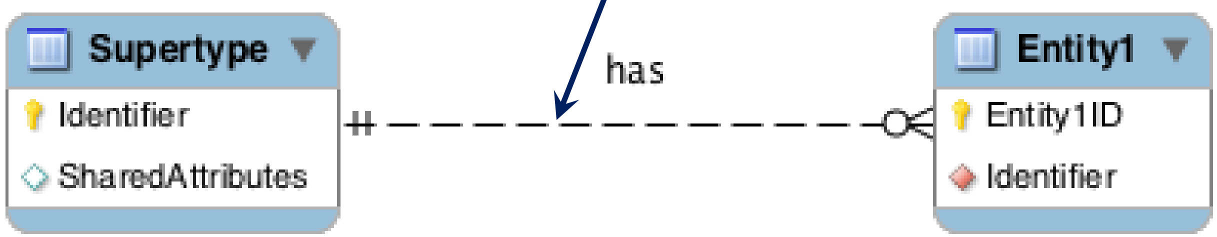


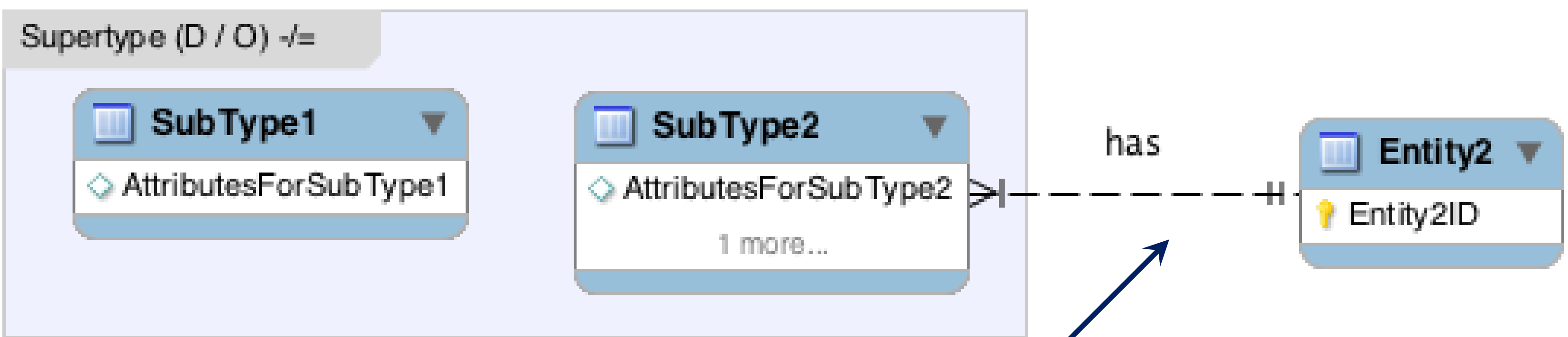
‘d’ = disjoint (can be one of these), and because of the double must be one of them

‘o’ = overlapping (can be more than one of these), and because of the double line must be at least one of them

General Form

Every instance of the entities are involved with this relationship (doesn't matter if it is a subtype or supertype)



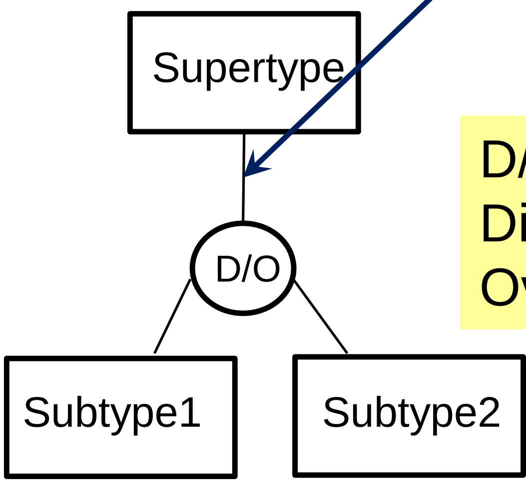


Only instances of Subtype2 are involved with this relationship

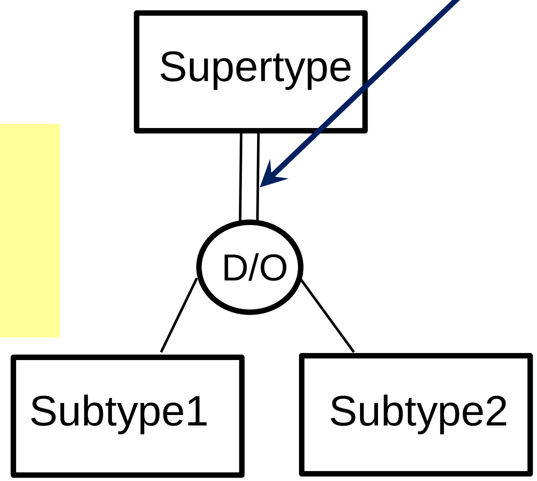
Supertype/subtype with Chen's notation

Single Line: the entity of type Supertype can be either Subtype1 or Subtype2 or neither

Double line: the entity of type Supertype MUST be either Subtype1 or Subtype2



D/O – Disjoint or Overlapping

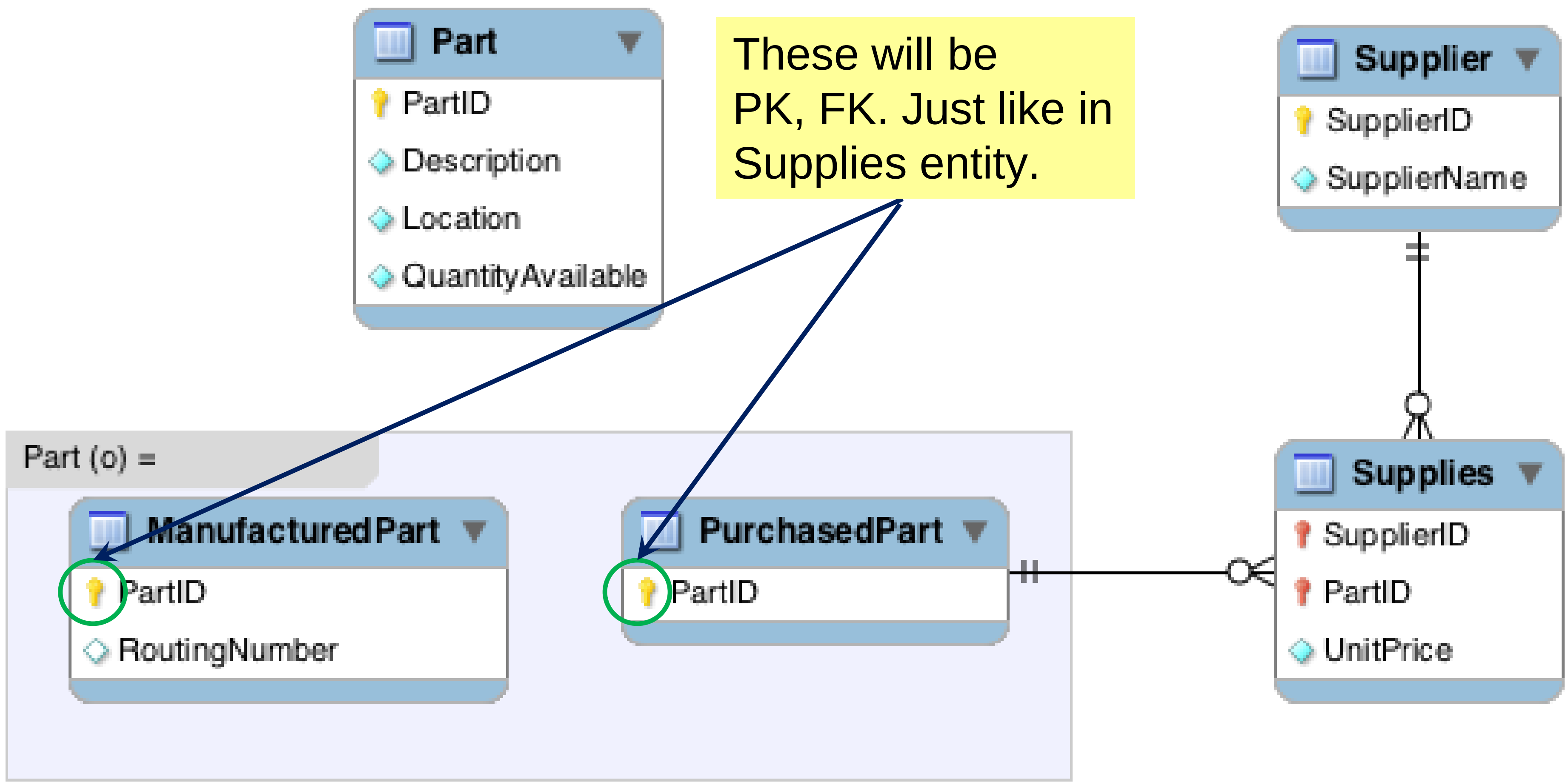


Identifying Super/Subtype Situations

- Bottom Up
  - Generalisation
  - Combing a number of entity sets with like attributes into a higher level entity set
    - Much like the Vehicle example
- Top Down
  - Specialisation
  - Process of defining one or more subtypes of the supertype and forming the relationships
  - Example
  - Part = (<u>ID</u>, Description, QuantityAvailable, Location, RoutingNumber, Supplier)
  - Becoming
    - Part = (<u>ID</u>, Description, QuantityAvailable, Location)
    - ManufacturedPart = (RoutingNumber)
    - SuppliedPart = (Supplier)

Note: These are incomplete relations

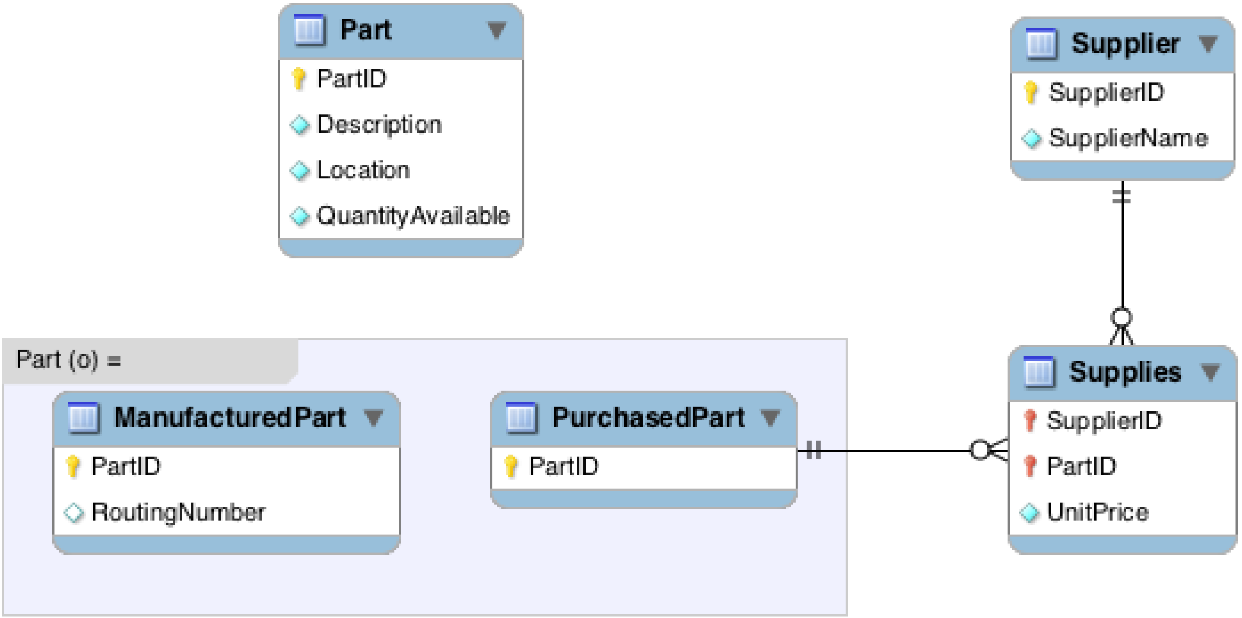
Example 1 – Parts – Logical Design



Example 1 – Parts – Logical Design

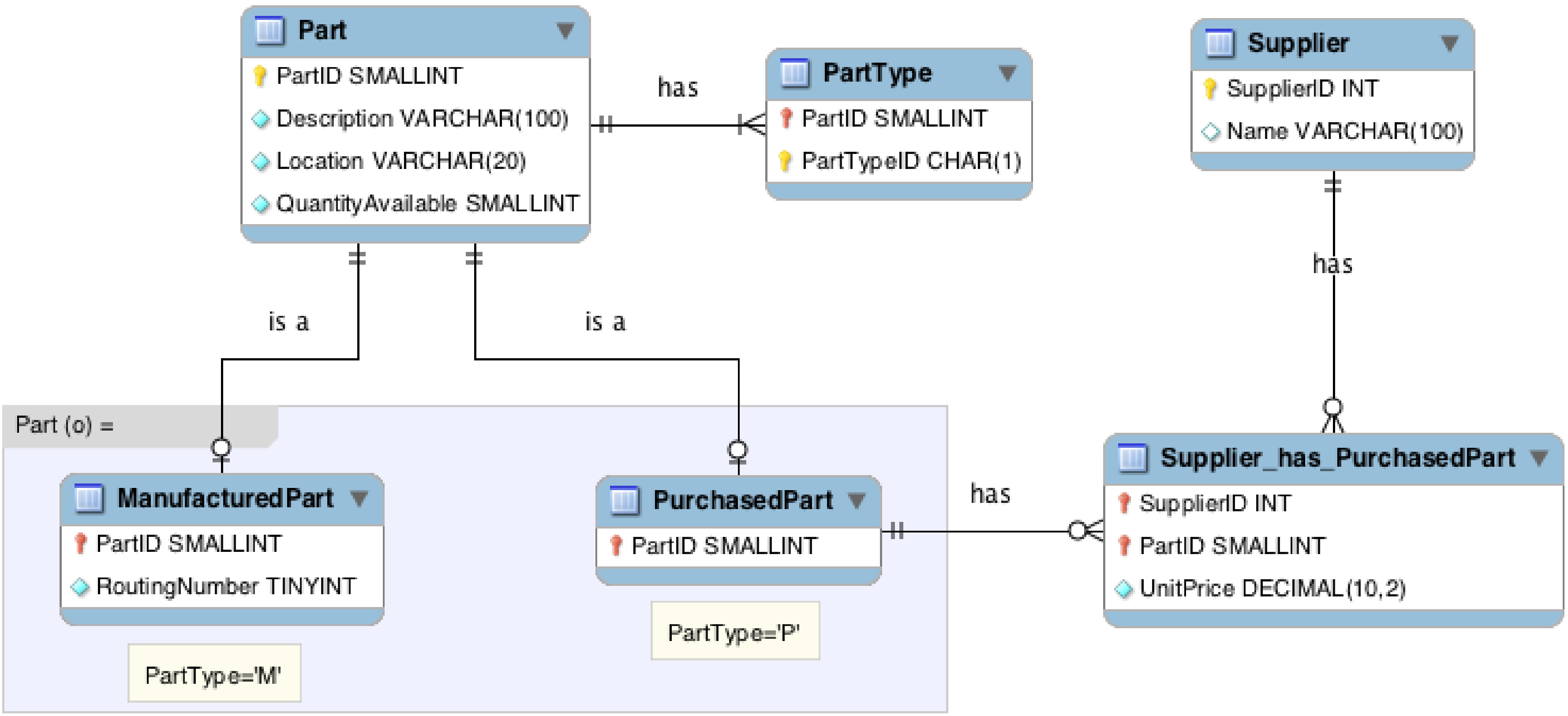
- Part = (<u>PartID</u>, Description, Location, QuantityAvailable)
- ManufacturedPart(**<u>PartID</u>**, RoutingNumber)
- PurchasedPart(**<u>PartID</u>**)
- Supplies(**<u>PartID</u>**, **<u>SupplierID</u>**)
- Supplier(<u>SupplierID</u>, Name)

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

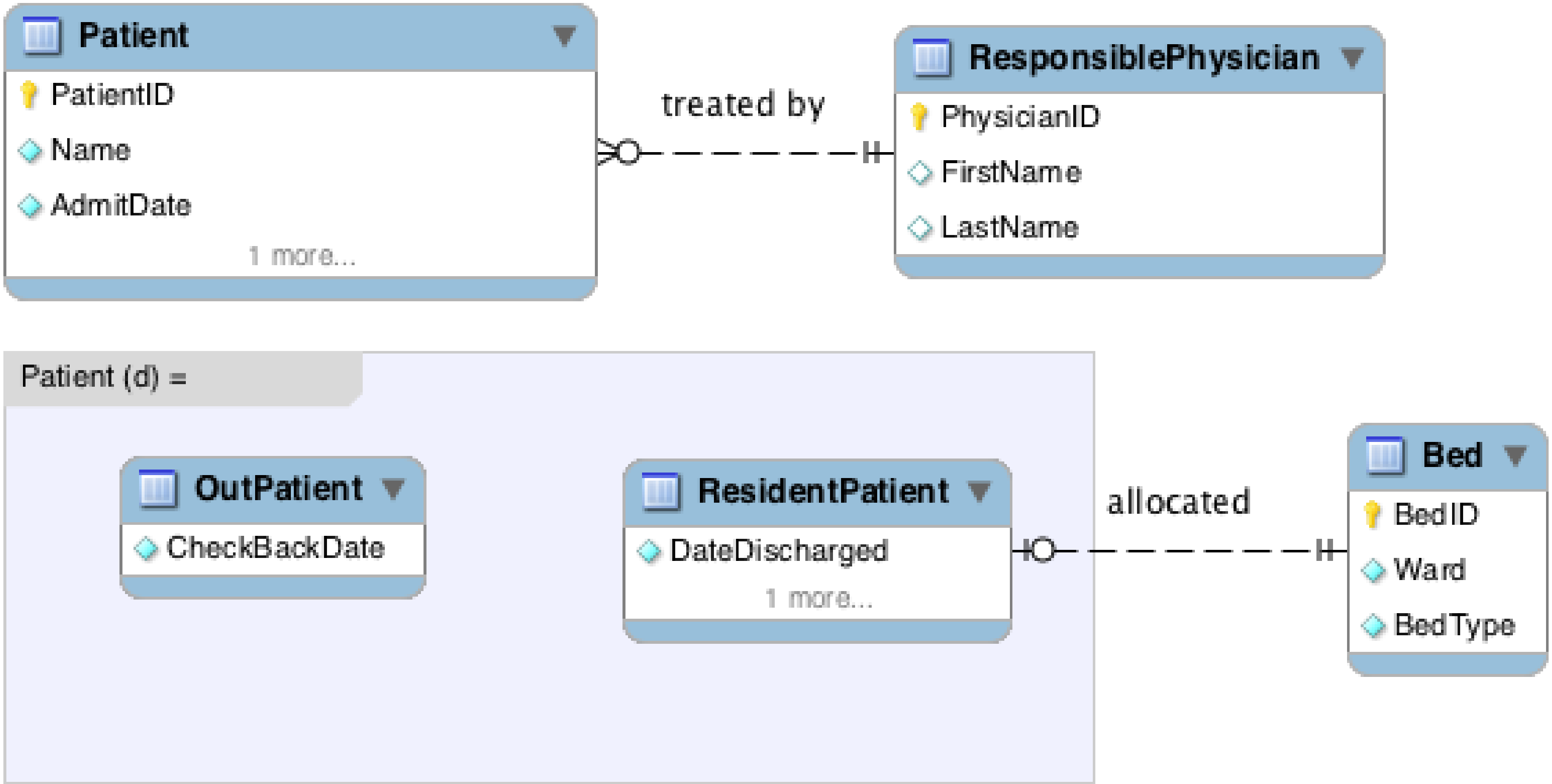


Example 1 – Parts – Physical Design

- Note new entity PartType – Because a part must be at least one type.



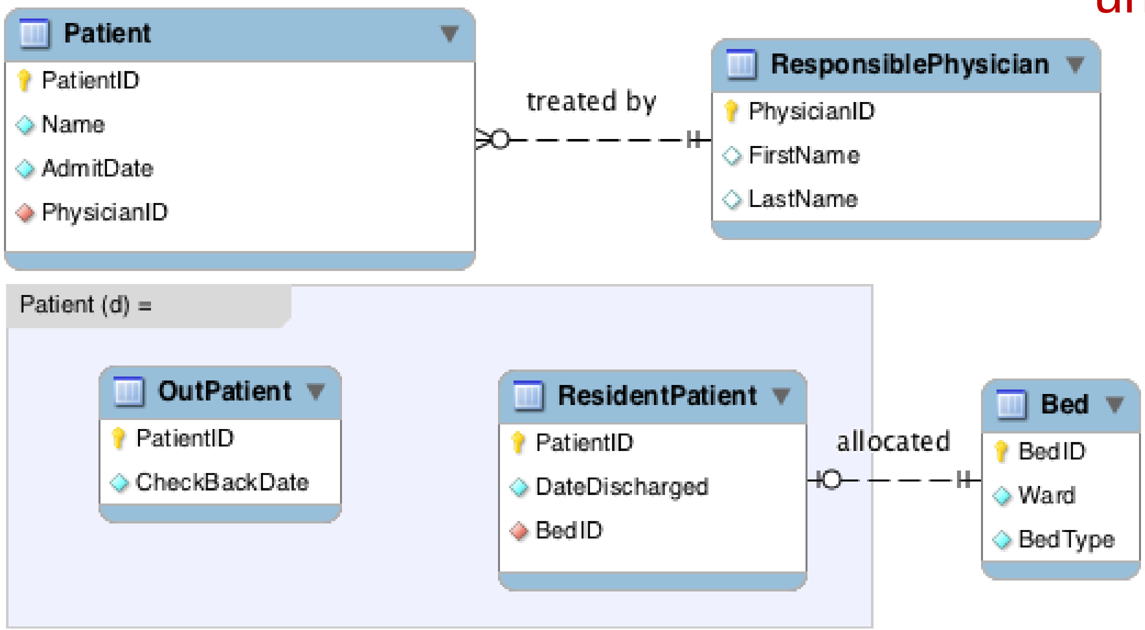
Example 2 - Patients (Conceptual)



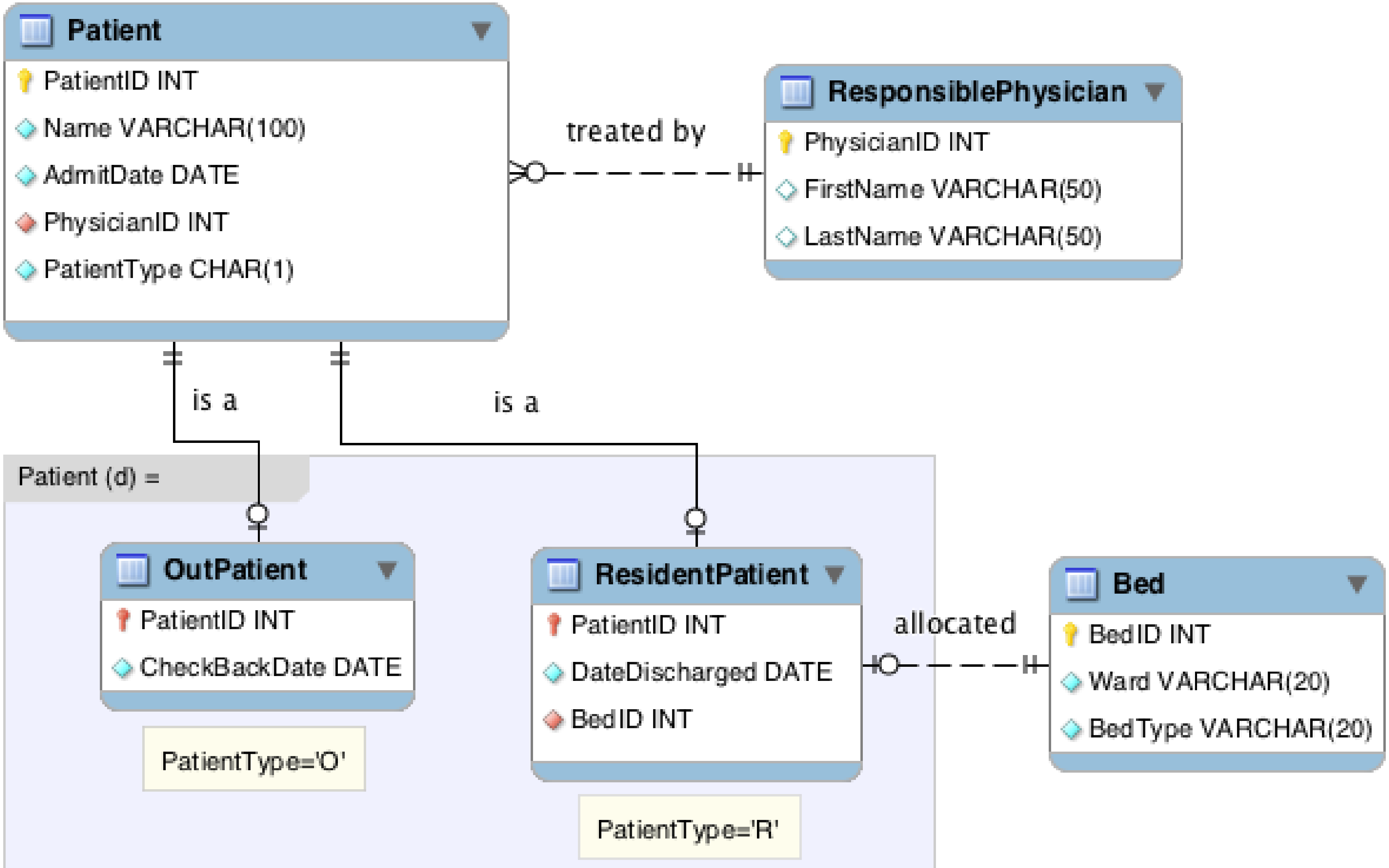
Example 2 – Patients – Logical Design

- Patient = (<u>PatientID</u>, Name, AdmitDate, *<u>PhysicianID</u>*)
- ResponsiblePhysician = (<u>PhysicianID</u>, FirstName, LastName)
- OutPatient = (**<u>PatientID</u>**, CheckBackDate)
- ResidentPatient = (**<u>PatientID</u>**, DateDischarged, *<u>BedID</u>*)
- Bed = (<u>BedID</u>, Ward, BedType)

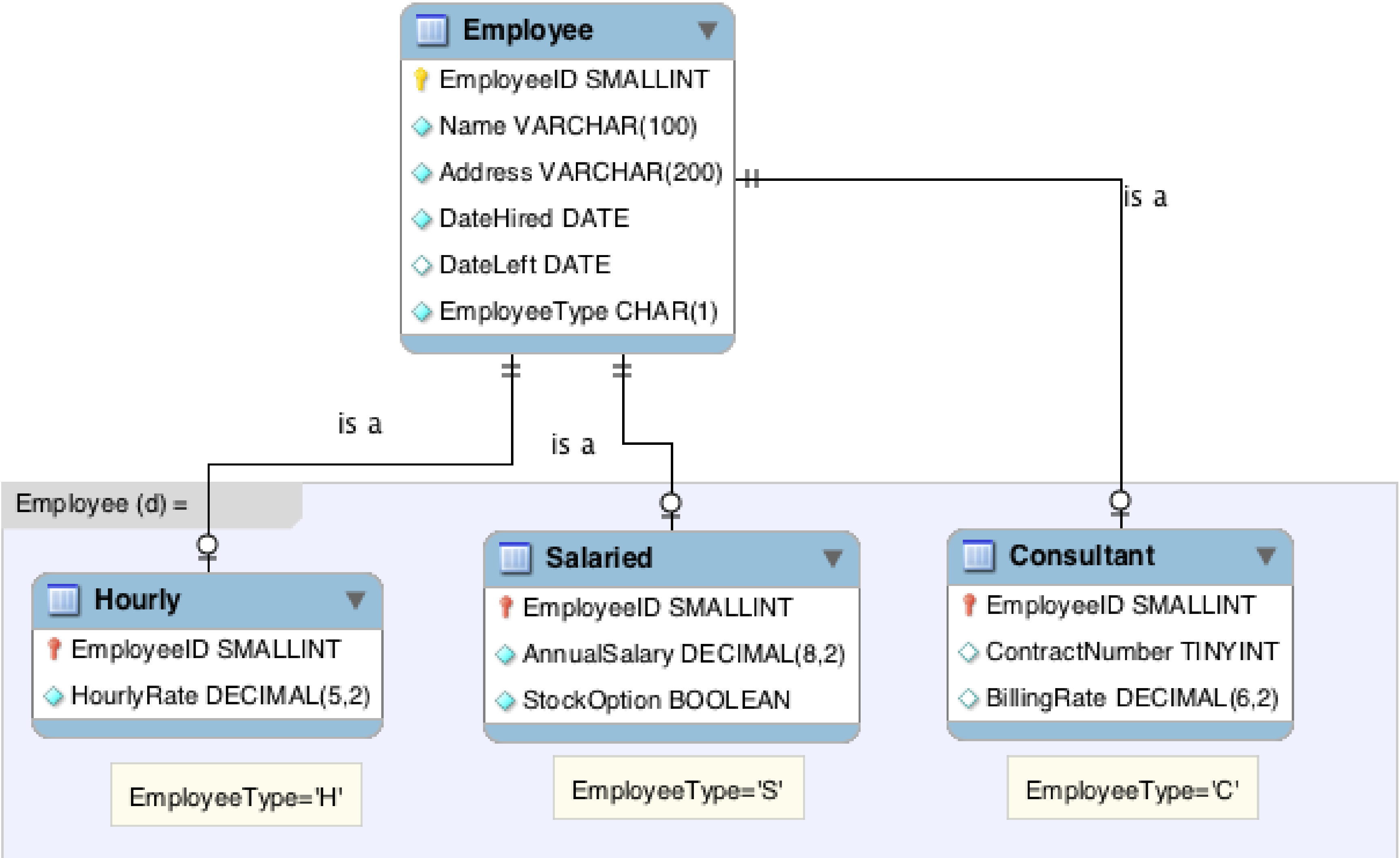
Note: Underline = PK, italic and underline = FK, underline and bold = PFK



Example 2 – Patients – Physical Design



Example 3 – Employee (showing physical only)



Create Table Statements

```sql
CREATE TABLE Employee (
    ID              SMALLINT        AUTO_INCREMENT,
    Name            VARCHAR(150)    NOT NULL,
    Address         VARCHAR(150)    NOT NULL,
    DateHired       DATE            NOT NULL,
    DateLeft        DATE,
    EmployeeType    CHAR(1)         NOT NULL,
    PRIMARY KEY  (ID)
) ENGINE=InnoDB;
```

```sql
CREATE TABLE Hourly (
    ID              SMALLINT,
    HourlyRate      DECIMAL(5,2)      NOT NULL,
    PRIMARY KEY  (ID),
    FOREIGN KEY (ID) REFERENCES Employee(ID)
          ON DELETE RESTRICT
          ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Create Table Statements

```
CREATE TABLE Salaried (
  ID                    SMALLINT,
  AnnualSalary          DECIMAL(8,2)                  NOT NULL,
  StockOption           CHAR(1)        DEFAULT "N"    NOT NULL,
  PRIMARY KEY  (ID),
  FOREIGN KEY (ID) REFERENCES Employee(ID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE Consultant (
  ID                    SMALLINT,
  ContractNumber        SMALLINT         NOT NULL,
  BillingRate           DECIMAL(6,2)     NOT NULL,
  PRIMARY KEY  (ID),
  FOREIGN KEY (ID) REFERENCES Employee(ID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB;
```

INSERTS Example

```sql
INSERT INTO Employee VALUES
    (DEFAULT, "Sean", "Sean's Address", "2012-02-02", NULL, "S");
SET @EID=LAST_INSERT_ID();
INSERT INTO Salaried VALUES (@EID, 92000, "N");
INSERT INTO Employee VALUES
    (DEFAULT, "Linda", "Linda's Address", "2011-06-12", NULL, "S");
SET @EID=LAST_INSERT_ID();
INSERT INTO Salaried VALUES (@EID, 92300, "Y");
INSERT INTO Employee VALUES
    (DEFAULT, "Alice", "Alice's Address", "2012-12-02", NULL, "H");
SET @EID=LAST_INSERT_ID();
INSERT INTO Hourly VALUES (@EID, 23.43);
INSERT INTO Employee VALUES
    (DEFAULT, "Alan", "Alan's Address", "2010-01-22", NULL, "H");
SET @EID=LAST_INSERT_ID();
INSERT INTO Hourly VALUES (@EID, 29.43);
INSERT INTO Employee VALUES
    (DEFAULT, "Peter", "Peter's Address", "2010-09-07", NULL, "C");
SET @EID=LAST_INSERT_ID();
INSERT INTO Consultant VALUES (@EID, 19223, 210);
INSERT INTO Employee VALUES
    (DEFAULT, "Rich", "Rich's Address", "2012-05-19", NULL, "C");
SET @EID=LAST_INSERT_ID();
INSERT INTO Consultant VALUES (@EID, 19220, 420);
```

INSERTS Example

ID	Name	Address	DateHired	DateLeft	EmployeeType
1	Sean	Sean's Address	2012-02-02	NULL	S
2	Linda	Linda's Address	2011-06-12	NULL	S
3	Alice	Alice's Address	2012-12-02	NULL	H
4	Alan	Alan's Address	2010-01-22	NULL	H
5	Peter	Peter's Address	2010-09-07	NULL	C
6	Rich	Rich's Address	2012-05-19	NULL	C

ID	AnnualSalary	StockOption
1	92000.00	N
2	92300.00	Y

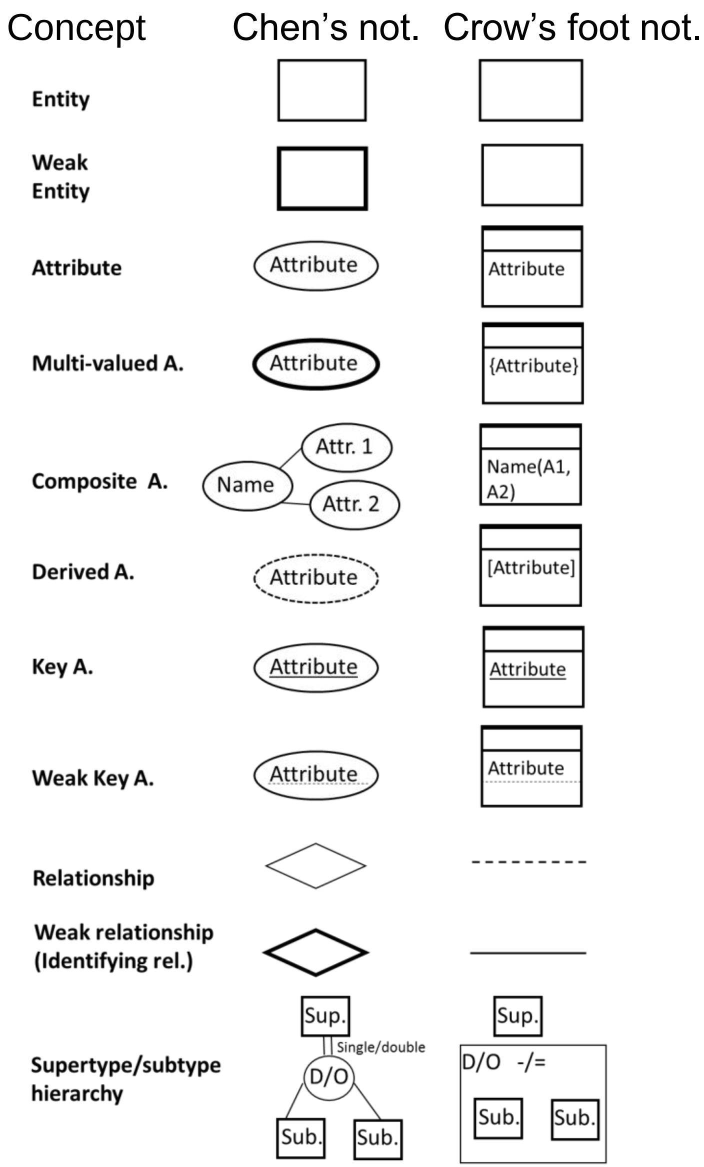
ID	HourlyRate
3	23.43
4	29.43

ID	ContractNumber	BillingRate
5	19223	210.00
6	19220	420.00

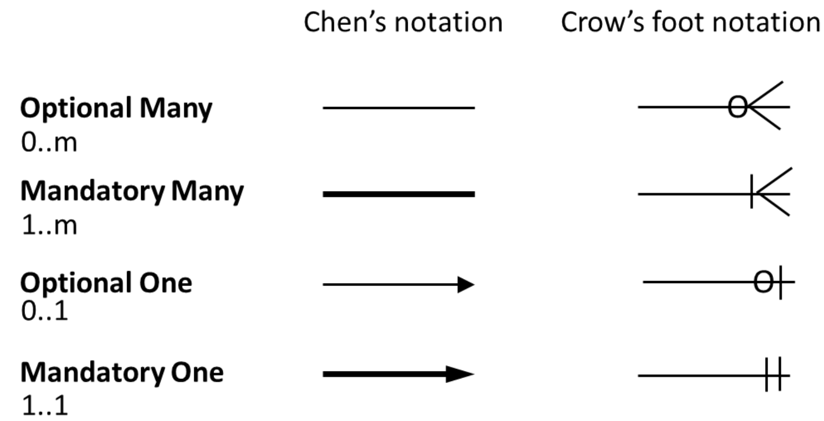
What's Examinable

- Need to be able to draw conceptual, logical and physical diagrams (now including EER)
- For conceptual both Chen's and Crow's foot notations are acceptable
- Create table SQL statements (with integrity constraints)

Conceptual Model Mapping

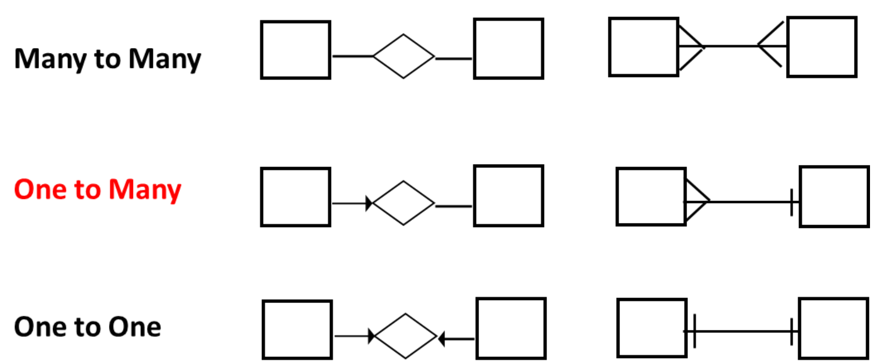


Relationship cardinalities and constraints



BINARY Relationship Cardinalities

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



Next Lecture

- Relational algebra and calculus
  - –How do we ask queries/interrogate a database
  - –Foundation of SQL queries
  - –Please come, it’s going to be a lot of fun