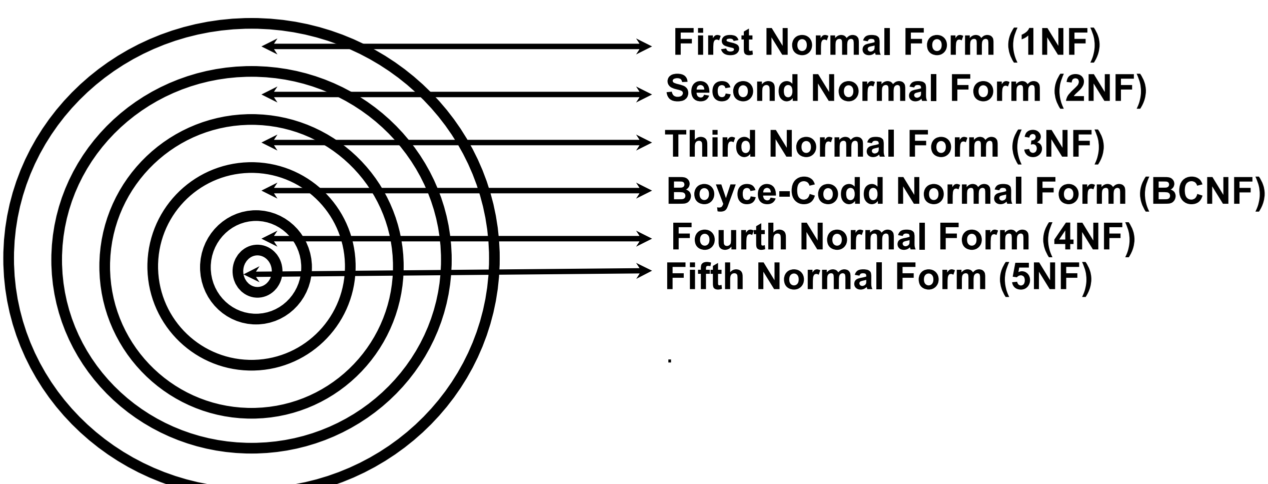
Good Database Design Principles

1. **no redundancy**
   - a field is stored in *only one table*, unless it happens to be a foreign key
   - replication of foreign keys is permissible, because they allow two tables to be joined together
2. **no "bad" dependencies**
   - in the dependency diagram of any relation in the database, the determinant should be the whole primary key, or a candidate key. Violations of this rule include:
     - partial dependencies
     - transitive dependencies

normalization is the process of eliminating "bad" dependencies by splitting up tables and linking them with foreign keys

- "normal forms" are categories that classify how completely a table has been normalized
- there are six recognized *normal forms* (*NF*):



First Normal Form

- a table is said to be in the <u>first normal form (1NF)</u> if all its attributes are *atomic*. Attributes that are *not* atomic go by the names
  - Nested relations, nested tables, or sub-tables
  - Repeating groups or repeating sections
  - List-valued attributes
- example of a table that is *not* in first normal form:

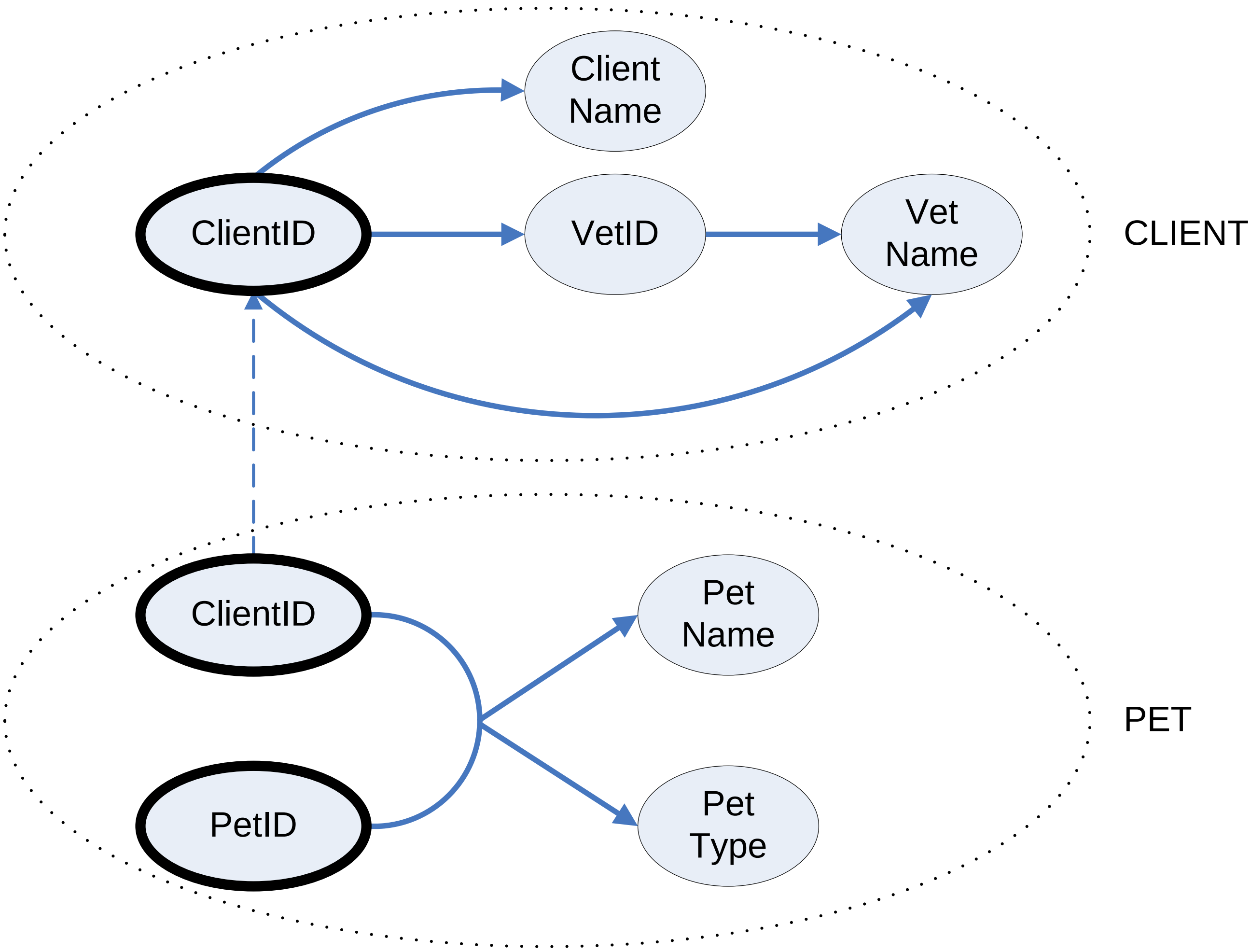
Client ID	Client Name	VetID	VetName	PetID	PetName	PetType
2173	Barbara Hennessey	27	PetVet	1 2 3	Sam Hoober Tom	Bird Dog Hamster
4519	Vernon Noordsy	31	PetCare	2	Charlie	Cat
8005	Sandra Amidon	27	PetVet	1 2	Beefer Kirby	Dog Cat
8112	Helen Wandzell	24	PetsRUs	3	Kirby	Dog

CLIENT(<u>ClientD</u>, ClientName, VetID,
VetName, PET(<u>PetID</u>, PetName, PetType))

- This kind of nested or hierarchical form is a very natural way for people to think about or view data.
- However, the relational database philosophy claims that it may not be a very good way for computers to *store* some kinds of data.
- Over the years, a lot of information systems have stored data in this kind of format – but they were not *relational* databases

RELATIONAL DATABASE DESIGN

- **In order to eliminate the nested relation, pull out the nested relation and form a new table**
- **Be sure to include the old key in the new table so that you can connect the tables back together.**



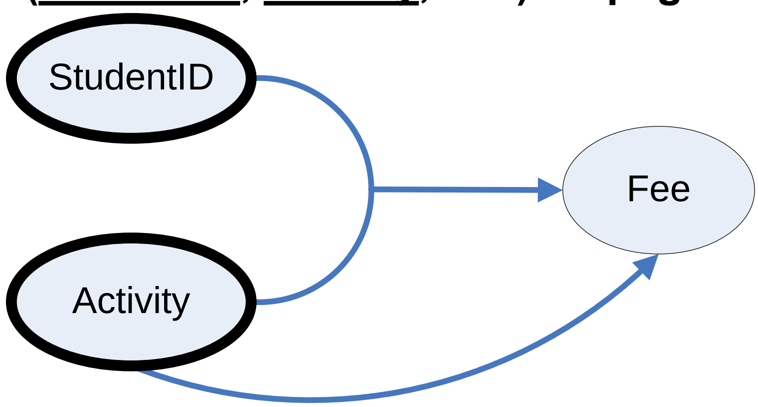
CLIENT(<u>ClientD</u>, ClientName, VetID, VetName)

PET(<u>ClientID</u>, <u>PetID</u>, PetName, PetType)
ClientID foreign key to CLIENT

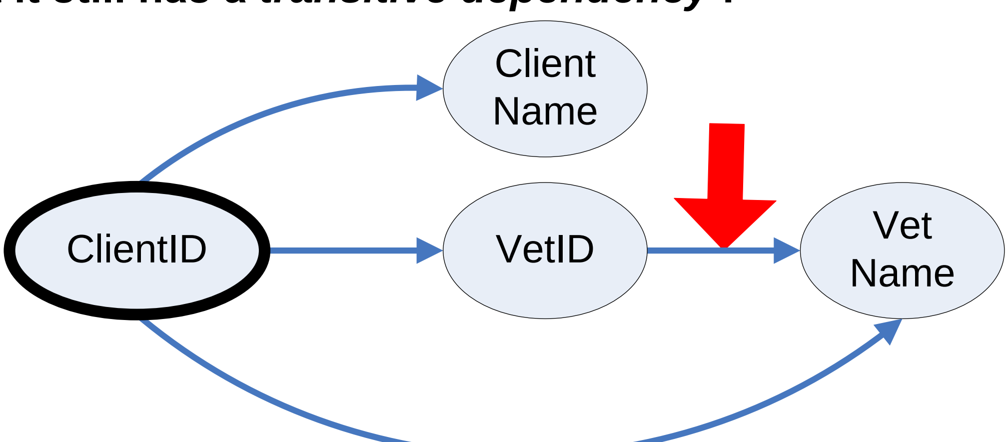
- **In this particular example, note that PetID is only unique within sets of pets with the same owner.**

Second Normal Form

- Recall: a *partial dependency* occurs when
  - You have a composite primary key
  - A non-key attribute depends on part of the primary key, but not all of it
- A table in 1NF is said to be in the **second normal form (2NF)** if it does not contain any partial dependencies.
  - Example of a partial dependency:
    ACTIVITY(StudentID, Activity, Fee) on pages 6, 7, and 9



- Our new CLIENT-PET database does not have any partial dependencies
- So, it already in second normal form
- But it still has a *transitive dependency* :



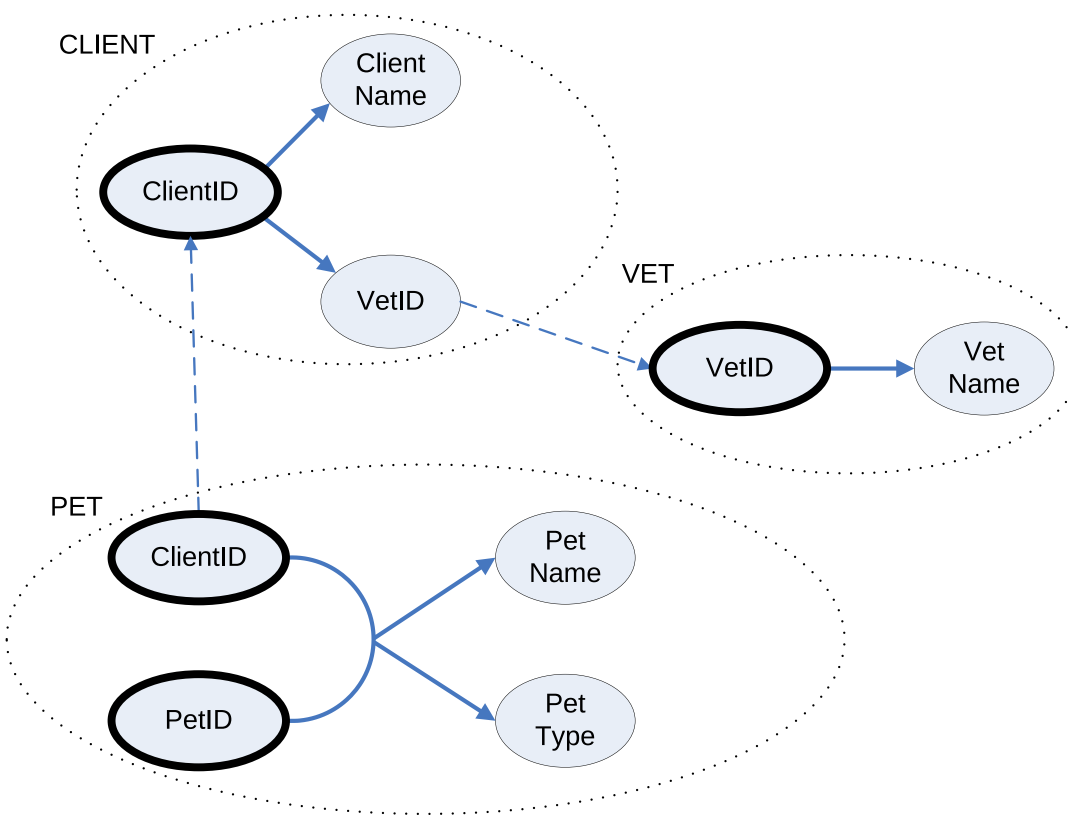
Third Normal Form

- **Recall: a *transitive dependency* happens when a non-key attribute depends on another non-key attribute, and that attribute could not have been used as an alternative primary key (or the same thing for a composition of several attributes).**
- **A table of 2NF is said to be in the third normal form (3NF) if it does not contain any transitive dependencies,**
- **In order to eliminate transitive dependency, we split the CLIENTS table again:**

CLIENTS(ClientID, ClientName, VetID)
VetID foreign key to VET

PETS(ClientID, PetID, PetName, PetType)
ClientID foreign key to CLIENT

VETS(VetID, VetName)



Third Normal Form (Cont.)

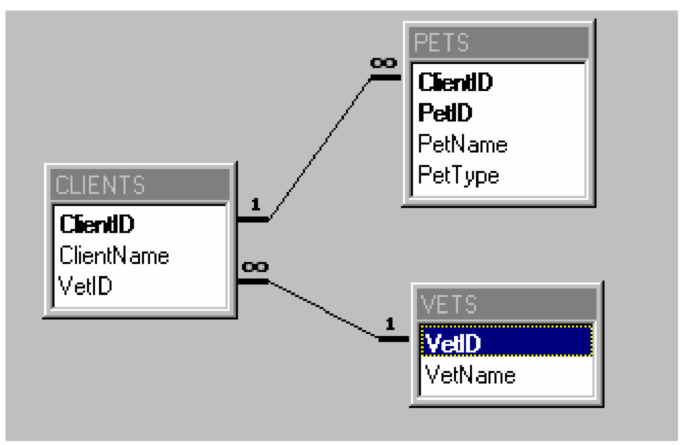
• **CLIENTS-PETS-VETS database in third normal form:**

Client ID	Client Name	VetID
2173	Barbara Hennessey	27
4519	Vernon Noordsy	31
8005	Sandra Amidon	27
8112	Helen Wandzell	24

Client ID	PetID	PetName	PetType
2173	1	Sam	Bird
2173	2	Hoober	Dog
2173	3	Tom	Hamster
4519	2	Charlie	Cat
8005	1	Beefer	Dog
8005	2	Kirby	Cat
8112	3	Kirby	Dog

VetID	VetName
27	PetVet
31	PetCare
24	PetsRUs

with MS Access table relationships



• **the database consists of three types of entities, stored as distinct relations in separate tables:**

  • ***clients*** **(CLIENTS)**

  • ***pets*** **(PETS)**

  • ***vets*** **(VETS)**

• **there is no *redundancy* (only *foreign keys* are replicated)**

• **there are no *partial* and *transitive dependencies***

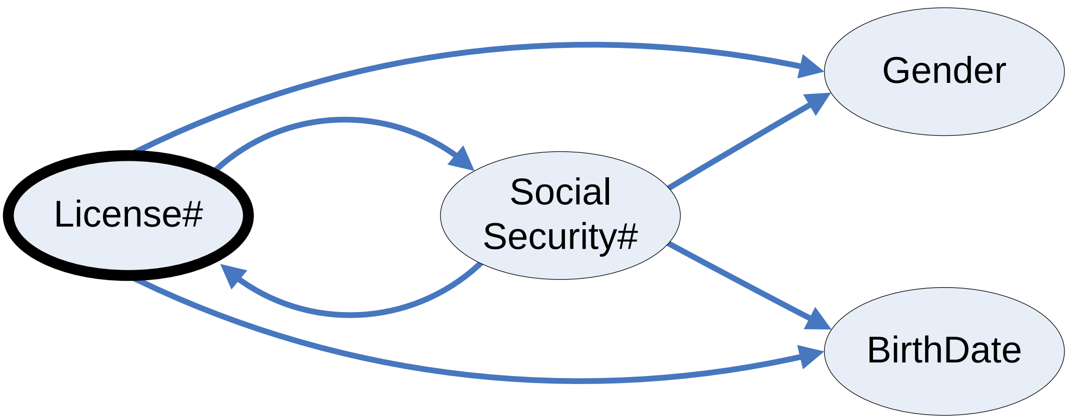
Normal Forms and Normalization

- The distinctions between third normal form (3NF), Boyce-Codd normal form (BCNF), fourth normal form (4NF), and fifth normal form (5NF) are subtle.
- They have to do with overlapping sets of attributes that could be used as primary keys (composite candidate keys).
- For our purposes, it's enough to know about 3NF.
    - You need to be able to put a database in 3NF.
    - That is more important than recognizing 1NF and 2NF
- Key factors to recognize 3NF:
    - All attributes atomic ─ gives you 1NF.
    - Every determinant in every relationship is the whole primary key (or could have been chosen as an alternative primary key) – guarantees no partial or transitive dependencies.

- Redesigning a database so it's in 3NF is called *normalization*.

Example With Multiple Candidate Keys

DRIVER(<u>License#</u>, SocialSecurity#, Gender, BirthDate)



- **The dependencies SocialSecurity# → Gender and SocialSecurity# → BirthDate are *not* considered transitive because we could have chosen SocialSecurity# as the primary key for the table.**
- **This kind of design will not give rise to anomalies.**

Normalization Example: Hardware Store Database

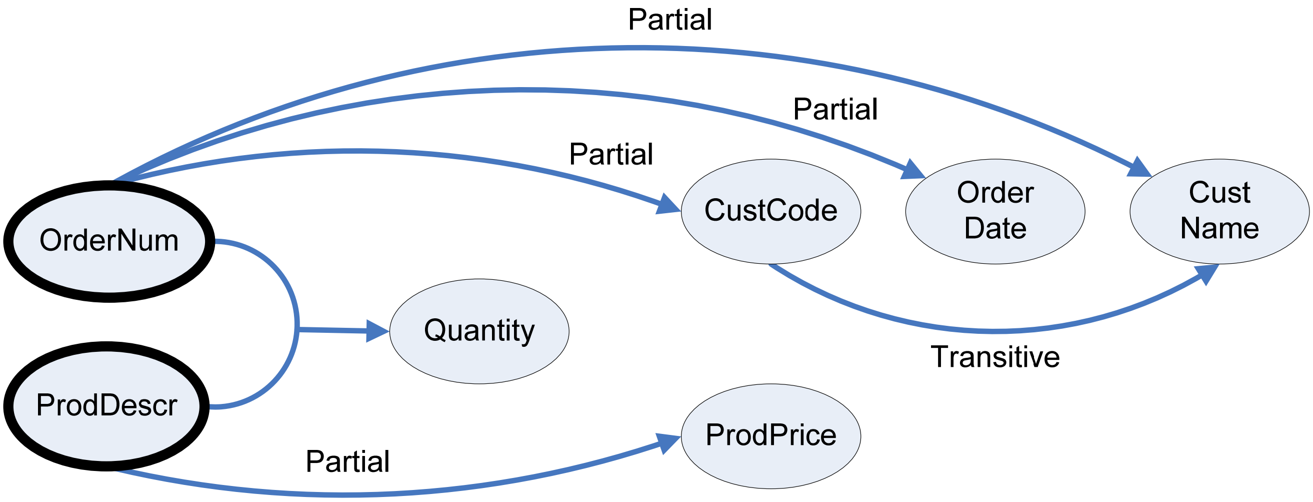
• the ORDERS table :

Order Numb	Cust Code	Order Date	Cust Name	ProdDescr	Prod Price	Quantity
10001	5217	11/22/94	Williams	Hammer	$8.99	2
10001	5217	11/22/94	Williams	Screwdriver	$4.45	1
10002	5021	11/22/94	Johnson	Clipper	$18.22	1
10002	5021	11/22/94	Johnson	Screwdriver	$4.45	3
10002	5021	11/22/94	Johnson	Crowbar	$11.07	1
10002	5021	11/22/94	Johnson	Saw	$14.99	1
10003	4118	11/22/94	Lorenzo	Hammer	$8.99	1
10004	6002	11/22/94	Kopiusko	Saw	$14.99	1
10004	6002	11/22/94	Kopiusko	Screwdriver	$4.45	2
10005	5021	11/23/94	Johnson	Cordlessdrill	$34.95	1

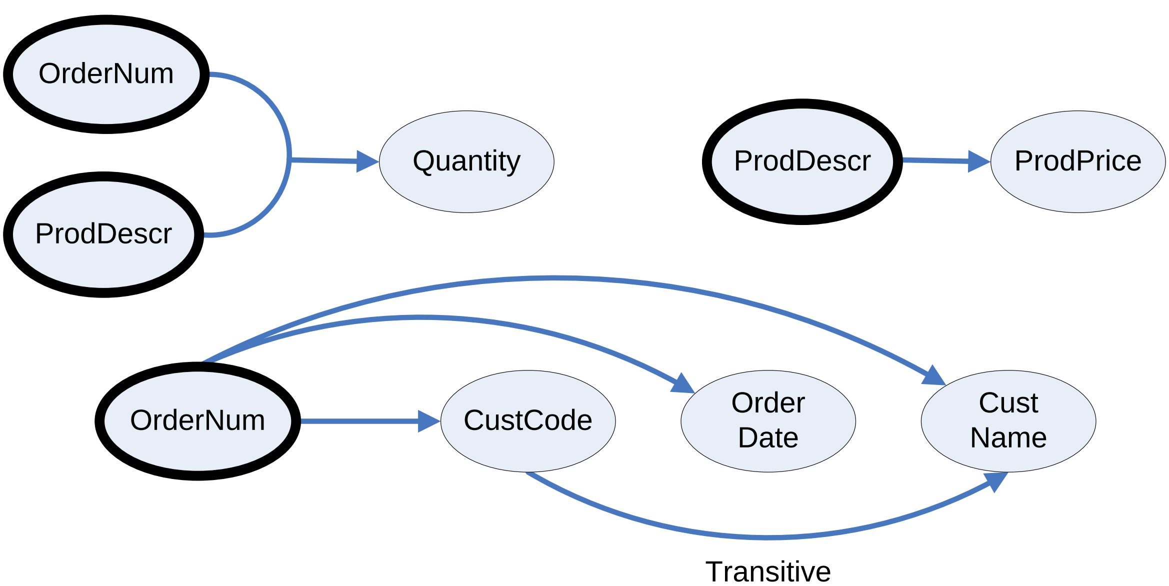
• **Note: in practice, we would also want to have product codes as well as descriptions, and use the product codes as keys to identify products. Here, we'll identify products by their *ProdDescr* to keep the number of fields down.**

Example: Hardware Store Database (Cont.)

ORDERS(<u>OrderNum</u>, <u>ProdDescr</u>, CustCode, OrderDate, CustName, ProdPrice, Quantity)



- **Conversion of the hardware store database to 2NF**
  - **QUANTITY(<u>OrderNum</u>, <u>ProdDescr</u>, Quantity)**
    - **OrderNum foreign key to ORDERS**
    - **ProdDescr foreign key to PRODUCTS**
  - **PRODUCTS(<u>ProdDescr</u>, ProdPrice)**
  - **ORDERS(<u>OrderNum</u>, CustCode, OrderDate, CustName)**



Example: Hardware Store Database (Cont.)

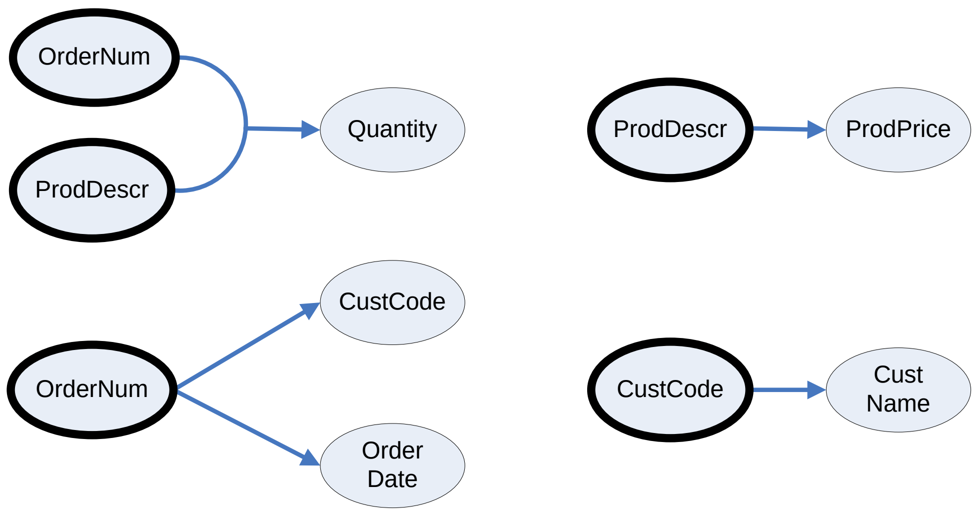
- **conversion of the ORDERS relation to 3NF**

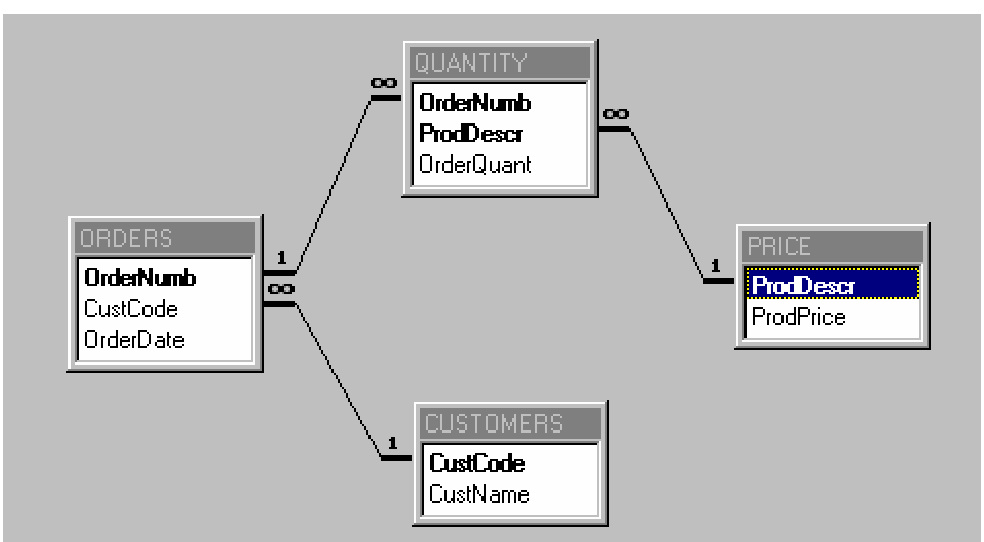
QUANTITY(<u>OrderNum</u>, <u>ProdDescr</u>, Quantity)
OrderNum foreign key to ORDERS
ProdDescr foreign key to PRODUCTS

PRODUCTS(<u>ProdDescr</u>, ProdPrice)

ORDERS(<u>OrderNum</u>, CustCode, OrderDate)
CustCode foreign key to CUSTOMERS

CUSTOMERS(<u>CustCode</u>, CustName)





Example: Video Store Database

• the CUSTOMER relation:

Customer ID	Phone	Last Name	First Name	Address	City	State	Zip Code
1	502-666-7777	Johnson	Martha	125 Main St.	Alvaton	KY	42122
2	502-888-6464	Smith	Jack	873 Elm St.	Bowling Green	KY	42101
3	502-777-7575	Washington	Elroy	95 Easy St.	Smith's Grove	KY	42171
4	502-333-9494	Adams	Samuel	746 Brown Dr.	Alvation	KY	42122
5	502-474-4746	Steinmetz	Susan	15 Speedway Dr.	Portland	TN	37148
….	…..	…..	…..	…..	….	….	….

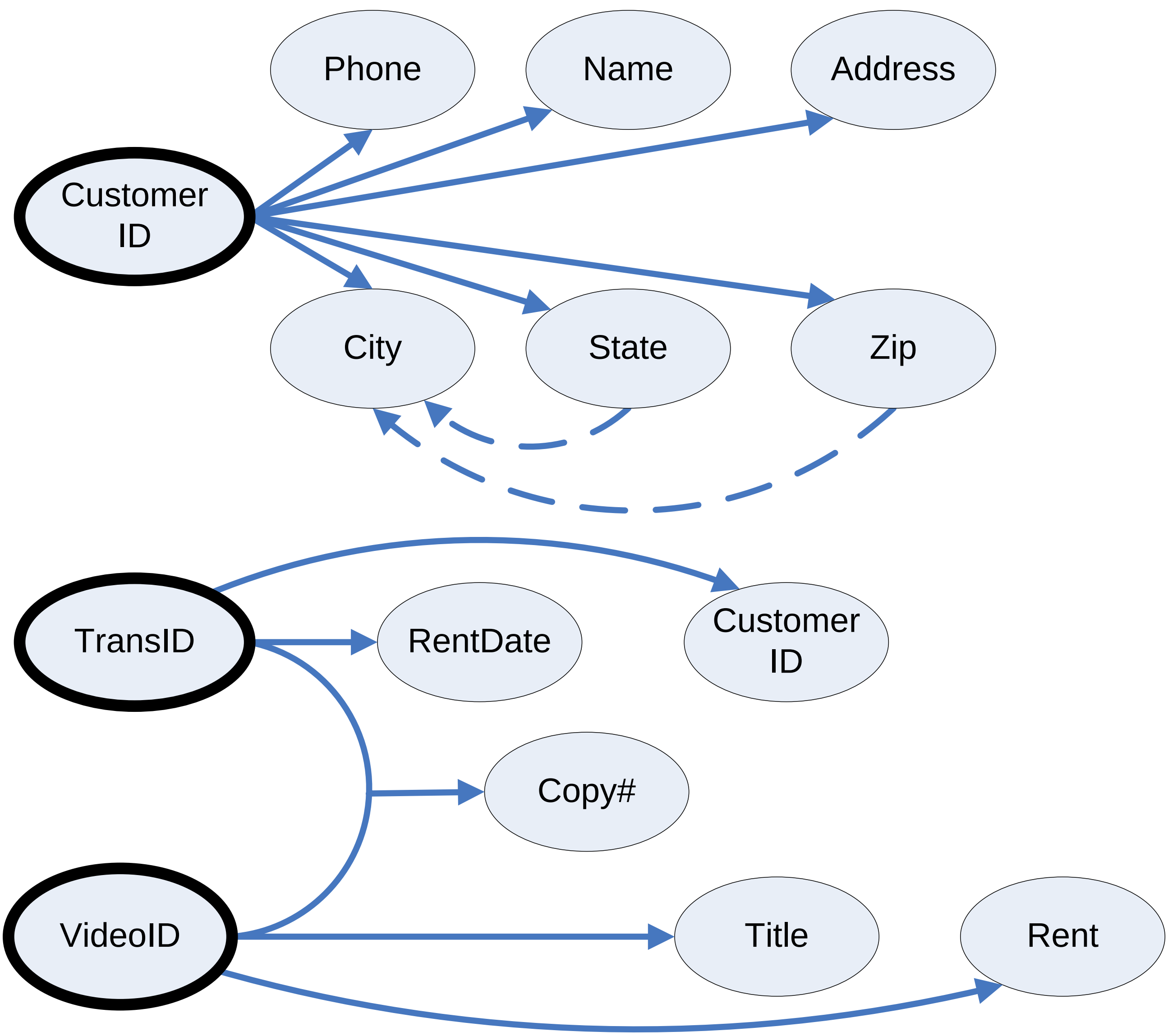
• the RENTALFORM relation:

Trans ID	Rent Date	Customer ID	Video ID	Copy#	Title	Rent
1	4/18/95	3	1	2	2001:SpaceOdyssey	$1.50
1	4/18/95	3	6	3	Clockwork Orange	$1.50
2	4/18/95	7	8	1	Hopscotch	$1.50
2	4/18/95	7	2	1	Apocalypse Now	$2.00
2	4/18/95	7	6	1	Clockwork Orange	$1.50
3	4/18/95	8	9	1	Luggage of the Gods	$2.50
…..	…….	……	……	……	…..	…..

- a customer can rent multiple videos as part of the same transaction
- multiple copies of the same video exist
  - the copy# field stores the number of the copy – unique only with copies of that same video
  - one customer cannot rent two copies of the same video at the same time
- although it has two tables, the database still contains some anomalies

Example: Video Store Database (Cont.)

- relations for the video store database
  - CUSTOMER(<u>CustomerID</u>, Phone, Name, Address, City, State, ZipCode)
  - RENTALFORM(<u>TransID</u>, RentDate, CustomerID, VideoID, <u>Copy#,</u> Title, Rent)
- dependency diagram for the video store database



Example: Video Store Database (Cont.)

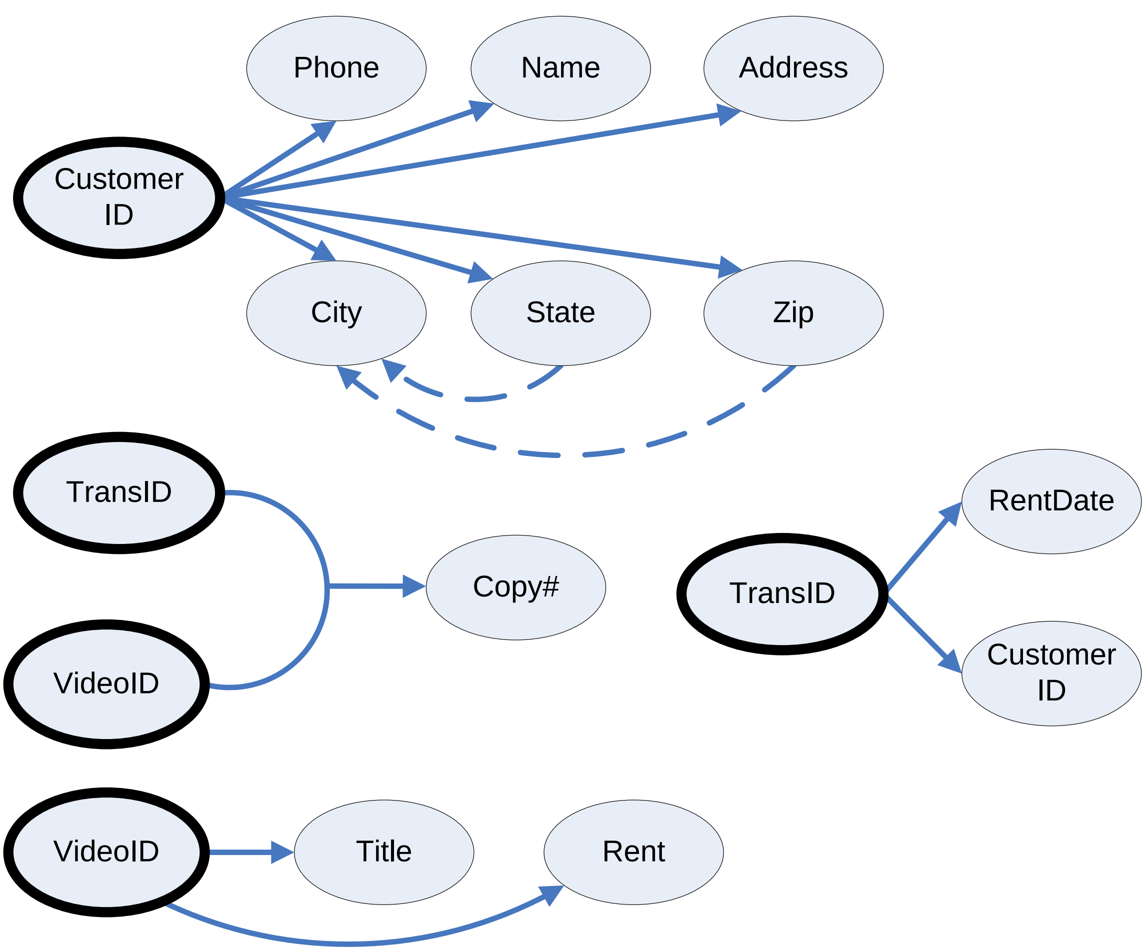
• video store database after eliminating partial and transitive dependencies

CUSTOMER(<u>CustomerID</u>, Phone, Name, Address, City, State, ZipCode)

RENTAL(<u>TransID</u>, RentDate, CustomerID)
CustomerID foreign key to CUSTOMER

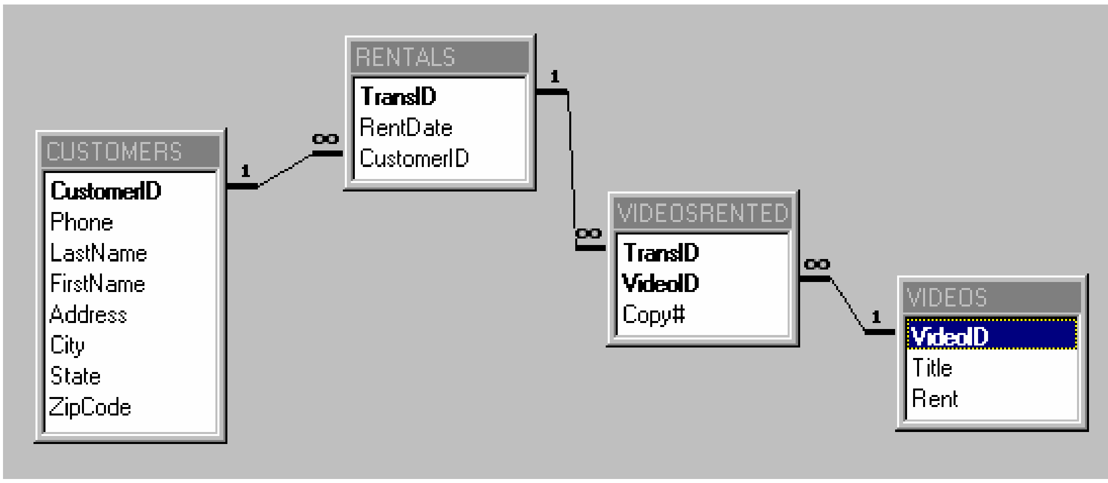
VIDEO(<u>VideoID</u>, Title, Rent)

VIDEOSRENTED(<u>TransID</u>, <u>VideoID</u>, Copy#)
TransID foreign key to RENTAL
VideoID foreign key to VIDEO



Example: Video Store Database (Cont.)

- table relationships for the video store database



Summary of Guidelines for Database Design

- identify the entities involved in the database
- identify the fields relevant for each entity and define the corresponding relations
- determine the primary key of each relation
- avoid data redundancy, but have some common fields so that tables can be joined together
- ensure that all the required database processing can be done using the defined relations
- normalize the relations by splitting them into smaller ones