Web Programming

YJ – Aug 2015

Course Outline

There’re 16 lectures (2.5 hrs each) in this course. We’ll mainly focus on web programming using LAMP model.

During the 24 hours tutorials in each week, our tutors will help you go through all the contents in the lecture notes and help you with the assignments

All scheduled lectures are learner-paced which means they could be altered depends on the situation. We help you gain rather than cram!

Course Outline

- L1 ~ Web programming intro
- L2 ~ HTML5
- L3 ~ CSS3
- L4 ~ JavaScripts
- L5 ~ PHP 1 Basic
- L6 ~ PHP 2 OO && Functions
- L7 ~ PHP 3 OO && Sessions
- L8 ~ PHP 4 Advanced features
- L9 ~ Mysql
- L10 ~ Mysql 2
- L11 ~ XML/JSON
- L12 ~ Ajax
- L13 ~ Framework: Bootstrap && Laravel
- L14 ~ CMS: Wordpress, Joomla, etc.
- L15 ~ Final Projects
- L16 ~ Final Projects

Join

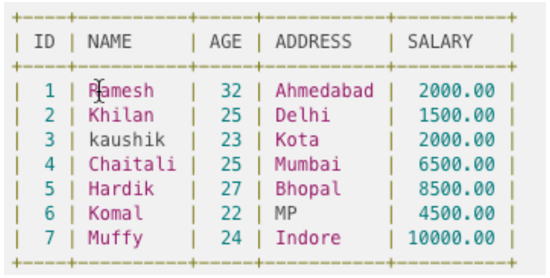
The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

The Joining Process

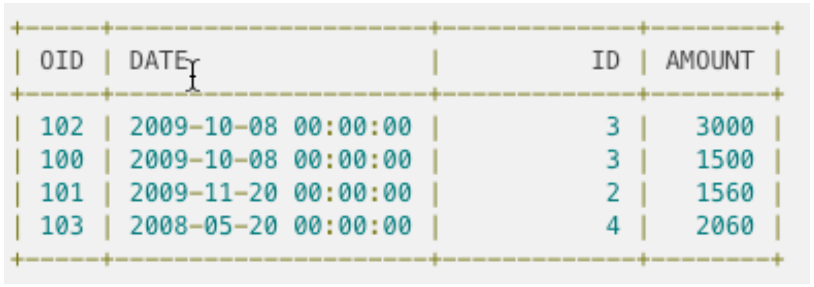
1. Combine every tuple in the first relation with every tuple in all other relations in the FROM clause.
2. Apply the joining condition from the WHERE clause.
3. Project onto the list of attributes and expressions in the SELECT clause.

- ❖ INNER JOIN: Returns all rows when there is at least one match in BOTH tables
- ❖ LEFT JOIN: Return all rows from the left table, and the matched rows from the right table
- ❖ RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table
- ❖ FULL JOIN: Return all rows when there is a match in ONE of the tables

Inner Join

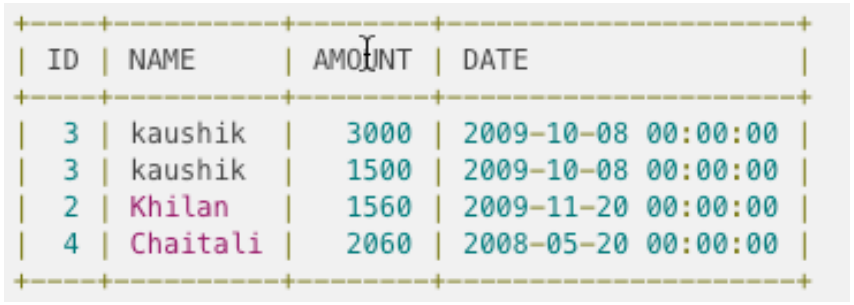


```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

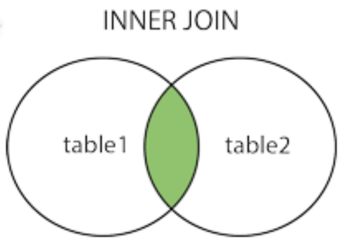


```
+-----+---------------------+-------------+--------+
| OID | DATE                |          ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+-------------+--------+
```

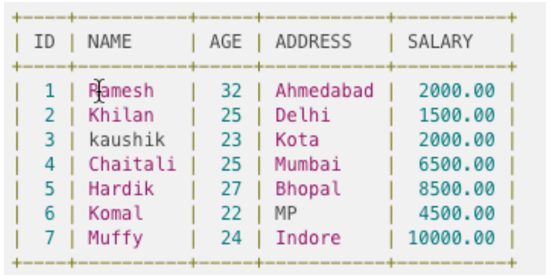
```
SELECT  ID, NAME, AMOUNT, DATE
    FROM CUSTOMERS
    INNER JOIN ORDERS
    ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```



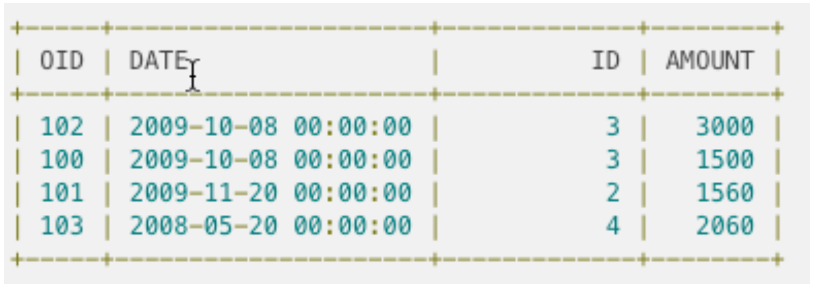
```
+----+----------+--------+---------------------+
| ID | NAME     | AMOUNT | DATE                |
+----+----------+--------+---------------------+
|  3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|  3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|  4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+----+----------+--------+---------------------+
```



Left Join

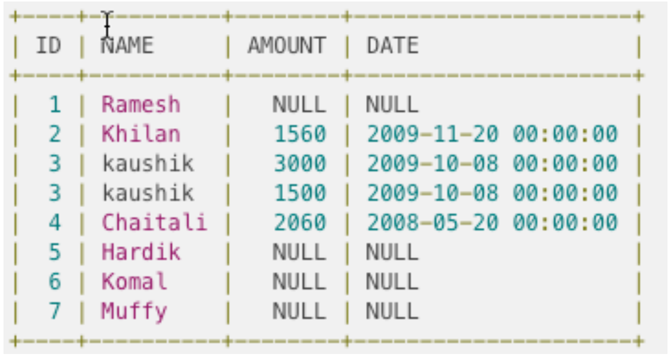


```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

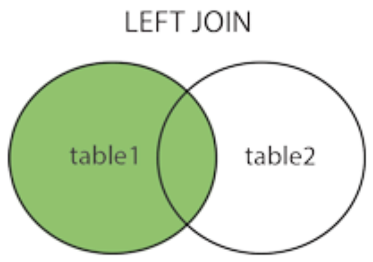


```
+-----+---------------------+------+--------+
| OID | DATE                |   ID | AMOUNT |
+-----+---------------------+------+--------+
| 102 | 2009-10-08 00:00:00 |    3 |   3000 |
| 100 | 2009-10-08 00:00:00 |    3 |   1500 |
| 101 | 2009-11-20 00:00:00 |    2 |   1560 |
| 103 | 2008-05-20 00:00:00 |    4 |   2060 |
+-----+---------------------+------+--------+
```

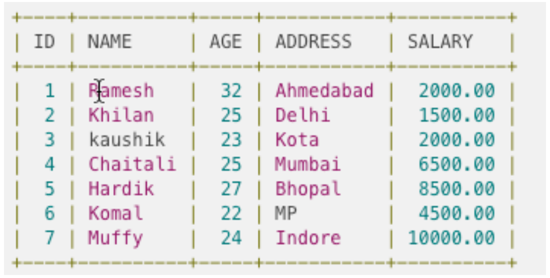
```
SELECT  ID, NAME, AMOUNT, DATE
    FROM CUSTOMERS
    LEFT JOIN ORDERS
    ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```



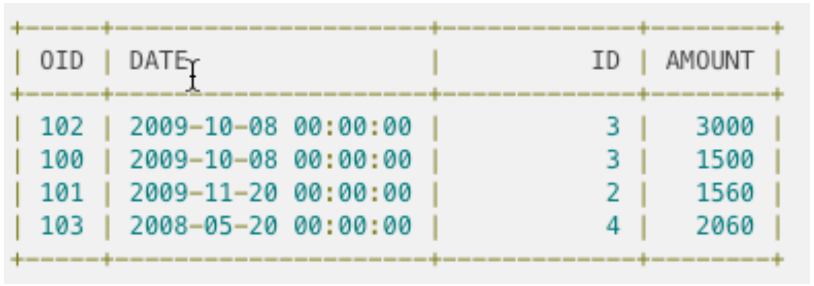
```
+----+----------+--------+---------------------+
| ID | NAME     | AMOUNT | DATE                |
+----+----------+--------+---------------------+
|  1 | Ramesh   |   NULL | NULL                |
|  2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|  3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|  3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|  5 | Hardik   |   NULL | NULL                |
|  6 | Komal    |   NULL | NULL                |
|  7 | Muffy    |   NULL | NULL                |
+----+----------+--------+---------------------+
```



Right Join

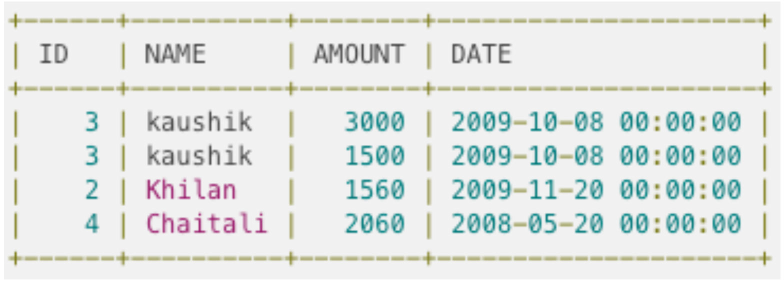


```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

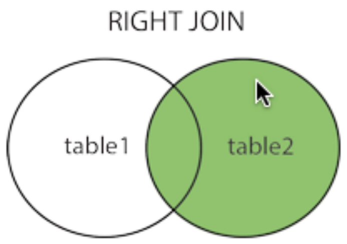


```
+-----+---------------------+-------------+--------+
| OID | DATE                |          ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+-------------+--------+
```

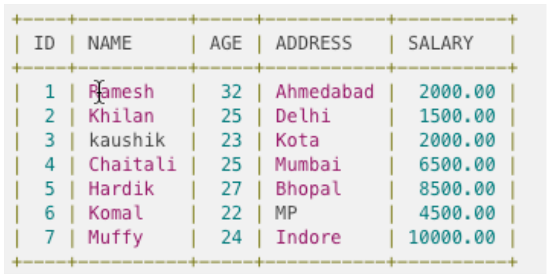
```
SELECT  ID, NAME, AMOUNT, DATE
    FROM CUSTOMERS
    RIGHT JOIN ORDERS
    ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```



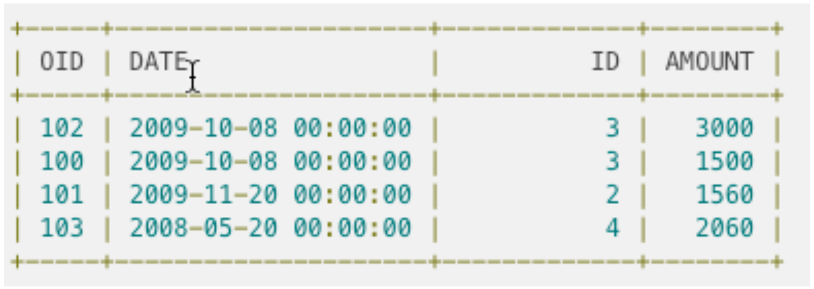
```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```



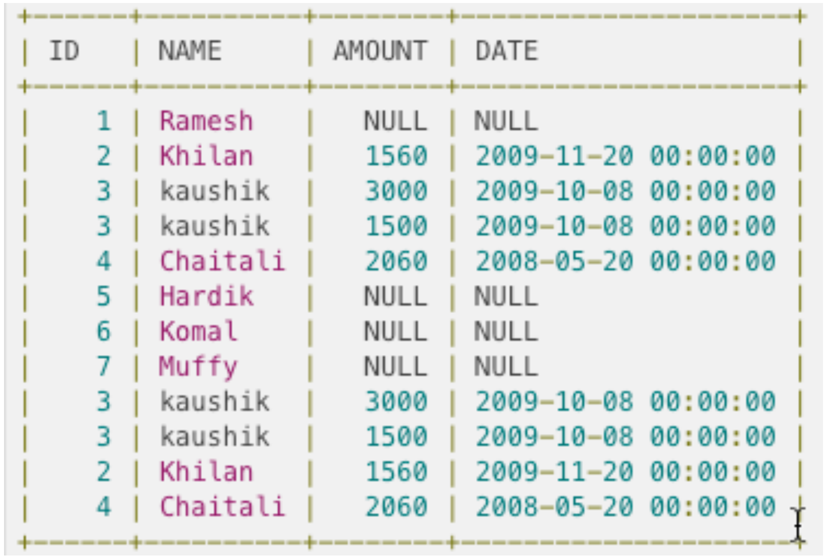
Full Join / Union All



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```



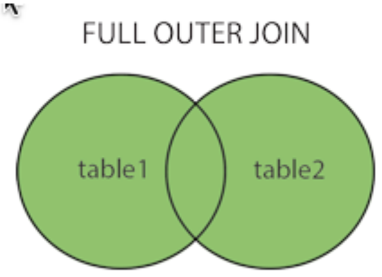
```
+-----+---------------------+-------------+--------+
| OID | DATE                |          ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+-------------+--------+
```



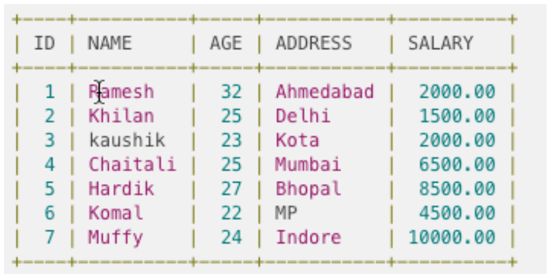
```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
|    1 | Ramesh   |   NULL | NULL                |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|    5 | Hardik   |   NULL | NULL                |
|    6 | Komal    |   NULL | NULL                |
|    7 | Muffy    |   NULL | NULL                |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```

```
SELECT  ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
LEFT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
  UNION ALL
SELECT  ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
RIGHT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

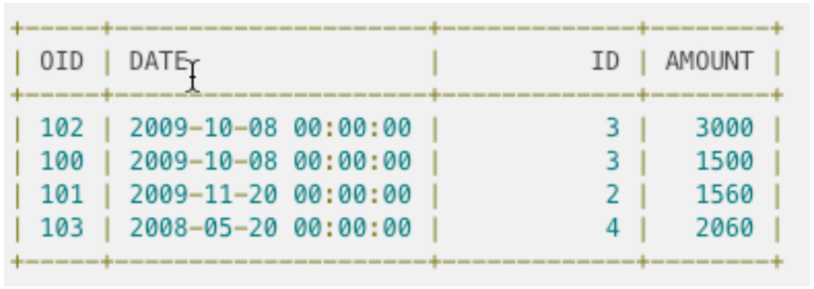
```
SELECT  ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
FULL JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```



Union



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

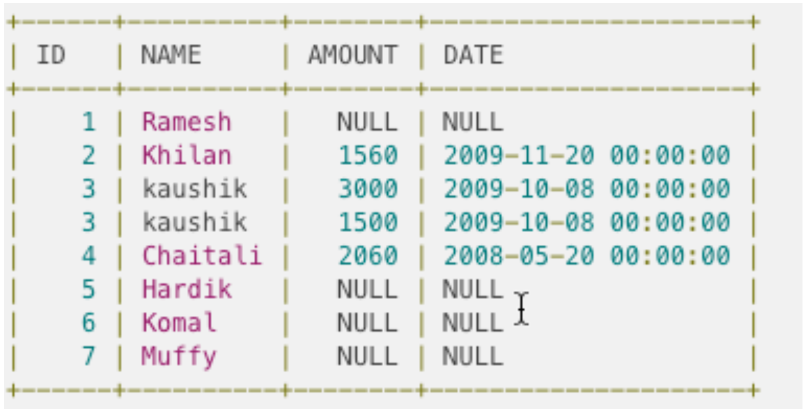


```
+-----+---------------------+-------------+--------+
| OID | DATE                |          ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+-------------+--------+
```

SELECT ID, NAME, AMOUNT, ***DATE***
FROM CUSTOMERS
LEFT JOIN ORDERS
ON CUSTOMERS.**ID** = ORDERS.CUSTOMER_ID

UNION

SELECT ID, NAME, AMOUNT, ***DATE***
FROM CUSTOMERS
RIGHT JOIN ORDERS
ON CUSTOMERS.**ID** = ORDERS.CUSTOMER_ID;

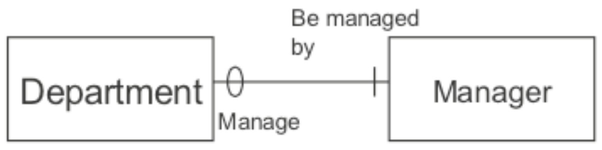


```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
|    1 | Ramesh   |   NULL | NULL                |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|    5 | Hardik   |   NULL | NULL                |
|    6 | Komal    |   NULL | NULL                |
|    7 | Muffy    |   NULL | NULL                |
+------+----------+--------+---------------------+
```

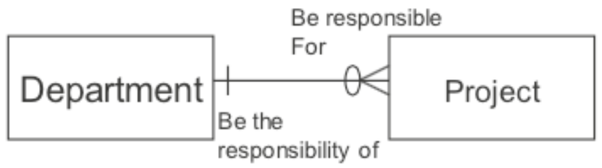
Entity Relationship

- ❖ Database is more than data, it has the relationship between data tables (Entities).
- ❖ An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It is a graphical approach to representing the structure of information. Sometimes known as a data structure diagram (DSD, ERD etc.)
- ❖ Actually quite simple, with two basic elements:
  - – Entities
  - – Relationships
- ❖ Models the things in the real world that the information system needs to represent, and the specific items of information about those things.

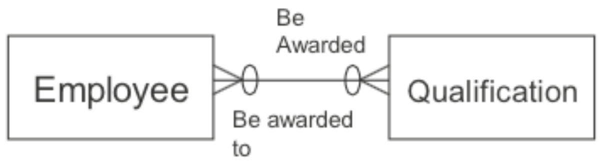
Entity Relationship



One-To-One

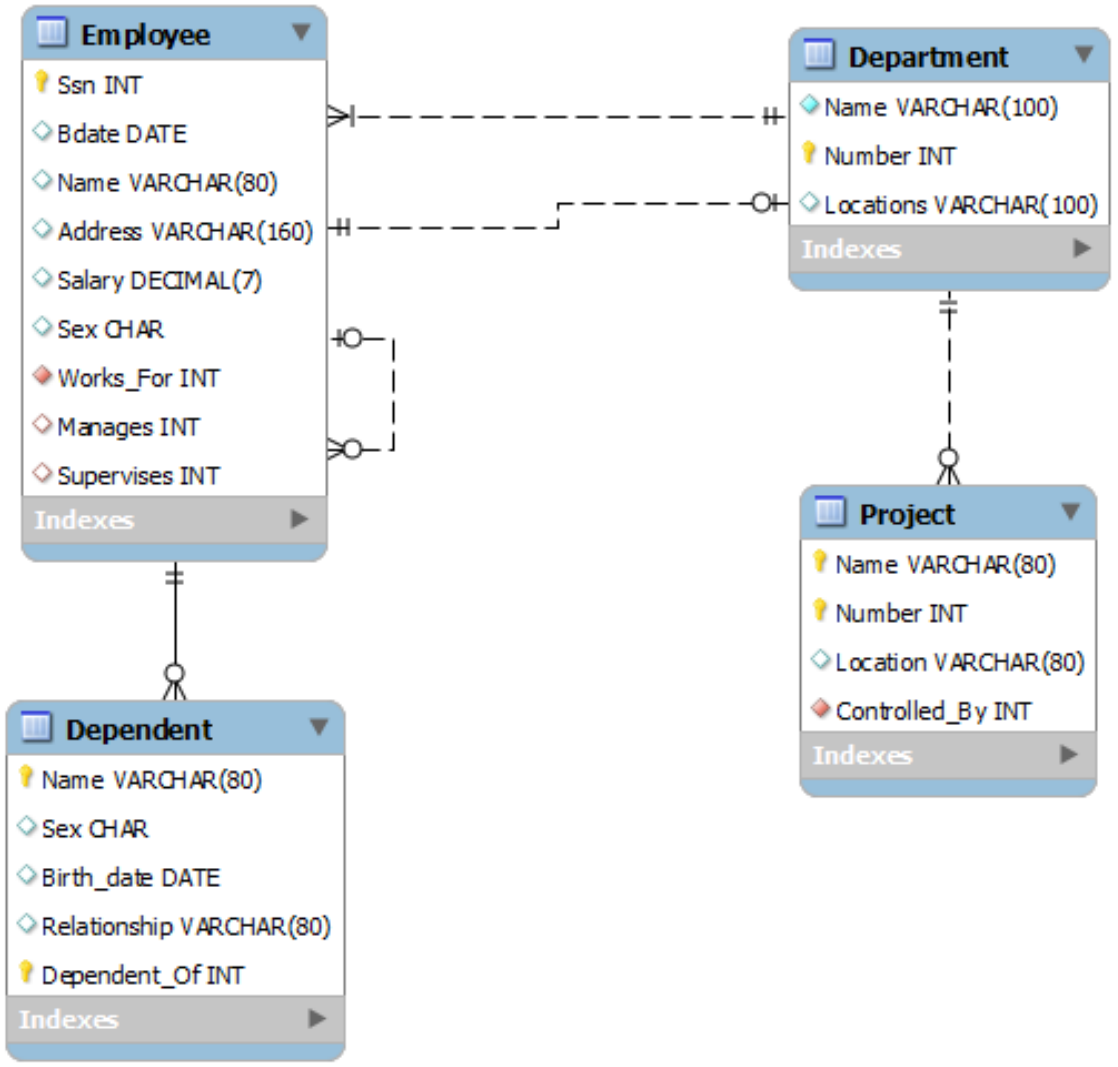


One-To-Many



Many-To-Many

Entity Relationship



Functional Dependency

An attribute B is FUNCTIONALLY DEPENDENT on another attribute A, if a value of A determines a single value of B at any one time.

ORDER-NUMBER➔ORDER-DATE

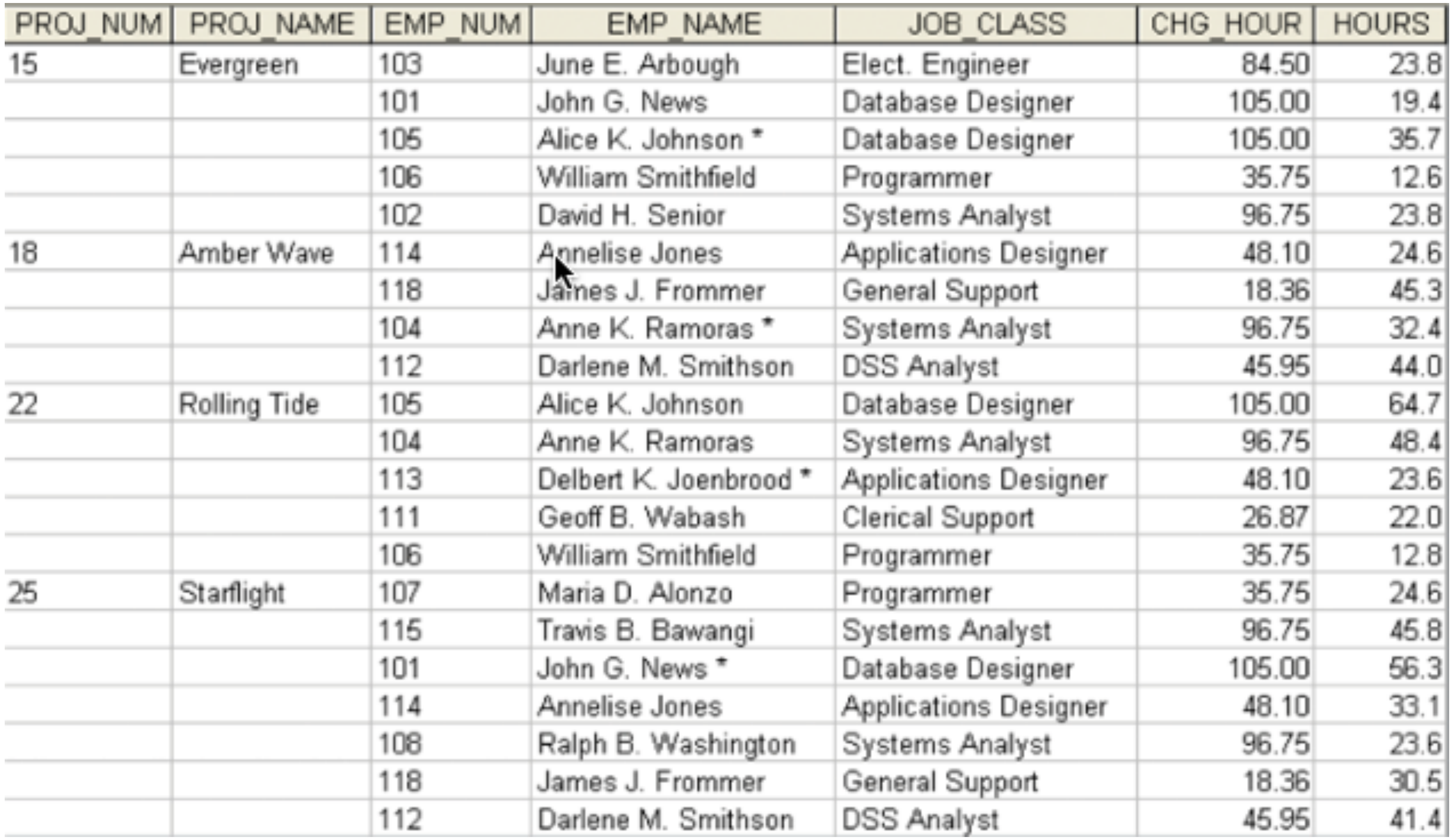
ORDER-NUMBER, PART-NUMBER ➔ QTY-ORDERED, PART-DESCRIPTION

– here although qty-ordered is **fully dependent** on order-number and part-number, only part-number is required to determine part-description
– part-description is said to be **partially dependent** on order-number and part- number

INVOICE-NUMB ➔ CUSTOMER-NUMB ➔ CUSTOMER-NAME

- **transitive dependency** occurs when Y depends on X, and Z depends on Y - thus Z also depends on X ie. X➔Y➔Z

Unormalised Form (UNF)



PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

First Normal Form (1NF)

A RELATION IS IN FIRST NORMAL FORM (1NF) IF

- ❖ a unique key has been identified for each tuple/row.
- ❖ it is a valid relation
  - > – Entity integrity (no part of PK is null)
  - > – Single value for each cell.
  - > – No repeating group.
- ❖ all attributes are functionally dependent on all or part of the primary key

2NF & 3NF

A RELATION IS IN 2NF IF -

- all non key attributes are functionally dependent on the entire key
- ie. no partial dependencies exist

A RELATION IS IN 3NF IF –

- all transitive dependencies have been removed - check for non key attribute dependent on another non key attribute
- Move from 2NF to 3NF by removing transitive dependencies

Entire Process UNF to 3NF

- UNF
  PROJECT (proj_num, proj_name {emp_num, emp_name, job_class, chg_hour, assign_hours})

- 1NF – remove repeating group
  PROJECT (proj_num, proj_name)
  ASSIGN (proj_num, emp_num, emp_name, job_class, chg_hour, assign_hours)

- 2NF – remove partial dependencies
  PROJECT (proj_num, proj_name)
  EMPLOYEE (emp_num, emp_name, job_class, chg_hour)
  ASSIGN (proj_num, emp_num, assign_hours)

- 3NF
  PROJECT (proj_num, proj_name)
  EMPLOYEE (emp_num, emp_name, job_class)
  ASSIGN (proj_num, emp_num, assign_hours)
  JOB (job_class, chg_hour)

Case Study

Pizza shop

Types of pizza

Orders

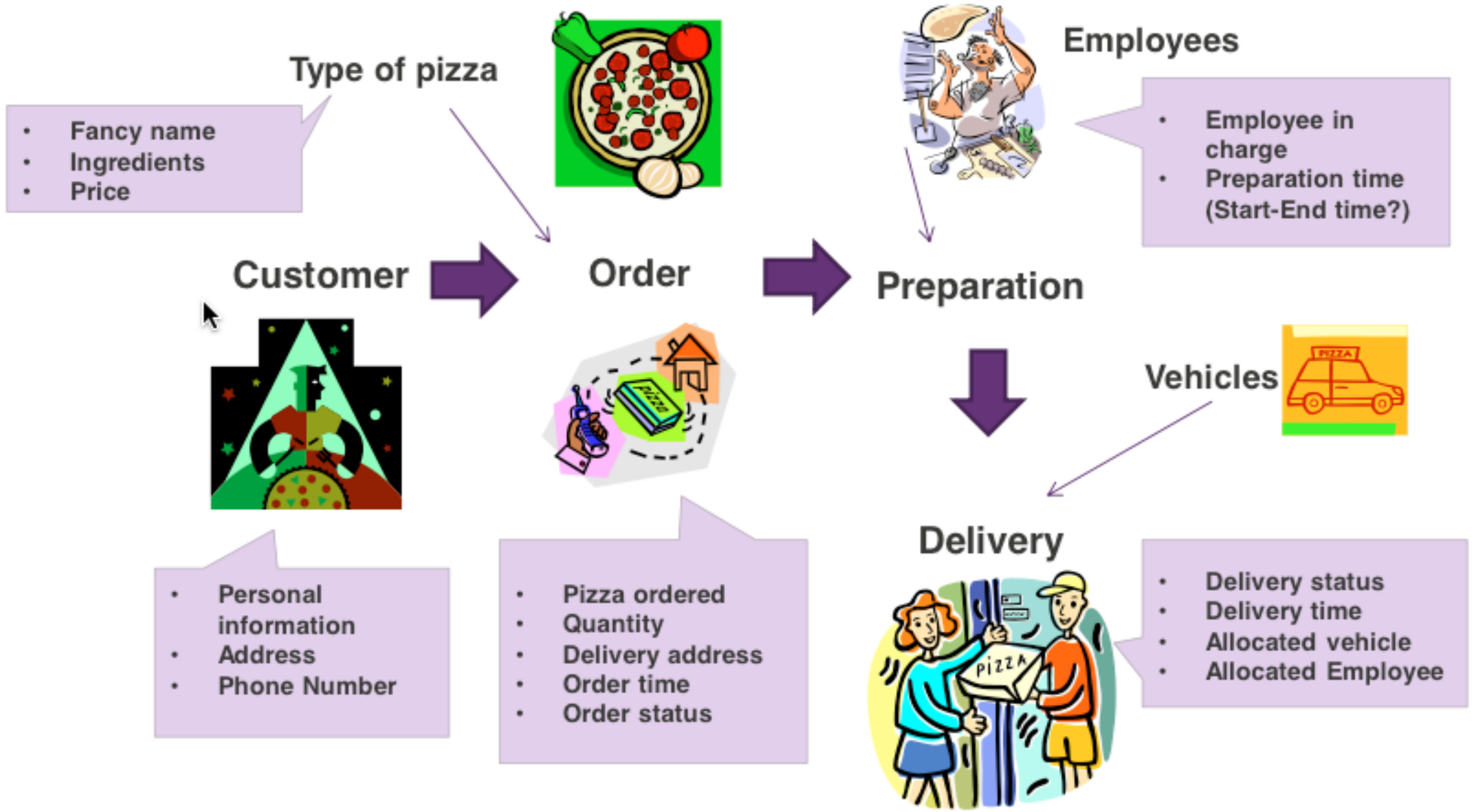
Delivery

Vehicles

Customers

Employees

Case Study



Type of pizza

- Fancy name
- Ingredients
- Price

Customer

- Personal information
- Address
- Phone Number

Order

- Pizza ordered
- Quantity
- Delivery address
- Order time
- Order status

Employees

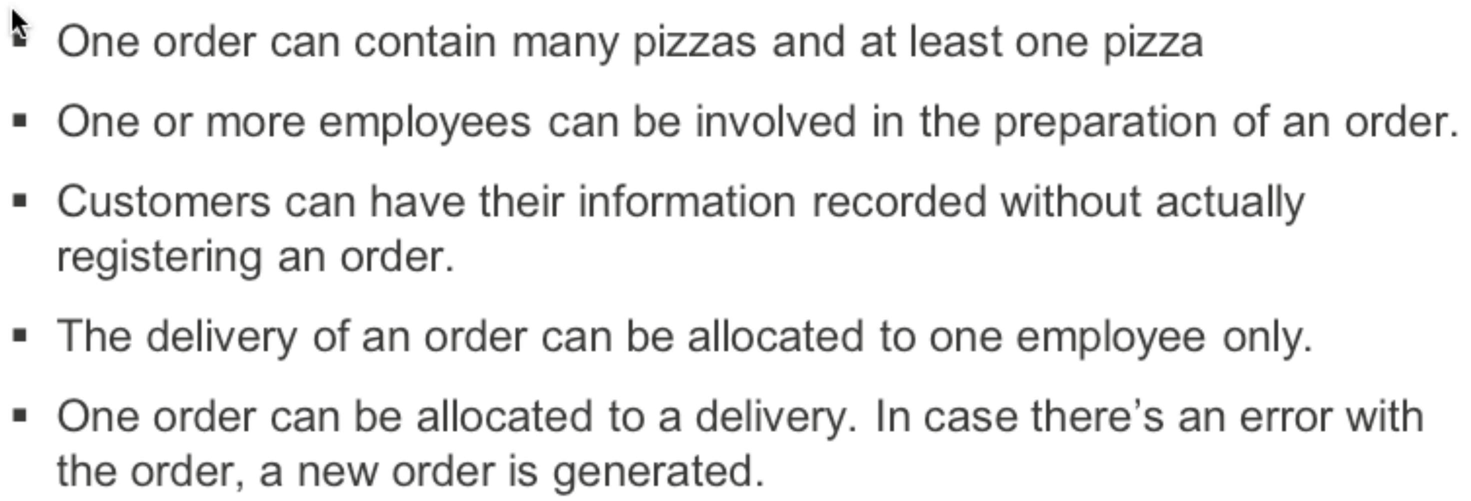
- Employee in charge
- Preparation time (Start-End time?)

Delivery

- Delivery status
- Delivery time
- Allocated vehicle
- Allocated Employee

Case Study

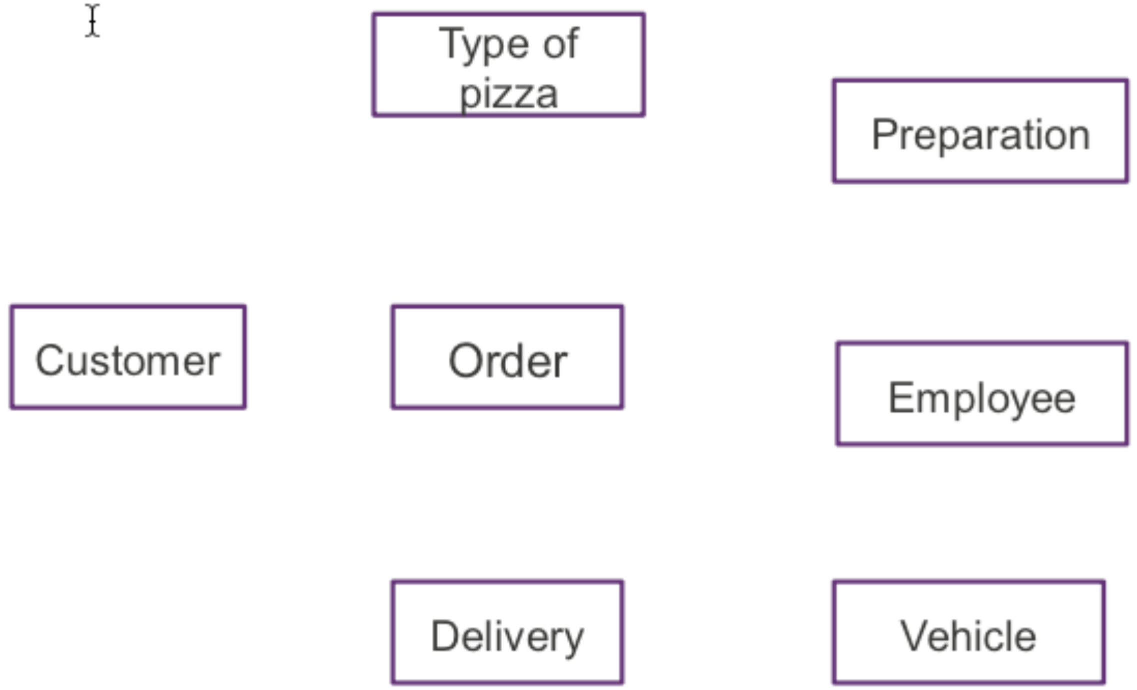
Understand the business



- One order can contain many pizzas and at least one pizza
- One or more employees can be involved in the preparation of an order.
- Customers can have their information recorded without actually registering an order.
- The delivery of an order can be allocated to one employee only.
- One order can be allocated to a delivery. In case there's an error with the order, a new order is generated.

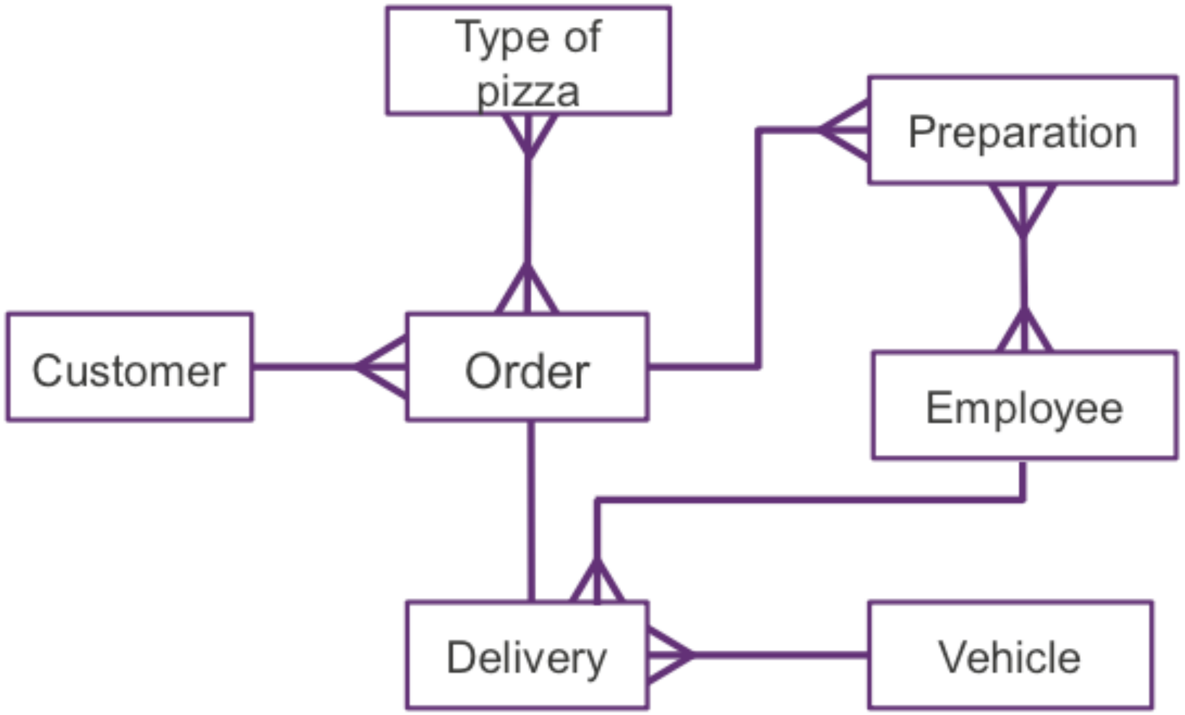
Case Study

Identify entities and put them on paper



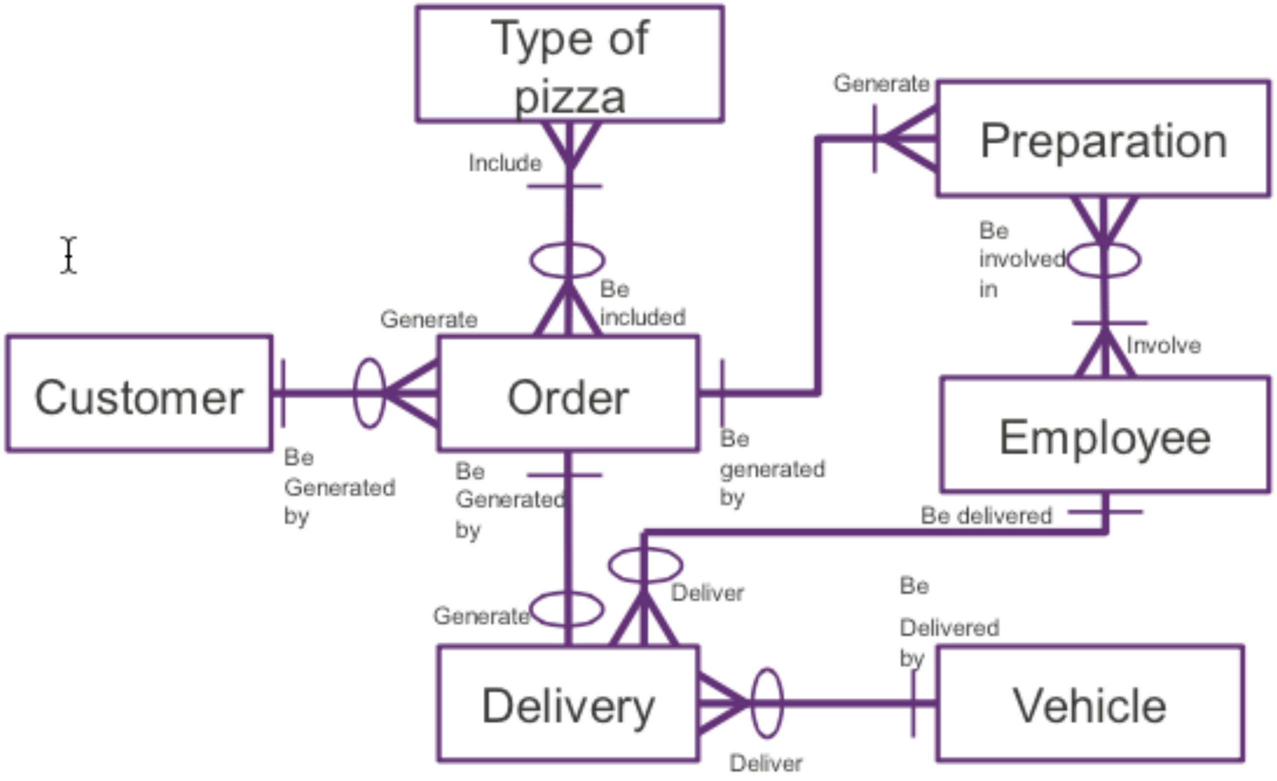
Case Study

Identify relationships



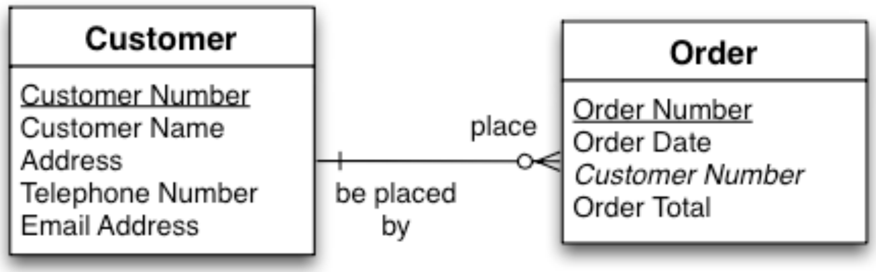
Case Study

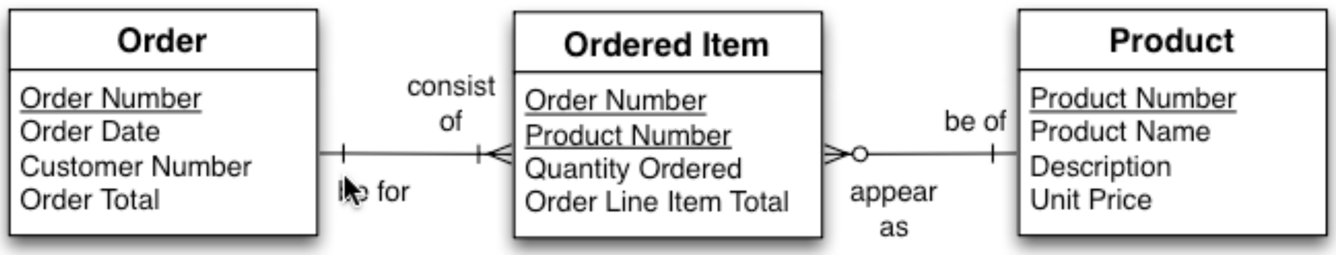
Identify Optionality



Case Study

Attributes and keys





Case Study

Physical implementation

Customer Table

Customer Number	Name	Postal Code	Age
24734	S Hayes	3000	34
33347	H Walsh	3065	43
37942	J O'Dea	3145	55
46745	B Rich	3184	39
78648	A De Silva	3507	27

Insurance Policy Table

Policy Number	Date Issued	Customer Number	Policy Type
1347	2/12/2003	46745	Car02
1487	14/5/2001	33347	Car02
9521	28/6/2004	46745	House01
3458	20/7/2003	78648	Car01
4876	19/4/2005	37942	Boat03