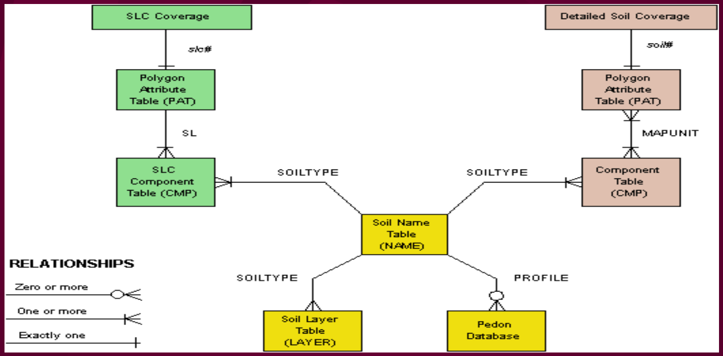
Advanced ER Model



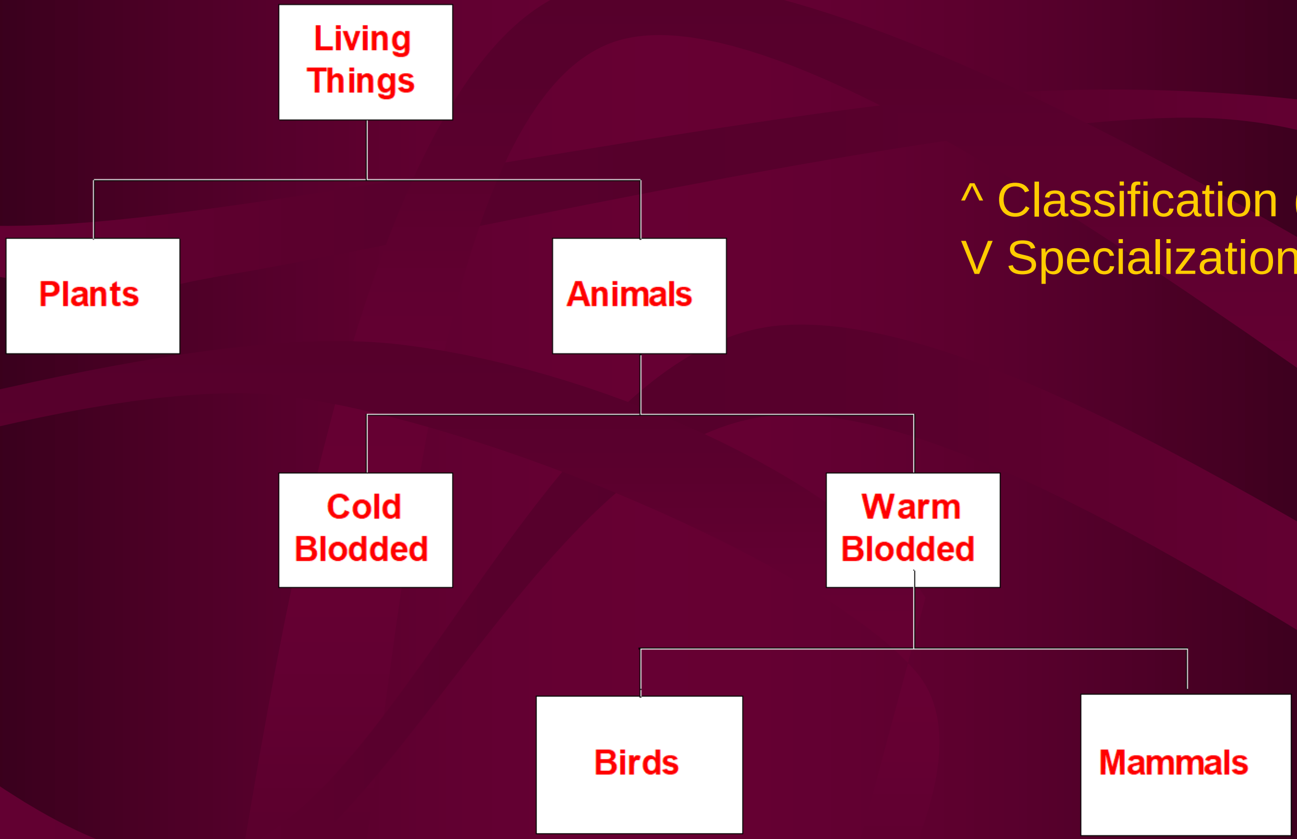
The Extended Entity Relationship Model (EERM)

- Enhanced (or extended) entity relationship model
  - Result of adding more semantic constructs to the original entity relationship (ER) model
  - Mostly involves specialization/generalization via “sub-types”
- Originally derived from IDEF1X model (see appendix)

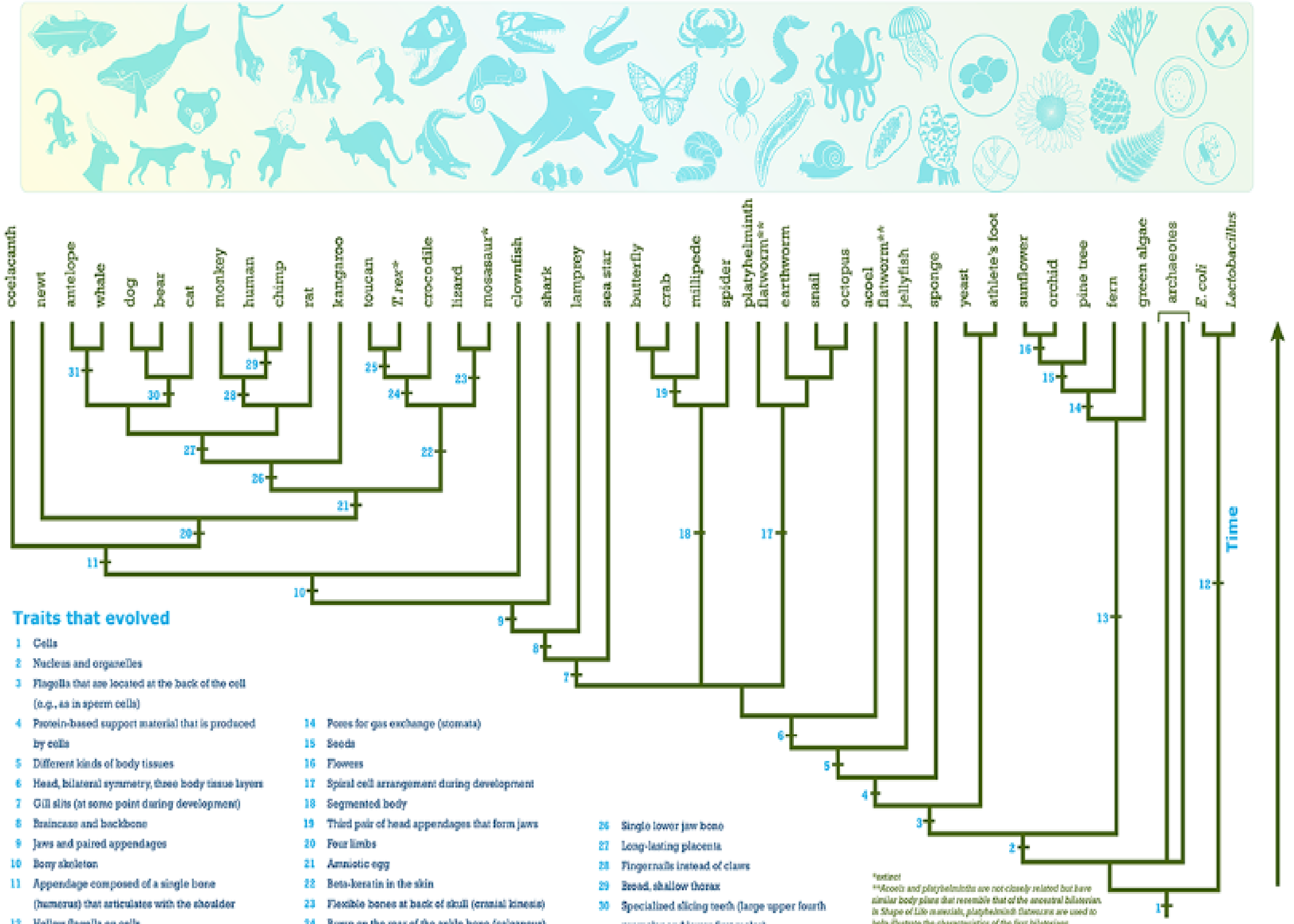
Specialization and Generalization

- Specialization
  - Top-down process
  - Identifies lower-level, more specific entity subtypes from a higher-level entity supertype
  - Based on grouping unique characteristics and relationships of the subtypes
- Generalization
  - Bottom-up process
  - Identifies a higher-level, more generic entity supertype from lower-level entity subtypes
  - Based on grouping common characteristics and relationships of the subtypes

Biological Inheritance Tree



^ Classification (up)
V Specialization (down)



Traits that evolved

1 Cells

2 Nucleus and organelles

3 Flagella that are located at the back of the cell (e.g., as in sperm cells)

4 Protein-based support material that is produced by cells

5 Different kinds of body tissues

6 Head, bilateral symmetry, three body tissue layers

7 Gill slits (at some point during development)

8 Braincase and backbone

9 Jaws and paired appendages

10 Bony skeleton

11 Appendage composed of a single bone (humerus) that articulates with the shoulder

12 Hollow flagella on cells

13 Cell wall made of cellulose

14 Pores for gas exchange (stomata)

15 Seeds

16 Flowers

17 Spiral cell arrangement during development

18 Segmented body

19 Third pair of head appendages that form jaws

20 Four limbs

21 Amniotic egg

22 Beta-keratin in the skin

23 Flexible bones at back of skull (cranial kinesis)

24 Bump on the rear of the ankle bone (calcaneus)

25 Wishbone or furcula (fused clavicle bones)

26 Single lower jaw bone

27 Long-lasting placenta

28 Fingernails instead of claws

29 Broad, shallow thorax

30 Specialized slicing teeth (large upper fourth premolar and lower first molar)

31 Double pulley ankle bone (astragalus)

*extinct

**Acoels and platyhelminths are not closely related but have similar body plans that resemble that of the ancestral bilaterian. In Shape of Life materials, platyhelminth flatworms are used to help illustrate the characteristics of the first bilaterians.

Modified from The Wellcome Tree of Life
http://www.wellcometreeoflife.org/interactive

Subtype Entities

- Entities with common identifier and some common attributes, but also some different attributes – "IS A" relationship
- Example: Patient who can be of several subtypes (outpatient, surgery, therapy, etc.); each with some common and some distinct attributes
- Subtypes may be exclusive; that is one and only one subtype instance is required for each supertype instance
- Subtypes "inherit" supertype ID and attributes

PATIENT - supertype entity class

- PatientNumber
- PatientName
- Address
- AmountDue

OUTPATIENT Subtype

- Date
- Procedure
- Doctor
- Facility

SURGERY PATIENT Subtype

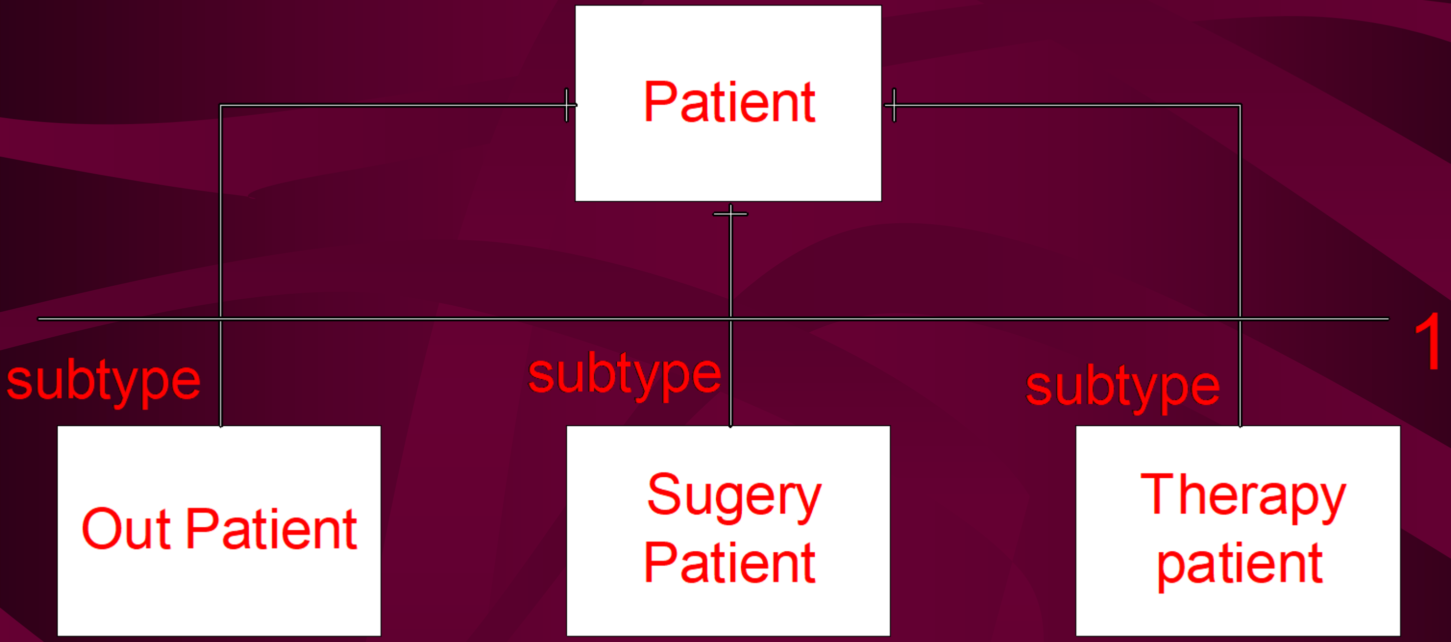
- Next of Kin
- CheckIn
- CheckOut
- Anesthesiologist
- Doctors
- Procedures

THERAPY PATIENT Subtype

- Therapist
- Therapy Schedule

Subtype Diagram

[exclusive subtypes, indicated with line and “1” or other notation]

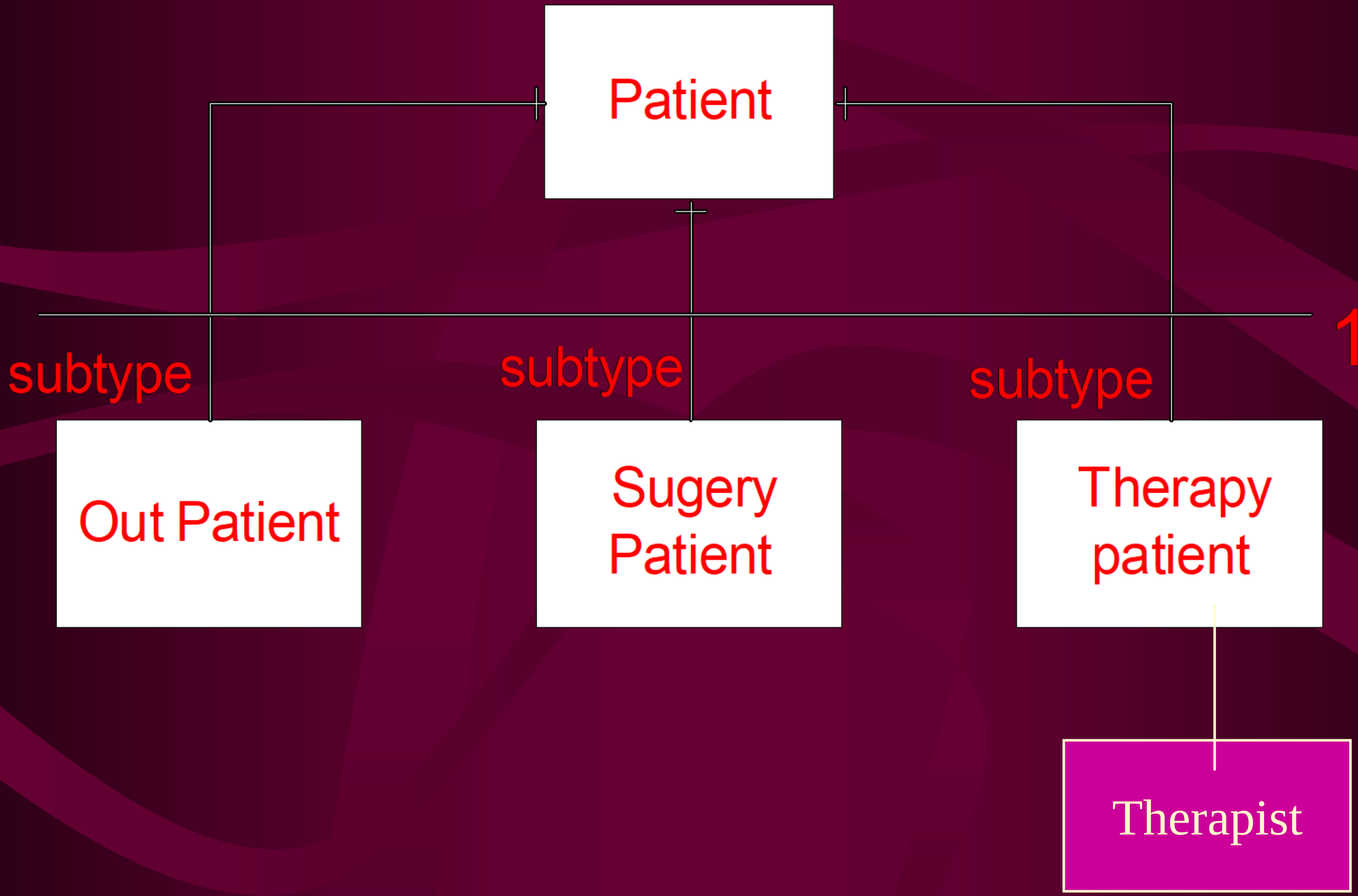


Copyright Dan Brandon, PhD,PMP   Christian Brothers University

Need for Subtypes

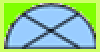
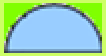
- If there were only an entity for patient in our database then we would have all the attributes in this one entity (PatientNumber, PatientName, Address, AmountDue, Date, Procedure, Doctor, Facility, Next of Kin, CheckIn, CheckOut, Anesthesiologist, Procedures, Therapist, Therapy Schedule); for any one type of patient much of the attributes would be unused and much space would be wasted
- Also some relationships might involve only certain subtypes

Relationships to Subtypes

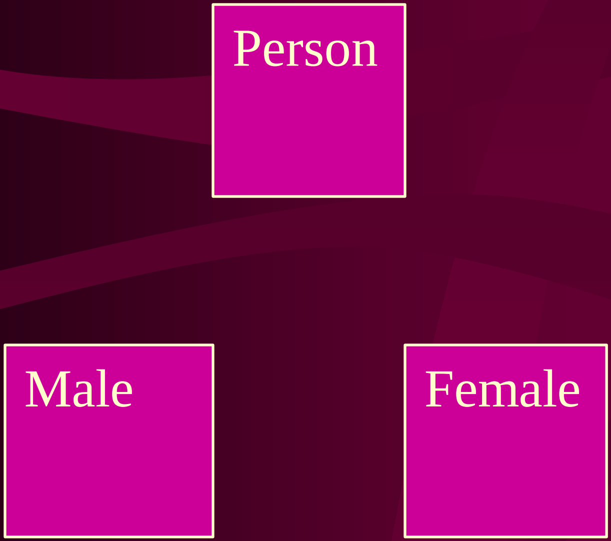


Exclusive and Inclusive Sub Types

- Each category or cluster (group) of subtypes can be exclusive or inclusive
- In an exclusive (disjoint) subtype group, the supertype is associated with at most one subtype
- In an inclusive (overlapping) subtype group, the supertype can be associated with more than one subtype
- Note that different authors or RDMS's may use different diagram symbols

	Exclusive Subtype Disjoint
	Inclusive Subtype Overlap

Exclusive and Inclusive Sub Types (con’t)

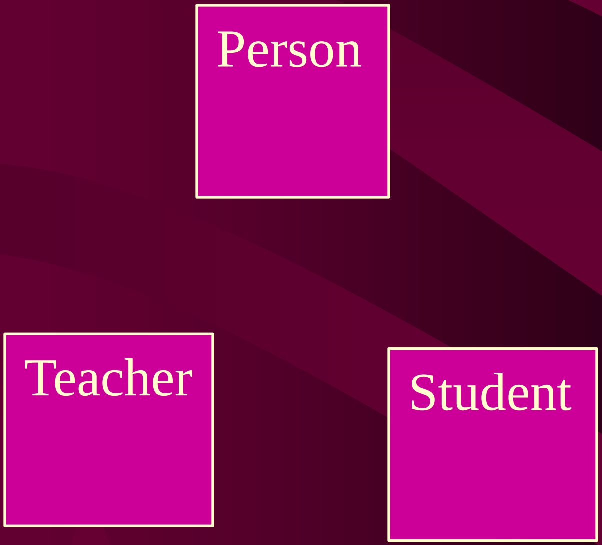


Person

Male

Female

Disjoint (Exclusive)– person can be male or female but not both



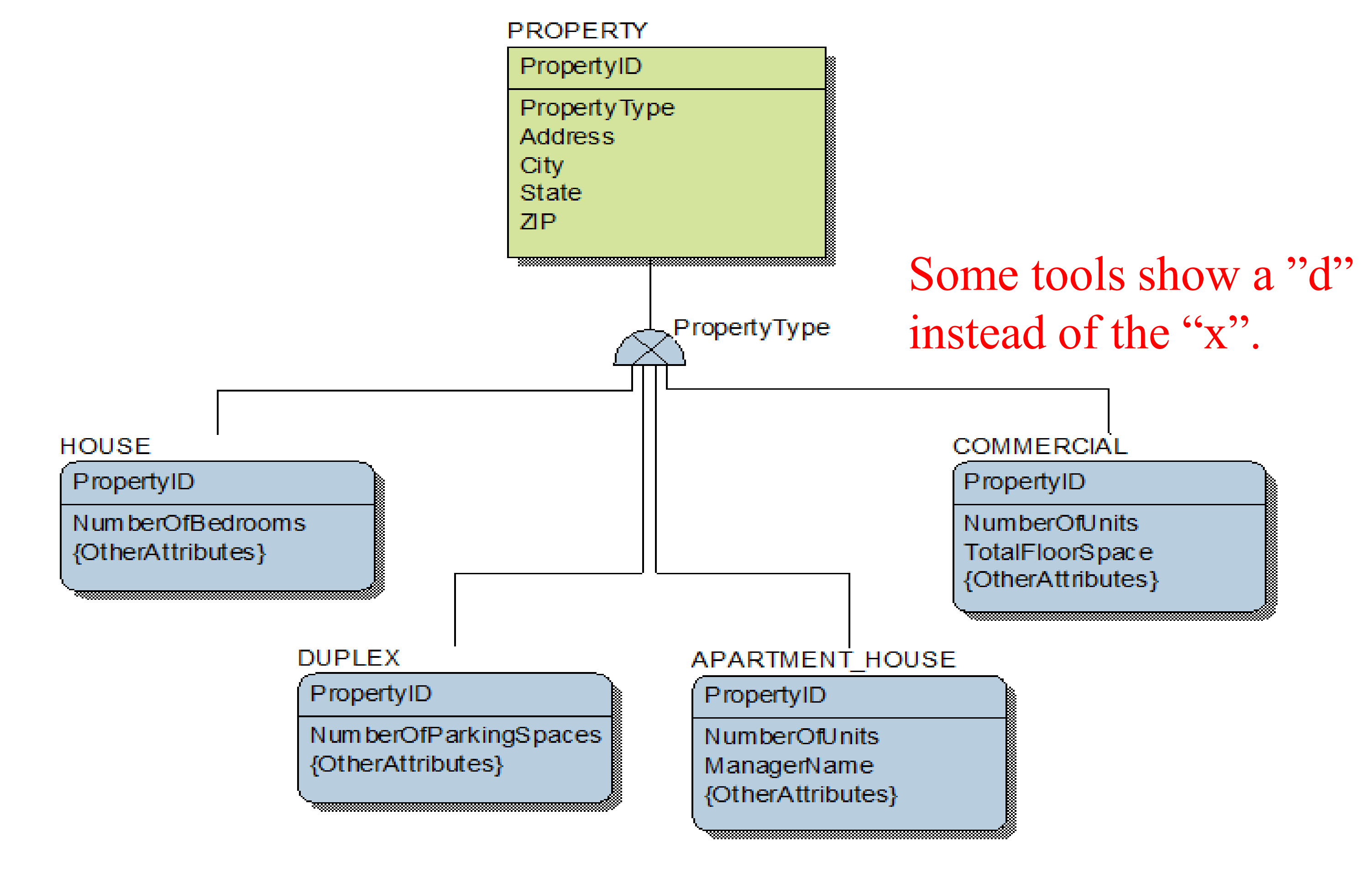
Person

Teacher

Student

Overlap (inclusive) – person may be both teacher and student

Exclusive (disjoint) Sub Type Group



PROPERTY
PropertyID
PropertyType
Address
City
State
ZIP

HOUSE
PropertyID
NumberOfBedrooms
{OtherAttributes}

COMMERCIAL
PropertyID
NumberOfUnits
TotalFloorSpace
{OtherAttributes}

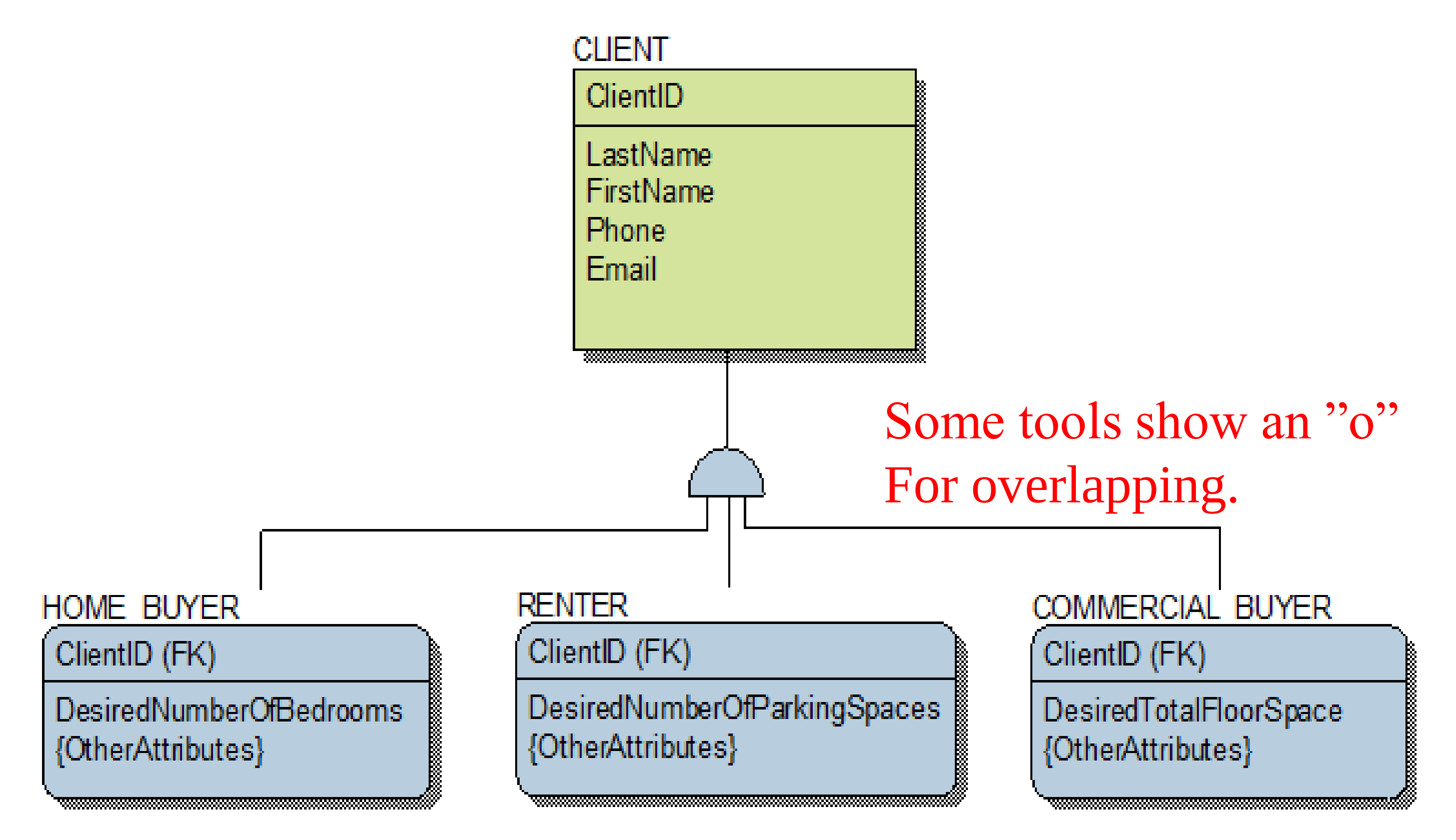
DUPLEX
PropertyID
NumberOfParkingSpaces
{OtherAttributes}

APARTMENT_HOUSE
PropertyID
NumberOfUnits
ManagerName
{OtherAttributes}

Some tools show a ”d” instead of the “x”.

Each property is just one subtype

Inclusive Sub Type Group



CLIENT
ClientID
LastName
FirstName
Phone
Email

HOME BUYER
ClientID (FK)
DesiredNumberOfBedrooms
{OtherAttributes}

RENTER
ClientID (FK)
DesiredNumberOfParkingSpaces
{OtherAttributes}

COMMERCIAL BUYER
ClientID (FK)
DesiredTotalFloorSpace
{OtherAttributes}

Some tools show an ”o”
For overlapping.

Each client can be more than one type

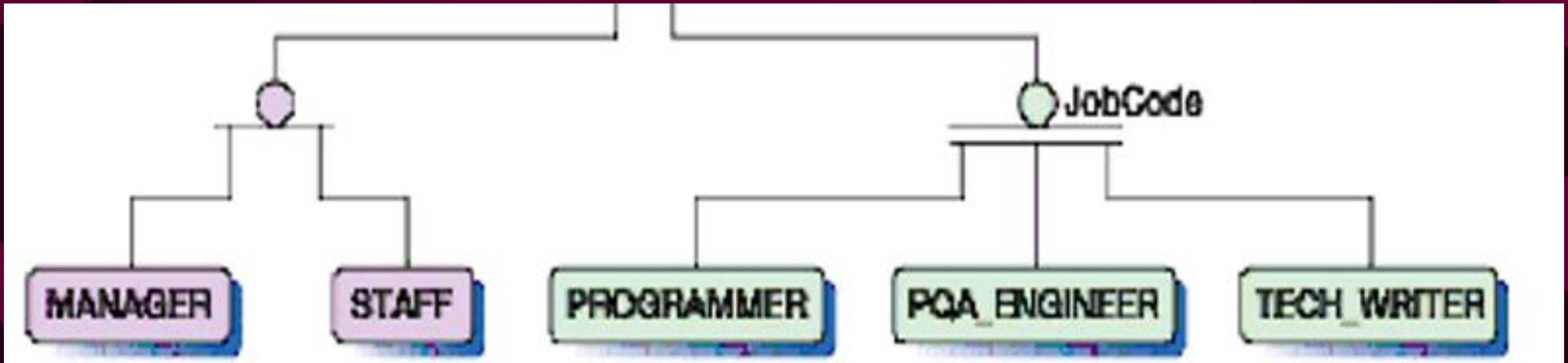
Completeness Constraint

- Specifies whether each supertype occurrence must also be a member of at least one subtype
  - Partial completeness (incomplete): not every supertype occurrence is a member of a subtype
  - Total completeness (complete): every supertype occurrence must be a member of at least one subtypes
  - Differing notations used

Completeness Constraint (con't)

- JobCode is a complete cluster – every employee is one of the 3 types
- Type is an incomplete cluster– an employee may be neither manager nor staff

EMPLOYEE



Exercise

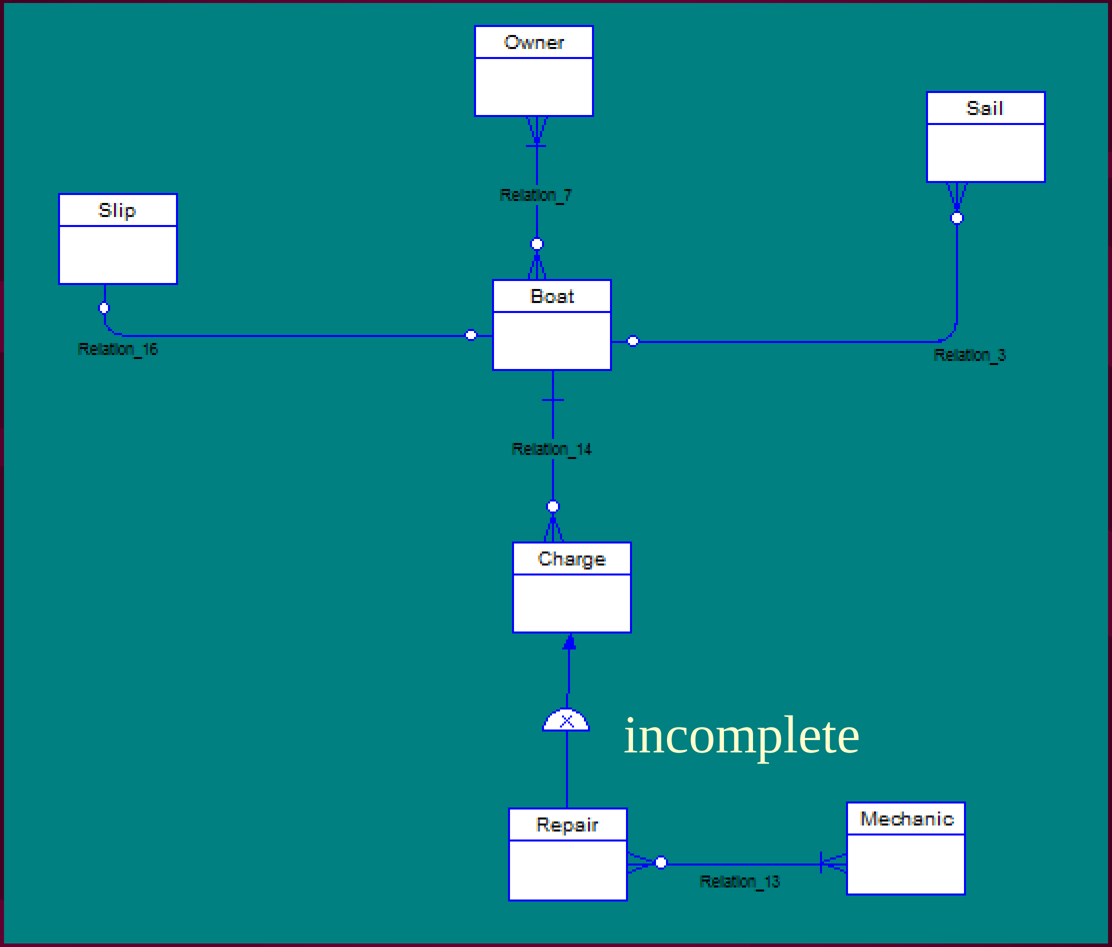
- Design a database for a marina which rents boat slips and sails, performs repairs to boats, and sells gasoline and other stuff for use on a boat
- The marina is not interested in a computer inventory system or otherwise tracking the stuff
- It really just wants to keep track of info to prepare monthly billing for rentals and other charges including who has rented what for which boat
- The marina assigns all credit charges to boats (so it can place liens on boats to collect money)
- It also has mechanics who do the repairs and needs to track which repairs involve which mechanics
- What might some of the entities be ?????

Exercise - Entities/Requirements

- Boat - a fiberglass lined hole in the water into which one pours money
- Slip - a parking place for a boat which is barely big enough to park the boat in
- Charge - slip rental, repairs, and other charges to boats
- Sail - an optional component of a boat
- Repairs - a repair is a charge, but not all charges are repairs
- Mechanics - More than one mechanic may be needed for a repair; keep track of hours per repair per mechanic
- Owner - the lucky guys who pay for it all
- Draw the E-R diagram !

Don’t look ahead !

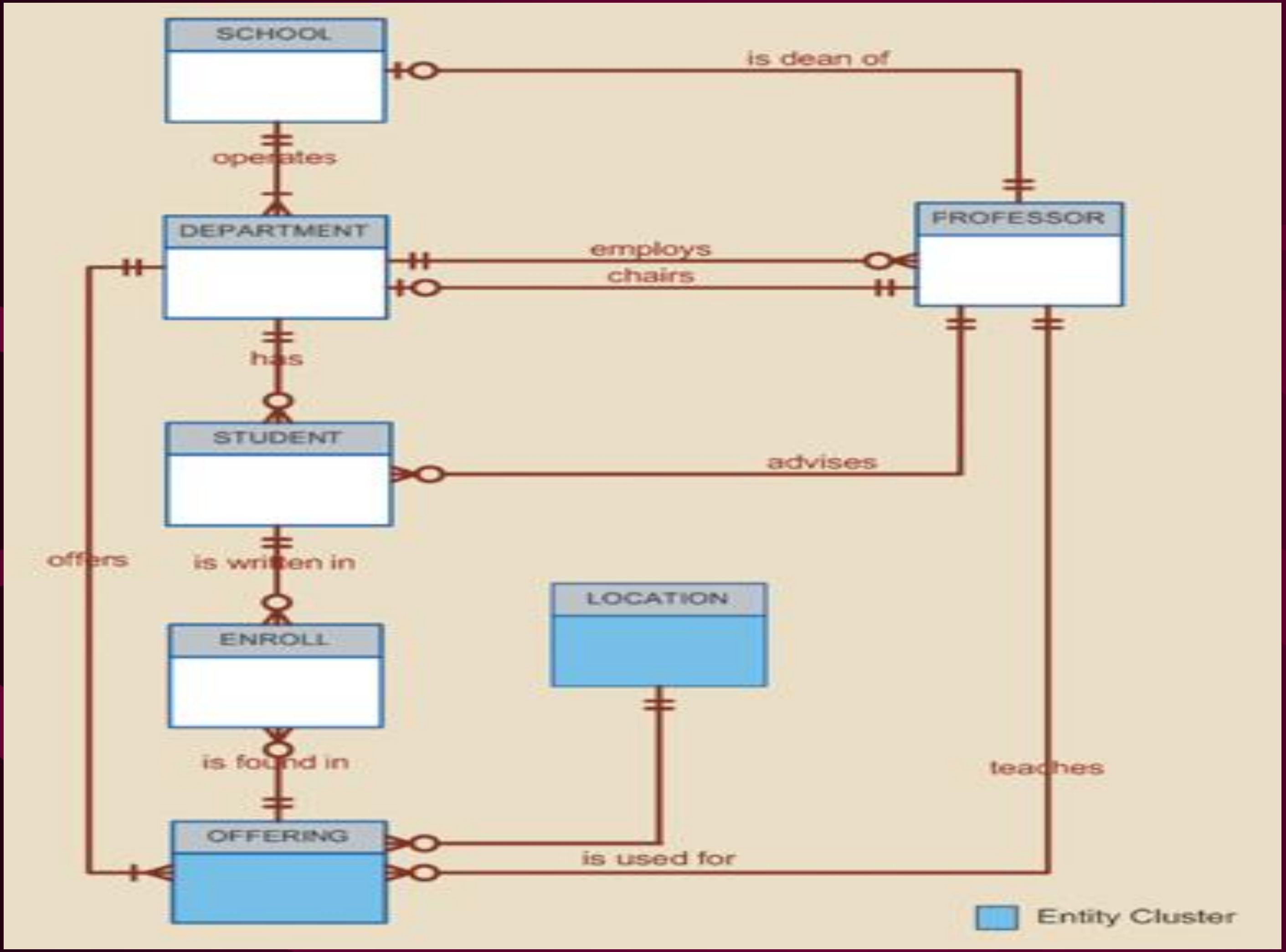
E - R Diagram


Owner
Sail
Relation_7
Slip
Boat
Relation_16
Relation_3
Relation_14
Charge
incomplete
Repair
Mechanic
Relation_13

Virtual (abstract) Entities

- An ER model may include virtual or abstract entities (often called entity clusters)
- A virtual entity is one that actually contains several entities
- Virtual entities must be separated into the underlying entities
- For example an ER diagram may contain a virtual entity for location that must be divided into city and state entities

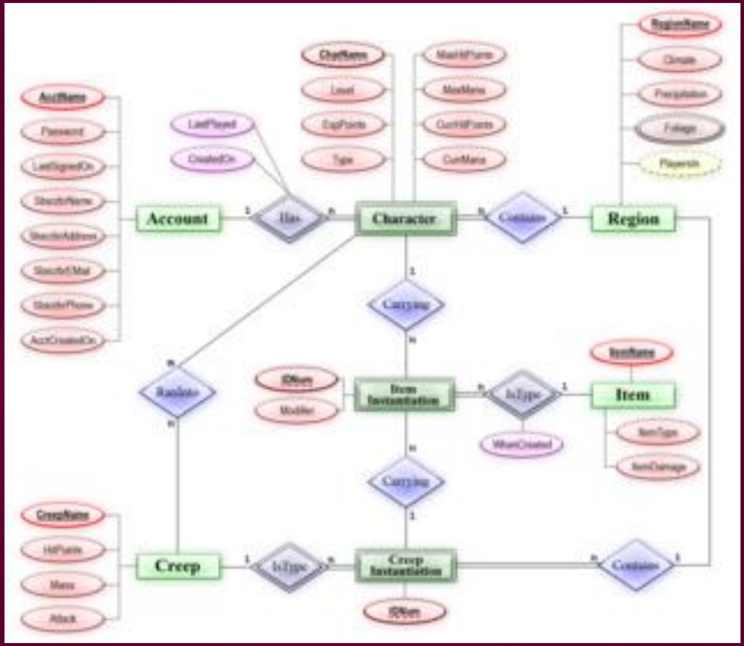
Virtual Entities (con’t)



Location would be divided into Building and Room

Offering would be Divided into Course and Class

Relational Database Design Using E-R Models



Moving from General Design to a Relational Design [from entity level to table level]

- Transformation of model information expressed in an E-R diagram into relational database
- Further Normalization (if necessary)
- Optimization
- General Constraints
- Other Business Rules/Processing
  - Defaults
  - Validation
  - Triggers & Stored Procedures
  - Recursion

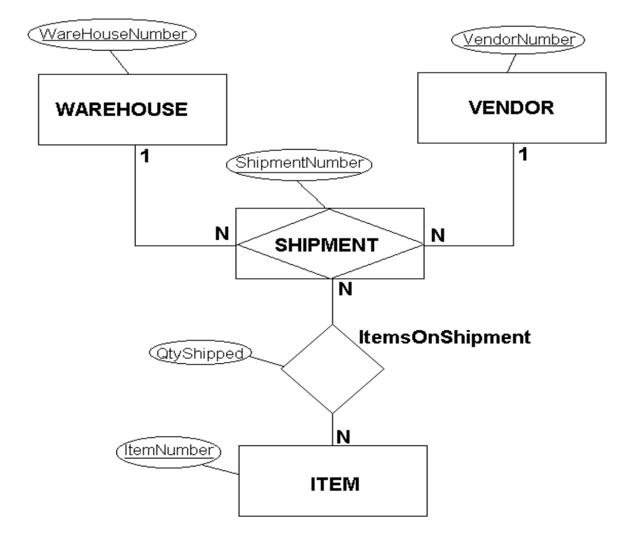
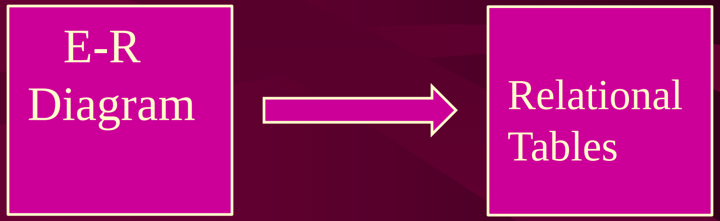


Figure 7.

- Transformation of model information expressed in an E-R diagram into relational database:



E-R Diagram

Relational Tables

  - Tables
  - Domains
  - Columns and Data Dictionaries
  - Primary Keys
  - Foreign Keys (relationships)
  - Defaults, valiation, masks
  - Indexes (for uniqueness, speed)
  - Constraints

Table Level Design Notation

- TABLENAME ($\underline{\text{field1}}$, *field2*, field3, …)

- Primary keys are underlined
- Foreign keys are shown in italics

Representing Entities

- E-R entities become relations (tables) in relational model
- Select a primary key (from candidate keys)
- Columns are entity attributes, ***and must be fully dependent on all of key***
- Specify column domains
- Specify any attribute level constraints, defaults, masks, validation

Choosing Primary Key

- Unique identifier of entity becomes primary key
- If there is more that one unique identifier (candidate or alternative keys), then one must be chosen as the primary key - ***every determinant must be a candidate key (Boyce Codd Normal Form) – normalization covered later in the course***
- All columns should be fully dependent upon all of key (Second Normal form)
- Constraints should be enforced by key or domains – DKNF (or by later code in application)

Primary Key Guidelines

- Desirable primary key characteristics
  - Non intelligent (no particular meaning)
  - No change over time (does not need to be modified)
  - Preferably single-attribute
  - Preferably numeric
  - Preferably less bytes
  - Security-compliant

Primary Key (con't)

- Consider an entity for items of equipment:
  - EQUIP (TagNumber, Manufacturer, Model, SerialNumber, Description, AcquisitionDate, PurchaseCost, UsefulLife)
- TagNumber would be the logical choice for a primary key – sometimes called a "natural key"

Primary Key (con't)

- What if there was no tag number?
- Then some other attribute(s) would have to be chosen for the primary key
- EQUIP (Manufacturer, Model, SerialNumber, Description, AcquisitionDate, PurchaseCost, UsefulLife)

Surrogate Keys

- If there was no suitable combination of unique attributes (for example there was no attribute for serial number), then a “surrogate” key would have to be created for the primary key
- In some products this is called an “auto-increment key”
- EQUIP (EQID, Manufacturer (Make), Model, SerialNumber, Description, AcquisitionDate, PurchaseCost)

Recommendations

- Use surrogate key not only when there is no suitable combination of attributes, but when multiple attributes are involved or when size of unique identifier is long
- ***Access time is proportional to log of key size***
- Create not-null unique index on natural keys to enforce uniqueness (Make, Model, SerialNumber)

Cautions on Surrogate Keys

- Many RDBMS have capability to auto-increment a key; be careful on:
  - Concurrency control, particularly in high volume operations
  - Skipped number problems
  - Consolidation of data from different databases and duplicate surrogate keys

Representing Relationships

- E-R relationships are represented by values called “foreign keys”
- ***A column is a foreign key in one table, if it is the primary key in another table:***
  - CUSTOMER (CID, name, …)
  - ORDER (orderNumber, *CID*, date, …)

Minimum Cardinality

- Foreign key columns are required (not null), if the minimum cardinality (child to parent) is one
  - CUSTOMER (CID, name, …)
  - ORDER (orderNumber, *CID*, date, …)
  - If every order needs a customer, then the minimum cardinality is 1 and the CID in ORDER cannot be null
- Or optional (null values are allowed), if the minimum cardinality (child to parent) is zero [not every order has a customer]

Representing Weak Entities

- A weak entity depends for its existence on another entity
- Need "referential integrity" constraints such as:
  - insert: cannot add weak entity without foreign key reference (parent) existence
  - deletion: cannot delete foreign key reference entity (parent), without first deleting children
- Handled directly by the DBMS, or by <u>triggers</u> on the relation, or in application code
- ***In IDEF model, the particular referential integrity constraints to be used are shown on the diagram***

ID Dependent Weak Entities

- Need to make the reference to the parent entity (foreign key) part of the primary key for the weak entity relation
- Consider information about cities:
  - STATE (stateCode, name)
  - CITY (cityName, *stateCode*, population)
  - CITY is dependent on STATE ("Jackson" is not sufficient to identify city, must indicate the state also)
  - The primary key on the CITY table is the combination of cityName and stateCode
  - stateCode is the primary key of the STATE table, and a foreign key in the CITY table; also stateCode in CITY table cannot be null

Relationship Classification

[based on max cardinality in each direction]

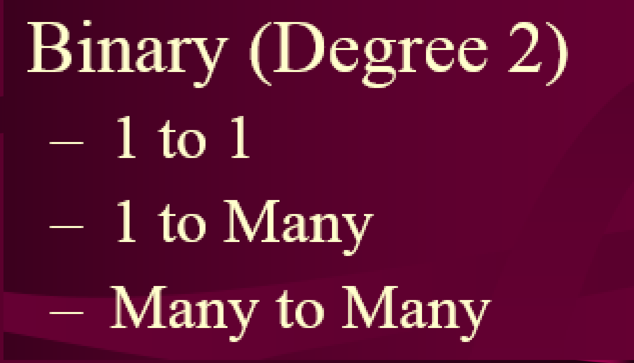
- Binary (Degree 2)
  - 1 to 1
  - 1 to Many
  - Many to Many
- Recursion (Degree 1)
  - 1 to 1
  - 1 to Many
  - Many to Many
- Higher Degree (> 2)
- Sub Types

RULES

- There are general rules on how to create relational tables for each of these relationship types
- The rules involve how many tables are created, and how foreign keys are created and used
- CASE products use the general rules to go from an E-R diagram to relational tables
- Other optimization rules and guidelines concern matters such as the use of indexes

Indexes

- Indexes are used to speed up access to data along a specific search path
- Unique indexes are also used to enforce uniqueness
- The primary key is typically automatically either indexed or hashed for fast access
- In some DBMS, the foreign key may be automatically indexed
- Need to specify other indexes through DBMS facilities or SQL Schema – more on this later in course

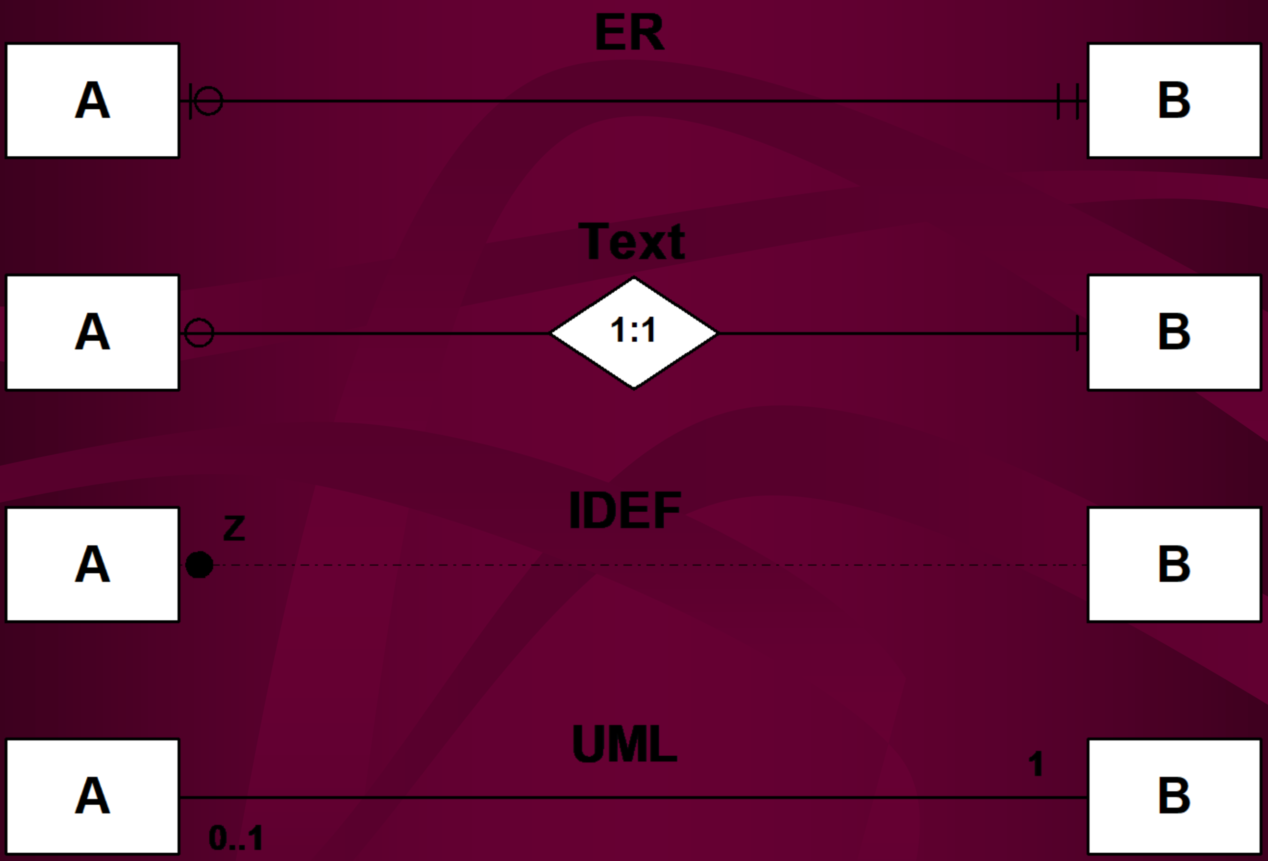


Binary (Degree 2)

- 1 to 1
- 1 to Many
- Many to Many

Degree 2 - Binary

One to One
[max cardinality in both directions is 1]

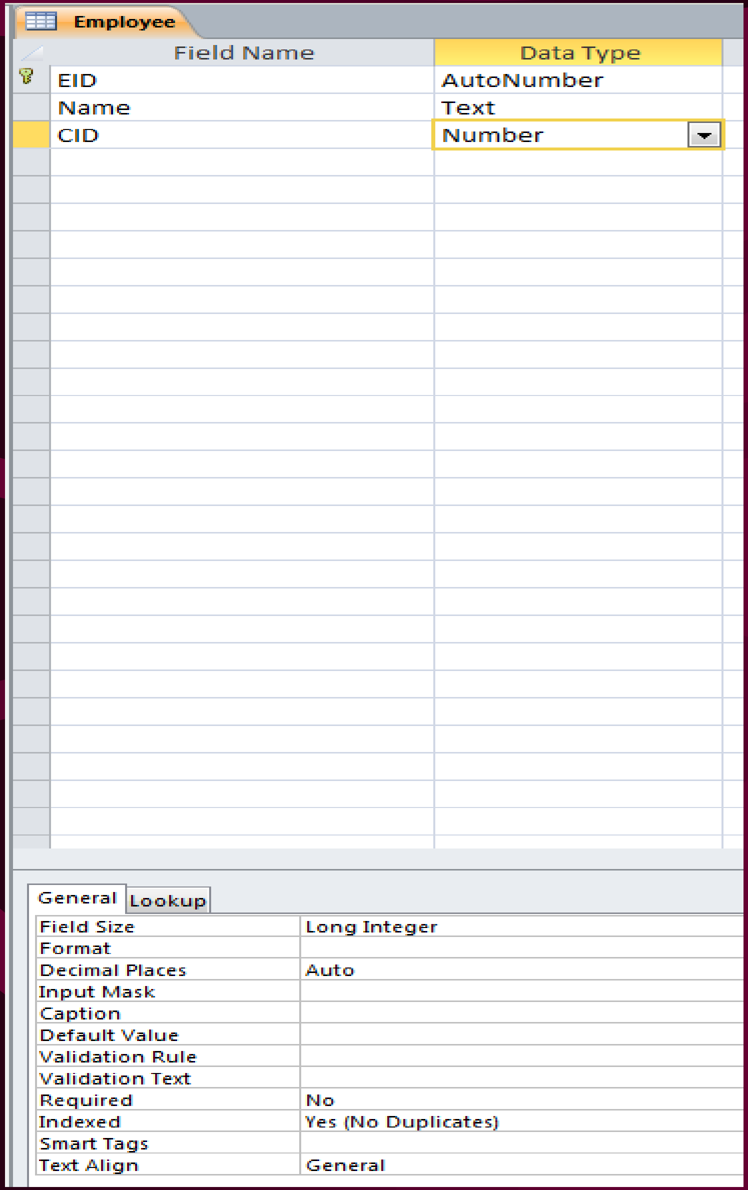


For example, each A has one and only one B, and each B has zero or 1 A.

One to One Rules

- Each entity is represented by a table
- Then the primary key of one table is included in the other table's attributes (becomes a foreign key)
- In ER diagrams, either (but not both) table can have the foreign key (child); but for IDEF the parent and child was selected when the diagram was drawn
- Consider employees and computers where each employee receives only one computer, and a computer is assigned to at most one employee
  - EMPLOYEE (EID, Name, ..., *CID*)
  - COMPUTER (CID, Type, ...)
- Use a foreign key in only one of the two tables, and place a unique index on that foreign key

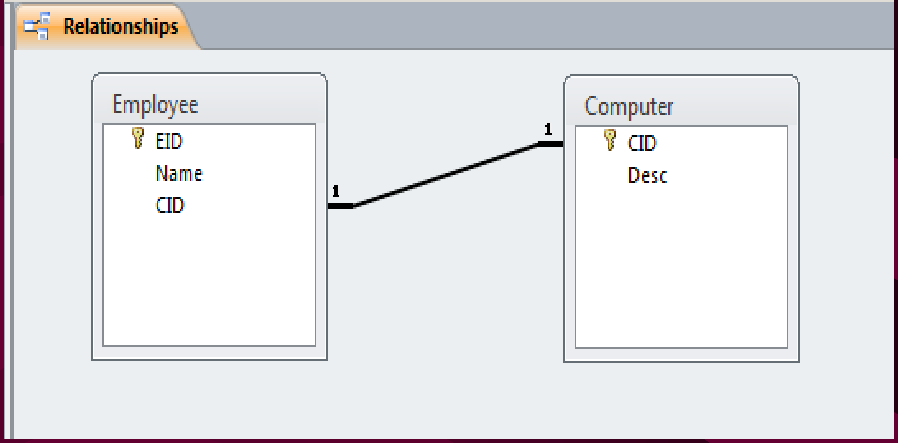
1 to 1 Relationship



Employee

Field Name	Data Type
EID	AutoNumber
Name	Text
CID	Number

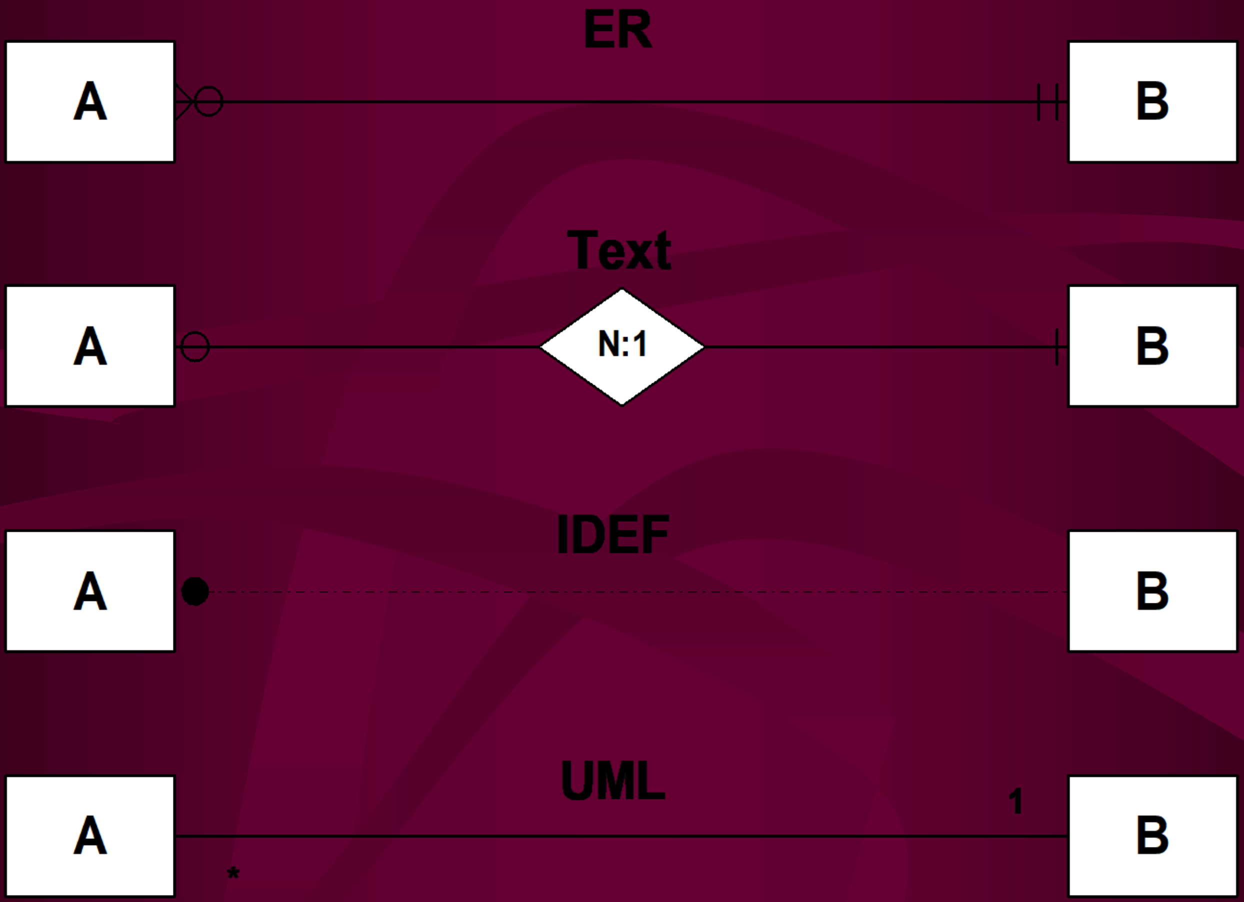
Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	
Text Align	General



One to One Rules (con't)

- In the employee-computer relationship the minimum cardinalities were zero
- If a minimum cardinality were one (i.e. each employee must be assigned a computer) , then that should be the table (employee) to hold the foreign key, and that column would not allow nulls
- In the case where both minimum cardinalities are one, the two entities may be combined in one table
- However if one entity is seldom accessed, or the use of the entities is independent, then it may be best to keep them in separate tables
- Also for security reasons, you may want two tables

One to Many



For example, each A has one and only one B, and each B has zero or many A.

One to Many Rules

- Sometimes called parent (one) to child (many) relationship
- Each entity is represented by its own table
- The parent (one side of relationship) primary key is included in the child’s (many side) attributes as a foreign key

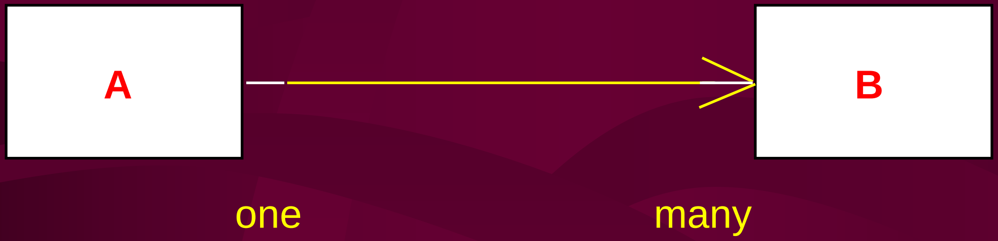
One to Many Rules (con't)

- Consider the relationship between publishers and books:
  - PUBLISHER (PID, Name, Address, ...)
  - TITLE (Isbn, Title, *PID*, ...)
- PID was moved into the TITLE table as FK, since each title has only one publisher
- Why can't we move Isbn to the Publisher table instead ????

One to Many Rules (con’t)

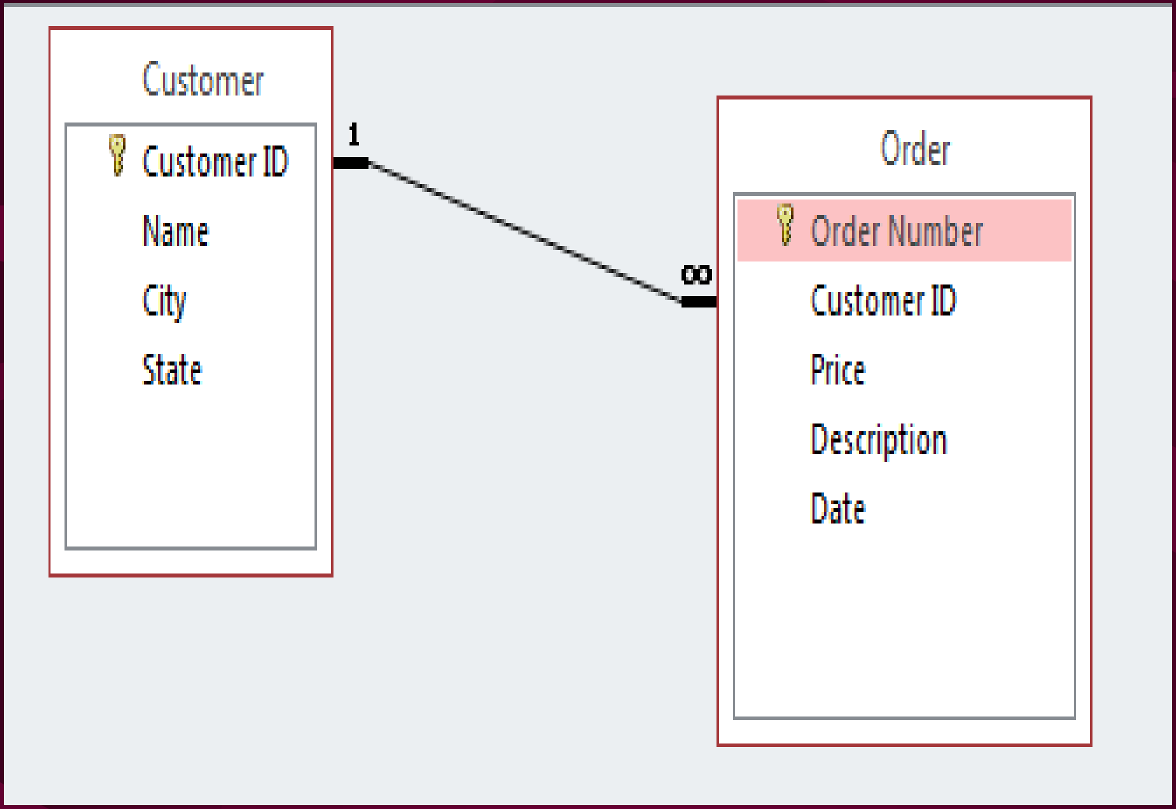
- An index on the FK speeds access in both directions
- ID dependent weak entities (as children) will already have the FK in their attributes (since it’s part of their primary key); for example STATE and CITY

One to Many Summary



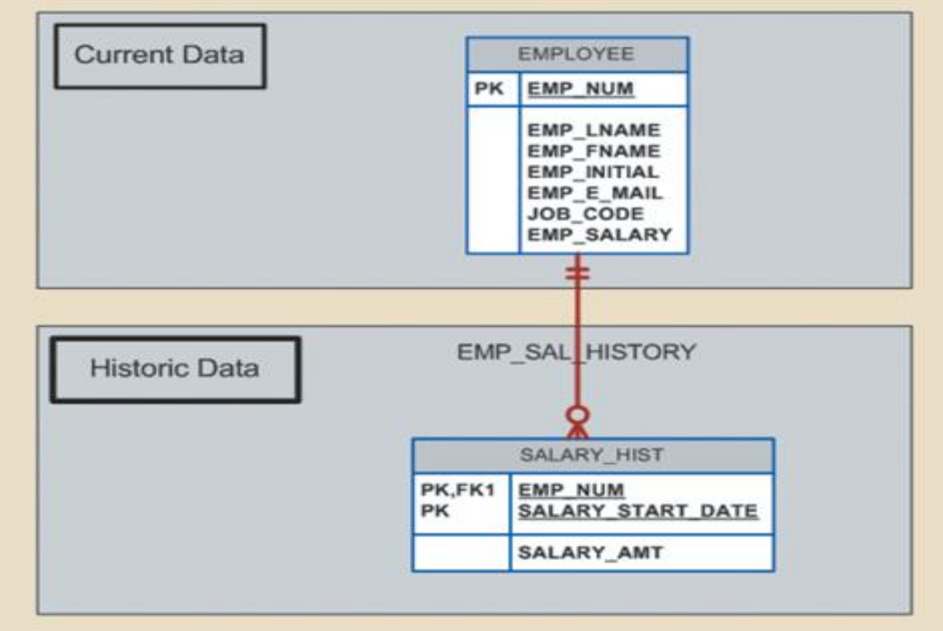
Primary key of parent entity A is placed in child entity B (many side)

1 to Many Relationship

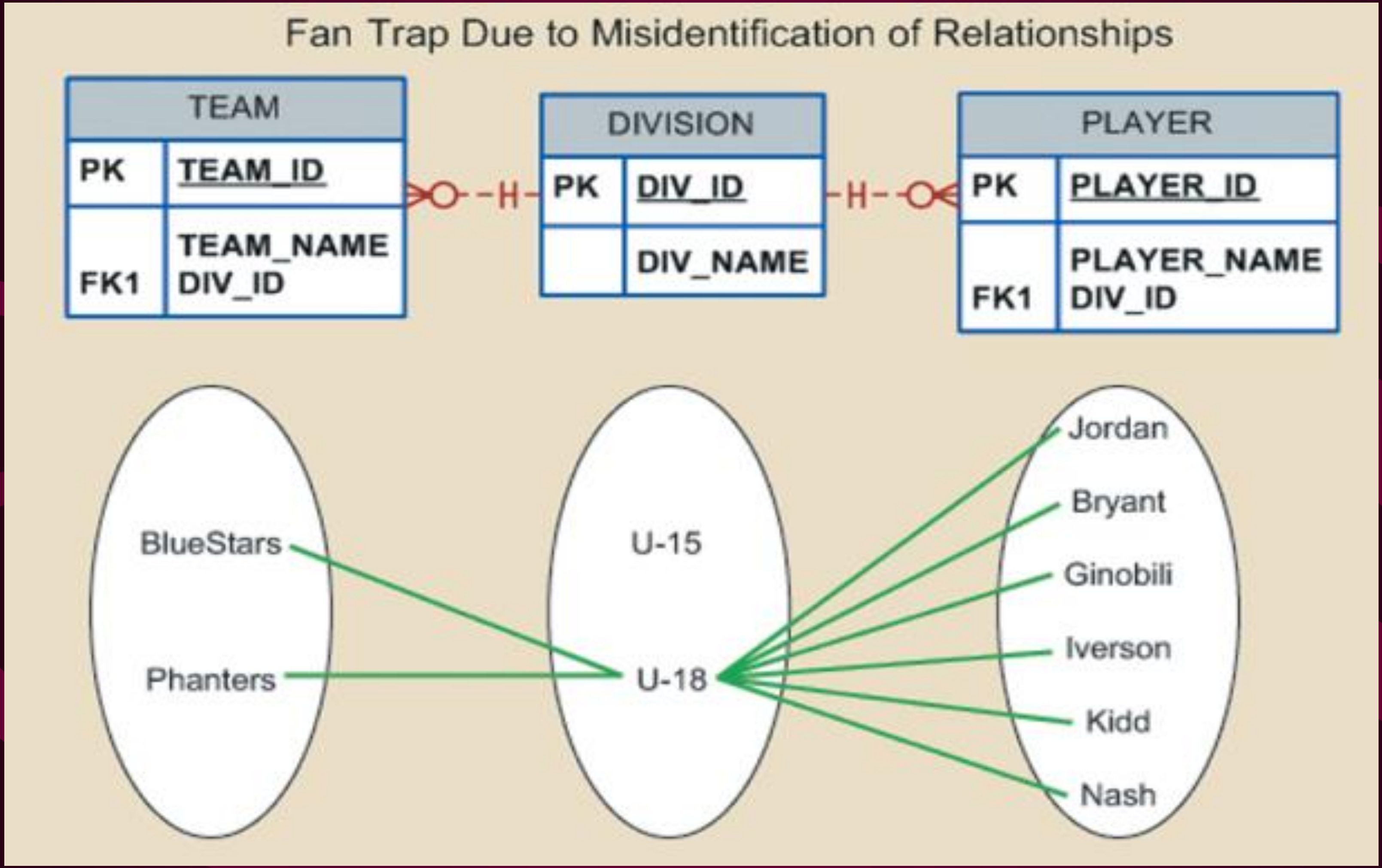


Handling History Data

- History data is typically set up as a 1 to many relationship
- In the diagram, both the current salary and salary history is maintained for each employee



“Fan Trap”



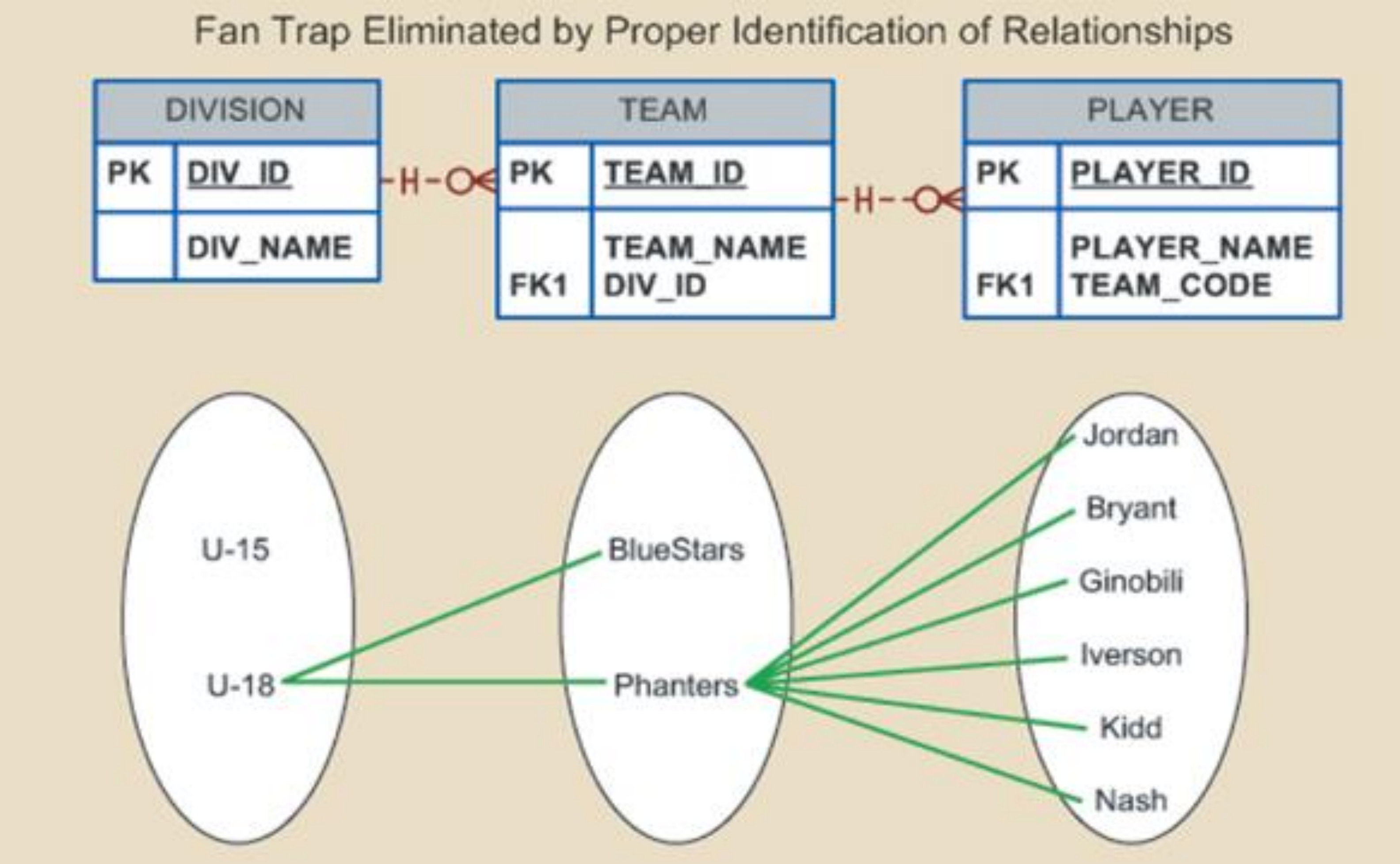
Fan Trap Due to Misidentification of Relationships

TEAM	
PK	TEAM_ID
 FK1	TEAM_NAME DIV_ID

DIVISION	
PK	DIV_ID
	DIV_NAME

PLAYER	
PK	PLAYER_ID
 FK1	PLAYER_NAME DIV_ID

Fan Trap Removed



Fan Trap Eliminated by Proper Identification of Relationships

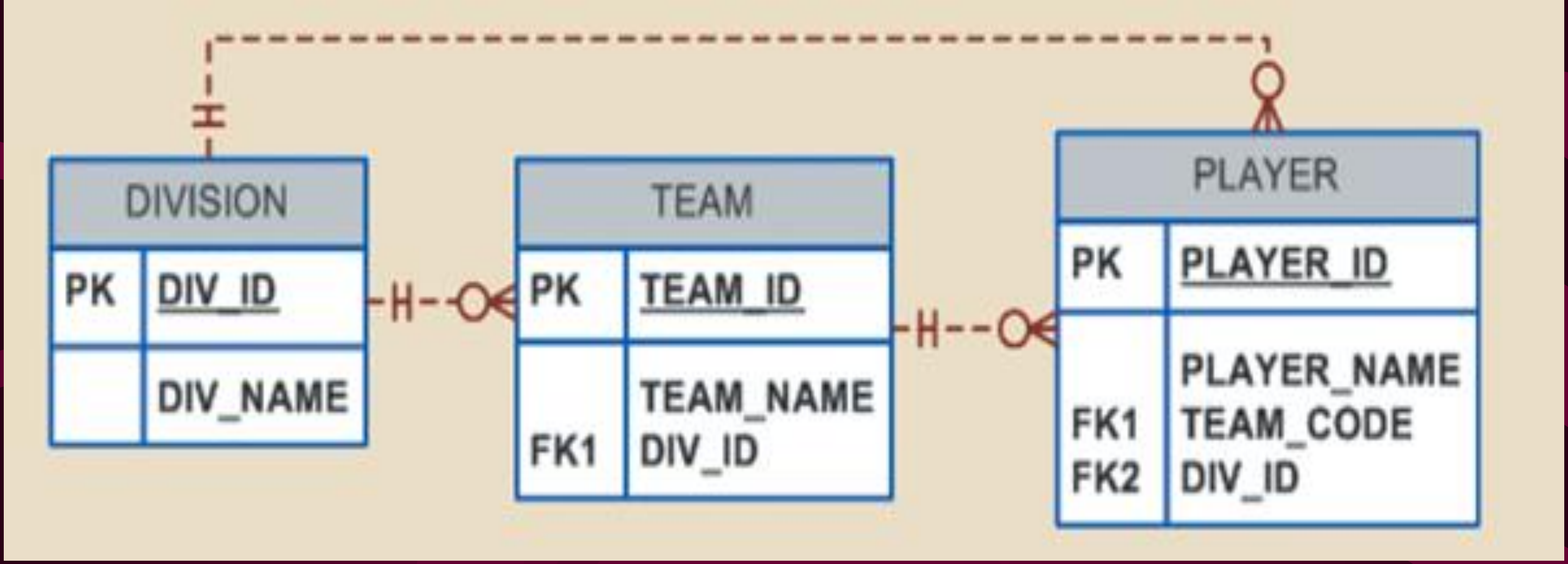
DIVISION	
PK	DIV_ID
	DIV_NAME

TEAM	
PK	TEAM_ID
	TEAM_NAME
FK1	DIV_ID

PLAYER	
PK	PLAYER_ID
	PLAYER_NAME
FK1	TEAM_CODE

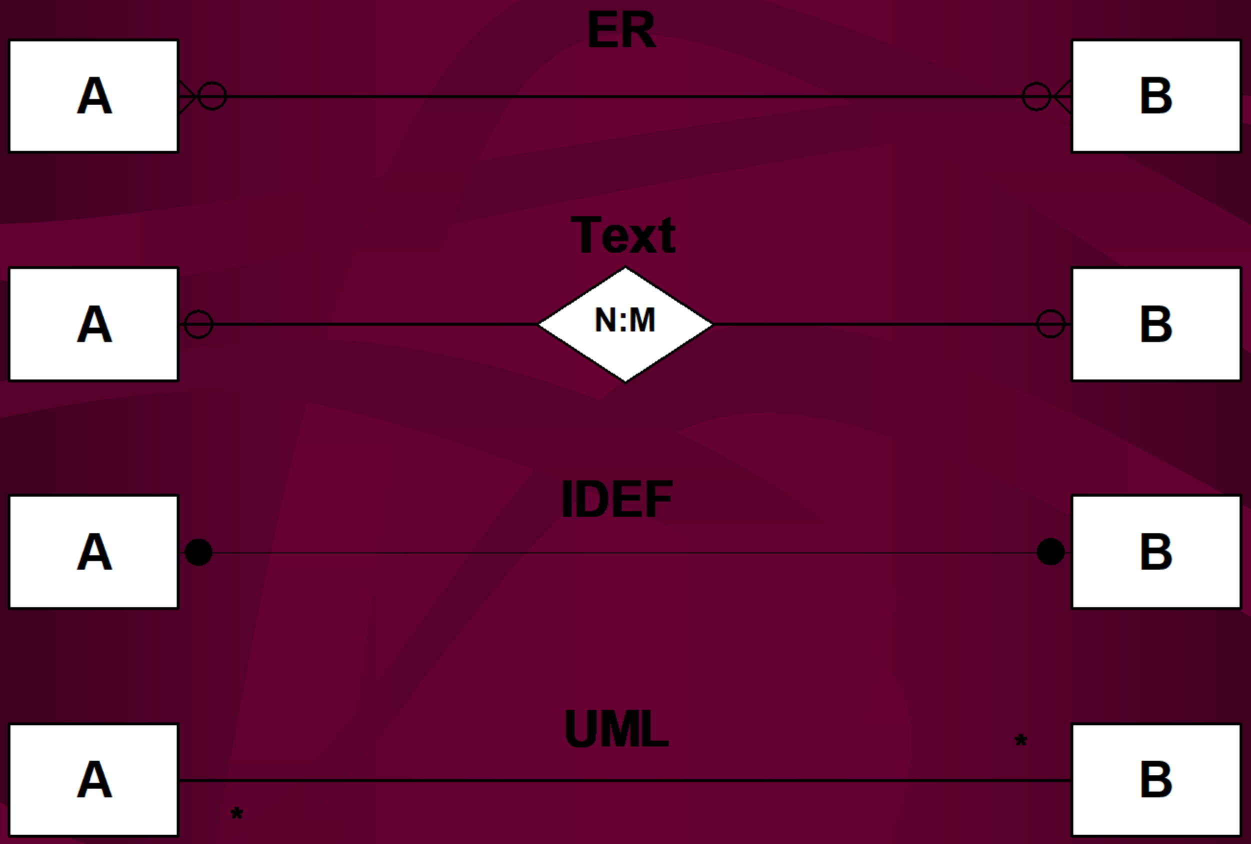
Redundant Relationships

(no need for relationship between Division and Player)



Since we know which players are on a team
and which teams are in a division, we can determine
which players are in a division.

Many to Many



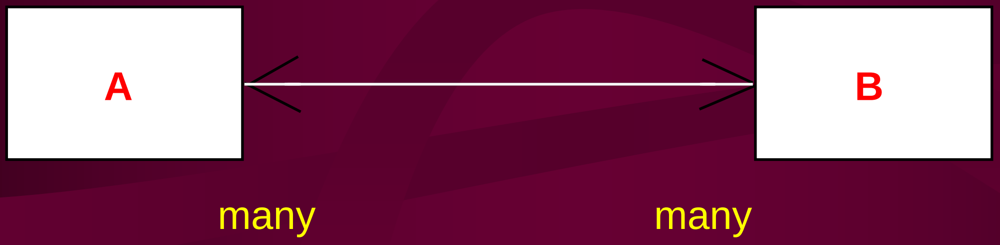
Many to Many

- Each entity is represented by its own table
- A third table needs to be created for the relationship itself
- This new table includes foreign keys to each of the other two tables
- This new table also includes attributes that are properties of the relationship (not the original entities)

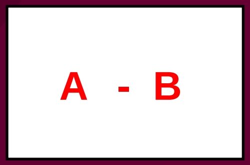
Many to Many (con’t)

- The name of the new table (sometimes called an intersection table or associative or composite entity) is the name of the relationship, if one; if not the table can be named by a combination of the names of the original two tables
- The primary key of the new table is the combination of the two foreign keys; the order is important and should be chosen for the order in which the table is normally processed ! (the inverse order can be set up as an index)

Many to Many Summary



Primary key of each table is combined to form primary key of new relationship table



Many to Many (con’t)

- Consider the relationship between authors and books, where an author can write many books and a book can be written by many authors:
  - – AUTHOR (AID, Name, ...)
  - – TITLE (Isbn, Title, *PID*, ...)
- Can we add the Isbn to the AUTHOR table to keep track of the books for each author ?
- Can we add the AID to the TITLE table to keep track of the authors for each title ?

Intersection Table

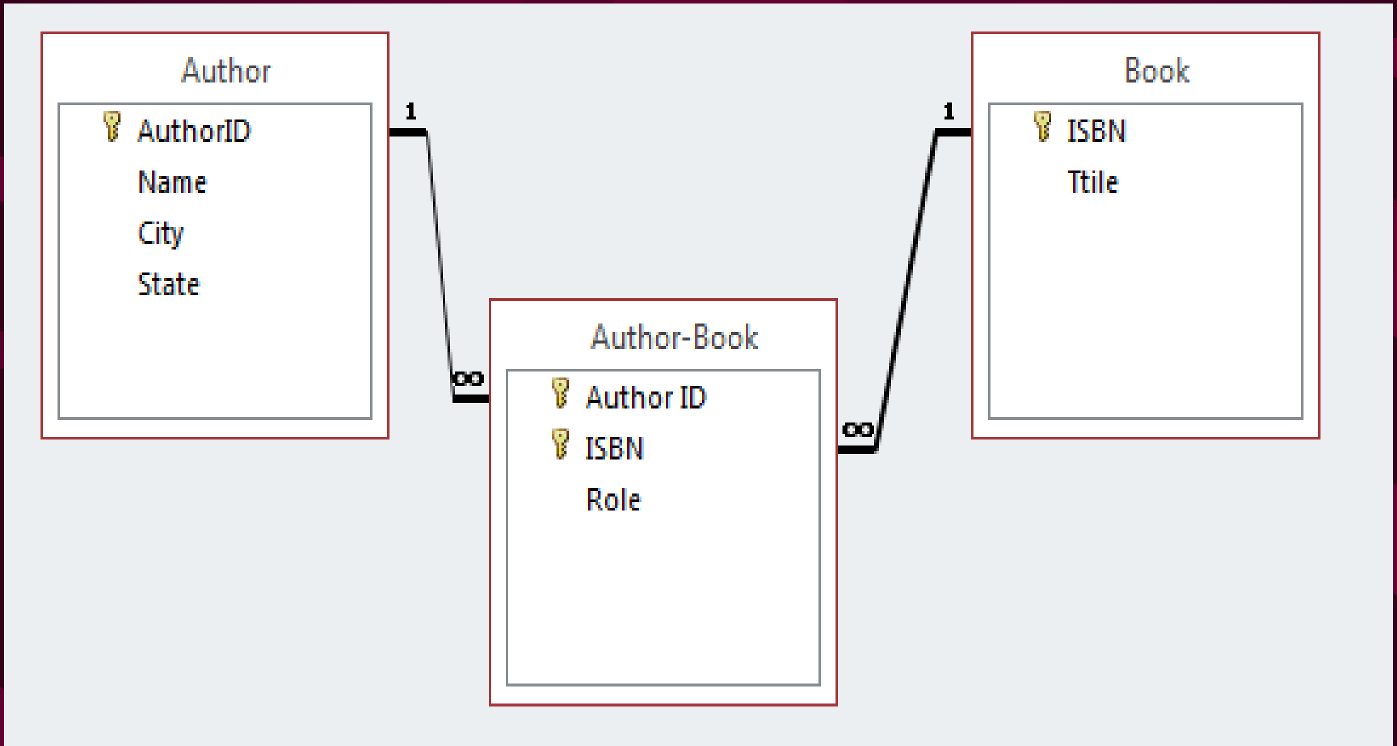
- AUTHOR-TITLE (*AID, Isbn*, Role, ...)

- Here role was an attribute on the relationship between titles and authors !
- AID should be first, if we more commonly access the table in author order: “list the books for each author”

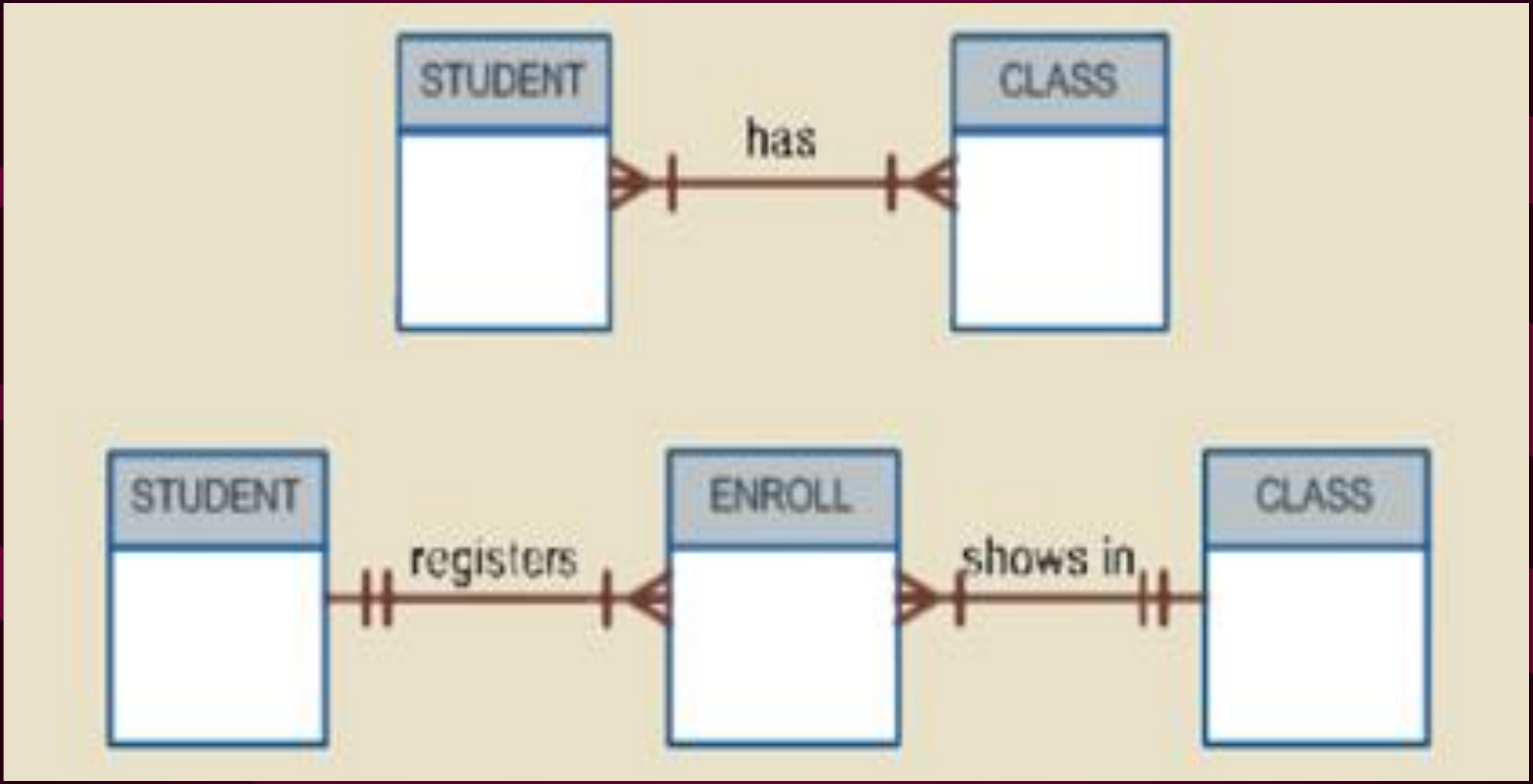
Intersection Table

AID	Isbn	Role
34	54321	author
21	54321	contributing author
34	12345	contributing author
16	98765	co-author
21	98765	co-author

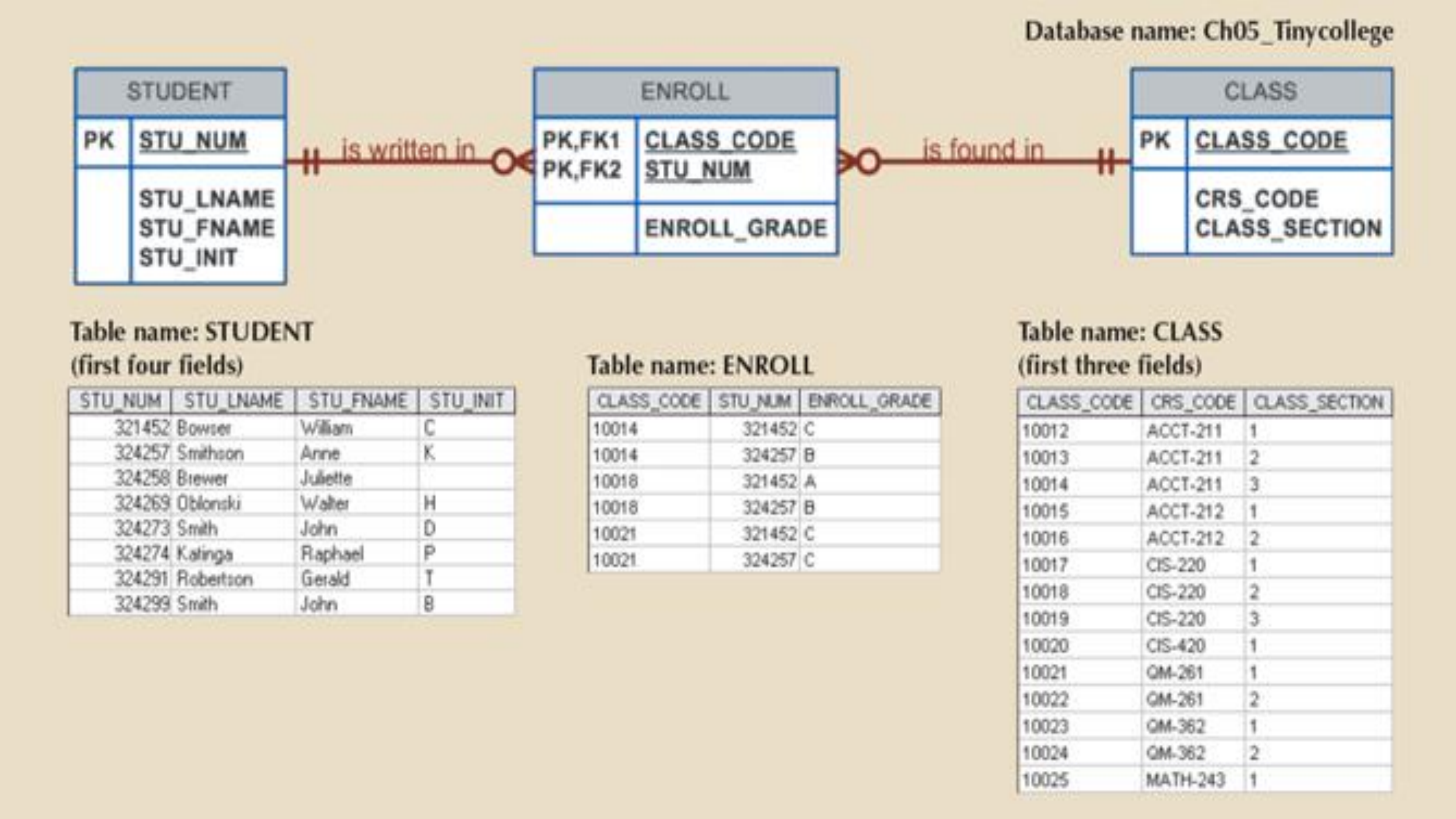
Many to Many Relationship



Enroll is an Intersection Table



Intersection Table Keys



Database name: Ch05_Tinycollege

Table name: STUDENT
(first four fields)

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT
321452	Bowser	William	C
324257	Smithson	Anne	K
324258	Brewer	Juliette	
324269	Oblonski	Walter	H
324273	Smith	John	D
324274	Katinga	Raphael	P
324291	Robertson	Gerald	T
324299	Smith	John	B

Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
(first three fields)

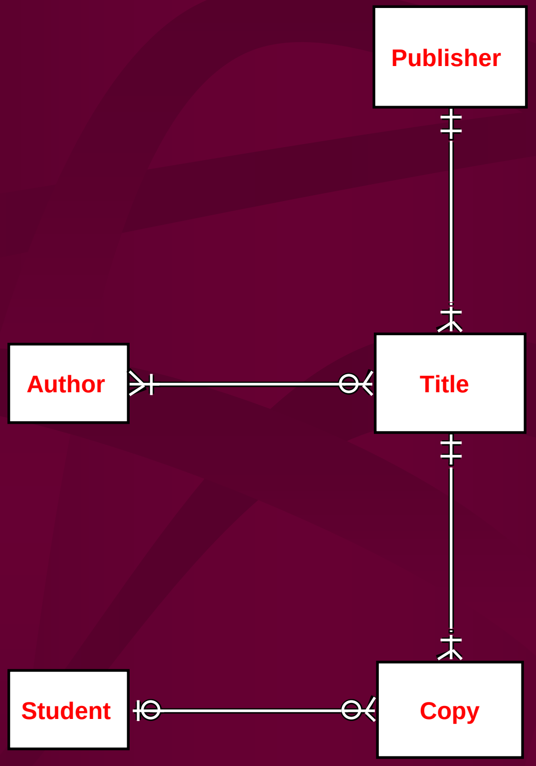
CLASS_CODE	CRS_CODE	CLASS_SECTION
10012	ACCT-211	1
10013	ACCT-211	2
10014	ACCT-211	3
10015	ACCT-212	1
10016	ACCT-212	2
10017	CIS-220	1
10018	CIS-220	2
10019	CIS-220	3
10020	CIS-420	1
10021	QM-261	1
10022	QM-261	2
10023	QM-362	1
10024	QM-362	2
10025	MATH-243	1

Exercise

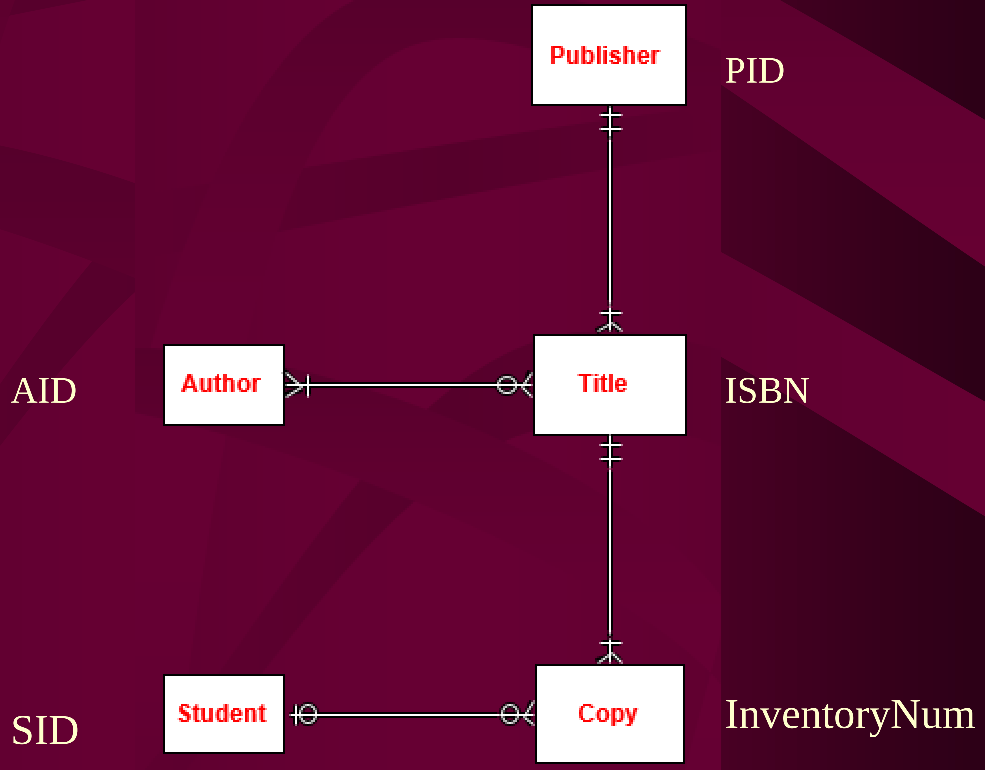
- Draw the E-R Diagram for the relationships involving:
  - Publishers
  - Titles (book titles)
  - Authors (write books)
  - Copies (copies of titles) - each copy is bar coded, and the library has at least one copy of each book
  - Students (check out copies)
    - for this model we are interested in who has the book is **currently** checked out

Don’t look ahead !

E-R Diagram

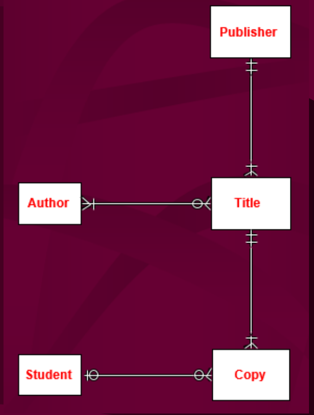


- Now specify the relational tables !
  - Underline PK's
  - Circle FK's (indicate null or not)



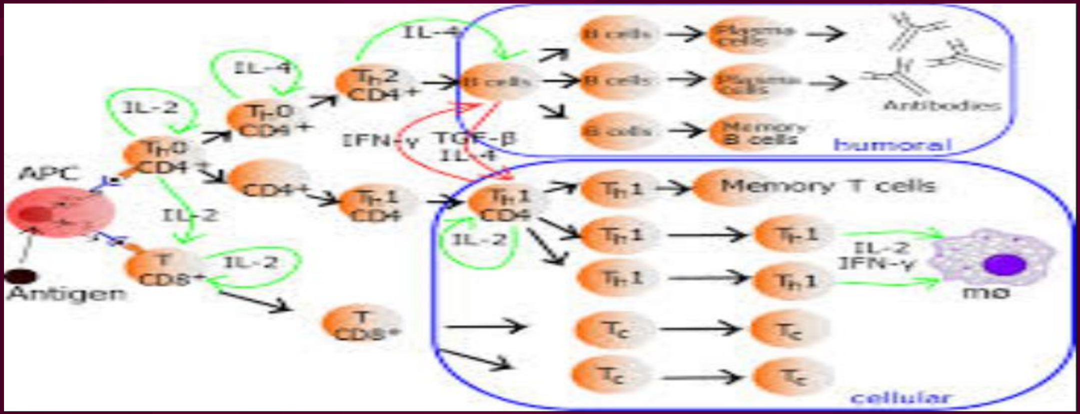
Don’t look ahead !

Relational Model



- PUBLISHER (PID, Name, ...)
- AUTHOR (AID, Name, ...)
- TITLE (Isbn, Title, *PID*, ...) {PID cannot be null}
- AUTHOR-TITLE (*AID, Isbn*, Role, ...)
  - Attributes on relationships added to PDM/Tables (ie. Role), just noted on CDM (E-R)
- STUDENT (SID, Name, ...)
- COPY (InventoryNum, *Isbn*, Condition, *SID*, DueBack,...) {SID can be null, Isbn cannot}

Sub-Type Relationship



Subtype Rules

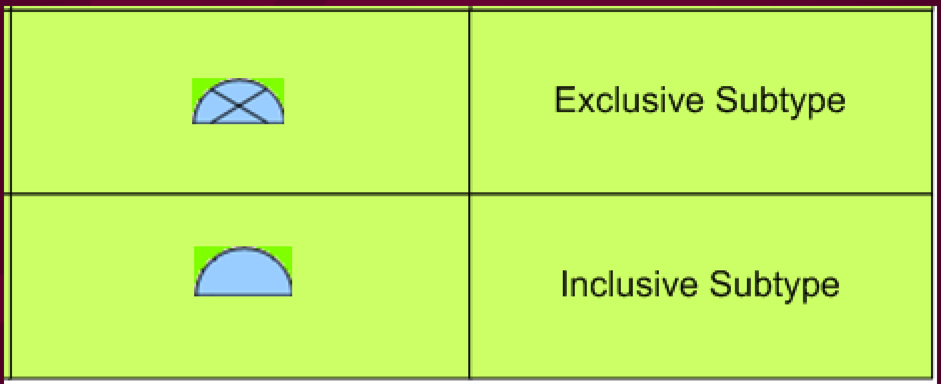
- An entity can have multiple subtypes (clusters)
- Depending on the cardinalities, the supertype (generic) entity may have more than one subtype and the relationship can have exclusivity constraints
- The supertype entity becomes one table and each subtype becomes another table
  - The supertype table contains all common attributes as its columns
  - The subtype tables contain their unique columns

Sub Type Keys

- The supertype PK is also the PK of each subtype
- In the subtype, that PK is typically is a foreign key also depending upon:
  - Whether the sub type can "stand alone" (exist without the super type)
  - You may need to do referential integrity checking between sub types and supertypes; normally maintenance (add, modify, delete) is done via forms which may automatically handle the super and subtype together and/or the referential integrity rules are handled in the code

Exclusive and Inclusive Sub Types

- Each category (group) of subtypes can be exclusive or inclusive
- In an exclusive (disjoint) subtype group, the supertype is associated with at most one subtype
- In an inclusive (overlapping) subtype group, the supertype can be associated with more than one subtype
- Note that different authors or RDMS’s may use different diagram symbols



⊗	Exclusive Subtype
◠	Inclusive Subtype

Or “D” for disjoint

Or “O” for overlap

Discriminator

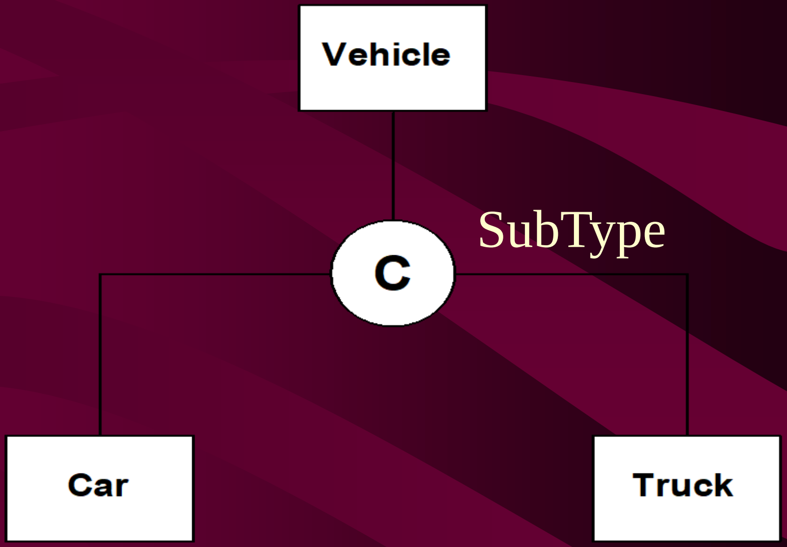
- A "type" indicator (IDEF discriminator) may be added to the parent record for each category cluster to indicate which sub type applies for faster access to subtype information
- For overlapping (inclusive) clusters, multiple discriminators are used and these are binary (true/false) attributes
- For a "complete" cluster (all subtypes have been specified), this is indicated by not allowing the discriminator to be null

Completeness and Disjoint Constraints

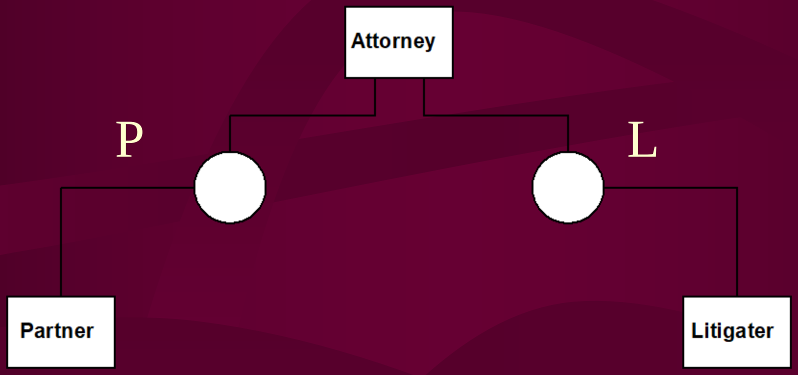
Complete	Disjoint Constraint	Overlapping Constraint
Partial	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

A vehicle is either a car or truck, cannot be both, must be one or the other

- One complete disjoint cluster:
  - VEHICLE (<u>VIN</u>, **SubType**, Make, Model,...
    - “Not Null” specification on SubType (required)
    - SubType constrained to be either ‘C’ or ‘T’
  - CAR (<u>*VIN*</u>, Color, ...
  - TRUCK (<u>*VIN*</u>, GVW, Axles, ...
- Note that without the “SubType” discriminator, we could still determine if a vehicle was a car or a truck, but with the discriminator we don’t have to search thru the subtype table



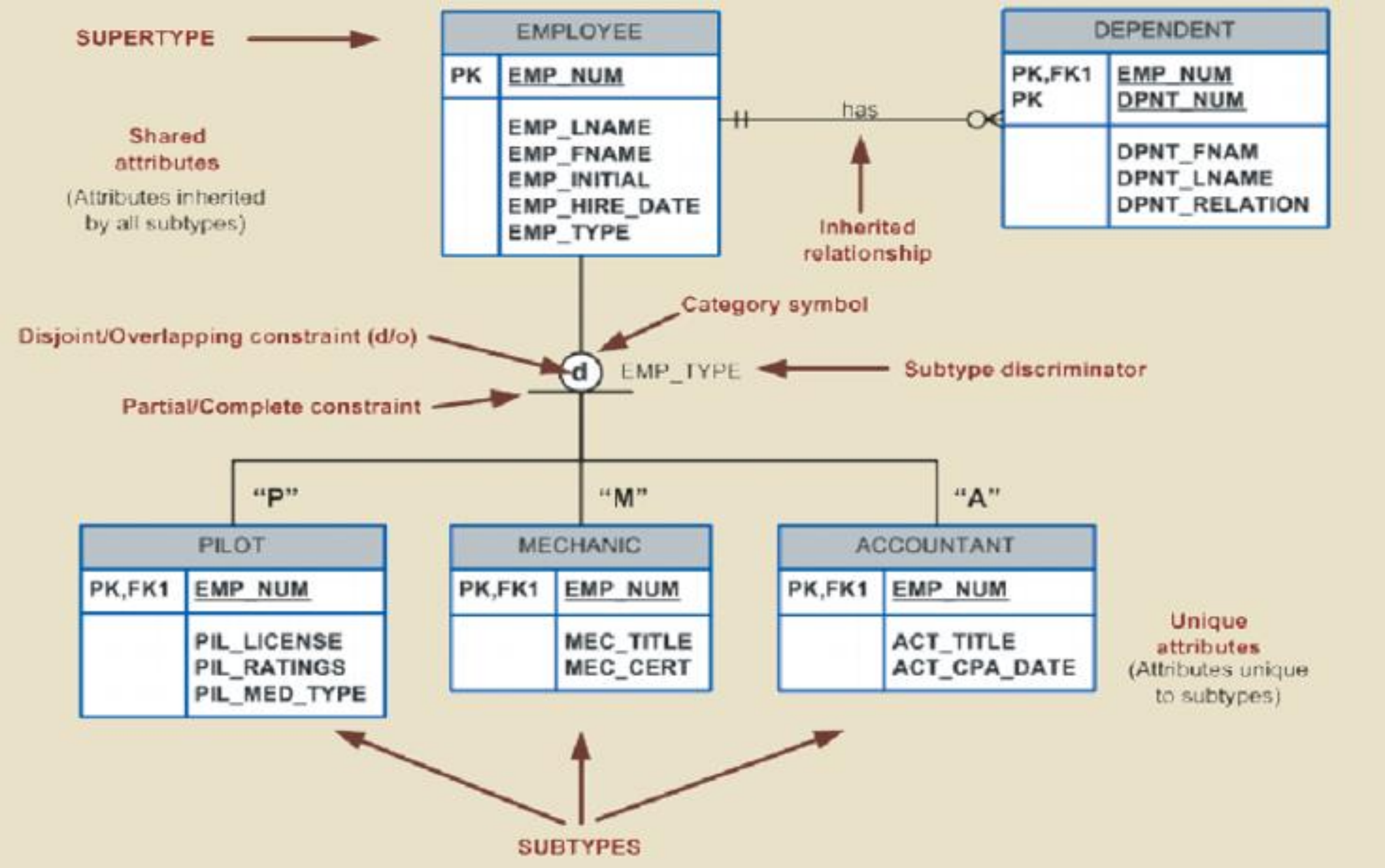
An Attorney can be either a litigater or partner, neither, or both



- Two <u>incomplete</u> overlapping clusters:
  - ATTORNEY (<u>ID</u>, Name, L, P, … ***{L & P are checklist}***
    - L & P constrained to be either 0 or 1 (or T or F)
  - LITIGATER (<u>*ID*</u>, win/loss, …
  - PARTNER (<u>*ID*</u>, Percent, …

Tables for Disjoint Subtypes (d)

[an employee cannot be more than one subtype (disjoint), but cluster is partial (not all subtypes are shown, every employee does not need to be one of the 3)]



SUPERTYPE

EMPLOYEE	
PK	EMP_NUM
	EMP_LNAME EMP_FNAME EMP_INITIAL EMP_HIRE_DATE EMP_TYPE

has

DEPENDENT	
PK,FK1 PK	EMP_NUM DPNT_NUM
	DPNT_FNAM DPNT_LNAME DPNT_RELATION

Shared attributes

(Attributes inherited by all subtypes)

Inherited relationship

Category symbol

Disjoint/Overlapping constraint (d/o)

d

EMP_TYPE

Subtype discriminator

Partial/Complete constraint

"P"

"M"

"A"

PILOT	
PK,FK1	EMP_NUM
	PIL_LICENSE PIL_RATINGS PIL_MED_TYPE

MECHANIC	
PK,FK1	EMP_NUM
	MEC_TITLE MEC_CERT

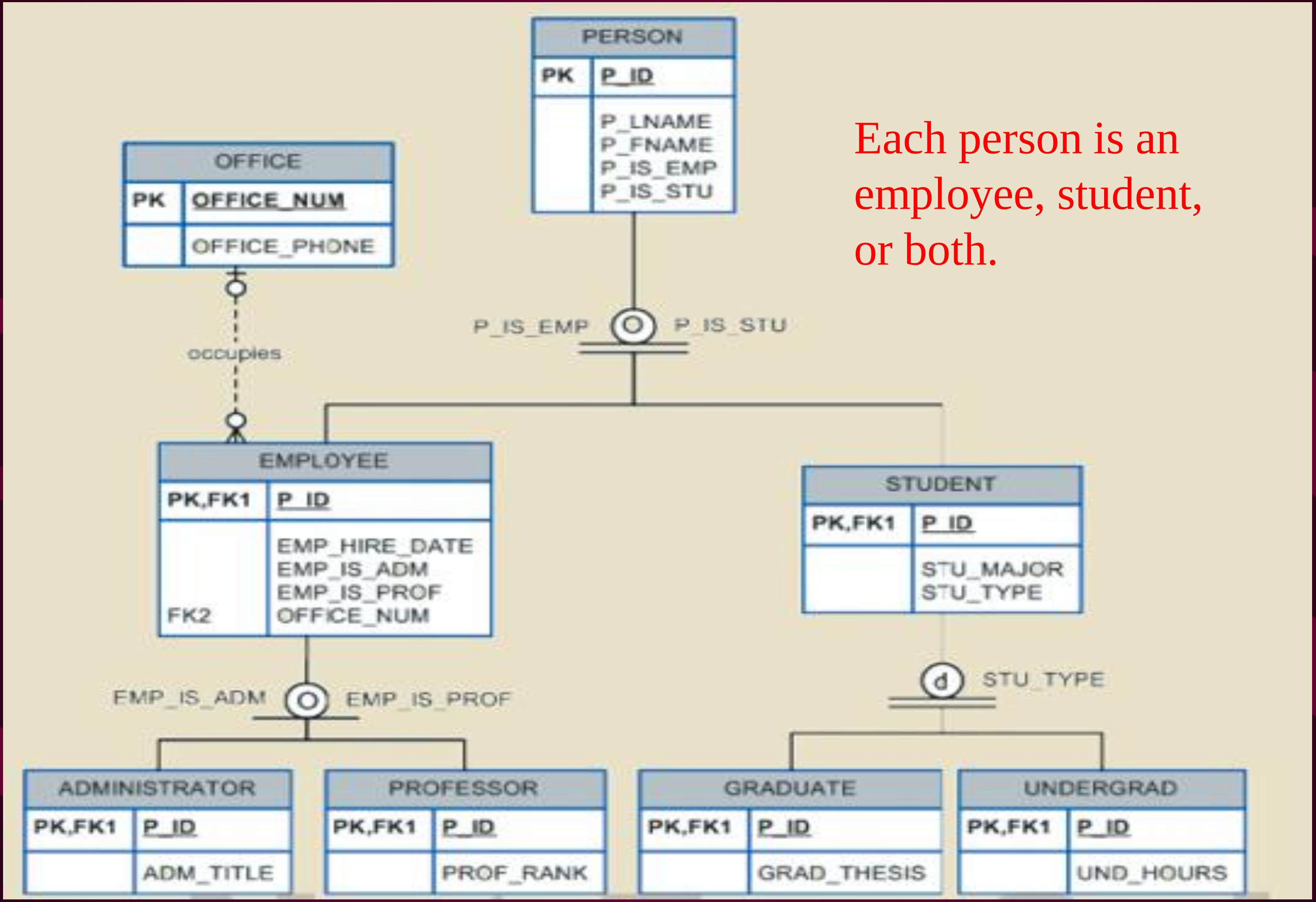
ACCOUNTANT	
PK,FK1	EMP_NUM
	ACT_TITLE ACT_CPA_DATE

Unique attributes

(Attributes unique to subtypes)

SUBTYPES

Tables for Overlapping Subtypes (multiple discriminators)



PERSON

PK	P_ID
	P_LNAME P_FNAME P_IS_EMP P_IS_STU

OFFICE

PK	OFFICE_NUM
	OFFICE_PHONE

EMPLOYEE

PK,FK1	P_ID
	EMP_HIRE_DATE EMP_IS_ADM EMP_IS_PROF
FK2	OFFICE_NUM

STUDENT

PK,FK1	P_ID
	STU_MAJOR STU_TYPE

ADMINISTRATOR

PK,FK1	P_ID
	ADM_TITLE

PROFESSOR

PK,FK1	P_ID
	PROF_RANK

GRADUATE

PK,FK1	P_ID
	GRAD_THESIS

UNDERGRAD

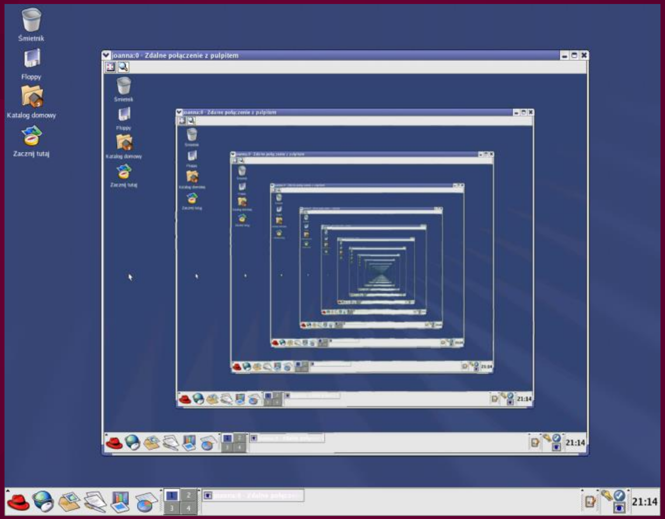
PK,FK1	P_ID
	UND_HOURS

Each person is an employee, student, or both.

Recursion (Degree 1)

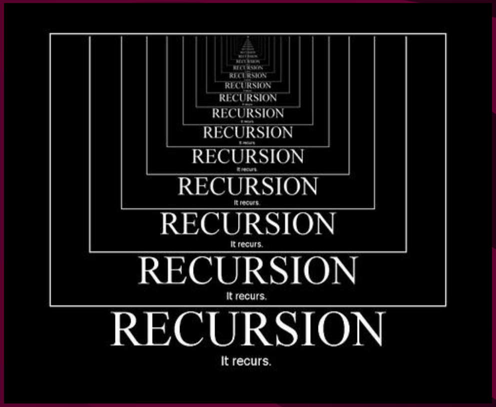
- 1 to 1
- 1 to Many
- Many to Many

Degree 1 - Recursive



Recursion

- A relationship between an entity an itself:
  - 1 to 1
  - 1 to many
  - many to many



- Caution: problems in application implementation –almost always need to do special coding !!!

1 to 1 Recursion

- Entity gets one table
- Include attribute (column) for foreign key in the same table, however, the foreign key is same physical domain as the primary key
- Pick one of the two search directions for the *semantic* portion of the domain (what does it mean)
- Use unique index on FK

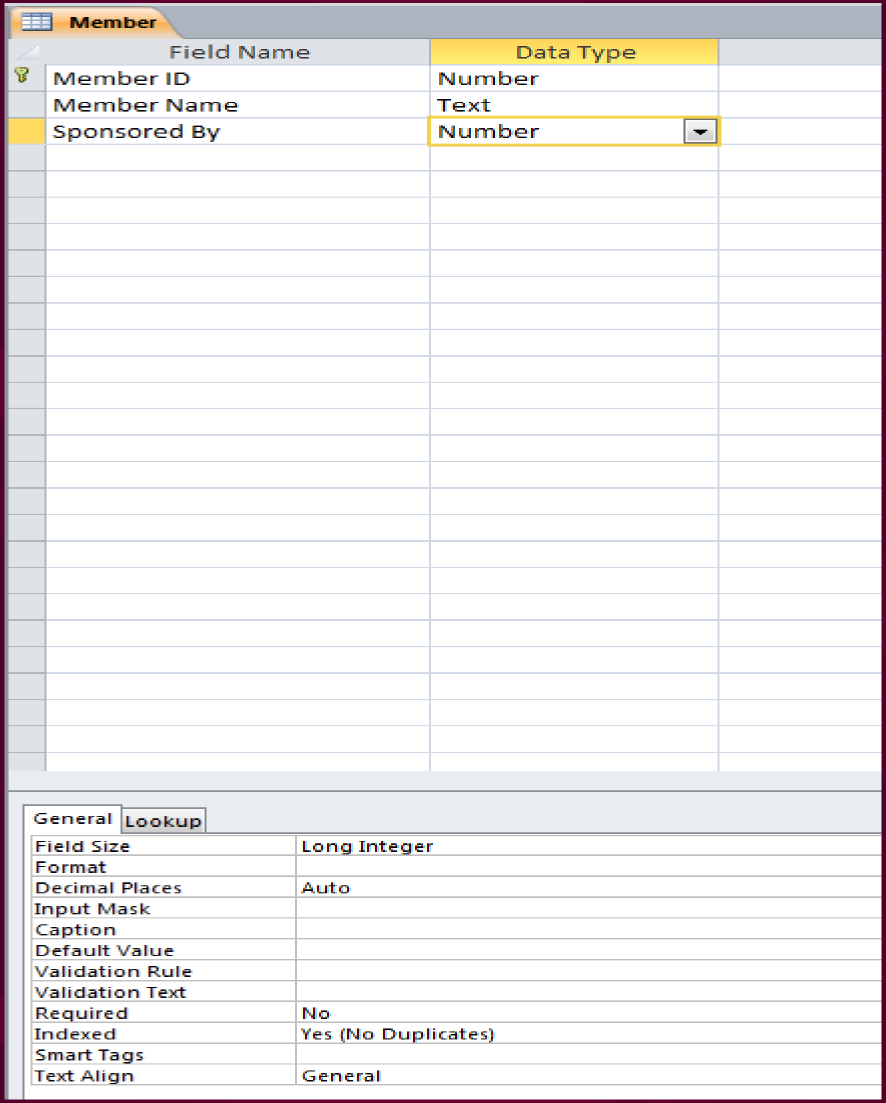
1 to 1 Recursion (con't)

- Consider the sponsor relationship where a person can sponsor only one other person, and a person can only be sponsored by one other person
- MEMBERS (MID, Name, *MID*[person sponsored],...)
- or
- MEMBERS (MID, Name, *MID*[person sponsoredBy],...)
- Pick one of the two domain semantics
- Referential integrity must still be dealt with, but cannot automatically set this up in many database products; also "chicken & egg" situation !
- Also need code so that a member cannot sponsor himself

Member Table

Member	Name	Sponsored By
12	Doe, John	14
13	Byte, Ima	23
14	Ivy, Dripper	2

Unique Index on Foreign Key

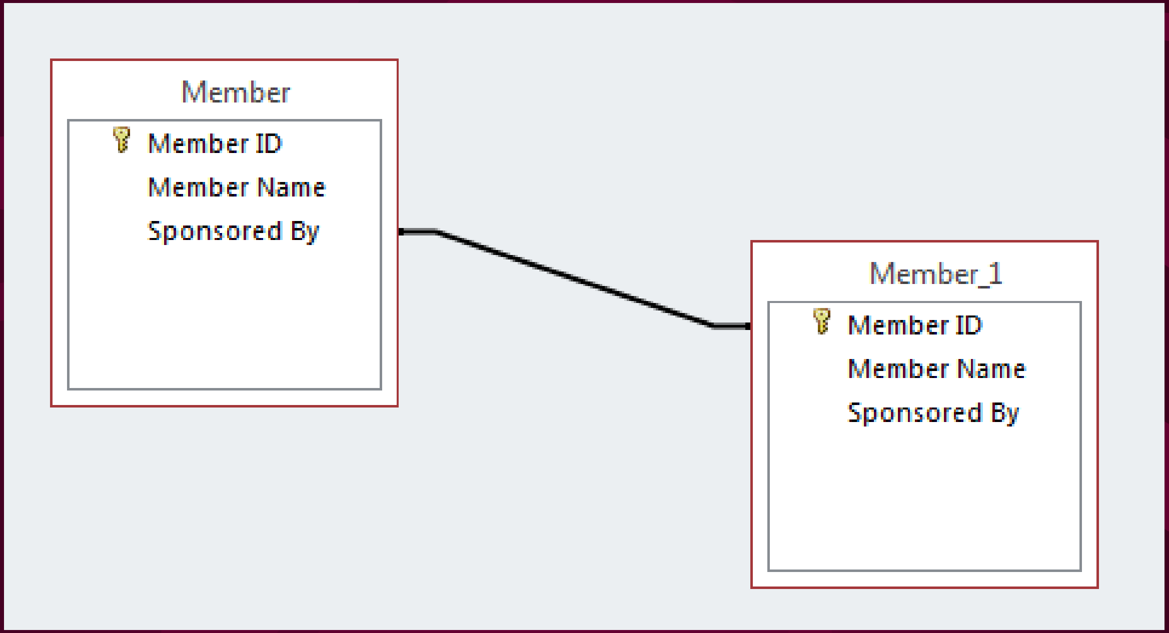


Member

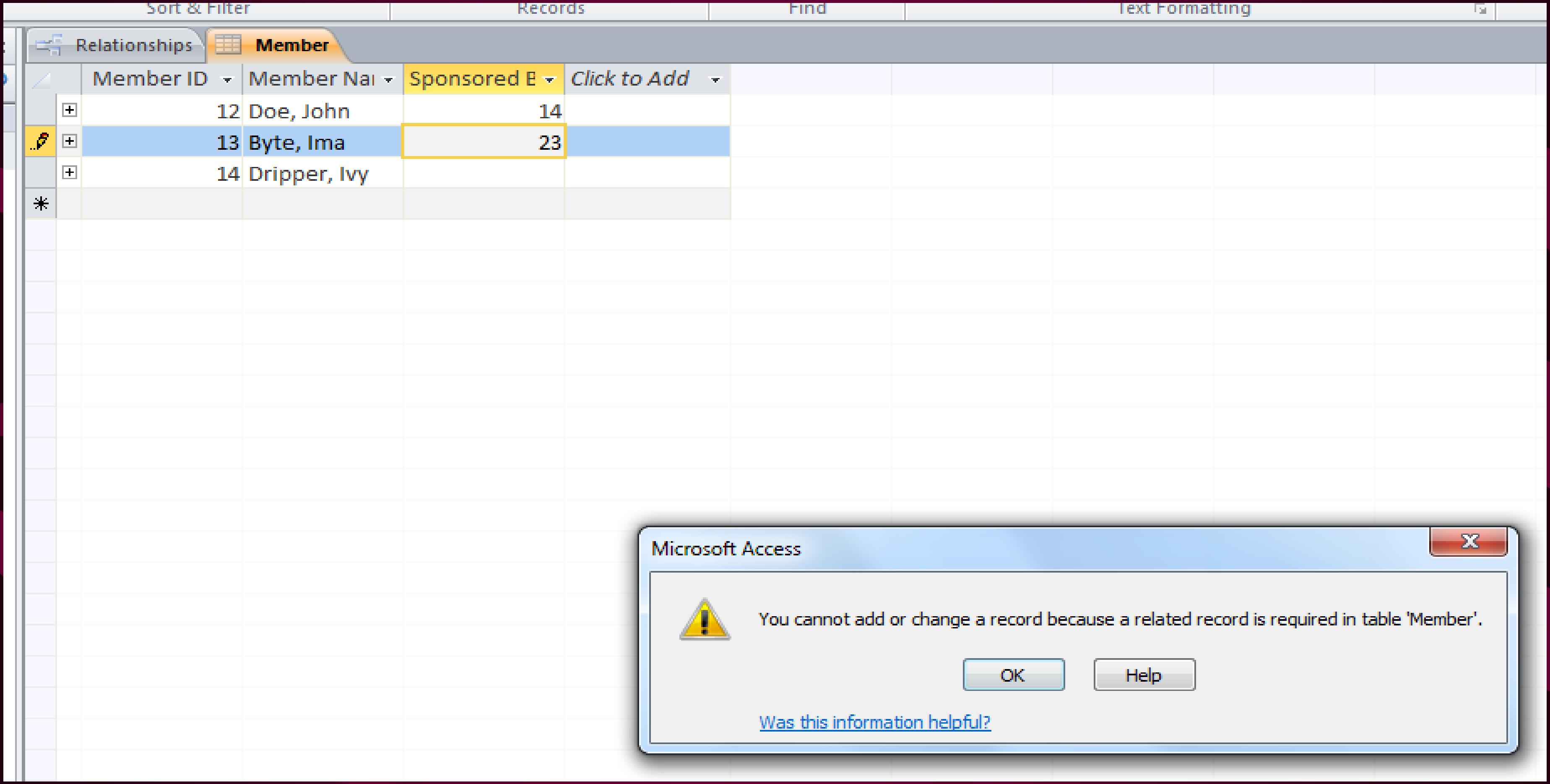
Field Name	Data Type
Member ID	Number
Member Name	Text
Sponsored By	Number

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	
Text Align	General

Recursion in MS Access (need alias table)



Need to add member 23 first…



Member ID	Member Nai	Sponsored E	Click to Add
12	Doe, John	14	
13	Byte, Ima	23	
14	Dripper, Ivy		

Microsoft Access

You cannot add or change a record because a related record is required in table 'Member'.

1 to Many Recursion

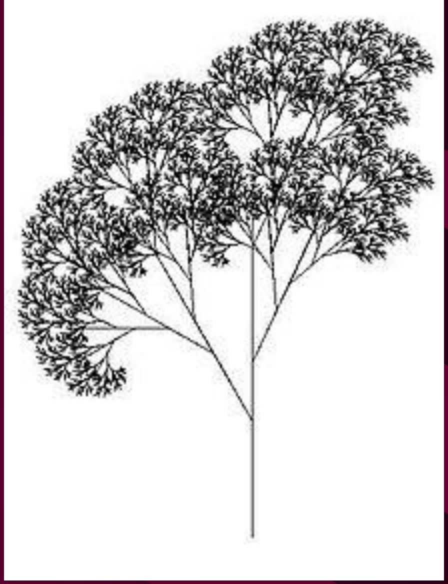
- Entity gets one table
- Include attribute (column) for foreign key in the child (many) table, however foreign key physical domain is same domain as the primary key
- Use index on FK

1 to Many Recursion

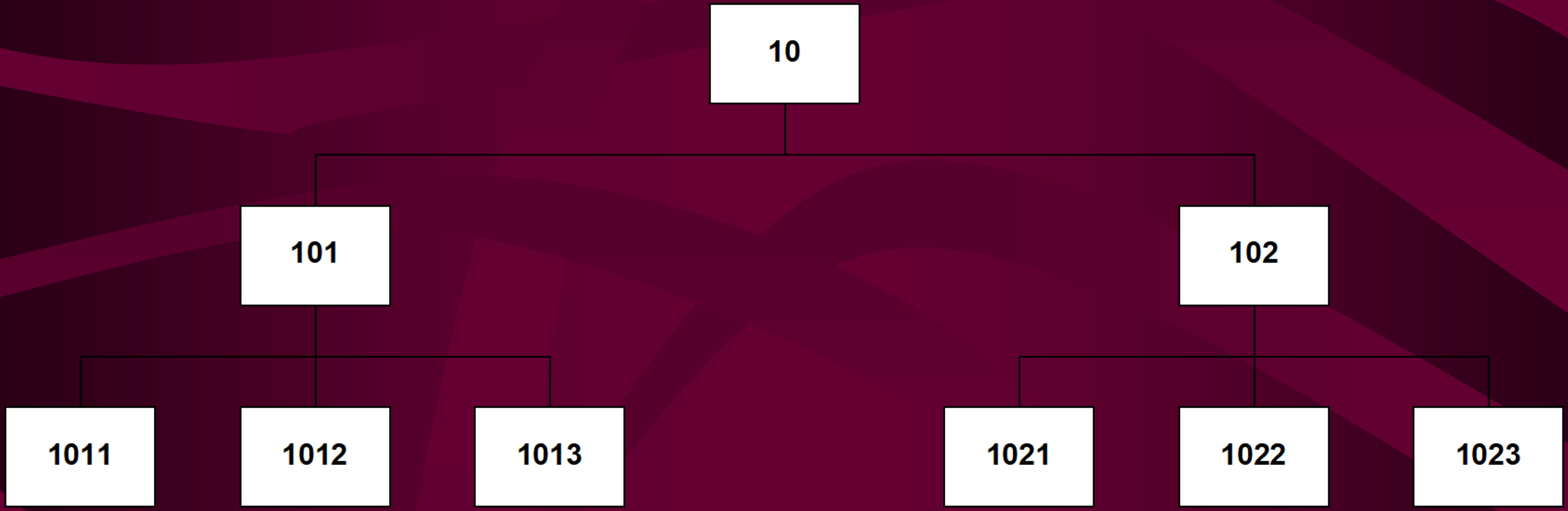
- Consider the case of a general ledger system, where each account can have a master (control) account, and each master account can have many subsidiary accounts
- Recursive tree or forest (multiple trees) structure
- LEDGER (<u>AccountNumber</u>, Type, Balance, *AccountNumber*[master],...)
- In a "forest" the foreign key may be null as with the general ledger accounts (an account does not have to have a master account)
- In a "tree", only the object at the top has no master account
- Referential integrity must still be enforced (typically in code)

Ledger Accounts

- 10 Cash
  - 101 Cash in Banks
    - 1011 First TN
    - 1012 NBC
  - 102 Petty Cash
    - 1021 John Doe
    - …
- 11 Accounts Receivable
  - 111 Accounts Receivable Trade
  - 112 Accounts Receivable Other



Tree Representation o G/L Accounts



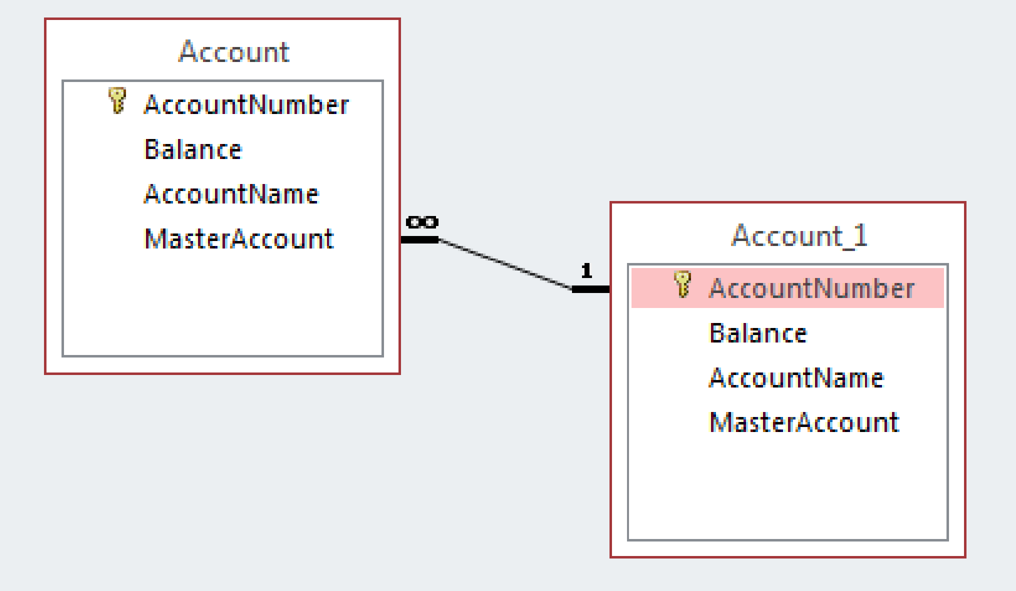
Normally account numbers must match hierarchy also.

Account Table

[Master is Foreign Key]

Account	Name	Master
10	Cash	
101	Cash in Banks	10
1011	First TN	101
1012	NBC	101

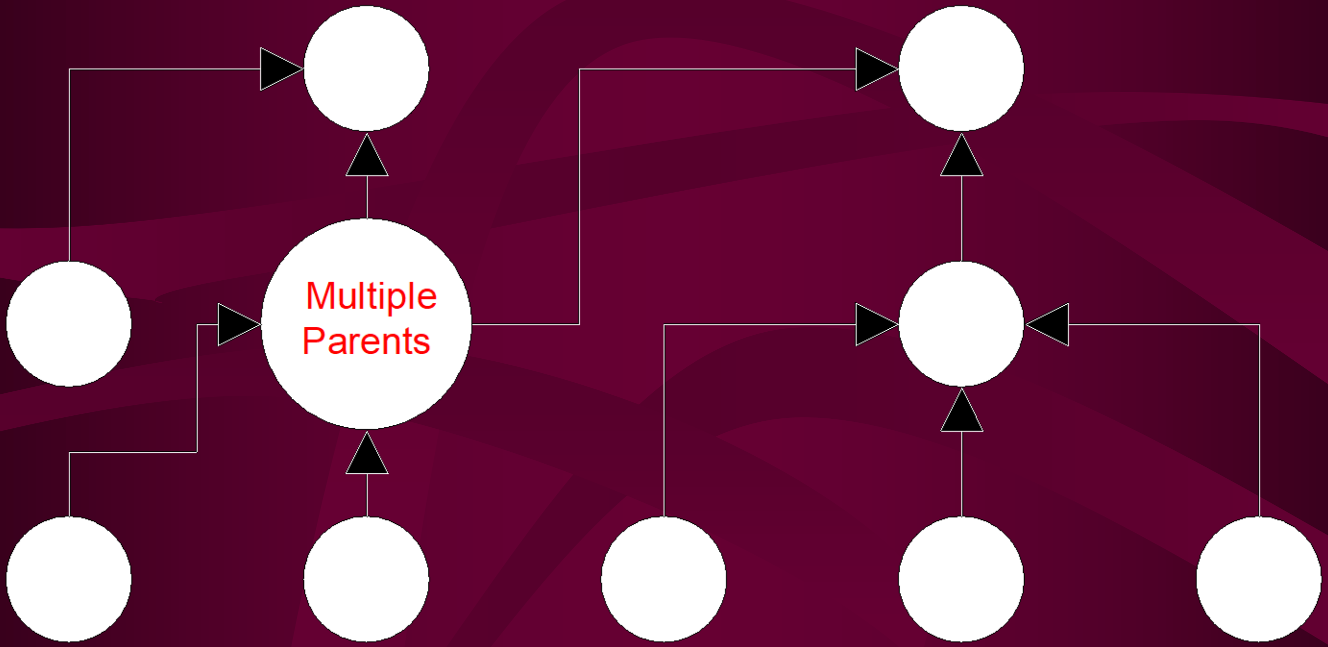
Recursion in MS Access (need alias table)



Many to Many Recursion
(Parts Explosion or Bill of Materials)

- Entity gets one table
- Intersection table is created for the relationship:
  - Composite (subassembly)
  - Components (parts of)
- Relationship table contains two columns of foreign keys both with the same physical domain as the primary key in the first table; the primary key of the relationship table is the combination of the two foreign keys, order is important!
- There may be attributes to the relationship itself

A part can be a component in many other parts, and a part can be made up of many other parts !!!



Parts Explosion

Many to Many Recursion

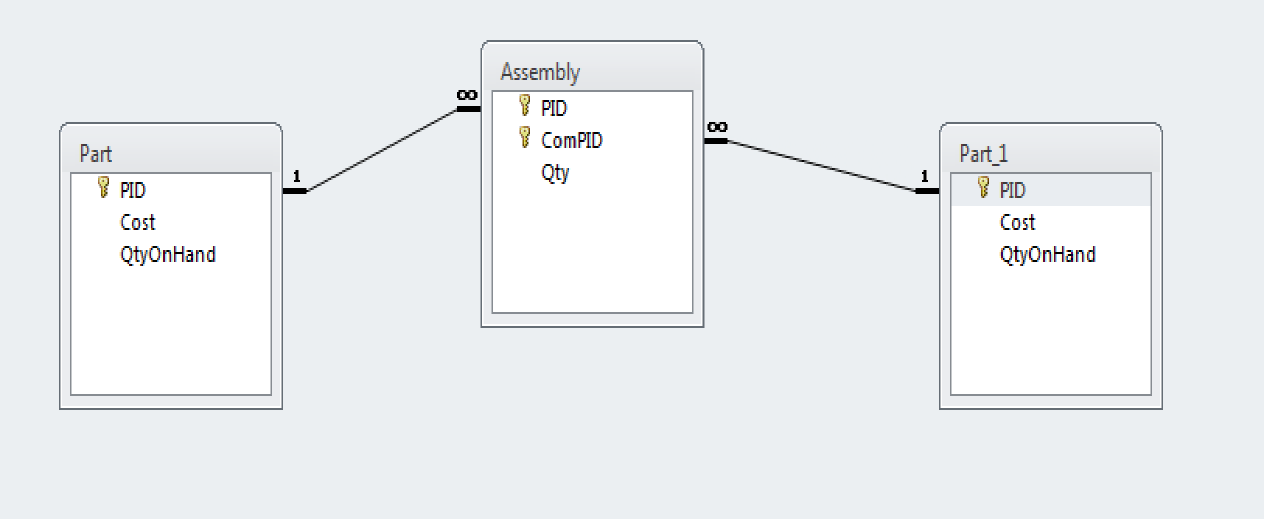
- PART (PartId, cost, QtyOnHand, ...)
- ASSEMBLY (*PartID, PartID*, Qty)
- In the second table, the first column represents the assembly and the second column represents the parts of which the assembly is composed, the third column is how many of the second column part is used in making the assembly

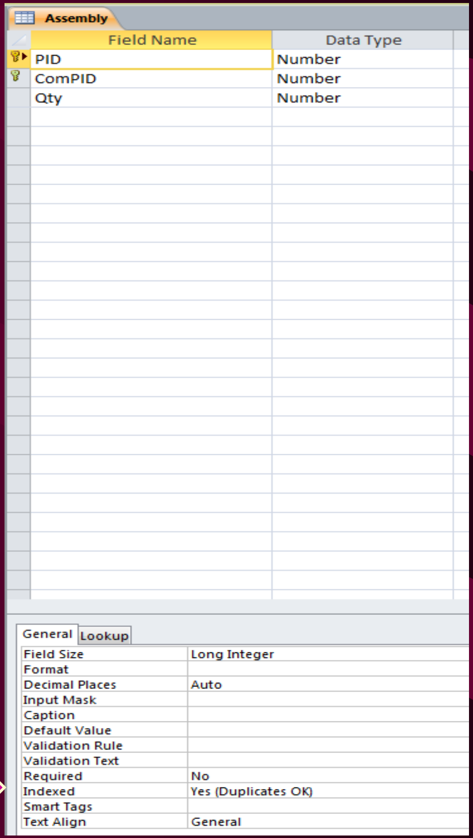
Assembly (Intersection Table)

Part	Component Part	Quantity
3	2	2
3	5	2
4	6	3
5	7	2
5	4	1
5	6	2

Also need restriction that a part cannot be composed of itself.

Access Relationship Grid
[part table added twice (alias)]





Field Name	Data Type
PID	Number
ComPID	Number
Qty	Number

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (Duplicates OK)
Smart Tags	
Text Align	General

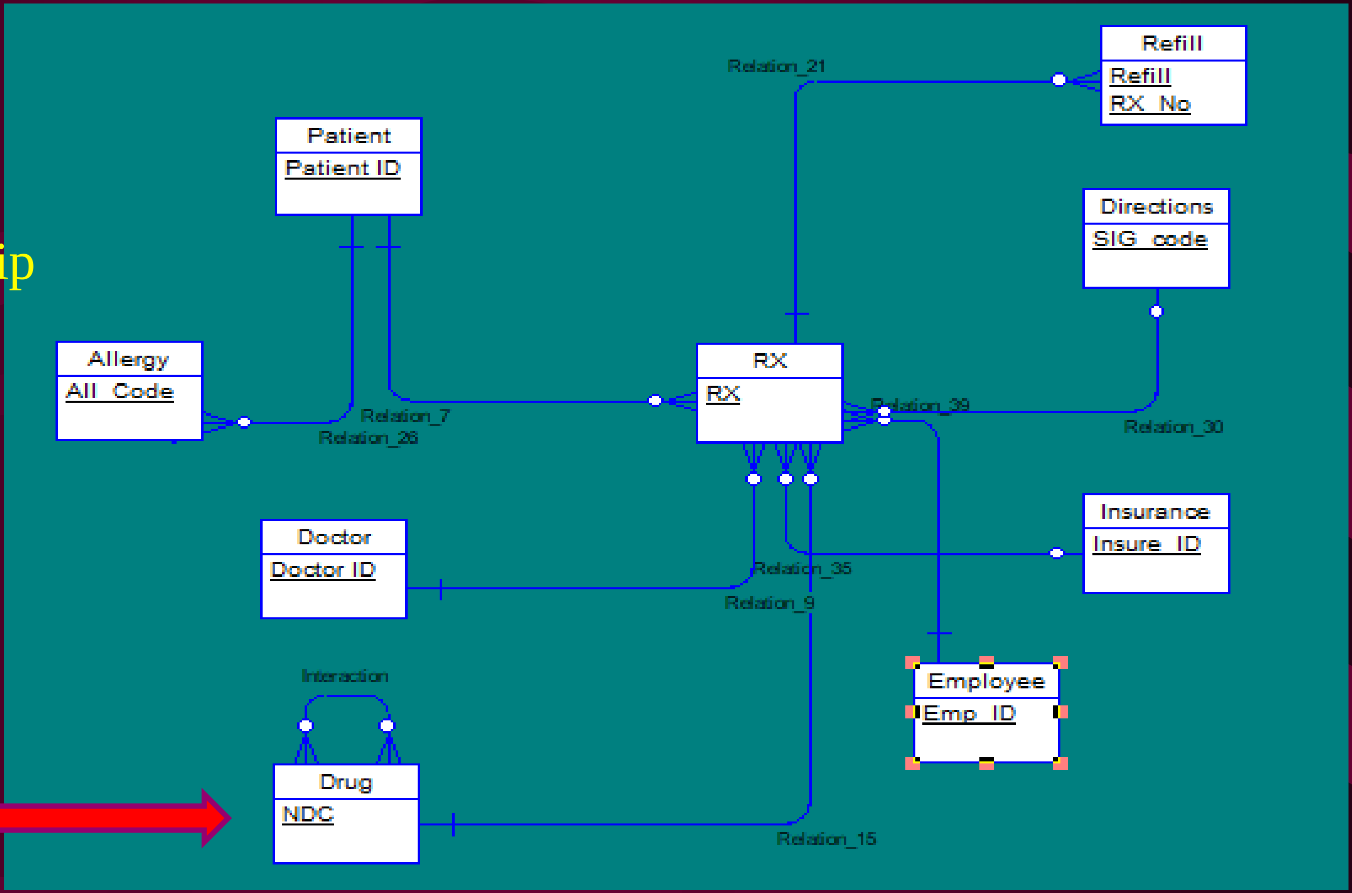
Recursive Relationships in Visio

Recursive Relationships in Access

Note: without adding code, there's nothing to stop a particular record from being its own parent

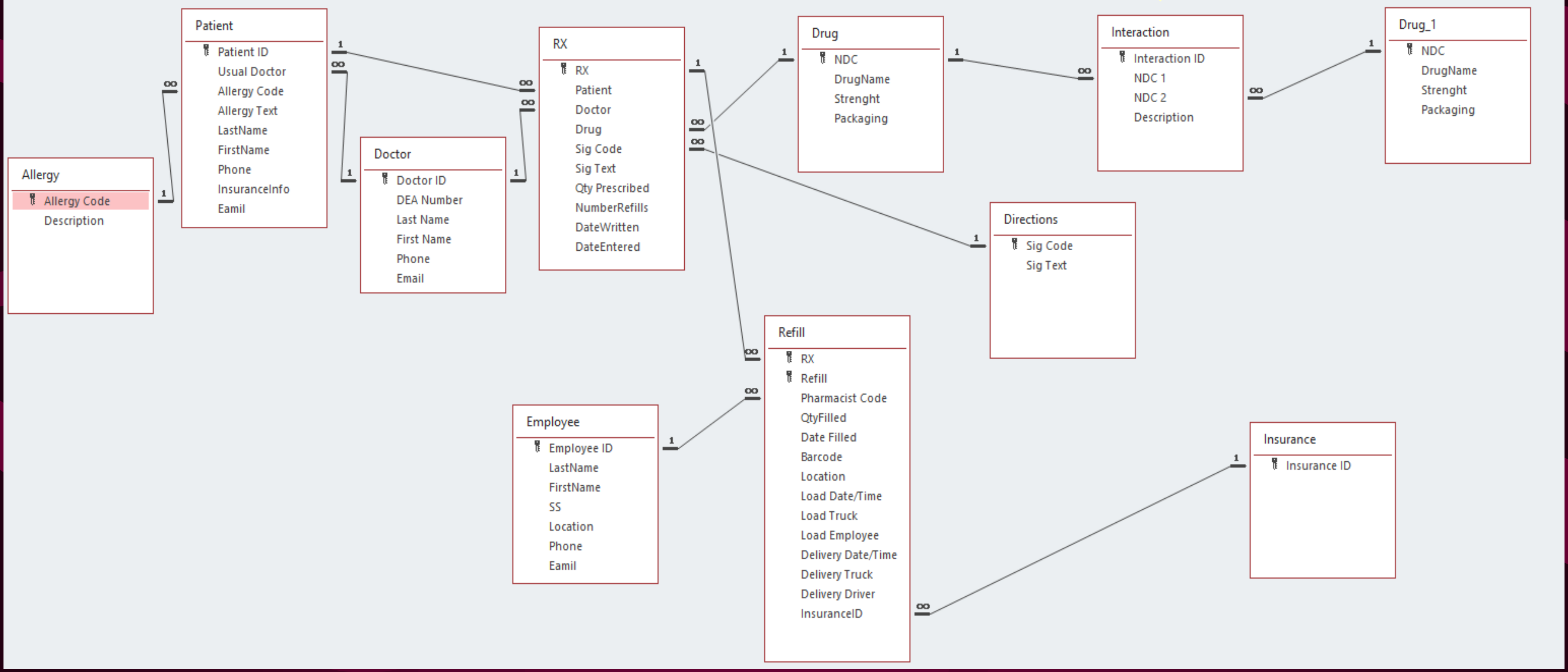
ER Model for Pharmacy

In Power Designer, have to specify the recursive relationship in the relationship dialog box not via dragging the tool.



Pharmacy Access Model

Have to show the Drug table twice.



Allergy
- Allergy Code
- Description

Patient
- Patient ID
- Usual Doctor
- Allergy Code
- Allergy Text
- LastName
- FirstName
- Phone
- InsuranceInfo
- Eamil

Doctor
- Doctor ID
- DEA Number
- Last Name
- First Name
- Phone
- Email

RX
- RX
- Patient
- Doctor
- Drug
- Sig Code
- Sig Text
- Qty Prescribed
- NumberRefills
- DateWritten
- DateEntered

Drug
- NDC
- DrugName
- Strenght
- Packaging

Interaction
- Interaction ID
- NDC 1
- NDC 2
- Description

Drug_1
- NDC
- DrugName
- Strenght
- Packaging

Directions
- Sig Code
- Sig Text

Refill
- RX
- Refill
- Pharmacist Code
- QtyFilled
- Date Filled
- Barcode
- Location
- Load Date/Time
- Load Truck
- Load Employee
- Delivery Date/Time
- Delivery Truck
- Delivery Driver
- InsuranceID

Employee
- Employee ID
- LastName
- FirstName
- SS
- Location
- Phone
- Eamil

Insurance
- Insurance ID

Relationships of Degree N

- Each entity is represented by its own table
- A new table needs to be created for the relation itself
- This new table includes foreign keys to each of the other tables
- This new table also includes attributes that are properties of the relationship (not the original entities)

Relationships of Degree N (con’t)

- The name of the new table is the name of the relationship, if one; if not the table can be named by a combination of the names of the original tables
- The primary key of the new table is the combination of the foreign keys; the order is important and should be chosen for the order in which the table is normally processed
- Other orders can be set up as an indexes

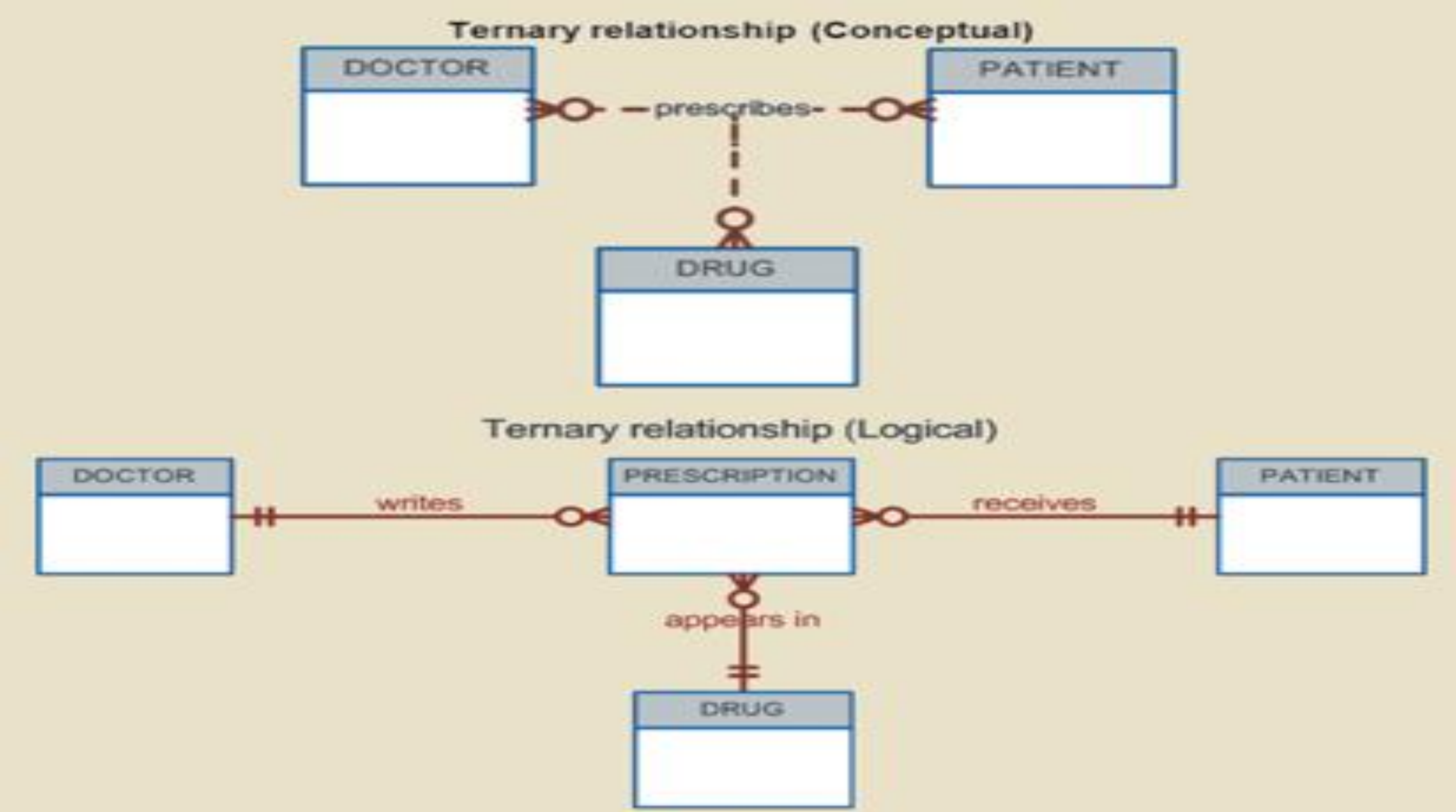
Relationships of Degree N (con’t)

- Consider the relationship between Car Dealers, Auto Types (cars, vans, trucks, etc), and Car Brands (Ford, GM, etc):
  - DEALERS (DID, Name, Address, ...)
  - TYPES (TID, Name, SafetyReq, ...)
  - BRANDS (BName, ...)
  - DEALER-TYPES-BRANDS (*DID,TID, BName*, QtyStocked, ...)

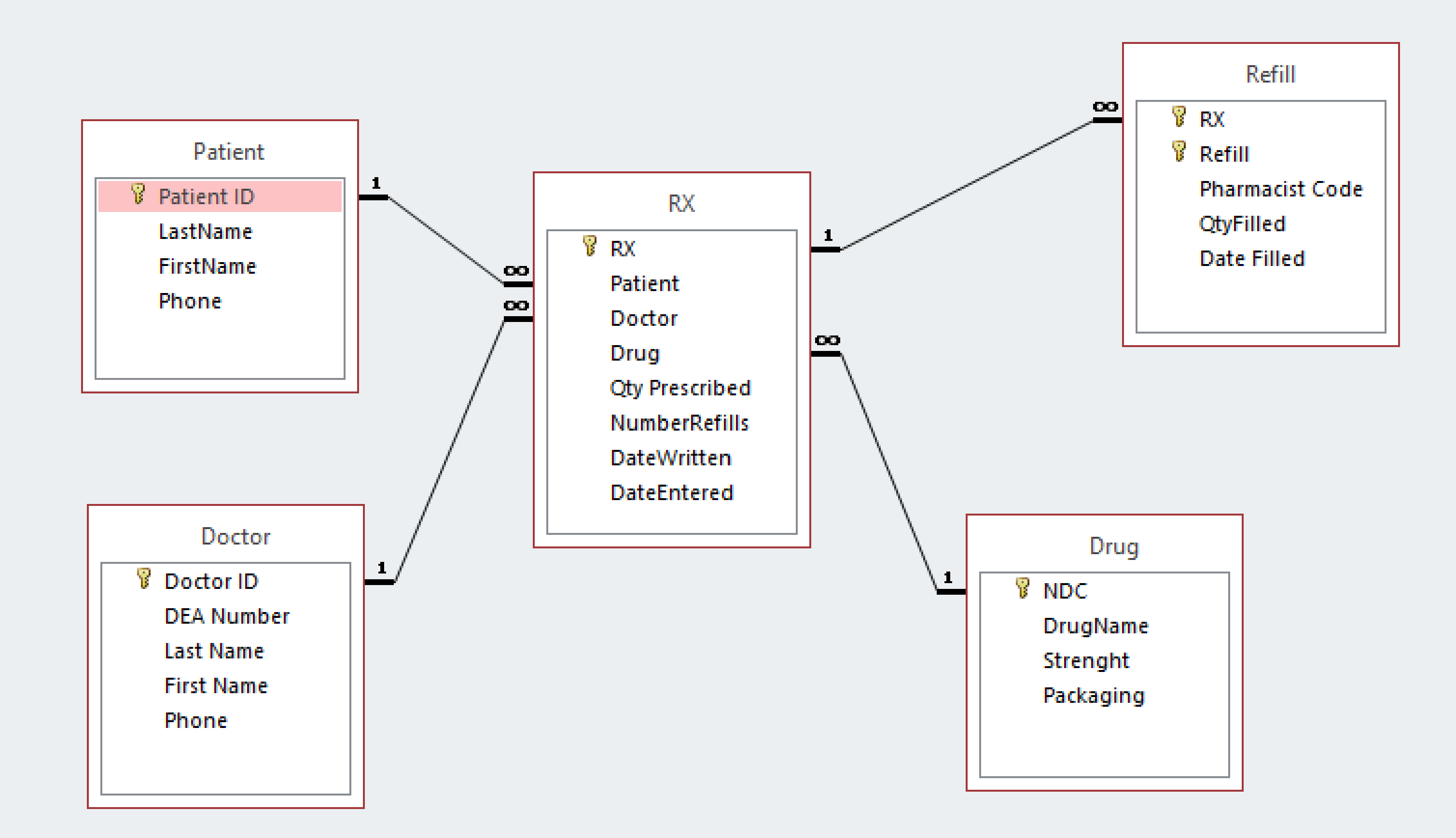
Relationships of Degree N (con’t)

- Sometimes the relationship has its own natural identifier, and probably name
- This natural identifier could be used as the primary key
- However, you still may need the combination of foreign keys to enforce uniqueness of that combination:
  - PRESCRIPTION (<u>RXNumber</u>, *DID*, *PID*, *NDC*, ...)

Relationships of Degree N (con't)



RX Three-way Intersection Table (including refills)



Patient
- Patient ID
- LastName
- FirstName
- Phone

RX
- RX
- Patient
- Doctor
- Drug
- Qty Prescribed
- NumberRefills
- DateWritten
- DateEntered

Refill
- RX
- Refill
- Pharmacist Code
- QtyFilled
- Date Filled

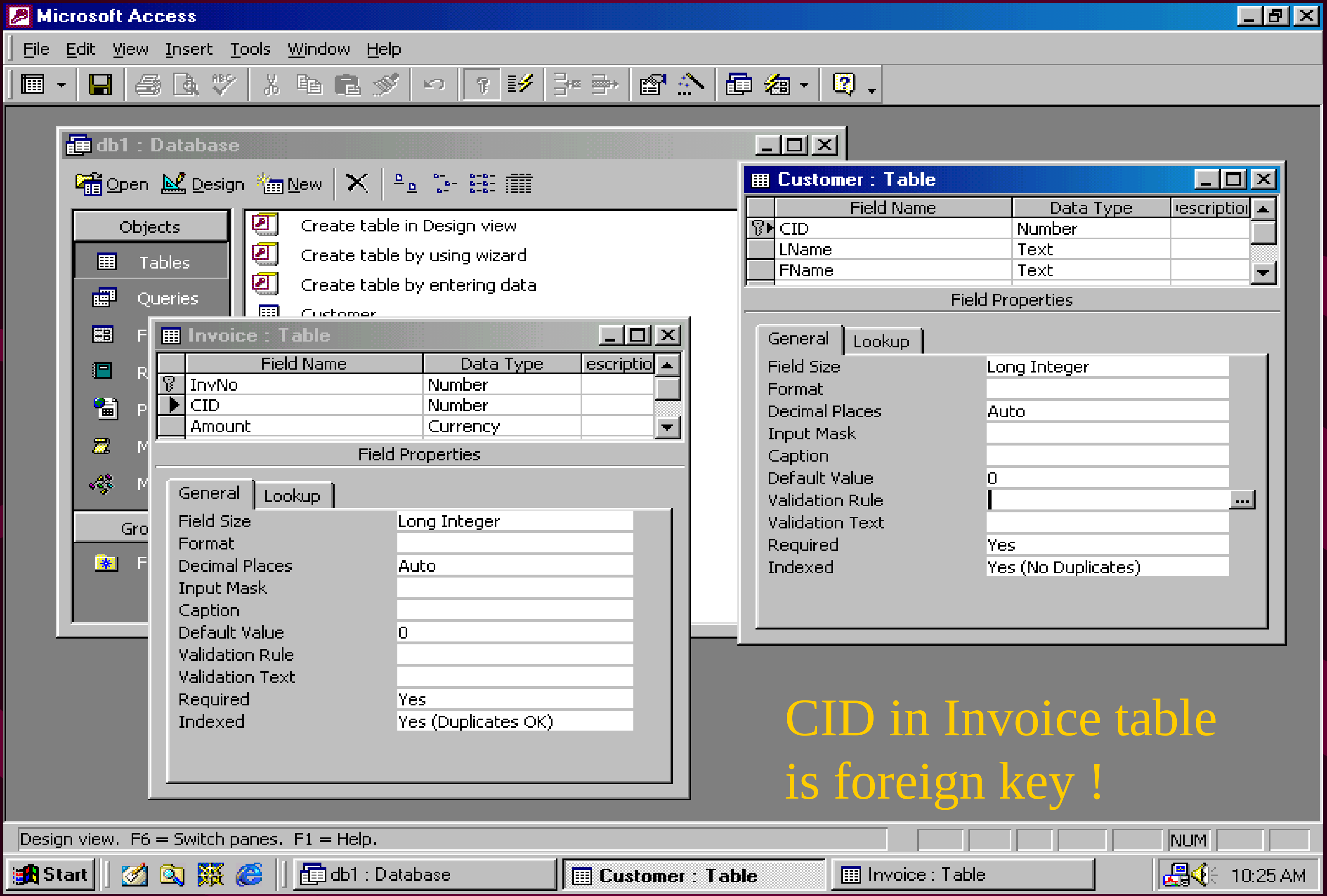
Doctor
- Doctor ID
- DEA Number
- Last Name
- First Name
- Phone

Drug
- NDC
- DrugName
- Strenght
- Packaging

Simplified Summary

- Create a table for each entity
- Create an attribute for each property with the unique identifier becoming the primary key
- Normalize if necessary
- Represent relationships with foreign keys; index foreign keys
  - 1 to 1: key of either as FK of other
  - 1 to N : key of parent as FK in child
  - M to N : intersection table having keys of both
- Handle referential integrity and other constraints

Access FK's and Referential Integrity


Microsoft Access
File Edit View Insert Tools Window Help
db1 : Database
Open Design New
Objects
Tables
Queries
Create table in Design view
Create table by using wizard
Create table by entering data
Customer
Design view. F6 = Switch panes. F1 = Help.
NUM
Start
db1 : Database
Customer : Table
Invoice : Table
10:25 AM

Invoice : Table

Field Name	Data Type	escriptio
InvNo	Number	
CID	Number	
Amount	Currency	

Field Properties (General)

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	Yes
Indexed	Yes (Duplicates OK)

Customer : Table

Field Name	Data Type	escriptio
CID	Number	
LName	Text	
FName	Text	

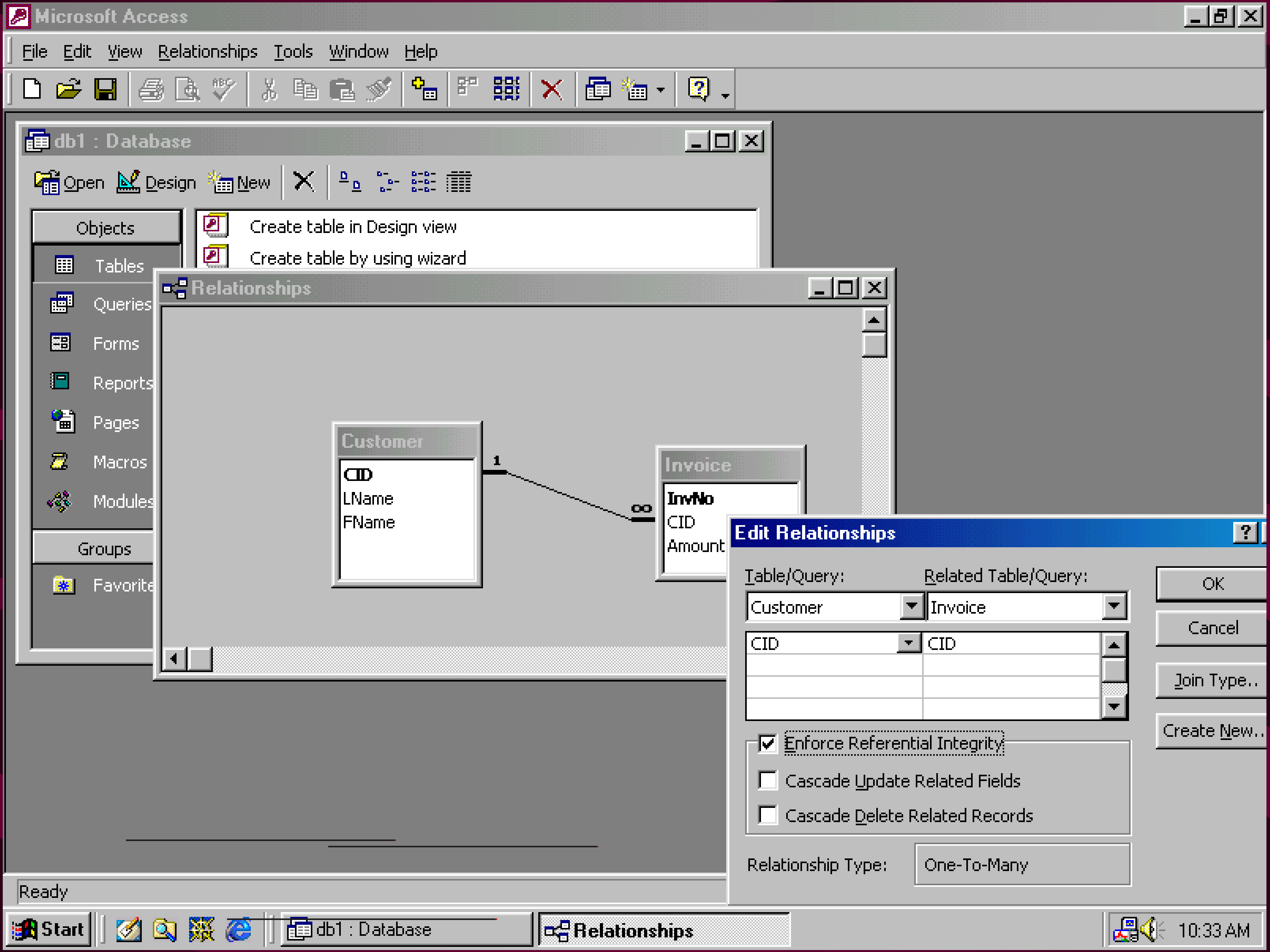
Field Properties (General)

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	Yes
Indexed	Yes (No Duplicates)

CID in Invoice table is foreign key !

Make sure that CID is defined as exactly the same type in both tables !

Use relationship grid in Access to set up foreign keys !



Table/Query:	Related Table/Query:
Customer	Invoice
CID	CID

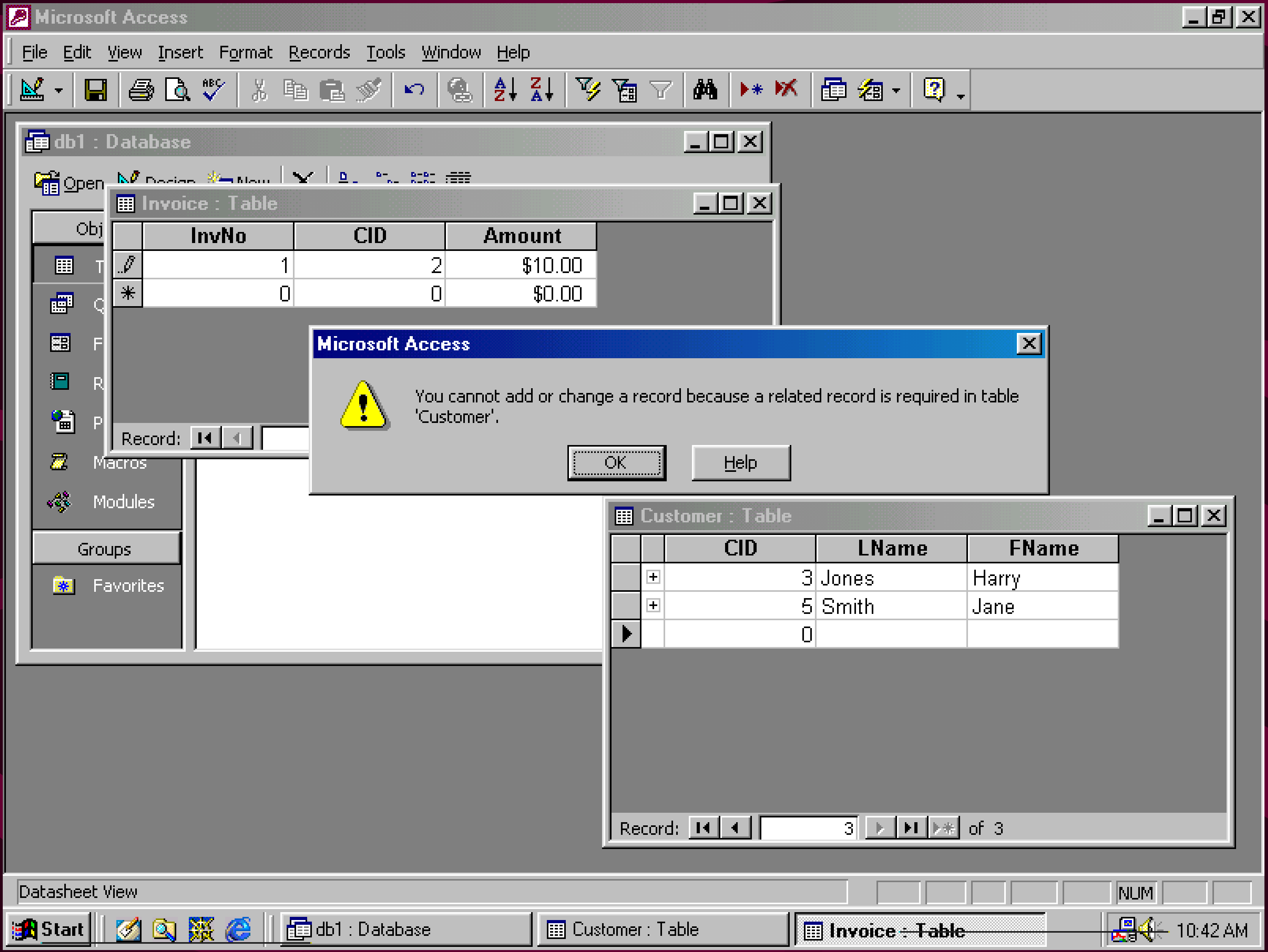
☑ Enforce Referential Integrity

□ Cascade Update Related Fields

□ Cascade Delete Related Records

Relationship Type: One-To-Many

Check “enforce referential integrity” !



Invoice : Table

	InvNo	CID	Amount
✎	1	2	$10.00
*	0	0	$0.00

Microsoft Access

You cannot add or change a record because a related record is required in table 'Customer'.

OK Help

Customer : Table

		CID	LName	FName
	+	3	Jones	Harry
	+	5	Smith	Jane
▶		0		

Record: 3 of 3

Error message trying to add invoice for customer not in customer table !

en you create a relationship between tables, the related fields don't have to have the same names. However, related fields must have the same data type unless the primary key field is an oNumber field. You can match an AutoNumber field with a Number field only if the **FieldSize** property of both of the matching fields is the same. For example, you can match an AutoNumber field and umber field if the **FieldSize** property of both fields is **Long Integer**. Even when both matching fields are Number fields, they must have the same **FieldSize** property setting.

)efine a one-to-many or a one-to-one relationship

. Close any tables you have open. You can't create or modify relationships between open tables.

. Press F11 to switch to the Database window.

. Click **Relationships** on the toolbar.

. If you haven't yet defined any relationships in your database, the **Show Table** dialog box is automatically displayed.

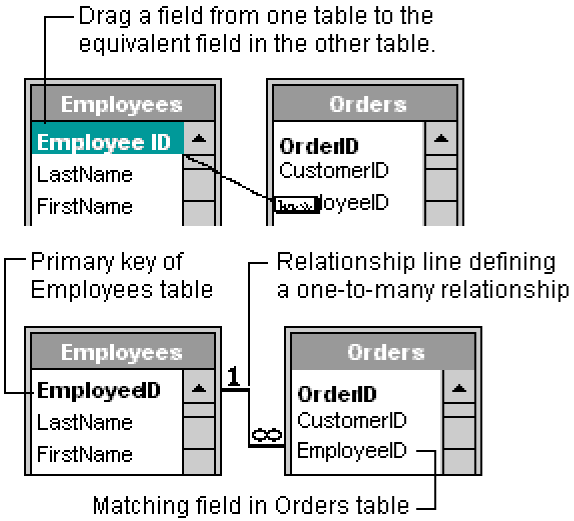
If you need to add the tables you want to relate and the **Show Table** dialog box isn't displayed, click **Show Table** on the toolbar.

. Double-click the names of the tables you want to relate, and then close the **Show Table** dialog box. To create a relationship between a table and itself, add that table twice.

. Drag the field that you want to relate from one table to the related field in the other table.

To drag multiple fields, press the CTRL key, click each field, and then drag them.

In most cases, you drag the primary key field (which is displayed in bold text) from one table to a similar field (often with the same name) called the foreign key in the other table.


Drag a field from one table to the equivalent field in the other table.
Employees
Employee ID
LastName
FirstName
Orders
OrderID
CustomerID
loyeeID
Primary key of Employees table
Relationship line defining a one-to-many relationship
Employees
EmployeeID
LastName
FirstName
1
∞
Orders
OrderID
CustomerID
EmployeeID
Matching field in Orders table

. The **Edit Relationships** dialog box is displayed. Check the field names displayed in the two columns to ensure they are correct. You can change them if necessary.

Set the relationship options if necessary.

. Click the **Create** button to create the relationship.

. Repeat steps 5 through 8 for each pair of tables you want to relate.

When you close the Relationships window, Microsoft Access asks if you want to save the layout. Whether you save the layout or not, the relationships you create are saved in the database.

e You can create relationships using queries as well as tables. However, referential integrity isn't enforced with queries.

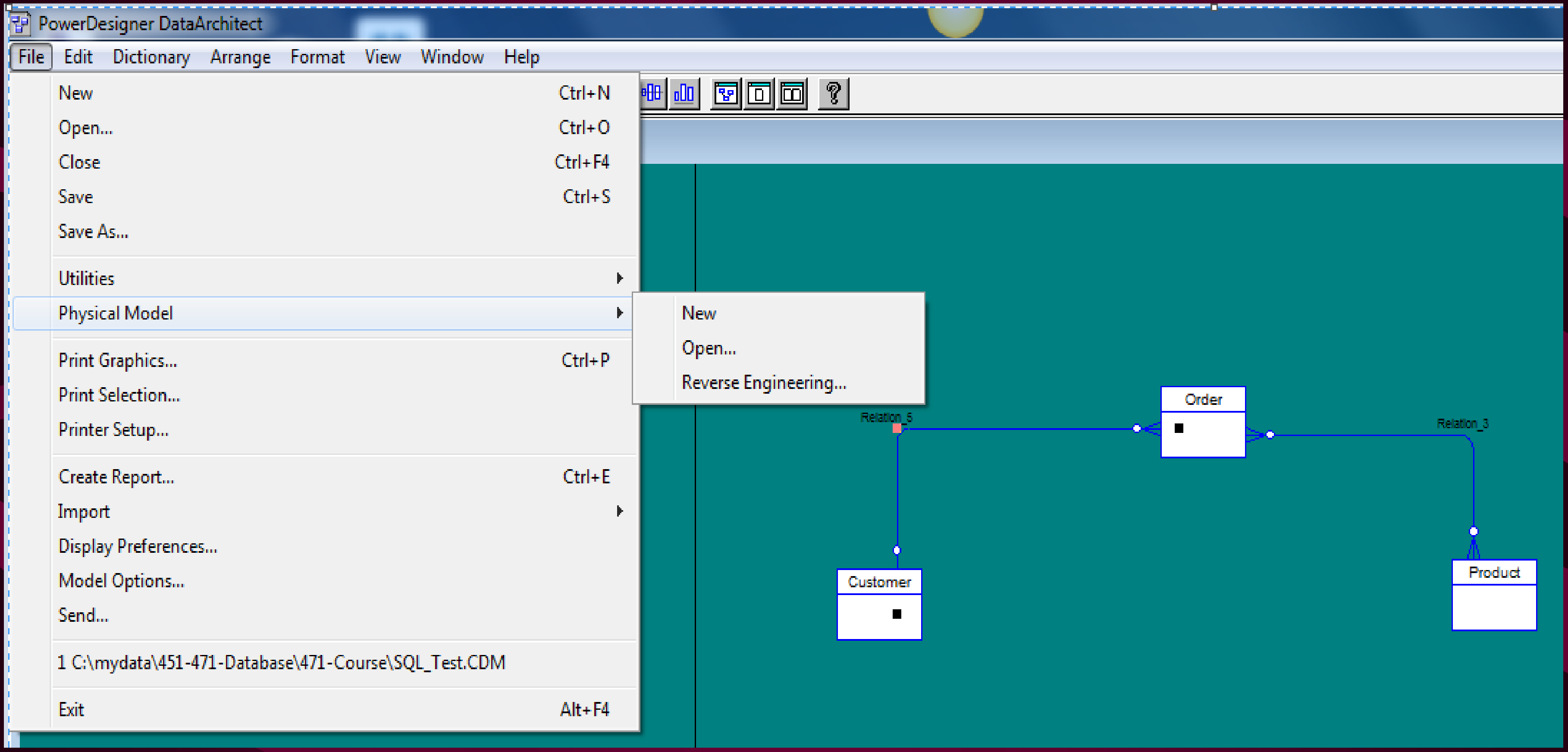
Access Referential Integrity

- Referential integrity is a system of rules that Microsoft Access uses to ensure that relationships between records in related tables are valid, and that you don't accidentally delete or change related data. You can set referential integrity when all of the following conditions are met:
- The matching field from the primary table (primary table: The "one" side of two related tables in a one-to-many relationship. A primary table should have a primary key and each record should be unique.) is a primary key (primary key: One or more fields (columns) whose values uniquely identify each record in a table. A primary key cannot allow **Null** values and must always have a unique index. A primary key is used to relate a table to foreign keys in other tables.) or has a unique index (unique index: An index defined by setting a field's **Indexed** property to **Yes (No Duplicates)**. A unique index will not allow duplicate entries in the indexed field. Setting a field as the primary key automatically defines the field as unique.).
- The related fields have the same data type (data type: The characteristic of a field that determines what type of data it can hold. Data types include Boolean, Integer, Long, Currency, Single, Double, Date, String, and Variant (default).). There are two exceptions. An AutoNumber (AutoNumber data type: In a Microsoft Access database, a field data type that automatically stores a unique number for each record as it's added to a table. Three kinds of numbers can be generated: sequential, random, and Replication ID.) field can be related to a Number field with a **FieldSize** property setting of **Long Integer**, and an AutoNumber field with a **FieldSize** property setting of **Replication ID** can be related to a Number field with a **FieldSize** property setting of **Replication ID**.
- Both tables belong to the same Microsoft Access database. If the tables are linked tables (linked table: A table stored in a file outside the open database from which Access can access records. You can add, delete, and edit records in a linked table, but you cannot change its structure.), they must be tables in Microsoft Access format, and you must open the database in which they are stored to set referential integrity. Referential integrity can't be enforced for linked tables from databases in other formats.
- The following rules apply when you use referential integrity:
- You can't enter a value in the foreign key (foreign key: One or more table fields (columns) that refer to the primary key field or fields in another table. A foreign key indicates how the tables are related.) field of the related table that doesn't exist in the primary key of the primary table. However, you can enter a **Null** (**Null**: A value you can enter in a field or use in expressions or queries to indicate missing or unknown data. In Visual Basic, the **Null** keyword indicates a **Null** value. Some fields, such as primary key fields, can't contain **Null**.) value in the foreign key, specifying that the records are unrelated. For example, you can't have an order that is assigned to a customer that doesn't exist, but you can have an order that is assigned to no one by entering a **Null** value in the CustomerID field.
- You can't delete a record from a primary table if matching records exist in a related table. For example, you can't delete an employee record from the Employees table if there are orders assigned to the employee in the Orders table.
- You can't change a primary key value in the primary table, if that record has related records. For example, you can't change an employee's ID in the Employees table if there are orders assigned to that employee in the Orders table.

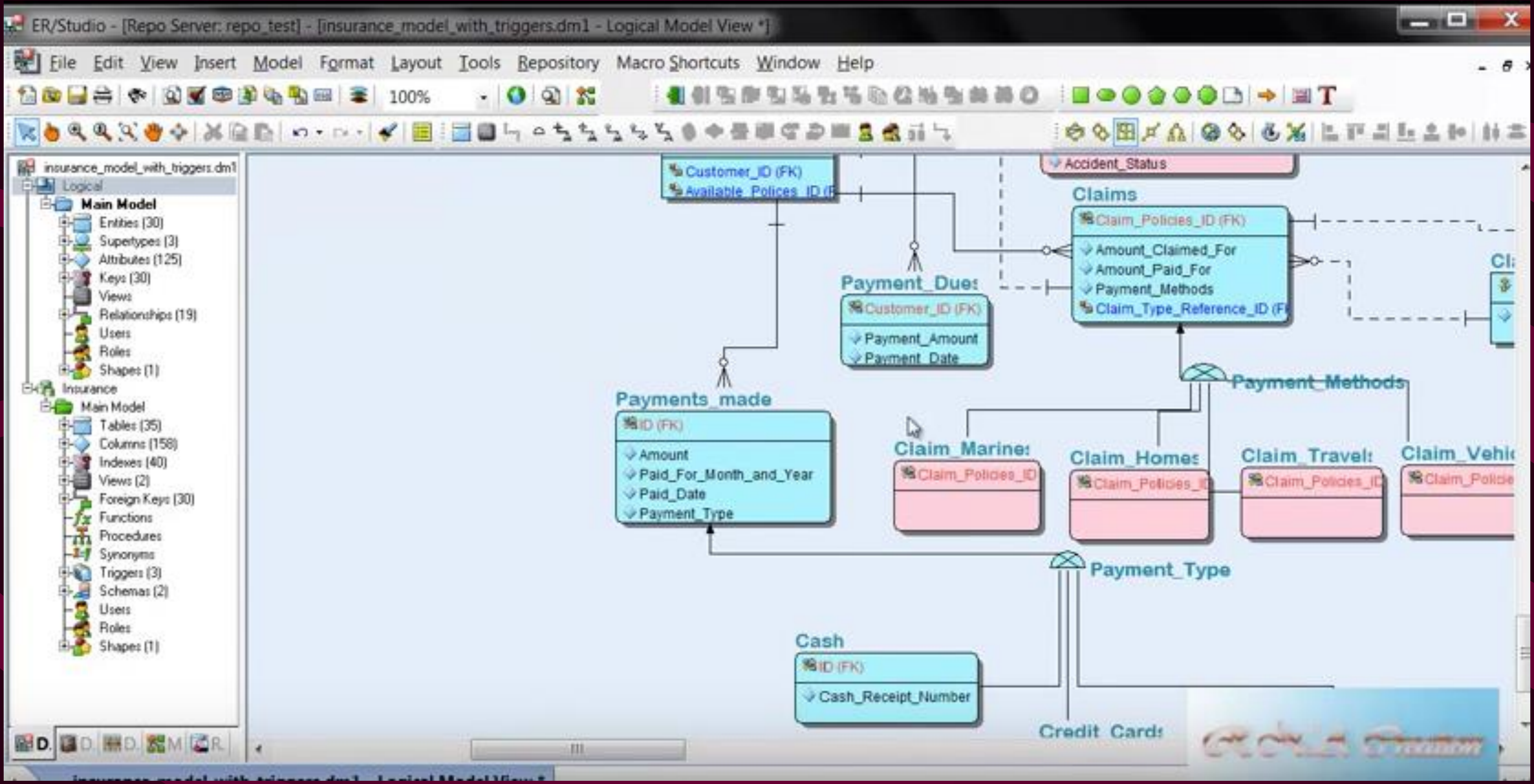
Relationships Created in Access

- The kind of relationship that Microsoft Access creates depends on how the related fields are defined:
- A one-to-many relationship is created if only one of the related fields is a primary key (primary key: one or more fields (columns) whose values uniquely identify each record in a table)
  - A primary key cannot allow **Null** values and must always have a unique index. A primary key is used to relate a table to foreign keys in other tables.) or has a unique index (unique index: An index defined by setting a field's **Indexed** property to **Yes (No Duplicates)**. A unique index will not allow duplicate entries in the indexed field. Setting a field as the primary key automatically defines the field as unique.).
- A one-to-one relationship is created if both of the related fields are primary keys or have unique indexes.
- A many-to-many relationship is really two one-to-many relationships with a third table whose primary key consists of two fields— the foreign keys (foreign key: One or more table fields (columns) that refer to the primary key field or fields in another table. A foreign key indicates how the tables are related.) from the two other tables.
- You can also create a relationship between a table and itself. This is useful in situations where you need to perform a Lookup within the same table. In the Employees table, for example, you can define a relationship between the EmployeeID and ReportsTo fields, so that the ReportsTo field can display employee data from a matching EmployeeID.
- **Note** If you drag a field that isn't a primary key and doesn't have a unique index to another field that isn't a primary key and doesn't have a unique index, an indeterminate relationship is created. In queries containing tables with an indeterminate relationship, Microsoft Access displays a default join (join: An association between a field in one table or query and a field of the same data type in another table or query. Joins tell the program how data is related. Records that don't match may be included or excluded, depending on the type of join.) line between the tables, but referential integrity (referential integrity: Rules that you follow to preserve the defined relationships between tables when you enter or delete records.) won't be enforced, and there's no guarantee that records are unique in either table.

Automatically Building Physical Model (relational tables) in Case Product



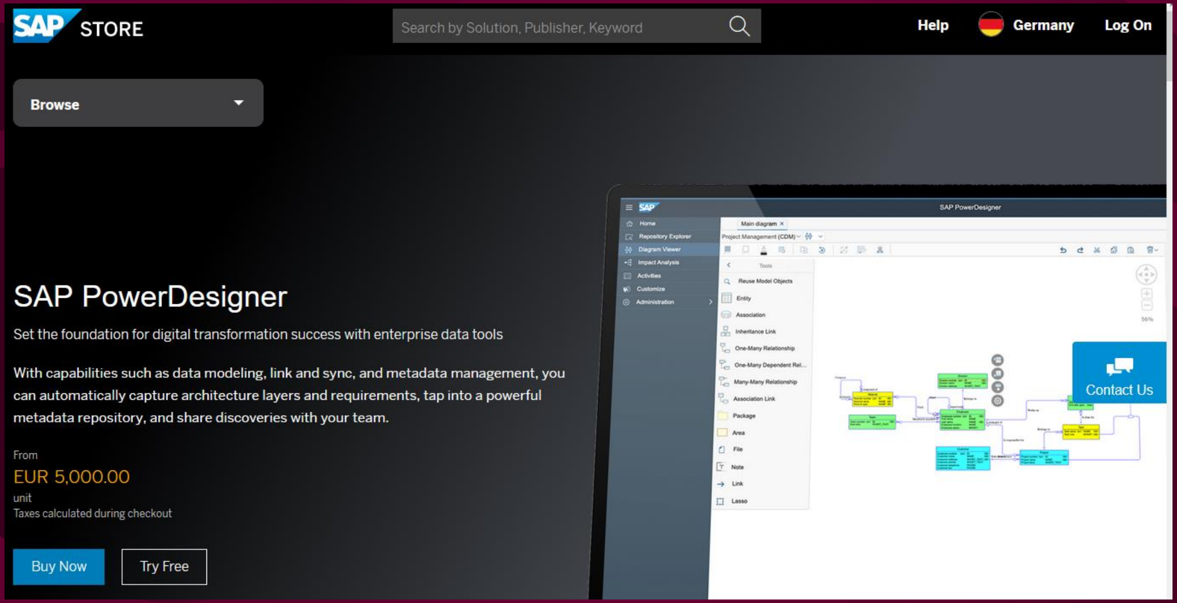
ER Studio


ER/Studio - [Repo Server: repo_test] - [insurance_model_with_triggers.dm1 - Logical Model View *]
File Edit View Insert Model Format Layout Tools Repository Macro Shortcuts Window Help
100%
insurance_model_with_triggers.dm1
Logical
Main Model
Entities (30)
Supertypes (3)
Attributes (125)
Keys (30)
Views
Relationships (19)
Users
Roles
Shapes (1)
Insurance
Main Model
Tables (35)
Columns (158)
Indexes (40)
Views (2)
Foreign Keys (30)
Functions
Procedures
Synonyms
Triggers (3)
Schemas (2)
Users
Roles
Shapes (1)
Customer_ID (FK)
Available_Polices_ID (F
Accident_Status
Claims
Claim_Policies_ID (FK)
Amount_Claimed_For
Amount_Paid_For
Payment_Methods
Claim_Type_Reference_ID (F
Payment_Dues
Customer_ID (FK)
Payment_Amount
Payment_Date
Payment_Methods
Payments_made
ID (FK)
Amount
Paid_For_Month_and_Year
Paid_Date
Payment_Type
Claim_Marines
Claim_Policies_ID
Claim_Homes
Claim_Policies_I
Claim_Travels
Claim_Policies_ID
Claim_Vehic
Claim_Polici
Payment_Type
Cash
ID (FK)
Cash_Receipt_Number
Credit_Cards
insurance_model_with_triggers.dm1 - Logical Model View *

SAP Power Designer

[https://www.sapstore.com/solutions/61111/SAP-PowerDesigner?url_id=ctabutton-UnifiedSearchResult]

- Power Designer was purchased by SAP and became SAP Power Designer in 2010



SAP PowerDesigner

Set the foundation for digital transformation success with enterprise data tools

With capabilities such as data modeling, link and sync, and metadata management, you can automatically capture architecture layers and requirements, tap into a powerful metadata repository, and share discoveries with your team.

From
EUR 5,000.00
unit
Taxes calculated during checkout

Referential Integrity Checking within Application

- Database independence
  - Different database products have different ways to handle referential integrity
- Can program any kind of business logic
  - Trees vs Forests (top level nodes do not have master accounts)
- Can give meaningful error messages
- Can check all the relationships for each entity involved
  - Check each FK for add’s/mod’s
  - Check dependencies for each delete; verify cascades

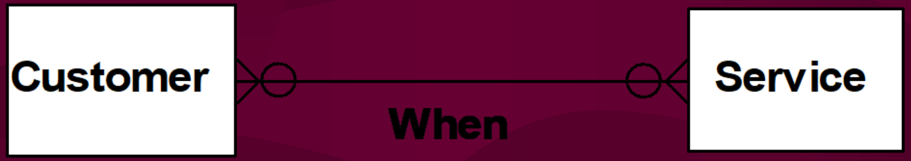
Referential Integrity Checking in Application Code (Deletion)

```
// referential integrity check in PHP/MySql for web application
        $query = "select wbsCode from $tableName ";
        $query = $query."where wbsMaster = '$id'"; // text key
        $result = mysql_query($query);
        if (mysql_affected_rows() > 0) {
                echo '<H2>Cannot delete this code,’;
                echo ‘ another code(s) has this code as a master’;
                echo ‘ !</H2>';
                exit;
        }
```

Referential Integrity Checking in Application Code (Insertion)

```
•   if ($master != "") {  // referential integrity check in PHP/MySql for Web App
•           if ($level == 0) {
•                echo '<h2>Error: If a Master WBS Code is specified,’;
•                echo ‘ the level must not be zero !</h2>';
•                exit;
•           }
•           $query2 = "select * from $tableName where wbsCode='".$master."';";
•           $result2 = mysql_query($query2);
•           $num_results2 = mysql_num_rows($result2);
•           if ($num_results2 == 0) {
                        echo '<h2>Error: Invalid/Non-existent Master WBS Code !</h2>';
•                       exit;
•           }
•   }
```

Services ER Diagram



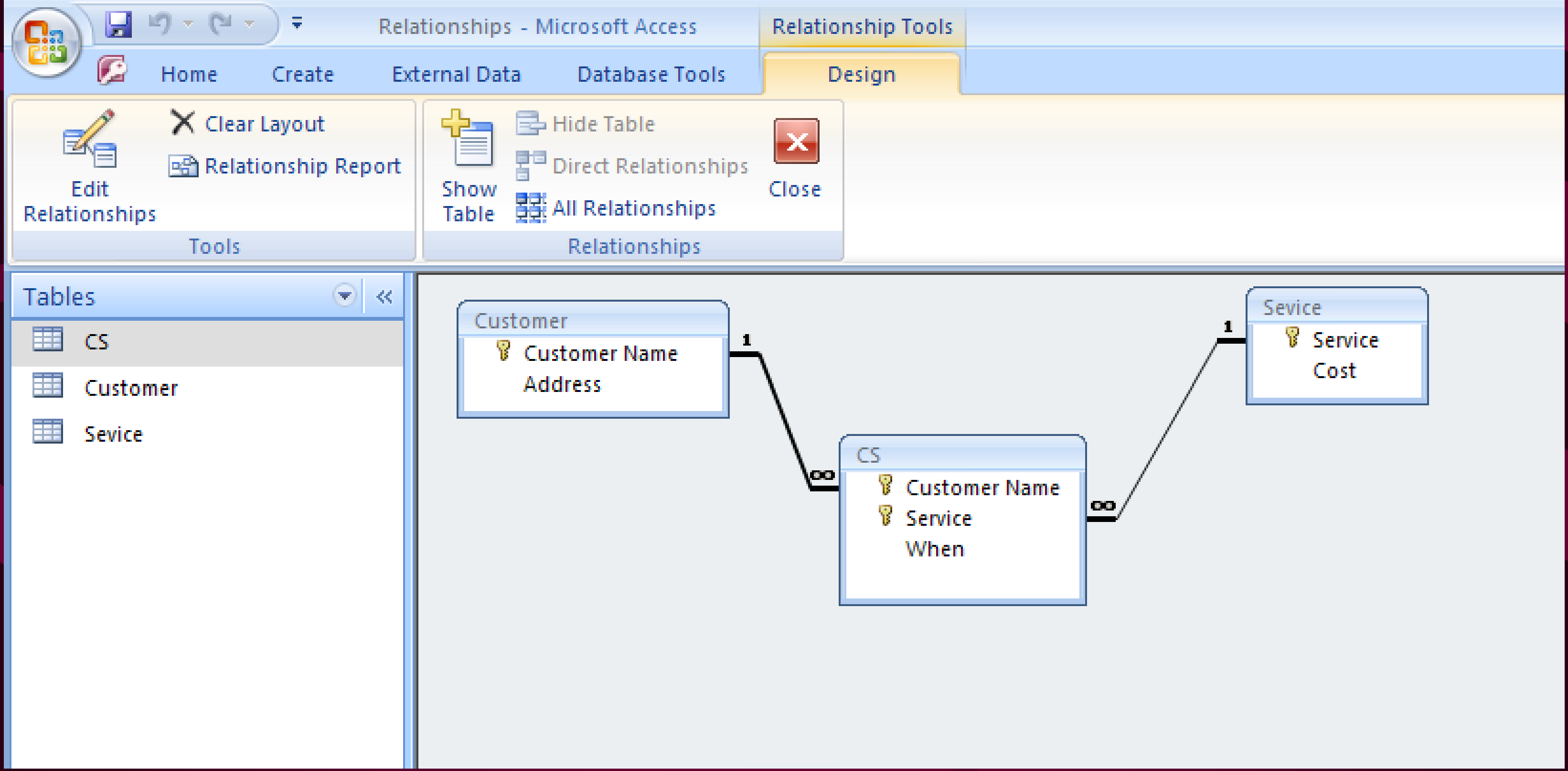
Specify Relational Tables

Don’t look ahead !

Services Model

[if charging a different cost to each customer, then cost would also go in intersection table]

CUSTOMER (<u>Name</u>, Address, …)
SERVICE (<u>ServiceCode</u>, Cost, …)
C_S (*<u>Name, ServiceCode</u>*, When)



Customer
- 🔑 Customer Name
- Address

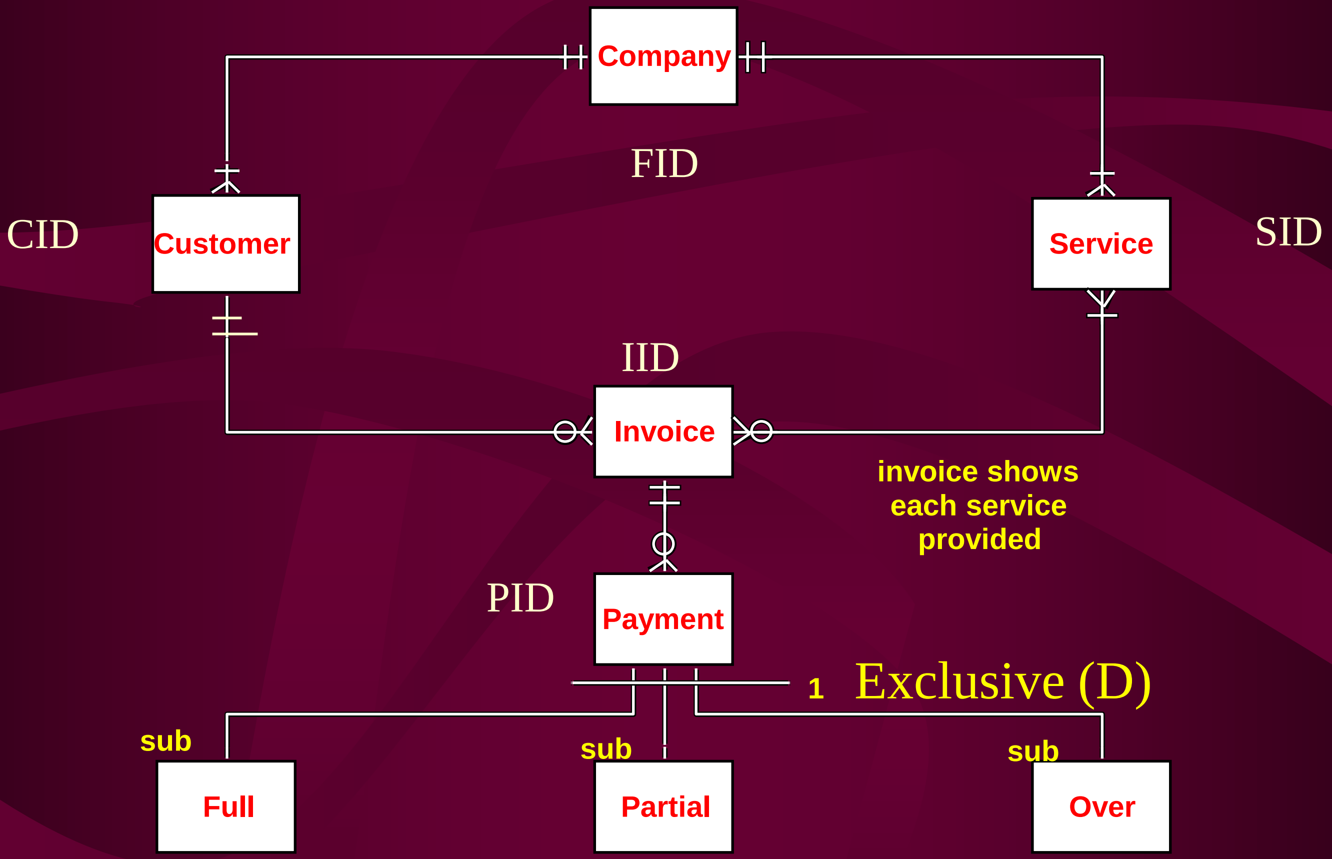
CS
- 🔑 Customer Name
- 🔑 Service
- When

Sevice
- 🔑 Service
- Cost

Class Exercise

- Create relational model for A/R office shown in the following diagram!
- Assume payments are applied to only one invoice

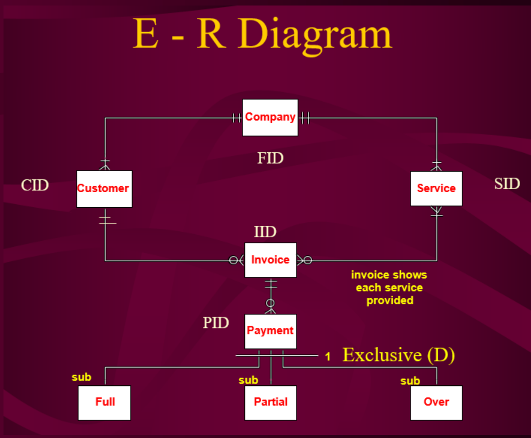
E - R Diagram


Company
FID
CID
Customer
Service
SID
IID
Invoice
invoice shows each service provided
PID
Payment
1
Exclusive (D)
sub
sub
sub
Full
Partial
Over

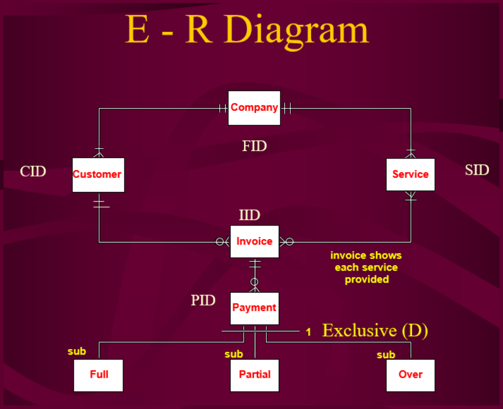
- Specify the relational tables !

Don’t look ahead !

Relational Model



- COMPANY (FID, Name, Address, ...)
- CUSTOMER (CID, Name, Address, ...,*FID*)
- SERVICE (SID, Desc, Basis, Rate, *FID*)
- INVOICE (IID, *CID*, TotalDue, TotalPaid)
- LINEITEM (*IID, SID*, Date, Qty, Charge)
- or if a service can appear more than once on an invoice (not known from E-R diagram):
  - – LINEITEM (*IID*,Line#, *SID*, Date, Qty, Charge)
  - – not handled directly by E-R CASE



E - R Diagram

- PAYMENT (PID, Type, Date, Amount, RefNum, *IID*)
  - PARTIAL (*PID*, Reason, ...)
  - OVER (*PID*, RebateCheckNumber, ...)
  - FULL (*PID*, ThankYouCallRef,...)
  - Type is not null, and must be either P, O, or F (each of these could trigger the insert into sub type table, as well as calculation of total paid in the INVOICE table)

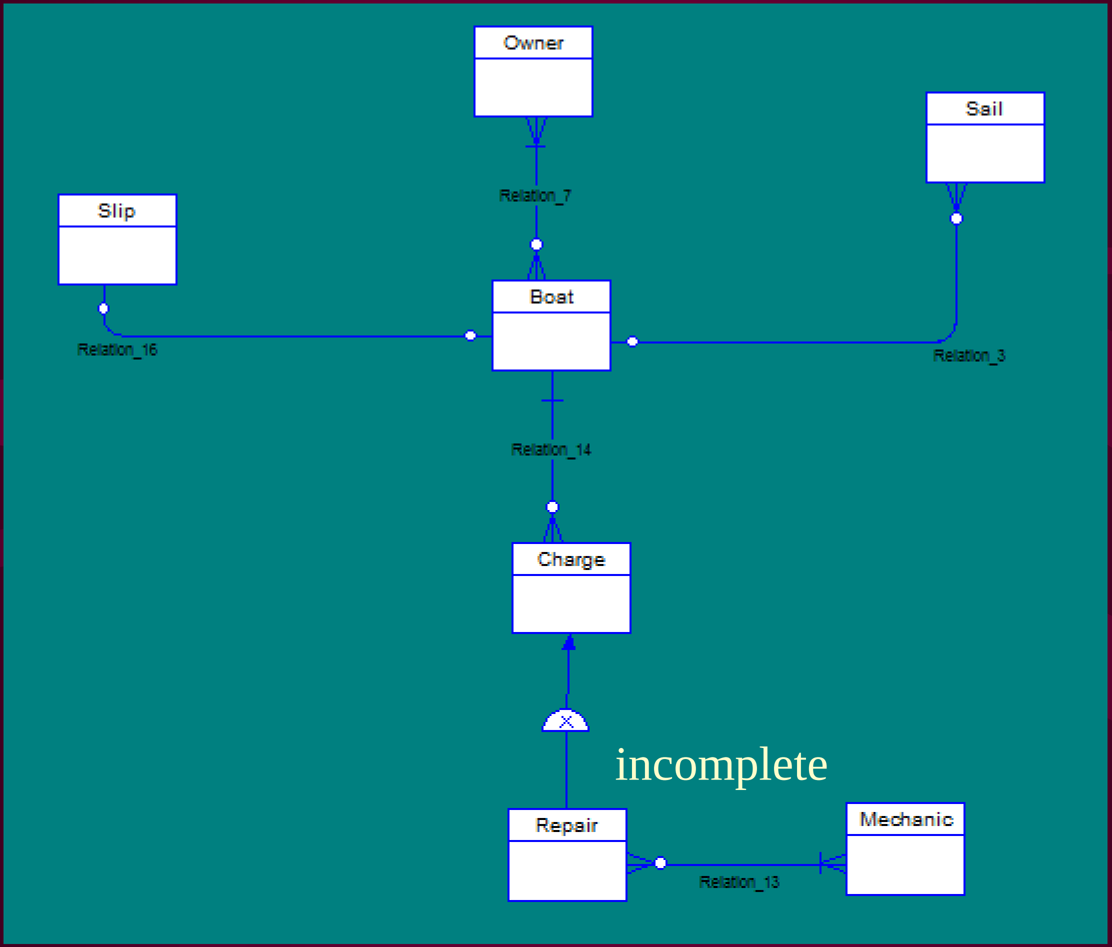
- What if we allow a payment to be applied to many invoices ?

Don’t look ahead !

Many to Many Relationship Between Payment and Invoice

- PAYMENT (PID, Type, Date, Amount, *IID*)
  - PARTIAL (*PID*, Reason, ...)
  - OVER (*PID*, RebateCheckNumber, ...)
  - FULL (*PID*, ThankYouCallRef,...)
- APPLICATION (*PID, IID*, AmountApplied)
  - APPLICATION is intersection table
  - Need constraint that all amounts applied for a payment add up to the payment amount

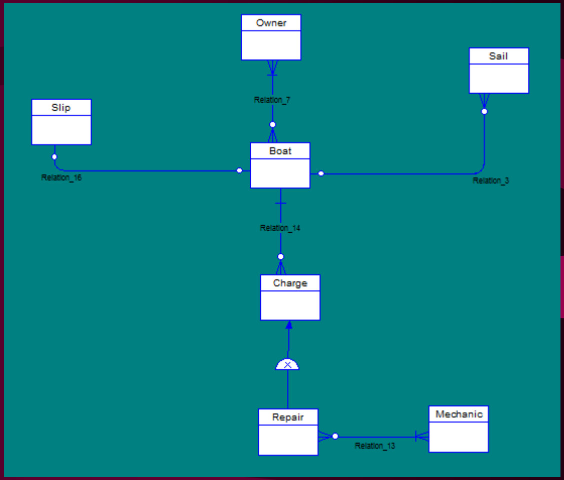
Marina E - R Diagram



Specify the relational tables !

Don’t look ahead !

Relational Model

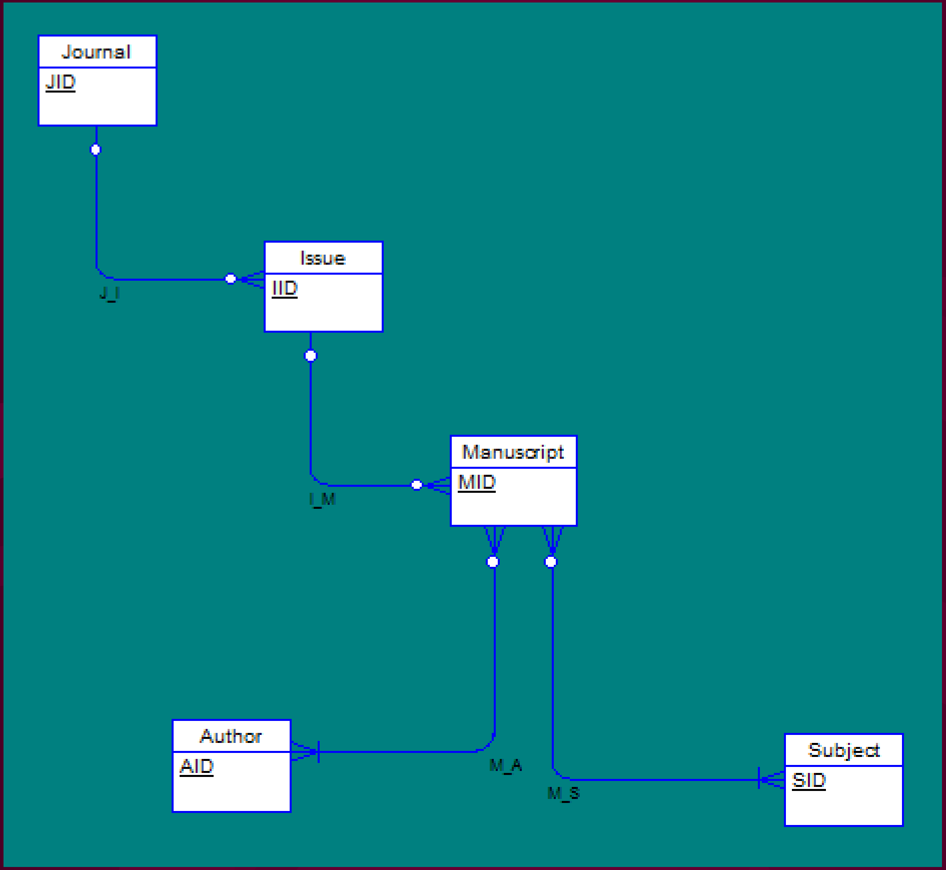


- BOAT (RNum, Type, Length, …)
- OWNER (SS, Name, …)
- BOAT-OWNER (*RNum, SS*, PrimaryContact)
- SLIP (SlipNum, *Rnum*, Size, Location) RNum can be NULL
- CHARGE (TicketNum, *RNum*, Date, Type, AmountDue, AmountPaid) RNum cannot be NULL, Type is discriminator
- REPAIR (TicketNum, Description) SUBTYPE
- MECHANIC (MID, Name, …)
- REPAIR-MECH(*TicketNum, MID*, Hours) Intersection Table
- SAIL (SID, *RNum*, Desc, …) RNum can be NULL

Problem Statement

- A database is required to maintain journal articles. Each journal is identified by its journal ID (JID), and has many issues published on a certain date
- The articles (called) manuscripts have a unique code number (MID) and only appear in one issue (IID)
- The manuscripts are written by one or more authors and concern one or more subjects
- The author information includes their code number (AID), name and city, and the subject information includes its code number (SID) and name
- The manuscript information also includes title, first page, and last page numbers

E-R Model

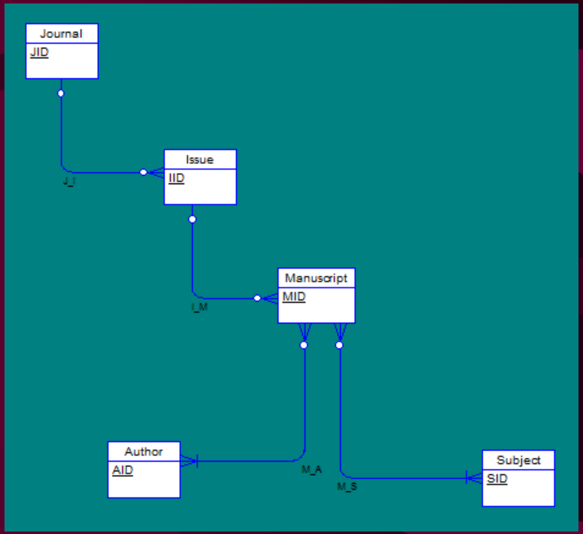


What are the relational tables ?

Don’t look ahead !

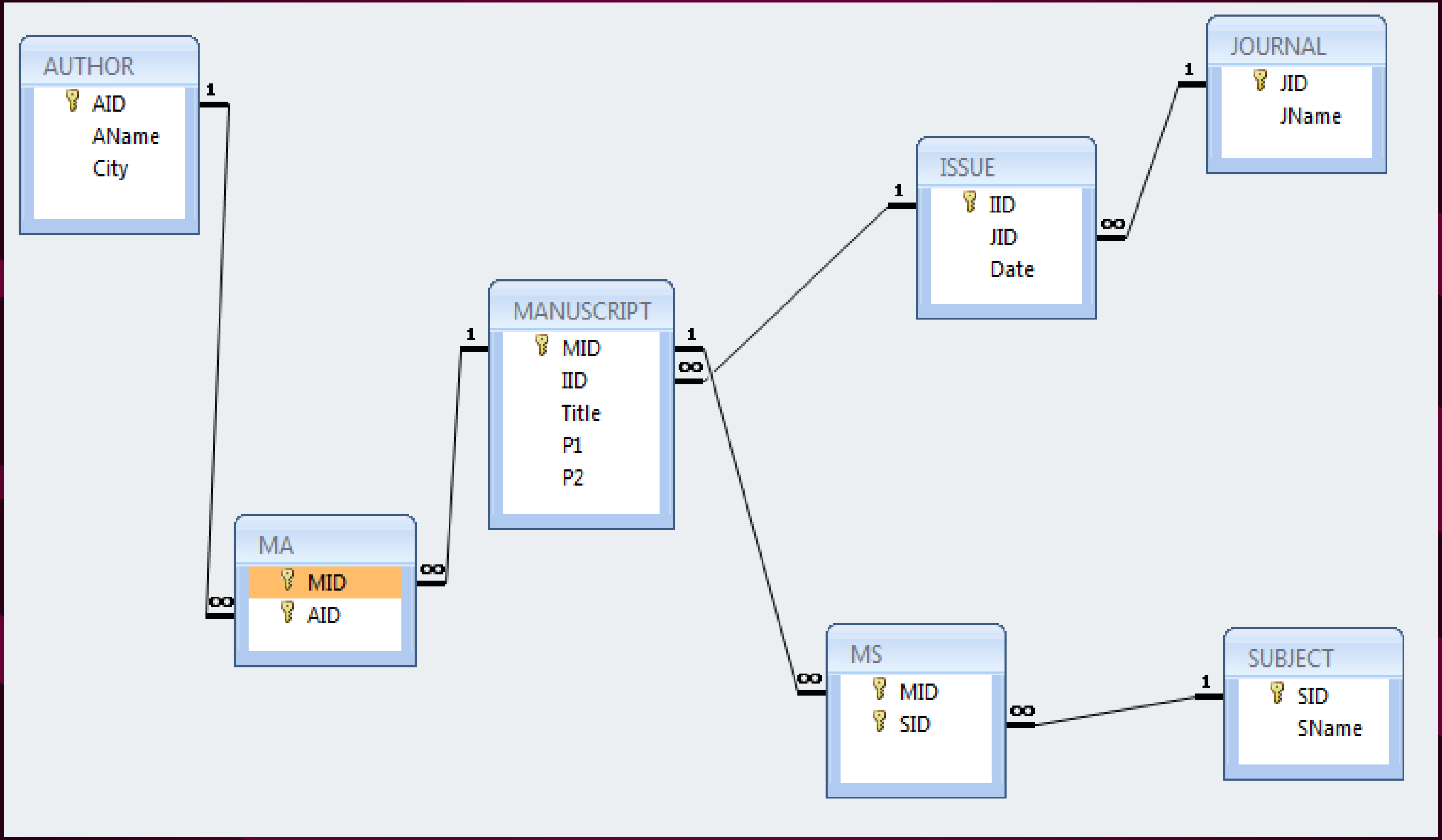
Relational Tables

(PK is underlined, FK's are italicized)



- JOURNAL (JID, JName)
- ISSUE(IID, *JID*, Date)
- MANUSCRIPT(MID, *IID*, Title, P1, P2)
- AUTHOR (AID, AName, City)
- SUBJECT (SID, SName)
- MA (*MID, AID*) **manuscripts by authors**
- MS (*MID, SID*) **subjects by authors**

Access Model



AUTHOR
- AID
- AName
- City

MA
- MID
- AID

MANUSCRIPT
- MID
- IID
- Title
- P1
- P2

ISSUE
- IID
- JID
- Date

JOURNAL
- JID
- JName

MS
- MID
- SID

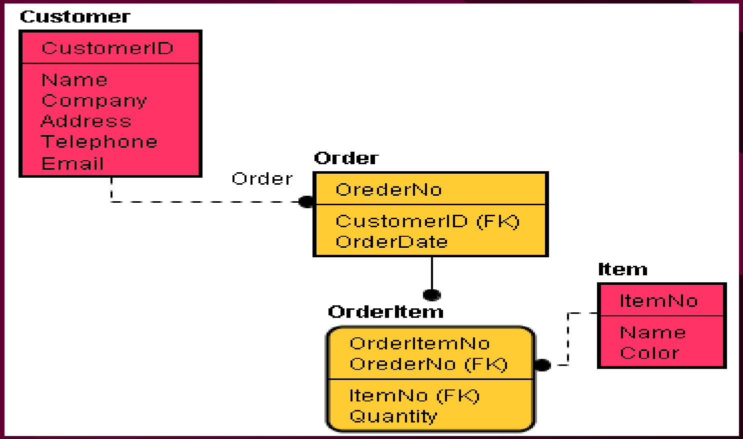
SUBJECT
- SID
- SName

References

- Database Design Using Entity-Relationship Diagrams (Foundations of Database Design) by Sikha Bagui and Richard Earp
- Data Modeling Diagrams: Entity, Entity-Relationship Model, Warnier|orr Diagram, Frsar, Data Structure Diagram by Books LLC
- Database Systems: Design, Implementation, and Management (with Bind-In Printed Access Card) by Carlos Coronel, Steven Morris, and Peter Rob
- Concepts of Database Management (Cengage Sam Compatible Products) by Philip J. Pratt and Joseph J. Adamski
- Data Modeling Essentials, Third Edition by Graeme Simsion and Graham Witt Database Development for Dummies by Allen G. Taylor

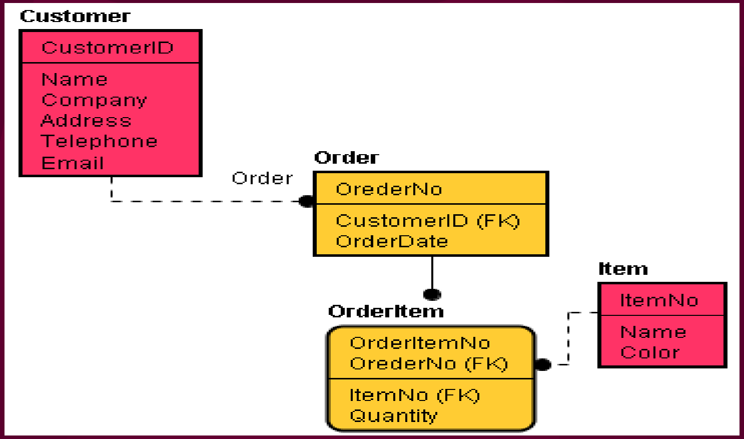
Homework

- Textbook Chapter 5
- Review questions 1 thru 7
- Test 1 upcoming
- IDEF1X Appendix



Appendix

IDEF1X



IDEF

- IDEF (Integrated Definition) became a U.S. national standard in 1993
  - IDEF1X (Definition 1, Extended)
  - Robert Brown 1979 [Lockheed]
- Based upon development in the US Air Force in 1980’s and eventually all of DoD
- Only used for relational databases, and includes the concept of a domain
- May be required in US government contract work

IDEF Models

- Two model types:
  - E-R (Logical) – shows entities and relationships
  - Key Based (Physical) – shows relational tables and keys

E-R Model	IDEF Model
Entity	Entity
Attribute	Attribute
Relationship	Relationship
1:1 & 1:N Relationships	Non-Identifying Connecting Relationship
N:M Relationship	Non-specific relationship
ID-Dependent Relationship	Identifying Connecting Relationship
Weak Entity (non ID-dept.)	None
Supertype Entity	Generic Entity
Subtype Entiry	Category Entity

IDEF Relationships

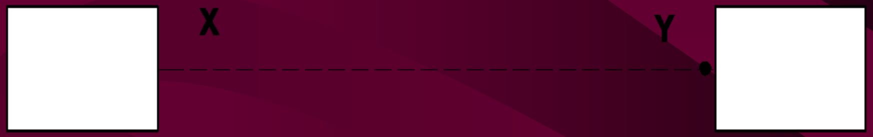
- Relationship types:
  - Non-Identifying Connection
  - Identifying Connection
  - Non-specific
  - Categorization
- All relationship types except for categorization can be named

Non-Identifying Connection

- An "association" of entities; "has-a" relationship
- Shown with dashed line with a small solid circle (dot) on the child (many) side
  - Even 1 to 1 relationships need a "parent" designation
- Default relationship of 1:N with mandatory parent (min cardinality of 1) and optional child (min cardinality of zero) needs no further symbols
- For minimum cardinality of one (or greater) on the child side, a "P" ("positive") is placed next to the dot
- For minimum cardinality of zero on the parent side, a diamond is placed at that end of the dashed line
- For 1-1 relationships, a "1" is placed near the dot indicating that 1 child is required (1 and only 1), or a "Z" is placed there indicating zero or 1 children

Non-Identifying Connection

- X (one/parent side)
  - Blank – min cardinality is 1
  - Diamond – min cardinality is zero
- Y (many/child side)
  - Blank – min cardinality is zero
  - P – min cardinality is 1 or more
  - 1 – a 1:1 relationship with a minimum cardinality of 1
  - Z – a 1:1 relationship with a minimum cardinality of 0

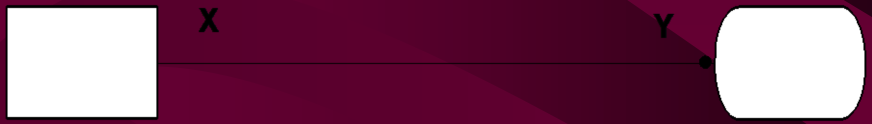


Identifying Connection

- Same as "ID-dependent" relationships in E-R
- A solid line is used, and the small filled circle (dot) is on the many (child) side of the relationship
- The identifier of the parent is part of the identifier of the child (the "FK" [foreign key] note may be applied to that identifier in the child entity)
- The entity symbol on the child side may also have rounded corners
- A 1, Z, or P may be placed next to the dot as was the case for the non-identifying connection
- Weak entities (that are not ID-dependent) cannot be specifically show in the IDEF model, except by stating that the minimum cardinality on the one (parent) side is one

Identifying Connection

- X
  - Blank – min cardinality is 1
  - Diamond – not allowed here !
- Y
  - Blank – min cardinality is zero
  - P – min cardinality is 1 or more
  - 1 – a 1:1 relationship with a minimum cardinality of 1
  - Z – a 1:1 relationship with a minimum cardinality of 0

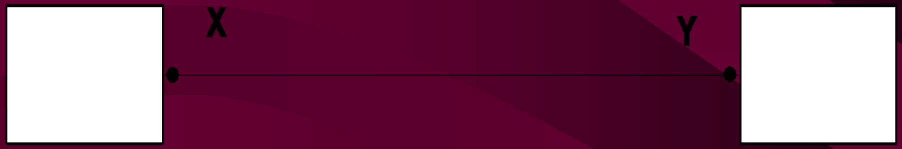


Non-specific

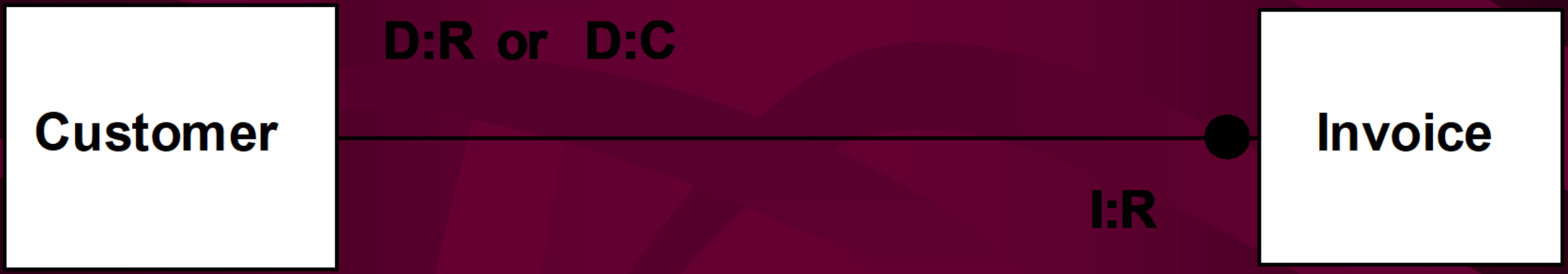
- A many to many relationship
- Uses a solid line with a small filled circle (dot) on both sides
- Can have a “P” on either side
- Some methodologies may force changing this into two 1:N relationships
- In IDEF, N degree relationships must be broken down into binary relationships

Non-Specific Connection

- X
  - Blank – min cardinality is 0
  - P – min cardinality is 1
- Y
  - Blank – min cardinality is 0
  - P – min cardinality is 1



Referential Integrity Options shown on IDEF



Delete -> Restrict
Delete -> Cascade

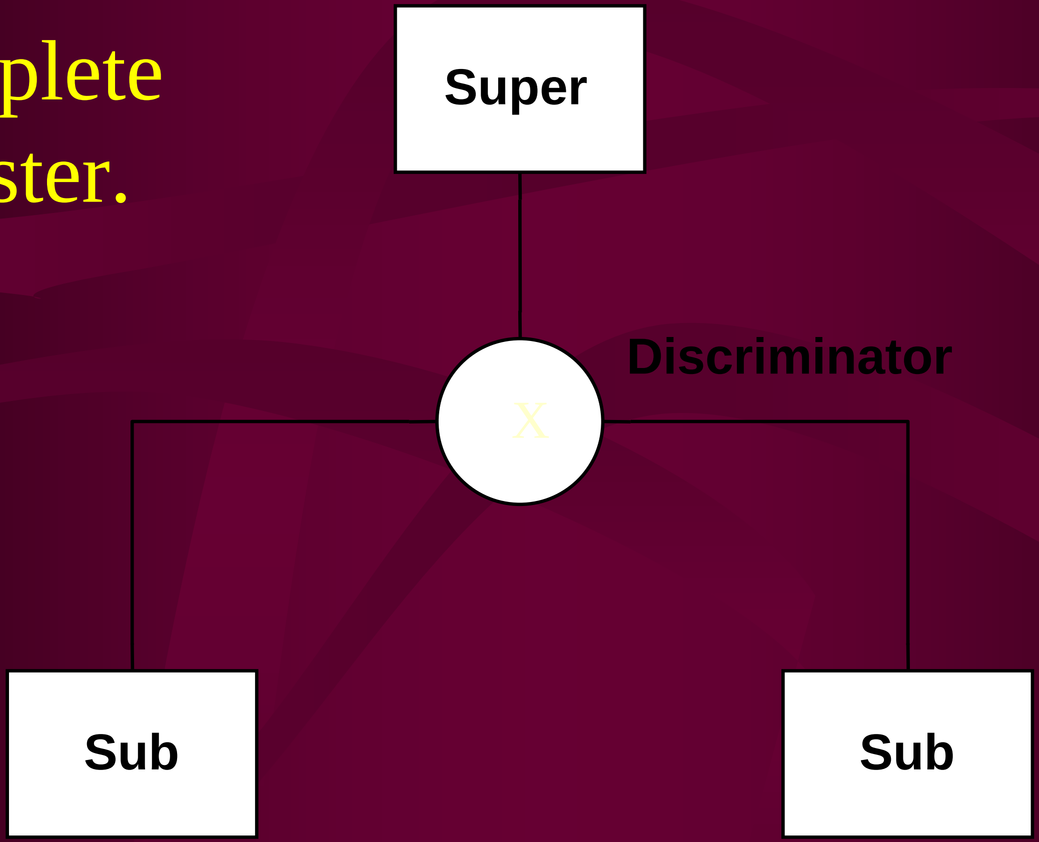
Insert -> Restrict

Categorization

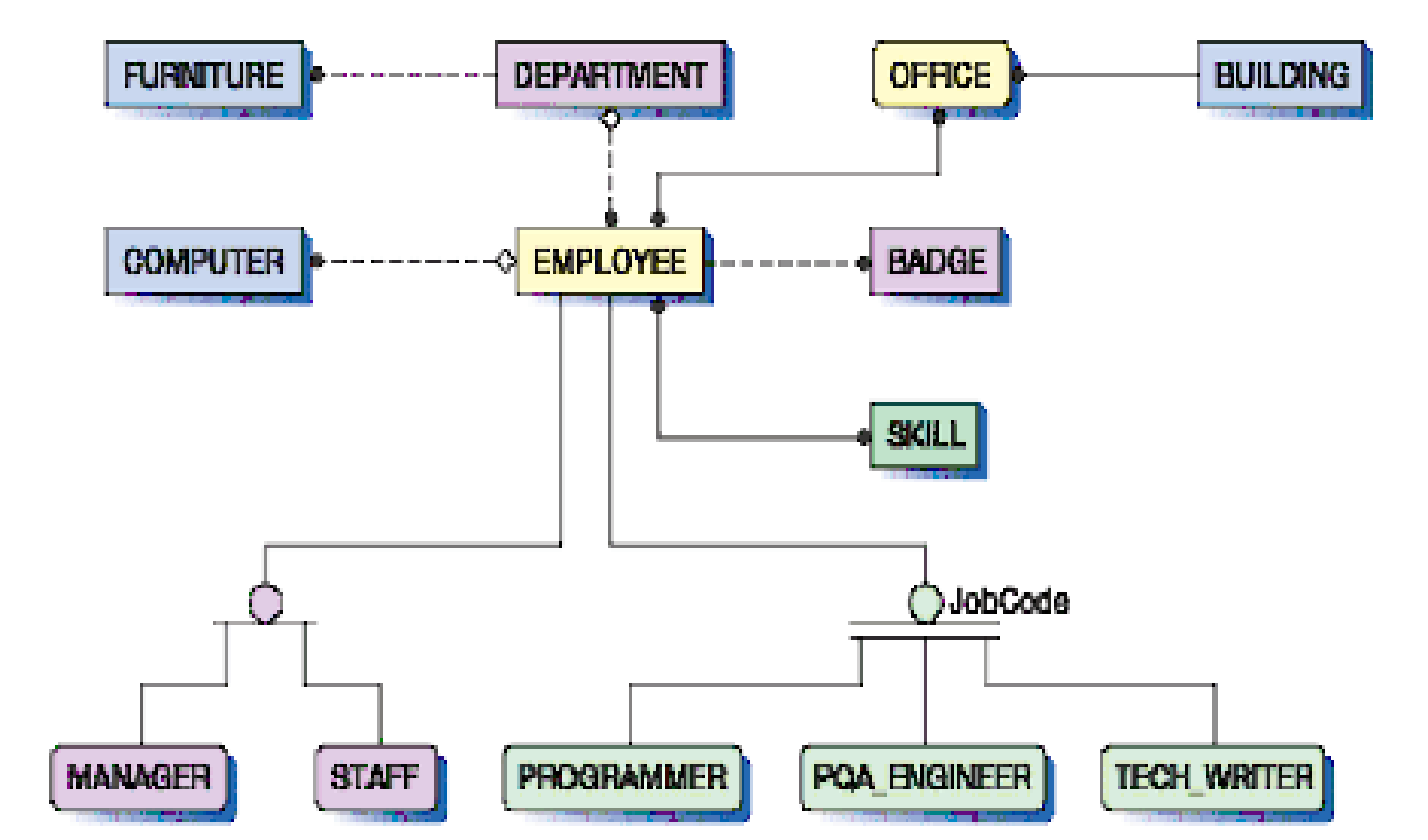
- Same as super/sub type relationship in ER
- Sub types can be grouped into "categorization clusters"; each cluster may have a "discriminator" which is an attribute of the supertype to indicate which subtype applies to each supertype entity
- A supertype cannot be associated with more than one subtype within a cluster, but can be associated with different subtypes in different clusters

- Clusters can be “complete” (double line under cluster symbol, or “C” within circle) where all subtypes are itemized or “incomplete” where not all subtypes are itemized (a generic type need not be one of the category types in this cluster)
- All subtypes require a supertype so the min and max cardinality in that direction is always one
- Currently in IDEF, a category entity cannot have more than one associated generic entity

A “C” in the circle would indicate a complete cluster.



IDEF Example


FURNITURE
DEPARTMENT
OFFICE
BUILDING
COMPUTER
EMPLOYEE
BADGE
SKILL
JobCode
MANAGER
STAFF
PROGRAMMER
PQA_ENGINEER
TECH_WRITER