Dr Renata Borovica-Gajic
David Eccles

INFO20003 Database Systems

Lecture 05
Modelling with MySQL Workbench
Data Types

Contacting Lecturers and Staff

- Include Subject code (INFO90002)
- Student ID (1234567)
- Be sent from your student.unimelb.edu.au email account

Check the Discussion wall first!

- Lecturer & Subject Coordinator :
  - David Eccles eccles.d@unimelb.edu.au
  - Prefer David or Mr Eccles (avoid Professor, Sir, Dr. etc)
- Tutors
  - Veronica Torres Pena maria.torres@unimelb.edu.au
  - Nick Howard nick.howard@unimelb.edu.au
  - Ibrahim AlMadi ibrahim.al@unimelb.edu.au
  - Dr Neven Tomov ntomov@unimelb.edu.au
  - Fraser Mcharg fraser.harg@unimelb.edu.au

News

- Assignment 1 has been published
- **Group Agreement**: Friday 21st August 2100H (6 p.m.) AEDT 2020 (**Assignment Hurdle**)
- **Assignment**: Friday 4th September 2020 0900H (9 a.m.) AEDT 2020

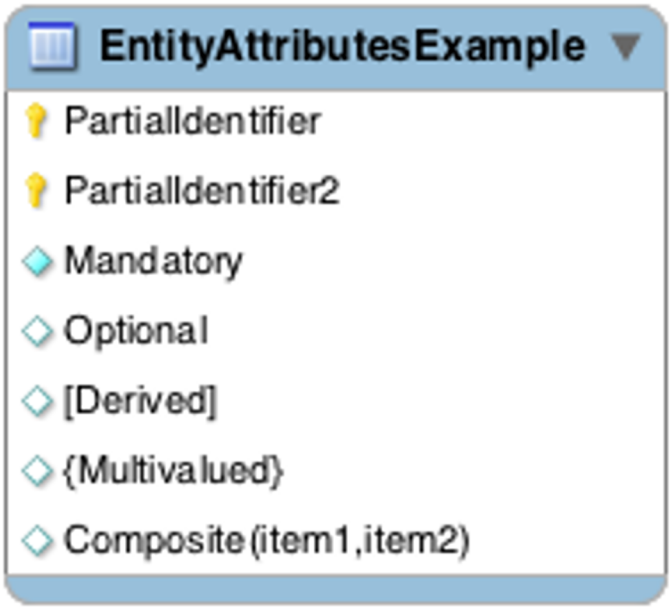
Lecture 5: Modelling with Workbench

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

Conventions of ER Modelling (Workbench)

- Entity

- Attributes

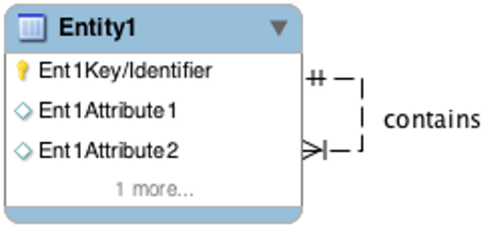


- **Identifier or key:**
  - Fully identifies an instance
- **Partial Identifier:**
  - Identifies an instance in conjunction with one or more partial identifiers
- **Attributes types:**
  - Mandatory (blue diamond)
  - Optional (empty diamond)
  - Derived []
    - [YearsEmployed]
  - Multivalued {}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

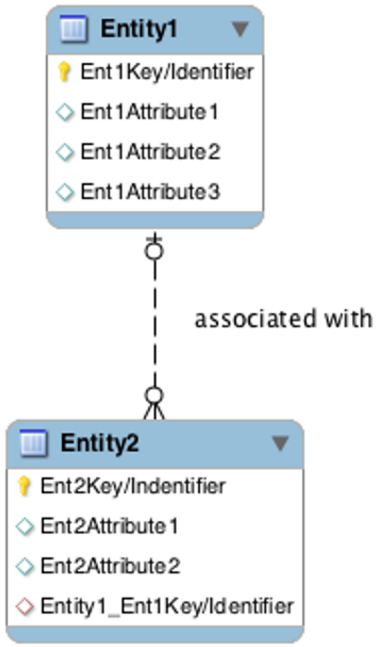
Conventions of ER Modelling (Workbench)

- Relationship Degrees

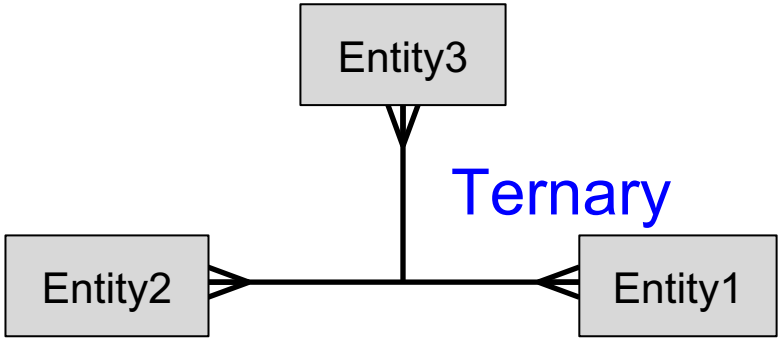
Unary



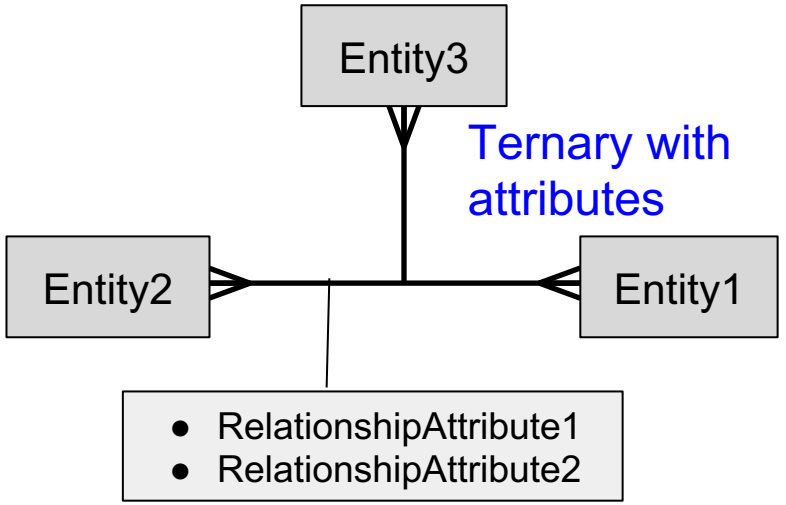
Binary



Ternary

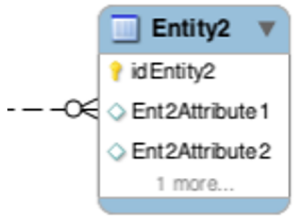


Ternary with attributes

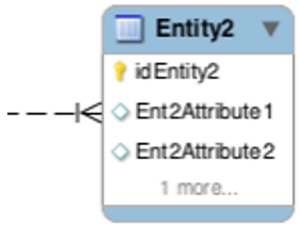


Conventions of ER Modelling (Workbench)

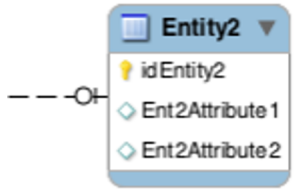
- Cardinality Constraints



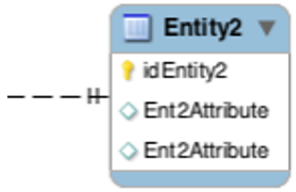
Optional Many
Partial participation
Without key constraint



Mandatory Many
Total participation
Without key constraint



Optional One
Partial participation
Key constraint



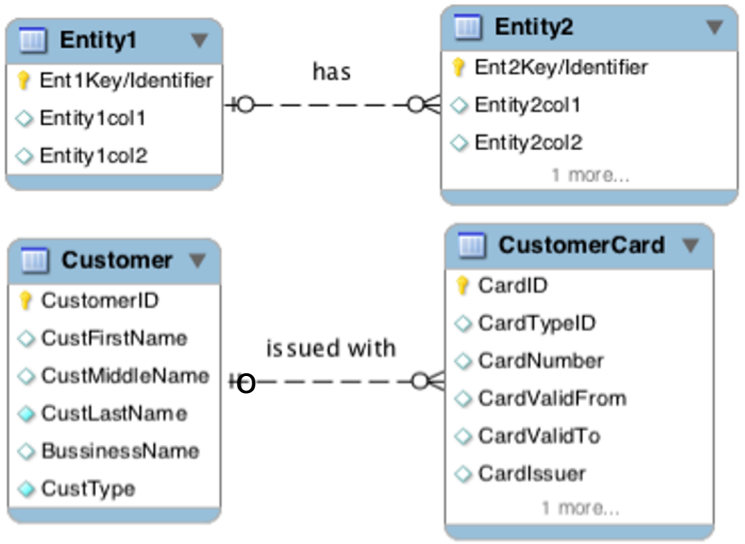
Mandatory One
Total participation
Key constraint

- 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)

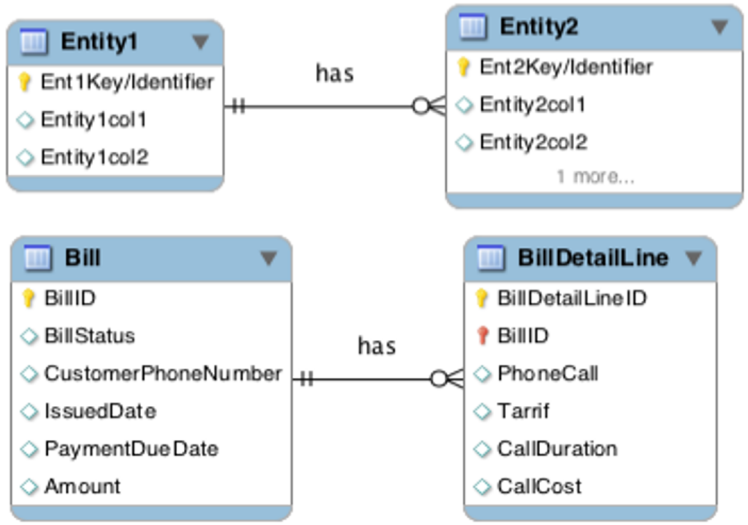
Strong Entity:

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



Weak Entity

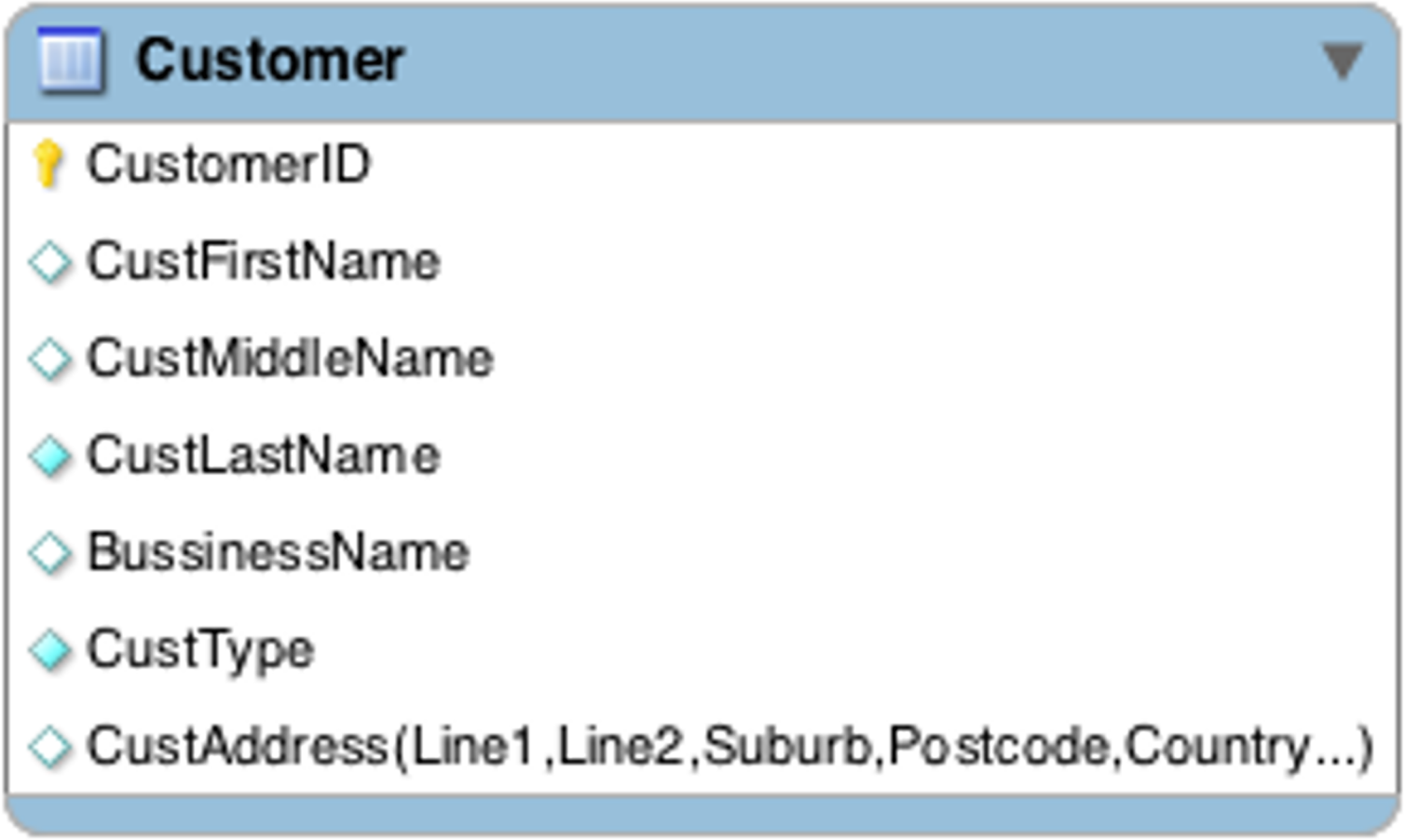
- Can't exist without the owner
- E.g. BillDetaiLine



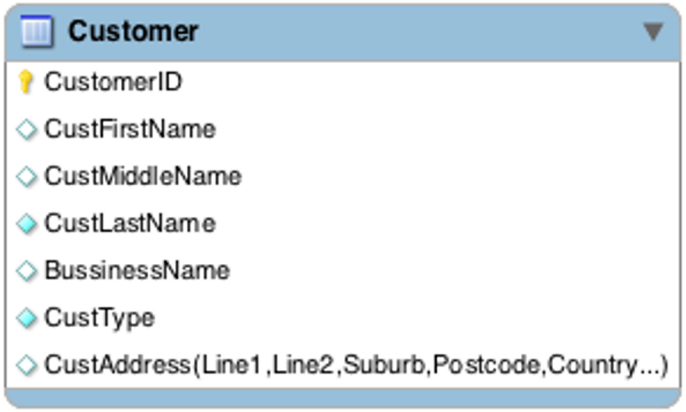
Good / Bad Entity Selection

- Entity
  - Will have many instances in the database
  - Is composed of many attributes
  - Is something needed for the system to work (something we are trying to model)
- Examples
  - Person: EMPLOYEE, STUDENT, PATIENT
  - Place: STORE, WAREHOUSE, STATE
  - Object: MACHINE, BUILDING, VEHICLE
  - Event: SALE, REGISTRATION, BROADCAST
  - Concept: ACCOUNT, COURSE, ROLE
- An Entity IS NOT
  - A user of the system
  - An output of the system (i.e. a report)

Single Entity (Conceptual)



Convert from Conceptual to Logical design (Single Entity)



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

- **Tasks checklist:**
  1. Convert composite and multi-valued attributes
     - Multi-Attribute values can become another table
  2. Resolve many-many relationships
  3. Add foreign keys at crows foot end of relationships (on the many side)

Convert from Logical to Physical Design

Inputs

- Normalised relations
  - Note: taught later
- Attribute definitions
- Response time expectations
- Data security needs
- Backup / recovery needs
- Integrity expectations
- DBMS technology used

Leads to

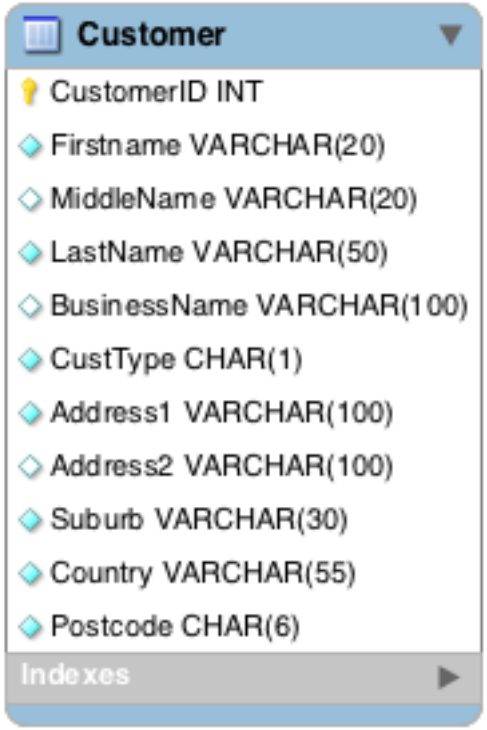
Decisions

- Attribute data types
- Physical record descriptions
  - These don't always match the logical design
- File organisations
- Indexes and database architectures
- Query optimisation

Convert from Logical to Physical Design

- **Generate attribute data types**

Physical Design:

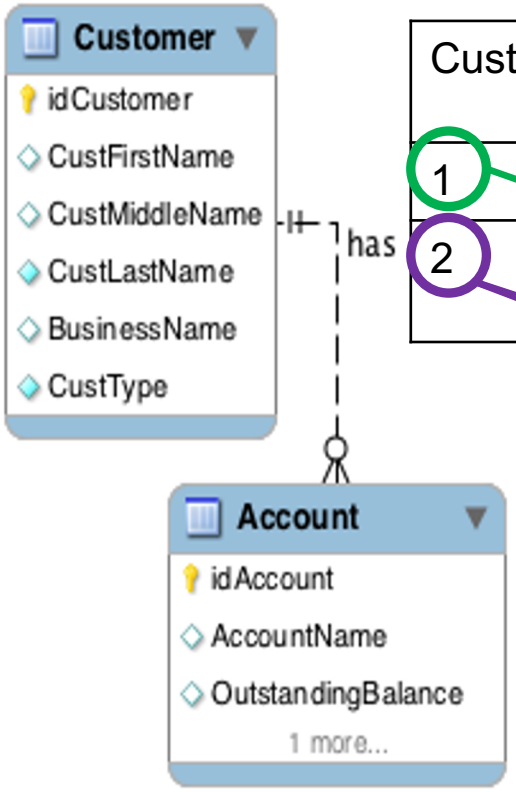


Implementation:

```
CREATE TABLE Customer(
  CustomerID INT NOT NULL,
  FirstName VARCHAR(20) NOT NULL,
  MiddleName  VARCHAR(20),
  LastName VARCHAR(50) NOT NULL,
  BusinessName VARCHAR(100),
  CustType      CHAR(1) NOT NULL,
  Address1 VARCHAR(100) NOT NULL,
  Address2 VARCHAR(100),
  Suburb VARCHAR(30) NOT NULL,
  Country VARCHAR(55) NOT NULL,
  Postcode CHAR(6) 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

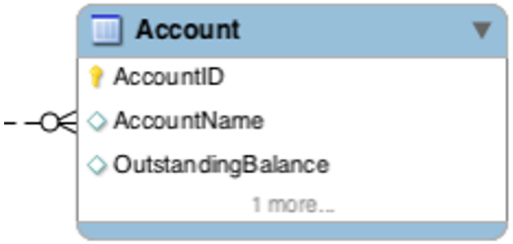


CustID	CustomerFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company

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

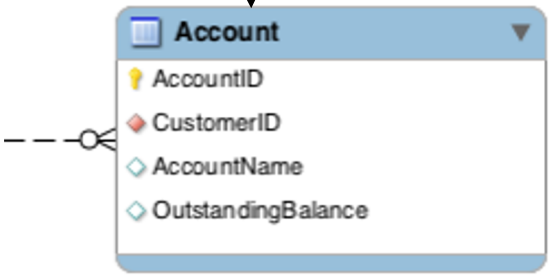
Conceptual Design:



Logical Design:

Account=(<u>AccountID</u>,
AccountName,
OutstandingBalance,
<u>CustomerID</u>)

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

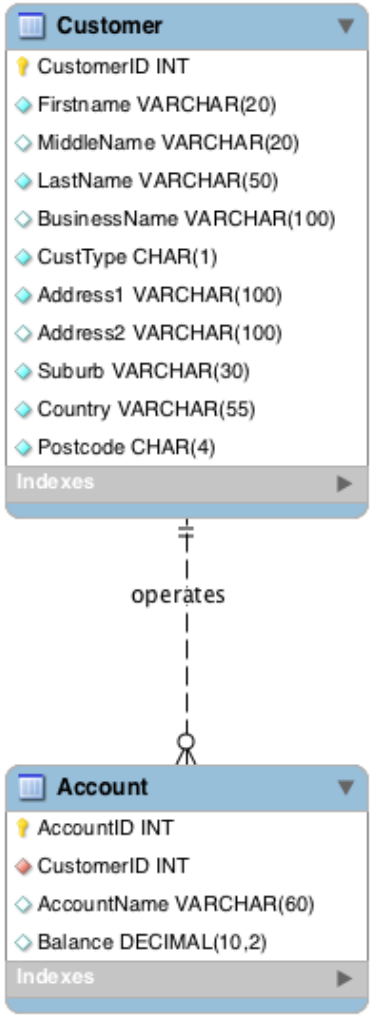


Tasks checklist:

1. Convert composite and multi-valued attributes
2. Resolve many-many relationships
3. Add foreign keys at crows foot end of relationships
   - See FK1 – CustomerID
   - Every row in the account table must have a CustomerID from Customer (referential integrity)

Physical Design & Implementation- Account

Physical design:

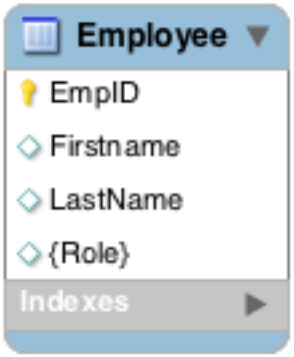


Implementation:

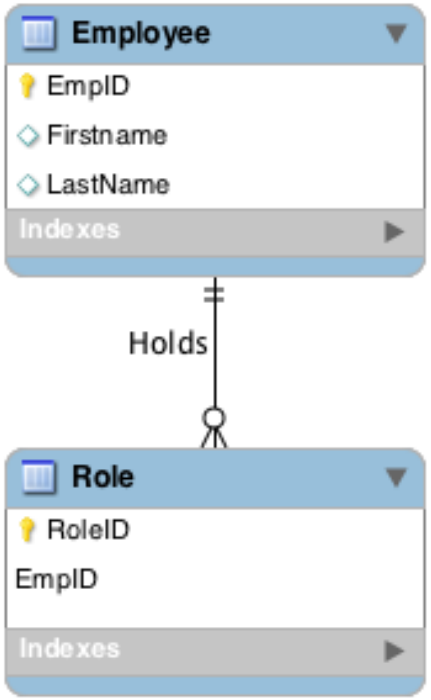
```
CREATE TABLE Account (
  AccountID        INTEGER auto_increment,
  CustomerID       INTEGER NOT NULL,
  AccountName      VARCHAR(60),
  Balance                  DECIMAL(10,2),
  PRIMARY KEY (AccountID),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
    ON DELETE RESTRICT
    ON UPDATE CASCADE);
```

Dealing with Multi-Valued Attributes

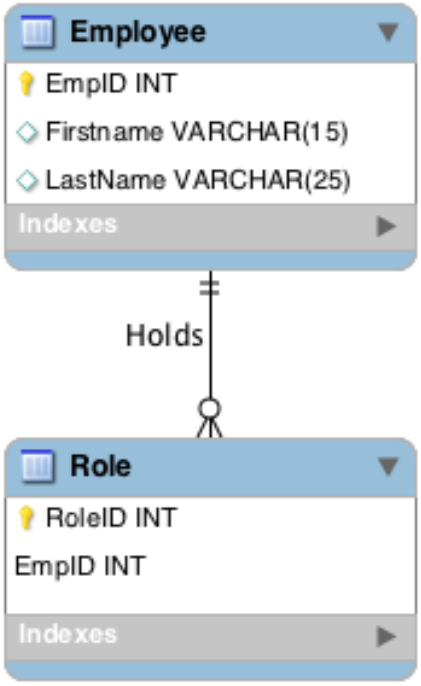
Conceptual Design:



Logical Design:



Physical Design:



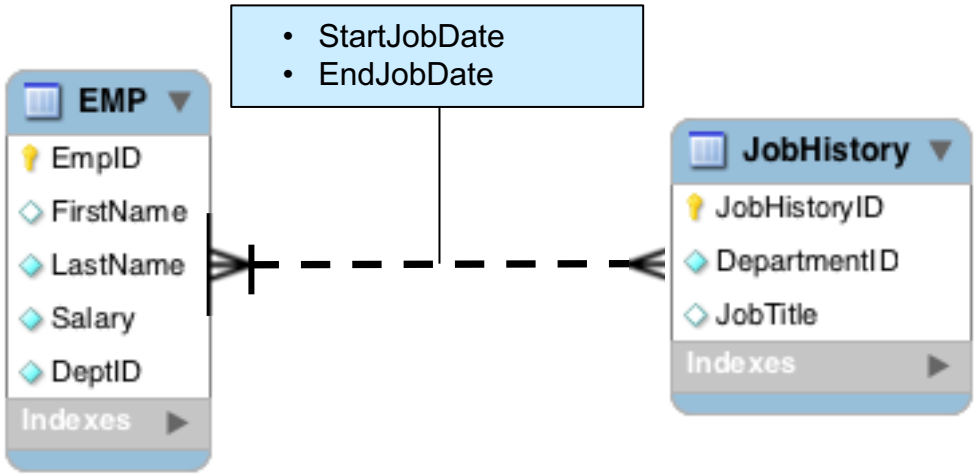
Role is an example of a *weak entity*

- 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 on "JOIN" time

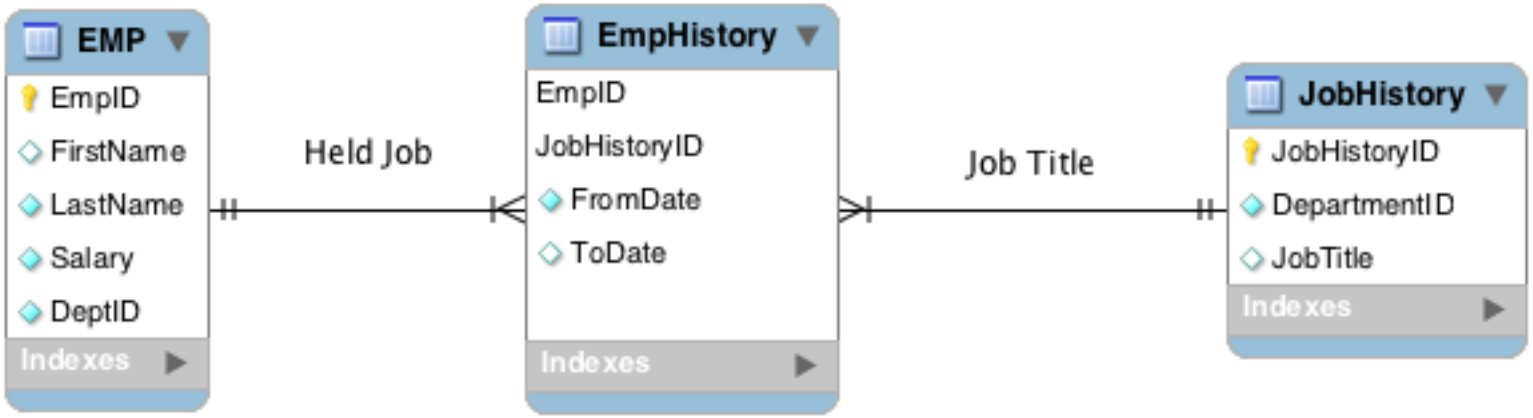
Many to Many Relationship

- 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



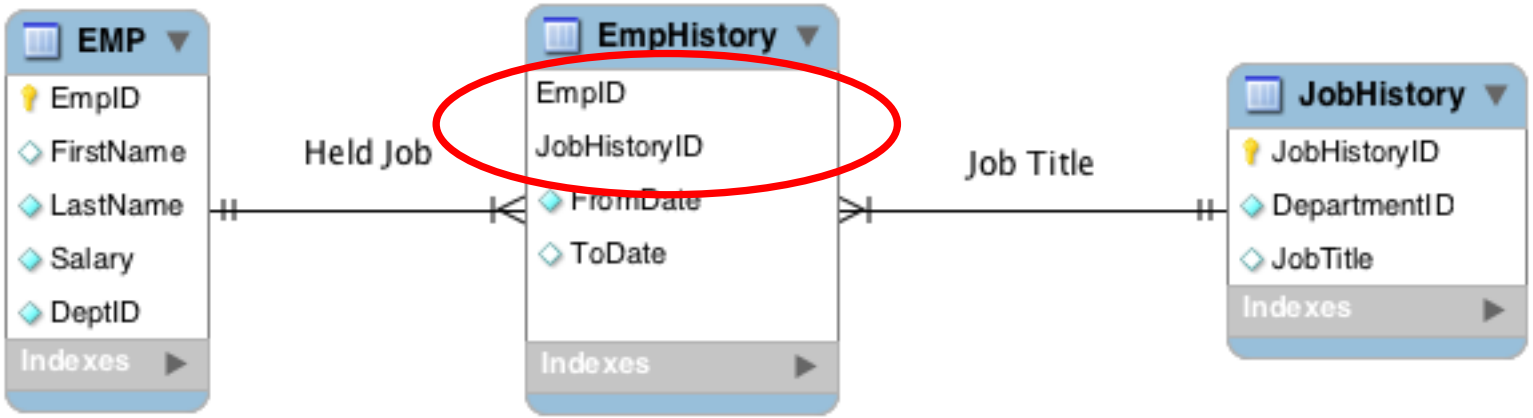
Note: FromDate/ToDate are descriptive attributes of the relationship

- They go into the associative entity for M-M

Many to Many - Logical Model

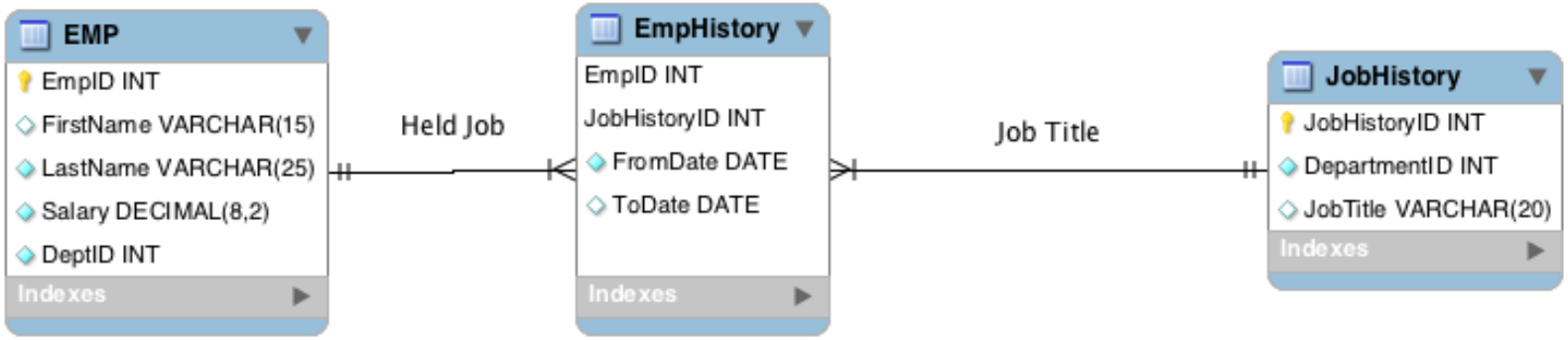
Emp(EmpID, FirstName, LastName, Salary, DeptID)
EmpHistory(**EmpID**, **JobHistoryID**, FromDate, ToDate)
JobHistory(JobHistoryID,DepartmentID, JobTitle)

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



In MySQL Workbench v8.0.x PFKs are displayed like this
They should be a RED KEY:

Many to Many - Physical Model & Implementation



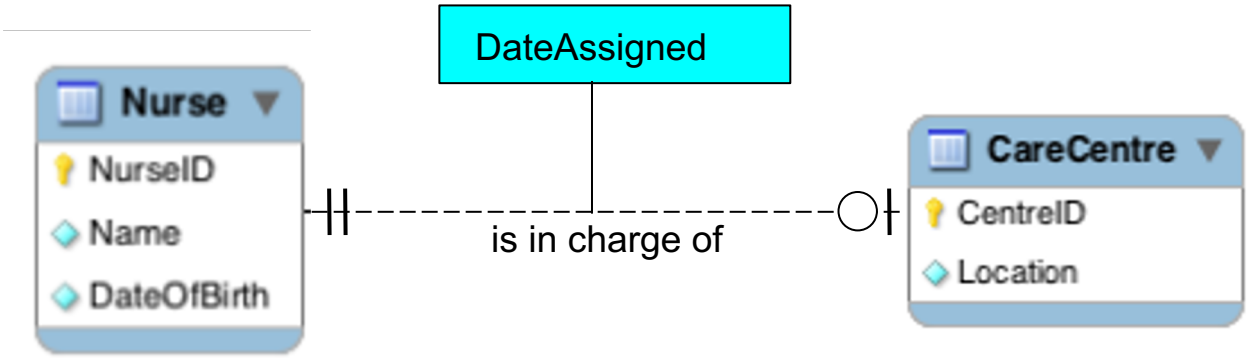

```
CREATE TABLE EMP
(EmpID integer,
FirstName varchar(15),
LastName varchar(25) NOT NULL,
Salary (8,2) NOT NULL,
DeptID integer NOT NULL,
PRIMARY KEY EmpID);
```

```
CREATE TABLE EmpHistory
(EmpID integer,
 JobHistoryID integer,
 FromDate DATE NOT NULL,
ToDate DATE,
PRIMARY KEY EmpID, JobHistoryID
FOREIGN KEY EmpID REFERENCES EMP(EmpID),
FOREIGN KEY JobHistoryID REFERENCES JobHistory(JobHistoryID));
```

```
CREATE TABLE JobHistory
(JobHistoryID  integer,
DepartmentID integer NOT NULL,
JobTitle varchar(20),
PRIMARY KEY JobHistoryID);
```

Binary One-One Relationship

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

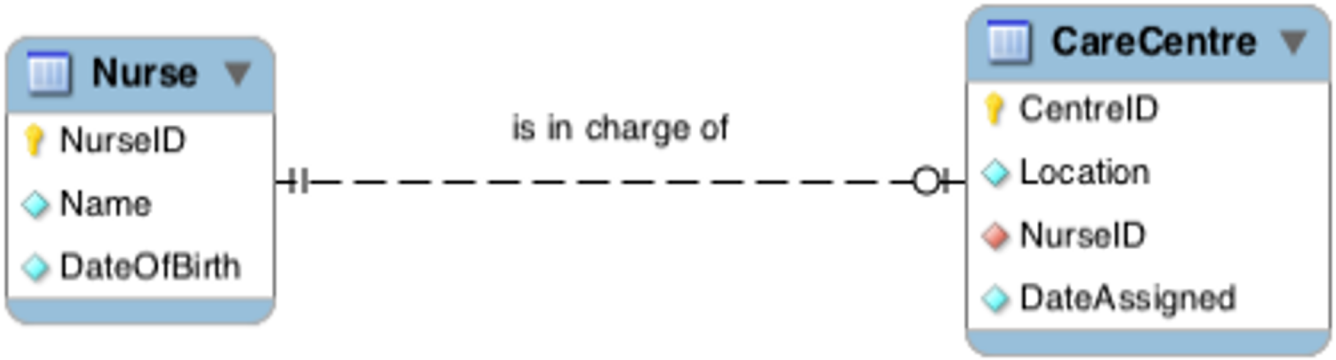


- But we have 2 “one” sides. Which one?

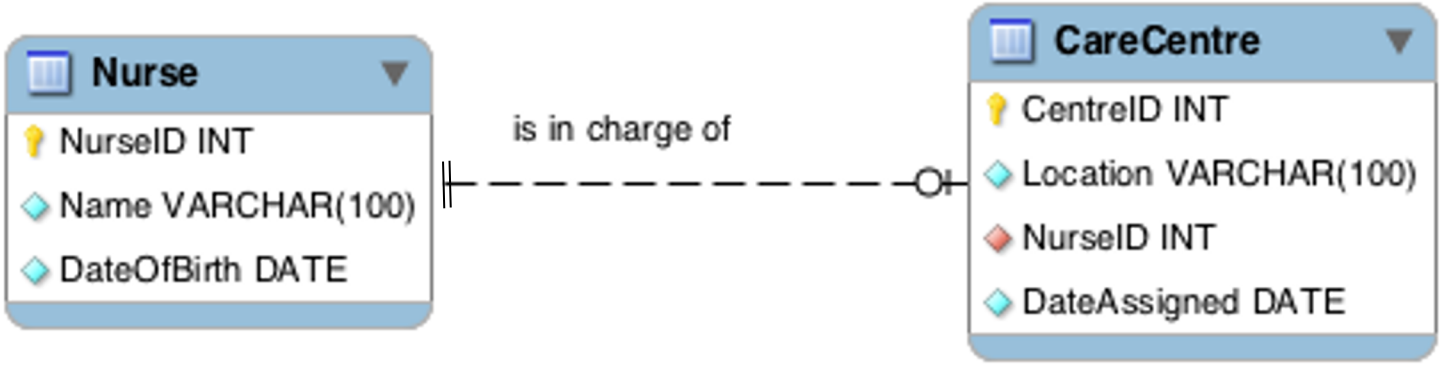
- Need to decide whether to put the foreign key inside Nurse or CareCentre (in which case you would have the Date_Assigned in the same location)
  - Where would the least NULL values be?
  - The rule is the OPTIONAL side of the relationship gets the foreign key

Binary One-One Relationship – Logical and Physical Design

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



- **Physical**

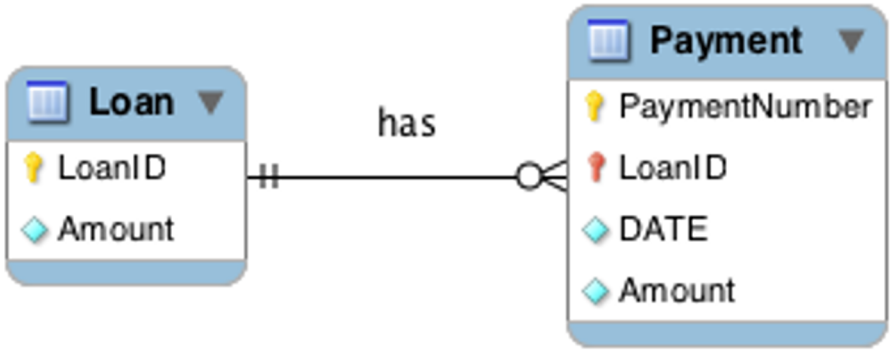


Summary of Binary Relationships

- **One-to-Many**
  - Primary key on the one side becomes a foreign key on the many
- **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

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**



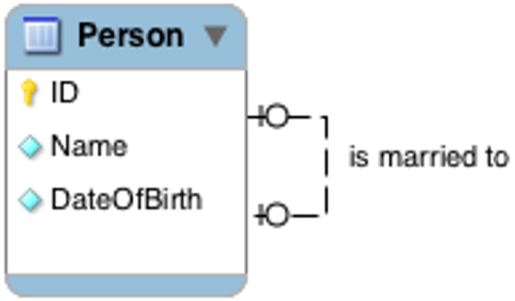
  - Logical Design
    - Loan = (LoanID, Amount)
    - Payment = (PaymentNumber, **LoanID**, Date, Amount)
  - Physical Design – as per normal one-to-many

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:

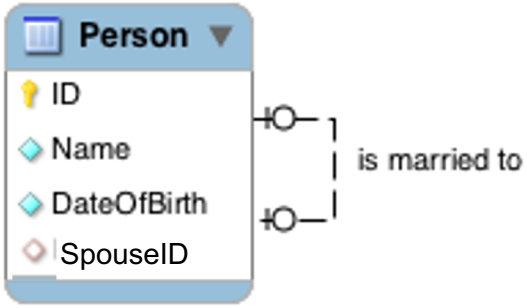


Implementation:

```
CREATE TABLE Person (
  ID INT NOT NULL,
  Name VARCHAR(15) NOT NULL,
  DateOfBirth DATE NOT NULL,
  SpouseID INT,
  PRIMARY KEY (ID),
  FOREIGN KEY (SpouseID)
  REFERENCES Person (ID)
  ON DELETE RESTRICT
  ON UPDATE CASCADE);
```

Logical Design:

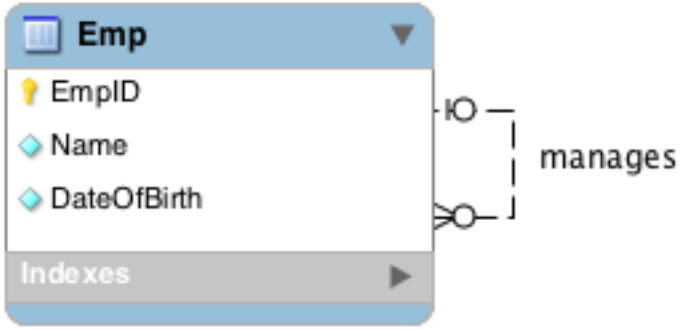
- Person = (ID, Name, DateOfBirth, *SpouseID*)



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

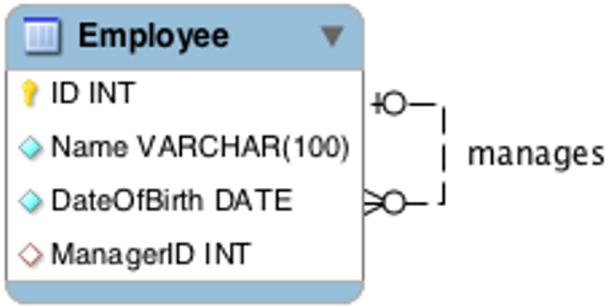
Conceptual Design:



Logical Design:

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

Physical Design:

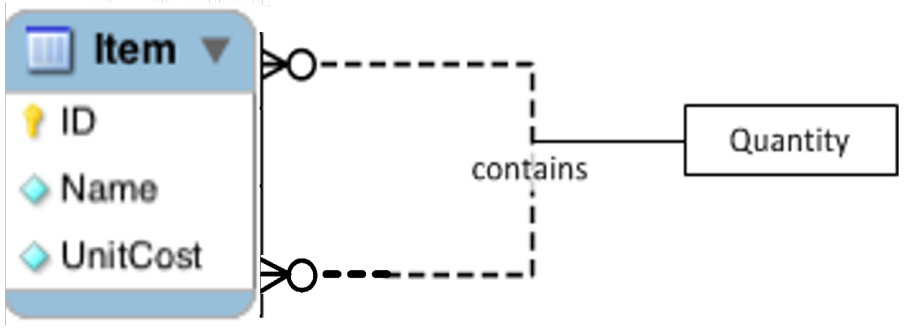


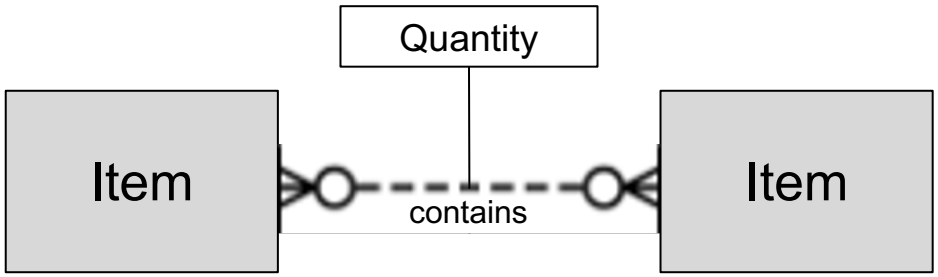
Implementation:

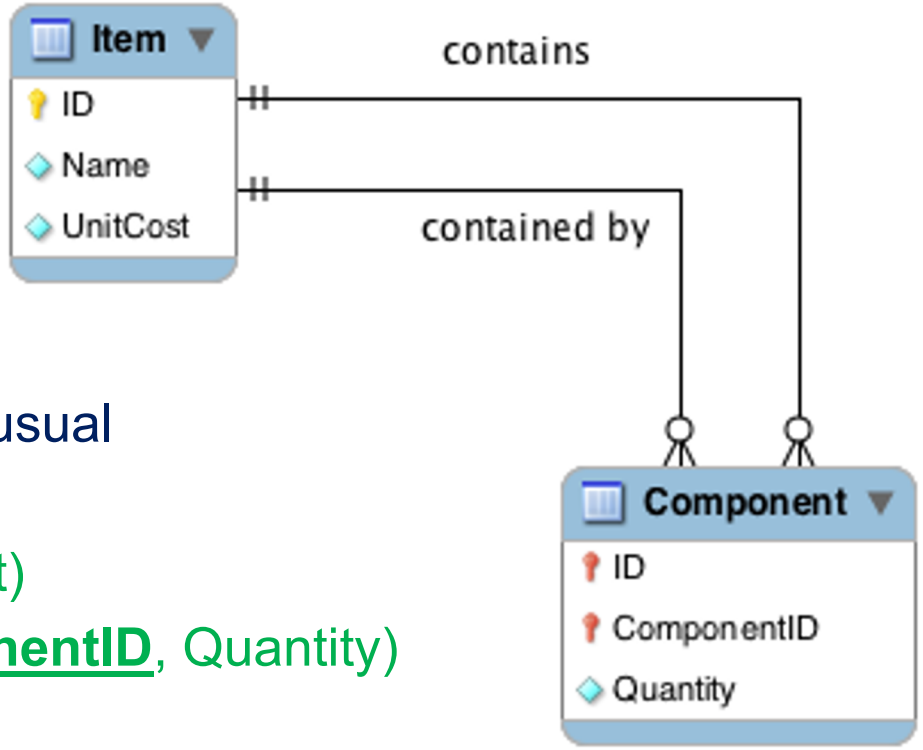
```
CREATE TABLE Employee(
  ID INT NOT NULL,
  Name VARCHAR(12) NOT NULL,
  DateOfBirth DATE NOT NULL,
  ManagerID INT ,
  PRIMARY KEY (ID),
  FOREIGN KEY (ManagerID)
  REFERENCES Employee(ID)
  ON DELETE RESTRICT
  ON UPDATE CASCADE);
```

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)

Unary: Many-to-Many Implementation

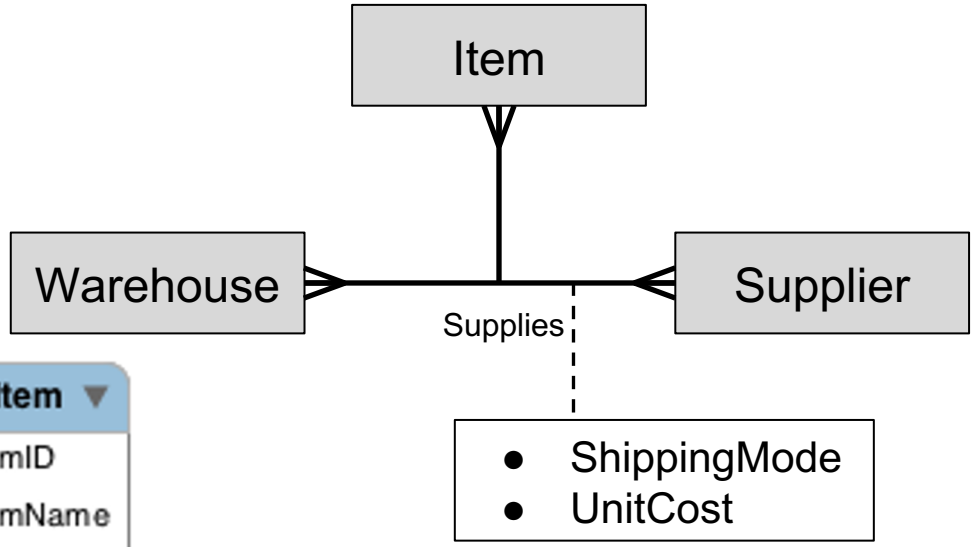
- Implementation

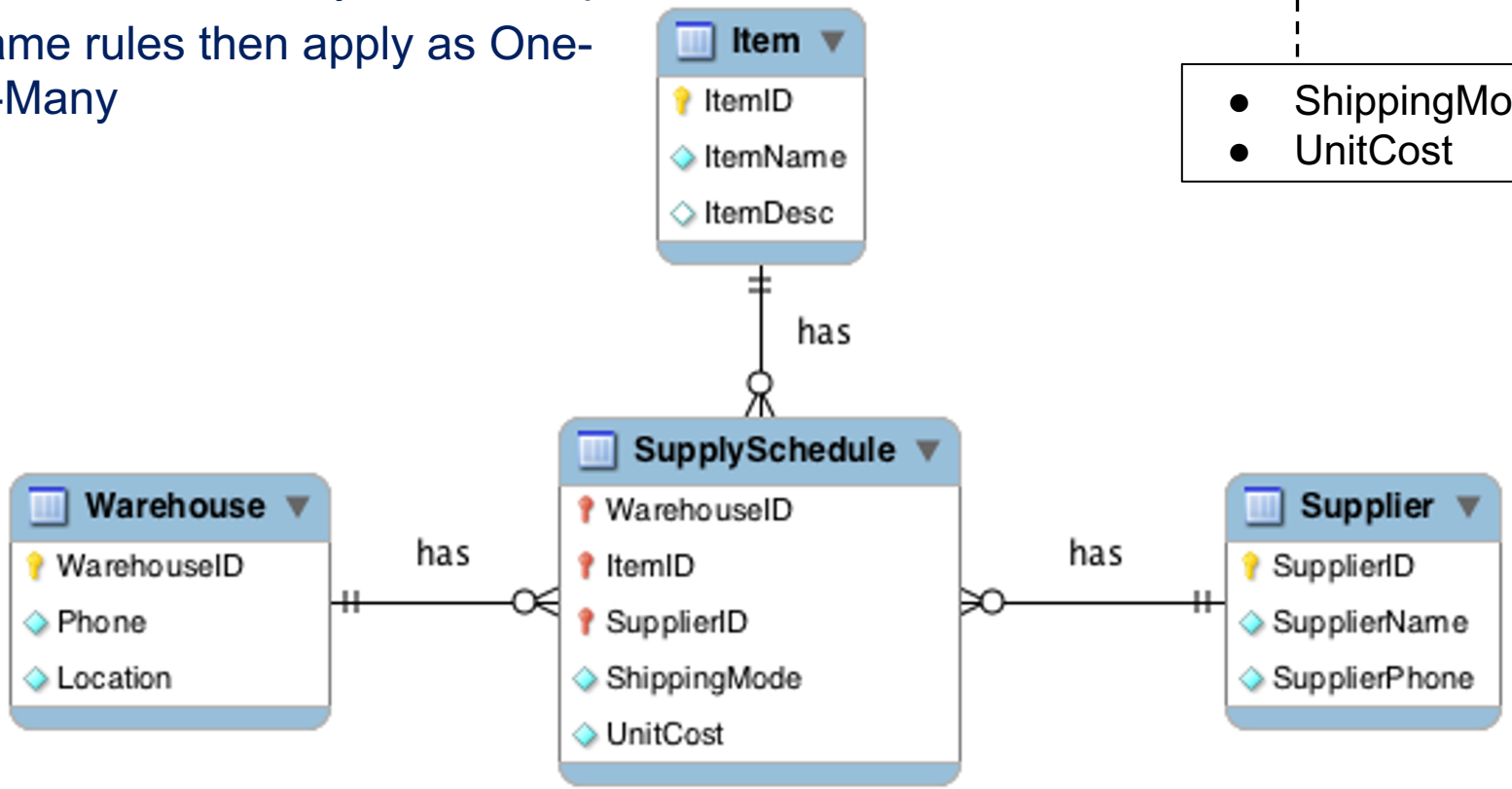
```
CREATE TABLE Part (
    ID                  smallint,
    Name                VARCHAR(100)    NOT NULL,
    UnitCost            DECIMAL(6,2)    NOT NULL,
    PRIMARY KEY  (ID)
) ENGINE=InnoDB;
```

```
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;
```

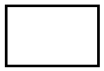
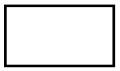
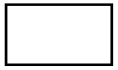
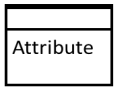
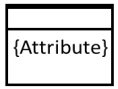
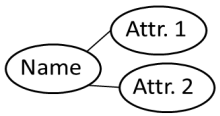
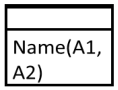
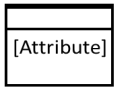
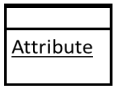
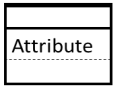
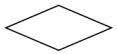
Ternary relationships

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

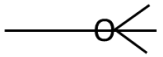
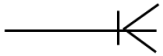
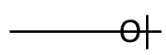
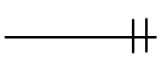




Conceptual Model Mapping

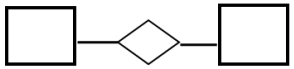
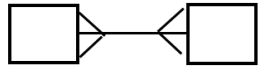
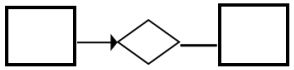
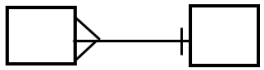
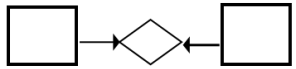
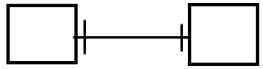
Concept	Chen's	Crow's foot
Entity		
Weak Entity		
Attribute		
Multi-valued A.		
Composite A.		
Derived A.		
Key A.		
Weak Key A.		
Relationship		
Weak relationship (Identifying rel.)		

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		

Conceptual to Physical checklist

	Many to Many resolved?	Database type (RDBMS, NoSQL)	Datatypes & DB brand (e.g. MySQL)	Key Name
Conceptual	NO	NO	NO	Key
Logical	YES	YES	NO	Primary Key + Foreign Key
Physical	YES	YES	YES	Primary Key + Foreign Key

Choosing data types

- Column: smallest unit of data in database
- Data types help DBMS to store and use information efficiently
- You should choose data types that:
  - enforce data integrity (quality)
  - can represent all possible values
  - support all required data manipulations
  - minimize storage space
  - maximize performance (e.g. fixed or variable length)
- The major data types are:
  - text or character
  - number
  - dates and time

MySQL Data Types

Character types (MySQL)

CHARACTER

- **CHAR(M):** A fixed-length string that is always right-padded with spaces to the specified length when stored on disc.
  The range of M is 1 to 255.
- **CHAR:** Synonym for CHAR(1).
- **VARCHAR(M):** A variable-length string.
  Only the characters inserted are stored – no padding.
  The range of M is 1 to 65535 characters.
- **BLOB, TEXT:** A binary or text object with a maximum length of 65535 (2^16) bytes (blob) or characters (text).
  Not stored inline with row data.
- **LONGBLOB, LONGTEXT:** A BLOB or TEXT column with a maximum length of 4,294,967,295 (2^32 - 1) characters.
- **ENUM** (‘value1’,’value2’,…) up to 65,535 members.

Number types (MySQL)

Integers

- **TINYINT**: Signed (-128 to 127), Unsigned (0 to 255)
- **BIT**, **BOOL:** synonyms for TINYINT
- **SMALLINT**:
  Signed (-32,768 to 32,767), Unsigned (0 to 65,535 – 64k)
- **MEDIUMINT**:
  Signed (-8388608 to 8388607), Unsigned (0 to 16777215 –16M)
- **INT / INTEGER**:
  Signed (-2,147,483,648 to 2,147,483,647),
  Unsigned (0 to 4,294,967,295 – 4G or 2^32)
- **BIGINT**:
  Signed (-9223372036854775808 to 9223372036854775807),
  Unsigned (0 to 18,446,744,073,709,551,615 - 2^64)
- **Don't** use the "(M)" number for integers

Number types (MySQL)

Real numbers (fractions)

- **FLOAT**: single-precision floating point, allowable values: -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38.

- **DOUBLE / REAL**: double-precision, allowable values: -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308.
- optional M = number of digits stored, D = number of decimals.
- Float and Double are often used for scientific data.

- **DECIMAL**[(M[,D])]: fixed-point type. Good for money values.
- M = precision (number of digits stored), D = number of decimals

Date Time types

Date/Time

- **DATE** 1000-01-01 to 9999-12-31
- **TIME** -838:59:59 to 838:59:59
  (time of day or elapsed time)
- **DATETIME** 1000-01-01 00:00:00 to 9999-12-31 23:59:59
- **TIMESTAMP** 1970-01-01 00:00:00 - ~ 2037
  Stored in UTC, converted to local
- **YEAR** 1901 to 2155

How is DATETIME different to TIMESTAMP?

– TIMESTAMP is for automatically adding “now” to a row

– DATETIME is for referring to a time in the real world

Use the Reference Manual

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

What's Examinable

- Need to be able to draw conceptual, logical and physical diagrams
  - Assignment 1: Conceptual Chen's pen and paper or Draw IO or any tool that can draw Chen conceptual model shapes
  - Physical Crow's foot with MySQL Workbench
- CREATE TABLE SQL statements

Next Lecture

- Lecture 6 Normalisation
- Lecture 7 Live!
- ER Modelling Please read the Medicare case study for L7!
  - Recorded Live on Tuesday 25th August at 11.00am AEST
  - Connect via the ZOOM link in the LMS
  - Questions in the Chat window