Database Design

Chapter 3

- Normalization

Chapter Objectives

- Explain the term normalization

- Explain the terms partial key dependency and transitive dependency

- Normalize a database to third normal form (3NF)

Limitations of E-R Designs

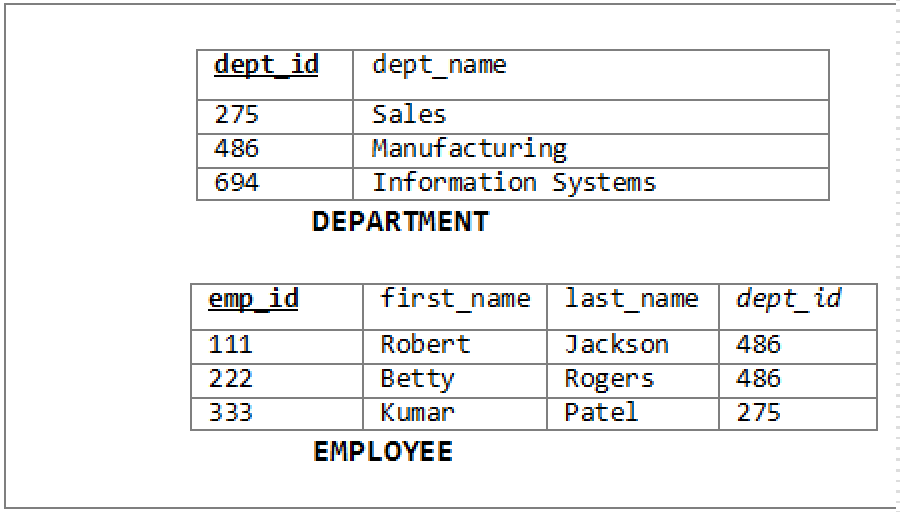
- Provides a set of guidelines, does not result in a unique database schema
- Does not provide a way of evaluating alternative schemas
- Normalization theory provides a mechanism for analyzing and refining the schema produced by an E-R design

Normalization

- Normalization is the process of restructuring the data model into logical database tables:
  - Eliminate redundant data
    - Eliminate the storing of the same data in more than one table
  - Ensure that data within a table are related

Representing Database Tables

- Table Form:
  - Capitalize table name
  - Bold and underline the primary key
  - Italicize foreign keys



dept_id	dept_name
275	Sales
486	Manufacturing
694	Information Systems

DEPARTMENT

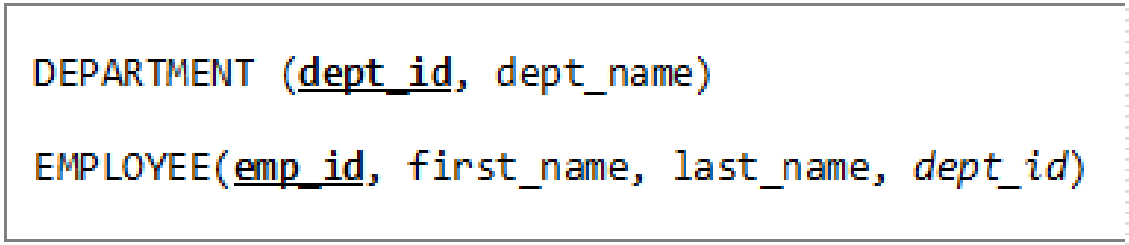
emp_id	first_name	last_name	*dept_id*
111	Robert	Jackson	486
222	Betty	Rogers	486
333	Kumar	Patel	275

EMPLOYEE

Representing tables in table form

Representing Database Tables

- Relational Schema:
  - Capitalize the table name
  - Put attributes in parentheses
  - Bold and underline the primary key
  - Italicize foreign keys



DEPARTMENT (**dept_id**, dept_name)

EMPLOYEE(**emp_id**, first_name, last_name, *dept_id*)

Relational schema

Normal Forms

- 1NF - First Normal Form
- 2NF - Second Normal Form
- 3NF - Third Normal Form
- BCNF – Boyce-Codd Normal Form
- 4NF –Fourth Normal Form
- 5NF – Fifth Normal Form

- Normal Forms are progressive. That is, to have 3NF we must have 2NF and to have 2NF we must have 1NF

Normal Forms: Summary

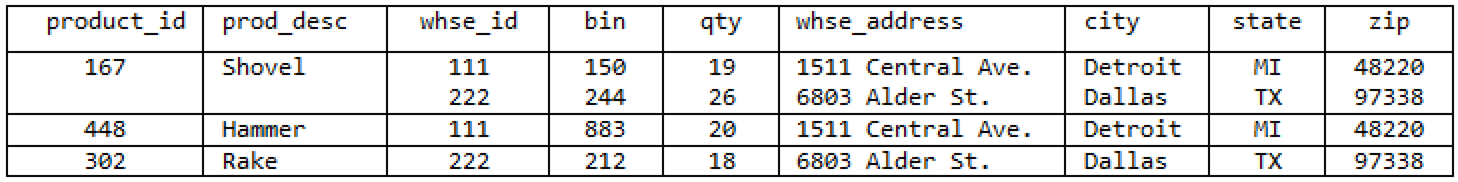
- Un-normalized There are multivalued attributes or repeating groups
- 1NF All columns contain a single value, no repeating groups and primary key assigned
- 2NF 1NF plus no partial dependencies
- 3NF 2NF plus no transitive dependencies

First Normal Form (1NF)

- A database table is in 1NF when:
  - Each attribute (column) contains a single value only
  - There are no repeating groups. That is, two columns do not store similar data
  - All values for a given attribute (column) are the same type
  - Each attribute (column) name is unique
  - The table has a primary key
    - One or more columns that uniquely identifies each row in the table
    - No two rows are identical
  - All columns in the table are dependent on the primary key
  - The order of the rows is insignificant

Un-Normalized Inventory Data

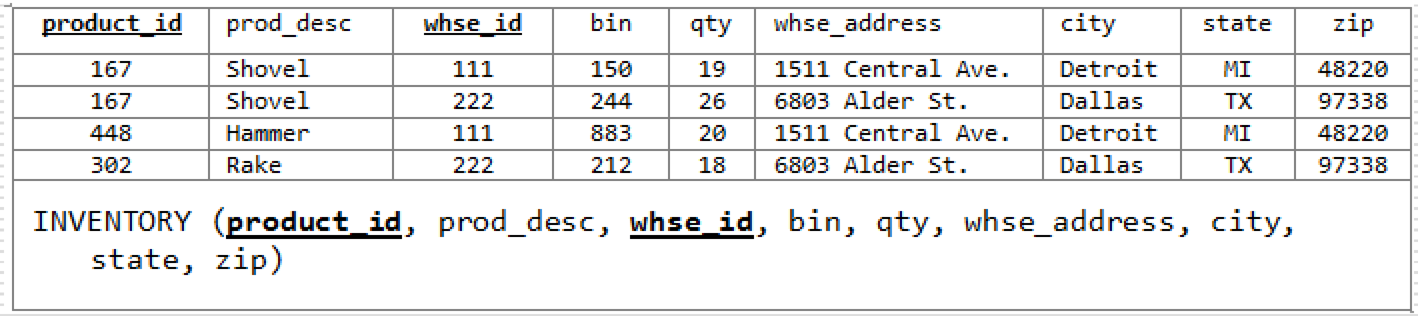
- The table below is in violation of the 1NF rules:
    - All attributes must contain a single value
    - No primary key



product_id	prod_desc	whse_id	bin	qty	whse_address	city	state	zip
167	Shovel	111 222	150 244	19 26	1511 Central Ave. 6803 Alder St.	Detroit Dallas	MI TX	48220 97338
448	Hammer	111	883	20	1511 Central Ave.	Detroit	MI	48220
302	Rake	222	212	18	6803 Alder St.	Dallas	TX	97338

First Normal Form (1NF)

- Each attribute contains a single value
- The primary key is a composite key consisting of **product_id** and **whse_id**



product_id	prod_desc	whse_id	bin	qty	whse_address	city	state	zip
167	Shovel	111	150	19	1511 Central Ave.	Detroit	MI	48220
167	Shovel	222	244	26	6803 Alder St.	Dallas	TX	97338
448	Hammer	111	883	20	1511 Central Ave.	Detroit	MI	48220
302	Rake	222	212	18	6803 Alder St.	Dallas	TX	97338

INVENTORY (**product_id**, prod_desc, **whse_id**, bin, qty, whse_address, city, state, zip)

1NF Example: There are no repeating groups
Two columns do not store similar data

Un-normalized STUDENT table:

student_id	adv_id	adv_name	adv_room	course1	course2
123	123A	James	555	102-8	104-9
124	123B	Smith	467	209-0	102-8

STUDENT table in 1NF:

student_id	adv_id	adv_name	adv_room	course_id
123	123A	James	555	102-8
123	123A	James	555	104-9
124	123B	Smith	467	209-0
124	123B	Smith	467	102-8

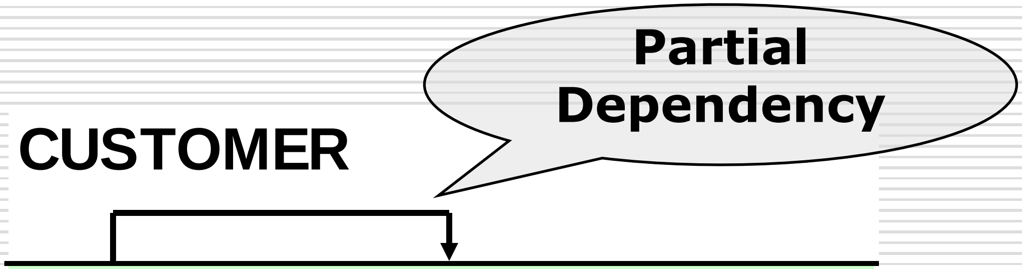
Second Normal Form (2NF)

- A database table is in 2NF when:
  - It is in 1NF
  - No partial key dependencies
    - Each non-key attribute (column) depends on the entire primary key

- Tables that have a single attribute (column) for a primary key are automatically in 2NF
  - This is one reason why artificial identifiers are used as primary keys
- 2NF is applies to tables that contain composite keys

Partial Key Dependency

- When an non-key attribute is determined by part, but not the whole, of a **COMPOSITE** UID (primary key)



CUSTOMER

Partial Dependency

Cust_ID	Name	Order_ID
101	AT&T	1234
101	AT&T	156
125	Cisco	1250

Consider an INVENTORY Example

INVENTORY			
product_id	supplier_id	cost	supplier_address

Partial Dependencies

INVENTORY			
product_id	supplier_id	cost	supplier_address

- There are two non-key attributes. Questions:
  - If I know just product_id, can I find out cost? **NO**, because we have more than one supplier for the same product
  - If I know just supplier_id, can I find out cost? **NO**, because I need to know what the product_id is as well
  - Therefore, cost is fully functionally dependent upon the ENTIRE primary key (UID) (product_id + supplier_id) for its existence

Continued…

INVENTORY			
product_id	supplier_id	cost	supplier_address

- If I know just product_id, can I find out supplier_address? **NO**, because there are more than one supplier for the same product
- If I know just supplier_id, can I find out supplier_address? **YES**, the address does not depend upon the product_id
- Therefore, supplier_address is NOT fully functionally dependent upon the ENTIRE primary key (UID) (product_id + supplier_id) for its existence
- To complete 2NF, a new entity is required

Converting Partial Dependencies

INVENTORY			
product_id	supplier_id	cost	supplier_address

To convert a table with partial dependencies to 2NF:

Step 1: Create a new table for each primary key attribute (or combination of attributes) that is a determinant or determining factor in a partial dependency. That attribute becomes the primary key in the new table

In this example, supplier_id is the primary key attribute that determines the partial dependency of supplier_address

SUPPLIER	
supplier_id	

Converting Partial Dependencies

INVENTORY			
product_id	supplier_id	cost	supplier_address

SUPPLIER	
supplier_id	

Step 2: Move the non-key attributes that are dependent on the primary key attribute from the original table to the new table

SUPPLIER	
supplier_id	supplier_address

Inventory Solution for 2NF

INVENTORY			
product_id	supplier_id	cost	supplier_address

- The entities below are in 2NF because there are no partial key dependencies
- The primary key in the SUPPLIER table becomes a foreign key in the INVENTORY table

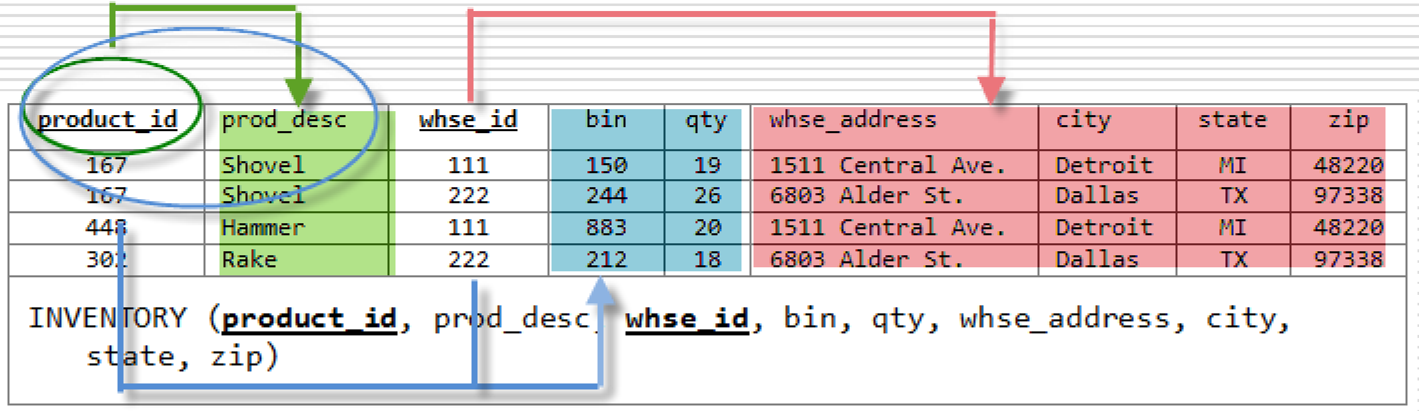
INVENTORY		
product_id	*supplier_id*	cost

SUPPLIER	
supplier_id	supplier_address

Partial Key Dependencies

product_id → prod_desc (green)

whse_id → whse_address, city, state, zip (pink)

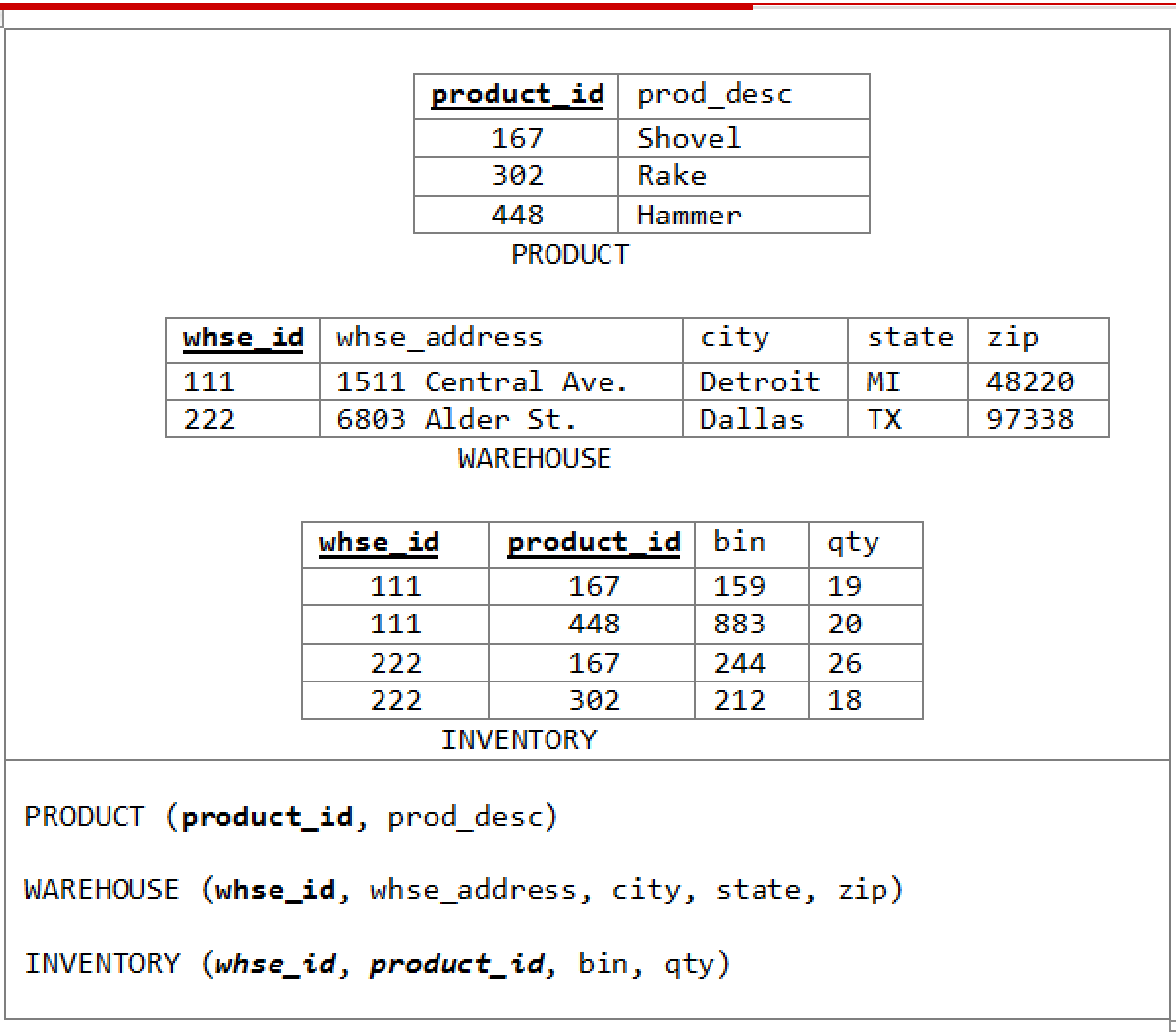


product_id	prod_desc	whse_id	bin	qty	whse_address	city	state	zip
167	Shovel	111	150	19	1511 Central Ave.	Detroit	MI	48220
167	Shovel	222	244	26	6803 Alder St.	Dallas	TX	97338
448	Hammer	111	883	20	1511 Central Ave.	Detroit	MI	48220
302	Rake	222	212	18	6803 Alder St.	Dallas	TX	97338

```
INVENTORY (product_id, prod_desc, whse_id, bin, qty, whse_address, city,
    state, zip)
```

product_id, whse_id → bin, qty (blue)

Second Normal Form



product_id	prod_desc
167	Shovel
302	Rake
448	Hammer

PRODUCT

whse_id	whse_address	city	state	zip
111	1511 Central Ave.	Detroit	MI	48220
222	6803 Alder St.	Dallas	TX	97338

WAREHOUSE

whse_id	**product_id**	bin	qty
111	167	159	19
111	448	883	20
222	167	244	26
222	302	212	18

INVENTORY

PRODUCT (**product_id**, prod_desc)

WAREHOUSE (**whse_id**, whse_address, city, state, zip)

INVENTORY (***whse_id***, ***product_id***, bin, qty)

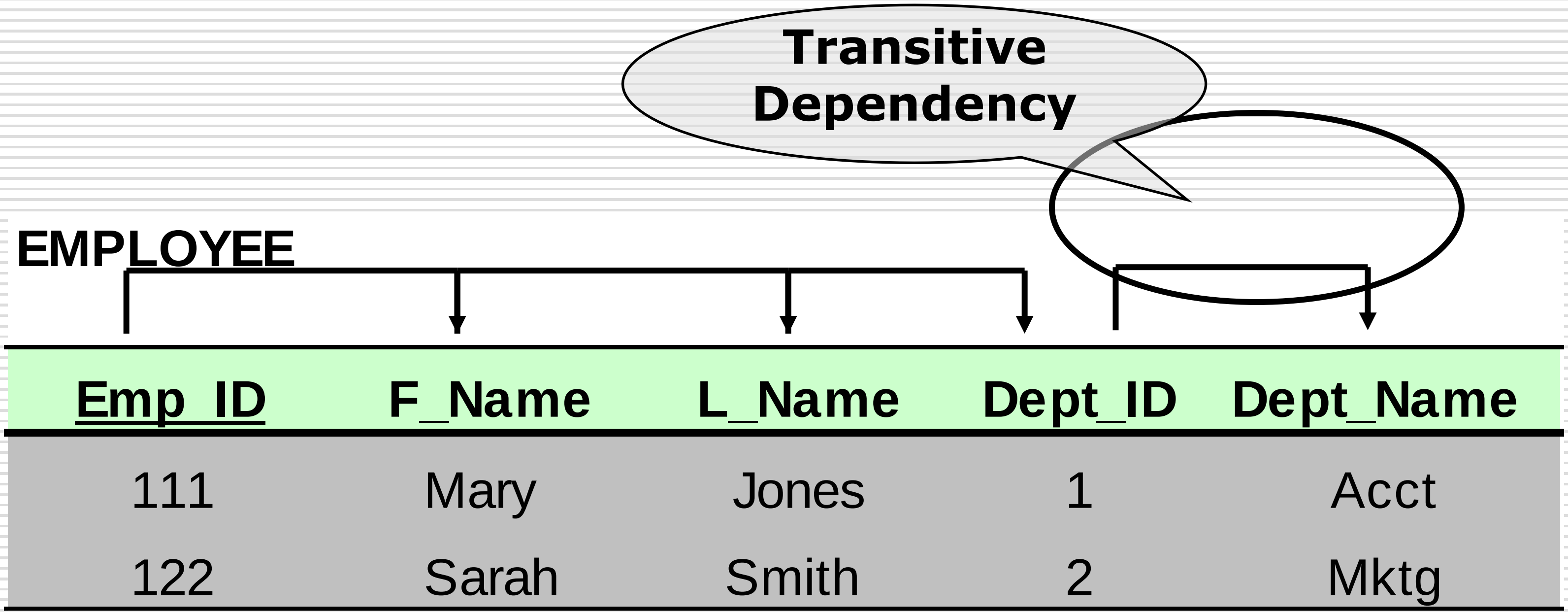
PRODUCT, WAREHOUSE, and INVENTORY tables represented in 2NF

Third Normal Form (3NF)

- A database table is in 3NF when:
  - It is in 2NF
  - It contains no transitive dependencies
    - Each nonkey column depends only on the primary key
    - Or, a nonkey attribute cannot be dependent on another nonkey attribute

Transitive Dependency

- When a non-key attribute determines another non-key attribute.

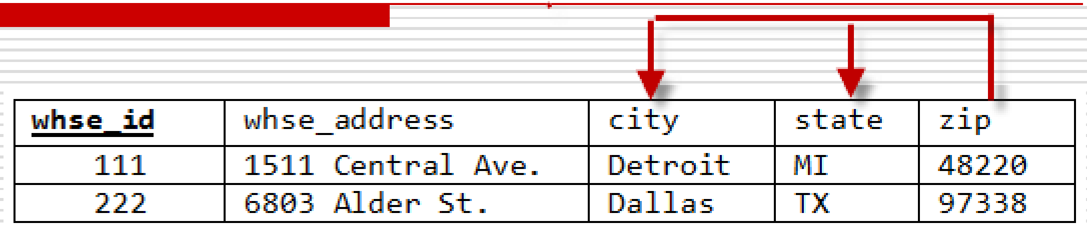


EMPLOYEE

<u>Emp_ID</u>	F_Name	L_Name	Dept_ID	Dept_Name
111	Mary	Jones	1	Acct
122	Sarah	Smith	2	Mktg

Remove Transitive Dependencies

zip → city, state



whse_id	whse_address	city	state	zip
111	1511 Central Ave.	Detroit	MI	48220
222	6803 Alder St.	Dallas	TX	97338

-11 WAREHOUSE table in second normal form

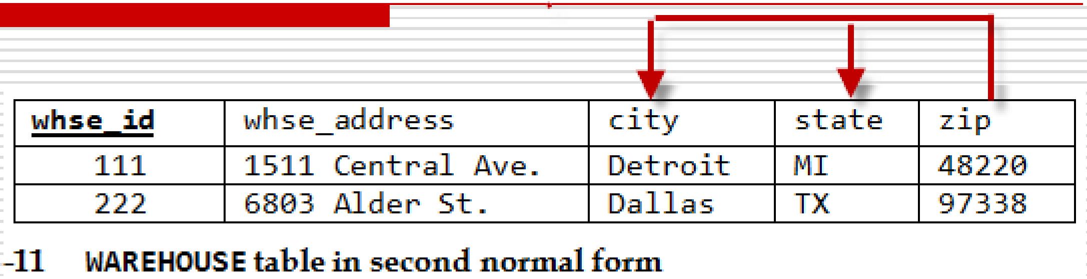
Step 1: Create a new table for each nonkey attribute that is a determinant or deciding factor of transitive dependents (other nonkey attributes). The determinant attribute becomes the primary key of the new table

In this example, zip is the nonkey attribute that determines the city and state attributes and becomes the primary key of the new ZIP table

ZIP		
zip		

Remove Transitive Dependencies

zip → city, state



whse_id	whse_address	city	state	zip
111	1511 Central Ave.	Detroit	MI	48220
222	6803 Alder St.	Dallas	TX	97338

-11 WAREHOUSE table in second normal form

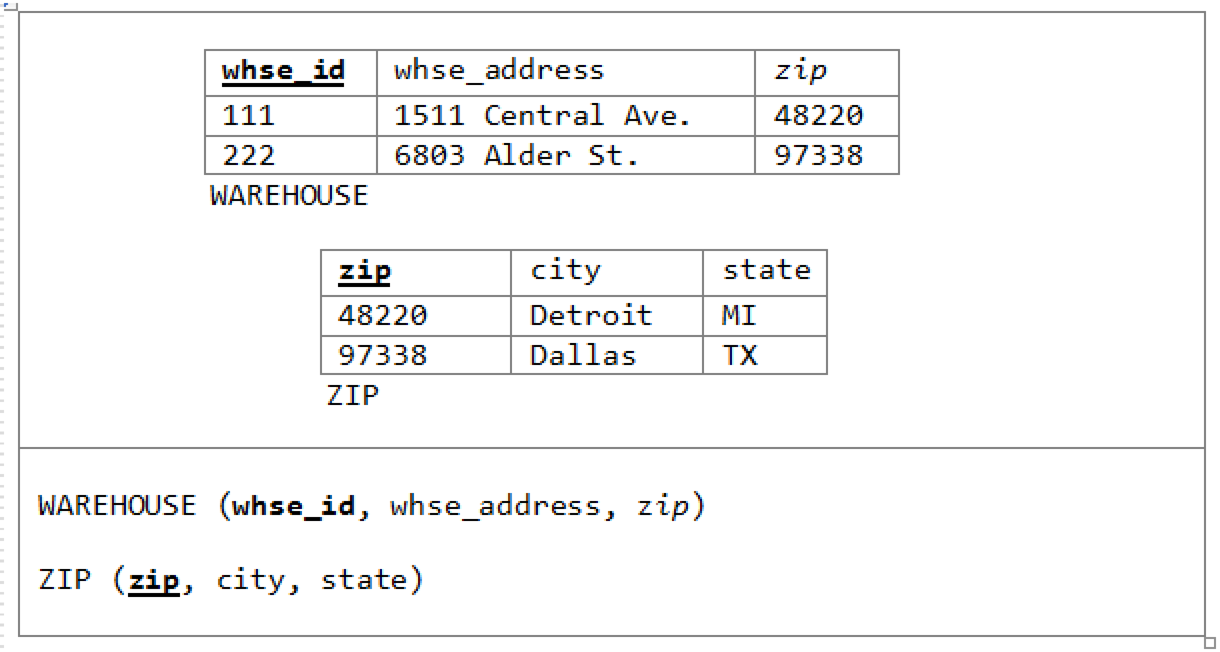
Step 2: Move the attributes from the original table that are functionally dependent on the nonkey attribute

In this example, city and state are moved to the new ZIP table because they are dependent on the zip attribute

ZIP		
zip	city	state

Remove Transitive Dependencies

Step 3: Leave the new table's primary key in the original table as a *foreign key*. This provides the relationship between the original table and the new table



whse_id	whse_address	*zip*
111	1511 Central Ave.	48220
222	6803 Alder St.	97338

WAREHOUSE

zip	city	state
48220	Detroit	MI
97338	Dallas	TX

ZIP

WAREHOUSE (**whse_id**, whse_address, *zip*)

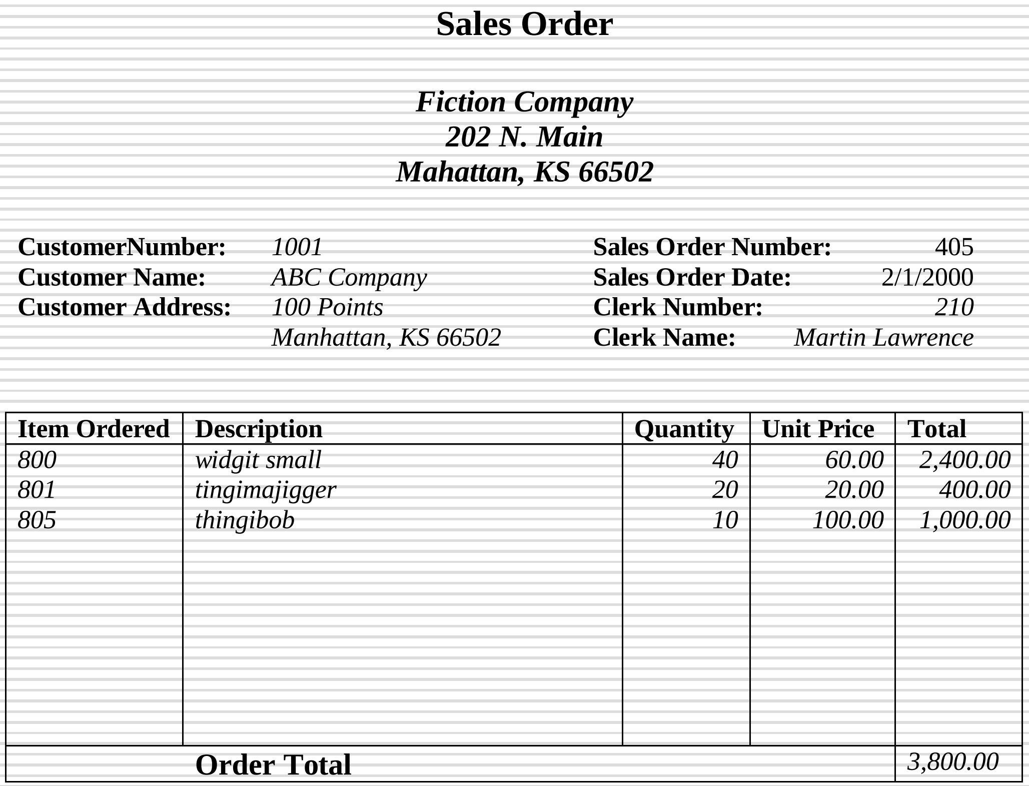
ZIP (**zip**, city, state)

WAREHOUSE and ZIP tables in third normal form

The End

The End

Example



Sales Order

Fiction Company
202 N. Main
Mahattan, KS 66502

CustomerNumber:	*1001*	**Sales Order Number:**	405
Customer Name:	*ABC Company*	**Sales Order Date:**	2/1/2000
Customer Address:	*100 Points*	**Clerk Number:**	*210*
	Manhattan, KS 66502	**Clerk Name:**	*Martin Lawrence*

Item Ordered	Description	Quantity	Unit Price	Total
800	*widgit small*	*40*	*60.00*	*2,400.00*
801	*tingimajigger*	*20*	*20.00*	*400.00*
805	*thingibob*	*10*	*100.00*	*1,000.00*
	Order Total			*3,800.00*

Un-Normalized

```
customer_number, customer_name, customer_address,
customer_city, customer_zip, clerk_number, clerk_name,
order_number, item_no, item_description,
quantity_ordered, unit_price
```

The End

The End

The End

customer(customer_number, date, customer_name, customer_address, clerk_no, clerk_name)

product(**item_no**, item_description, item_price)

order(**order_no**, date, customer_no, *clerk_no*)

~~order_detail(***order_no***, ***item_no***, quanity_ordered, unit_price~~

2NF

customer(**customer_number**, customer_name, customer_address

clerk(**clerk_no**, clerk_name)

~~product(**item_no**, item_description, item_price)~~

order(**order_no**, date, customer_no, *clerk_no*)

order_detail(***order_no***, ***item_no***, quanity_ordered, unit_price

3NF

customer(**customer_number**, customer_name, customer_address

clerk(**clerk_no**, clerk_name)

product(**item_no**, item_description, item_price)

order(**order_no**, date, customer_no, *clerk_no*)

order_detail(***order_no***, ***item_no***, quanity_ordered, unit_price

First Normal Form (1NF)

A table repre

repeating gro

Repeating group = (property_id, address, rent_start, rent_finish, rent, owner_id, o_name)

client_id	c_name	property_id	address	rent_start	rent_finish	rent	owner_id	o_name
CR76	John Kay	PG4	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

First Normal Form (1NF)

There are two approaches to removing repeating groups from unnormalized tables:

1. Removes the repeating groups by entering appropriate data in the empty columns of rows containing the repeating data.
2. Removes the repeating group by placing the repeating data, along with a copy of the original key attribute(s), in a separate table. A primary key is identified for the new table.

1NF with the first approach

With the first approach, we remove the repeating group (property rented details) by entering the appropriate client data into each row.

1NF with the first approach

A table representing client rental data in 1NF

CLIENT_RENTAL (client_id, property_id, c_name, address, rent_start, rent_finish, rent, owner_id, o_name)

client_id	property_id	c_name	address	rent_start	rent_finish	rent	owner_id	o_name
CR76	PG4	John Kay	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

1NF with the second approach

With the second approach, we remove the repeating group (property rented details) by placing the repeating data along with a copy of the original key attribute (client_id) in a separte table.

1NF with the second approach

CLIENT (client_id, c_name)

RENTAL (client_id, property_id, name, rent_start, rent_finish, rent, owner_id, o_name)

client_id	c_name
CR76	John Kay
CR56	Aline Stewart

client_id	property_id	address	rent_start	rent_finish	rent	owner_id	o_name
CR76	PG4	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
CR76	PG16	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	PG4	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
CR56	PG36	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
CR56	PG16	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

2NF Partial Dependencies

Partial dependencies:

client_id → c_name

property_id → address, rent, owner_id, o_name

<u>client_id</u>	<u>property_id</u>	c_name	address	rent_start	rent_finish	rent	owner_id	o_name
CR76	PG4	John Kay	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

2NF ClientRental

CLIENT (client_id, c_name)

PROPERTY (property_id, address, rent, owner_id, o_name)

RENTAL (client_id, property_id, rent_start, rent_finish)

CLIENT

client_id	c_name
CR76	John Kay
CR56	Aline Stewart

RENTAL

client_id	property_id	rent_start	rent_finish
CR76	PG4	1-Jul-00	31-Aug-01
CR76	PG16	1-Sep-02	1-Sep-02
CR56	PG4	1-Sep-99	10-Jun-00
CR56	PG36	10-Oct-00	1-Dec-01
CR56	PG16	1-Nov-02	1-Aug-03

PROPERTY

property_id	address	rent	owner_id	o_name
PG4	6 lawrence St,Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	370	CO93	Tony Shaw

3NF ClientRental

The transitive dependencies are :

owner_id → o_name

property_id	address	rent	owner_id	o_name
PG4	6 lawrence St,Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	370	CO93	Tony Shaw

3NF ClientRental

The resulting 3NF tables have the forms:

CLIENT	(client_id, c_name)
RENTAL	(client_id, property_id, rent_start, rent_finish)
PROPERTY	(property_id, address, rent, owner_id)
OWNER	(owner_id, o_name)

3NF ClientRental

CLIENT

Client_id	c_name
CR76	John Kay
CR56	Aline Stewart

RENTAL

client_id	property_id	rent_start	rent_finish
CR76	PG4	1-Jul-00	31-Aug-01
CR76	PG16	1-Sep-02	1-Sep-02
CR56	PG4	1-Sep-99	10-Jun-00
CR56	PG36	10-Oct-00	1-Dec-01
CR56	PG16	1-Nov-02	1-Aug-03

PROPERTY

property_id	address	rent	owner_id
PG4	6 lawrence St,Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	370	CO93

OWNER

owner_id	o_name
CO40	Tina Murphy
CO93	Tony Shaw

Lab Assignment

2NF ClientRental

The ClientRental table has the following functional dependencies:

fd1	client_id, property_id → rent_start, rent_finish	(Primary Key)
fd2	client_id → name	(Partial dependency)
fd3	property_id → address, rent, owner_id, name	(Partial dependency)
fd4	owner_id → name	(Transitive Dependency)
fd5	client_id, rent_start → property_id, address, rent_finish, rent, owner_id, name	(Candidate key)
fd6	property_id, rent_start → client_id, name, rent_finish	(Candidate key)

3NF ClientRental

The functional dependencies for the Client, Rental and PropertyOwner tables are as follows:

Client

fd2	client_id → name	(Primary Key)

Rental

fd1	client_id, property_id → rent_start, rent_finish	(Primary Key)
fd5	client_id, rent_start → property_id, rent_finish	(Candidate key)
fd6	property_id, rent_start → client_id, rent_finish	(Candidate key)

PropertyOwner

fd3	property_id → address, rent, owner_id, name	(Primary Key)
fd4	owner_id → name	(Transitive Dependency)