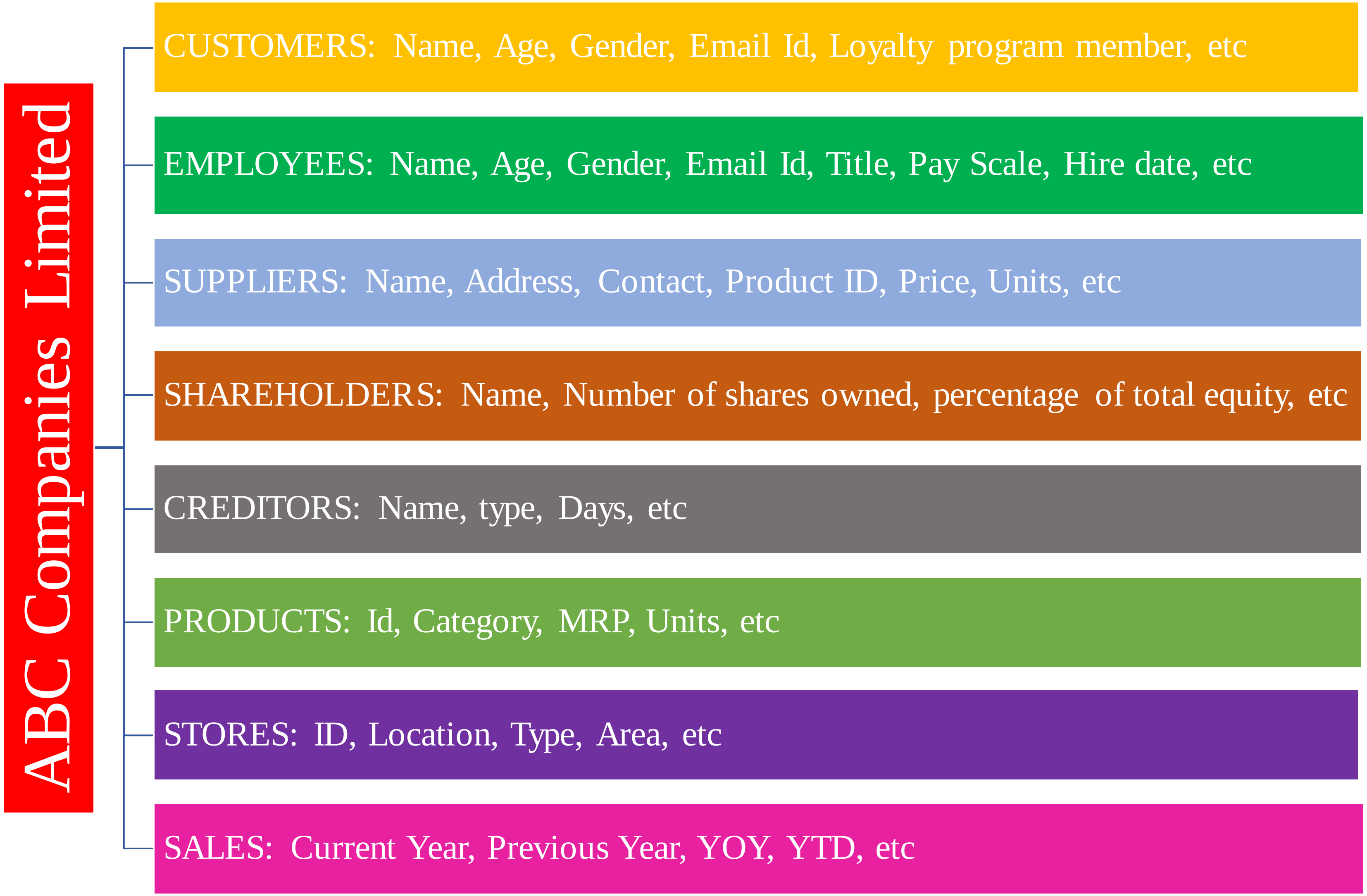
Practical guide to SQL basics

WHAT IS DATA?

Let us consider a large retailer named ABC Companies limited. The data for such a retailer would majorly be about its customers, employees, suppliers, shareholders, creditors, products, stores and sales as shown: ☛

These facts, ex: Name-Vin, Age-26, Units-40, Price-$34.95, etc, or in other words, these set of values with respect to the variables is Data.



ABC Companies Limited

CUSTOMERS: Name, Age, Gender, Email Id, Loyalty program member, etc

EMPLOYEES: Name, Age, Gender, Email Id, Title, Pay Scale, Hire date, etc

SUPPLIERS: Name, Address, Contact, Product ID, Price, Units, etc

SHAREHOLDERS: Name, Number of shares owned, percentage of total equity, etc

CREDITORS: Name, type, Days, etc

PRODUCTS: Id, Category, MRP, Units, etc

STORES: ID, Location, Type, Area, etc

SALES: Current Year, Previous Year, YOY, YTD, etc

ORGANIZED DATA

A few product Id’s and Units at a particular store of ABC Companies Ltd is as follows:

FD139, 250, FD760 , 375, JH565, 410, KU206, 190, SY408, 200, BK213, 255, FR908, 384, SL650, 293, ZM987, 212, LK209, 368

- This form of representation of data becomes confusing when the number of products increase.

Representation of data in a tabular form as shown 👉 is easy to interpret and analyse the data.

Product Id	Units
FD139	250
FD760	375
JH565	410
KU206	190
SY408	200
BK213	255
FR908	384
SL650	293
ZM987	212
LK209	368

Suppose we have data for 1000 product Ids and the company wants to know the product Ids having more than 250 units.

Unorganized data: Will take a lot of time and resource to gather the required information

Organized data in ascending order of units: Will be very quick and easy to sort out the required information.

DATABASE

Database can be considered as a cupboard of organized data.

Each **drawer** of the cupboard represents a table also called an **Entity**.
Ex: Employees, Products, Packaging, etc

Each Entity has variables known as **Fields**.
Ex: EmployeeID, Title, DOB, etc are the fields in the Employees Entity.

Each field has values known as **Data**.
Ex:

BrandID	CategoryID
162	SD125
208	GF326
363	FF118
284	DP251
189	RM318

DATA

EMPLOYEES
EmployeeID
Emp_Department_ID
Manager_ID
LastName
FirstName
Title
DOB
HireDate

DEPARTMENT
Emp_Department_ID
Department Name
Department Code
Department Description
Department Head
Department_Start_Date

STORE
Store_ID
Store_Code
Store_Name
Store_Region_ID
Store_Type

STORE BILLING
Bill_Number
Customer_ID
EmployeeID
Store_Billing_Date
Store_Billing_Time
Store_Start_Date
Store_ID

STORE BILLING DETAILS
Bill_Number
ProductID
UnitPrice
Quantity
Total Amount
Discount
Storage cost per item
Cost price per item
Labour cost per item

STORE REGION
Store_Region_ID
Store_Region_Nmae
Store_County
Store_State

CUSTOMERS
CustomerID
Customer_Type_ID
Customer_Region_ID
Customer_Shopper_ID
Customer_Full_Name
Customer_First_Name
Customer_Last_Name
Address
City
Region

CUSTOMER TYPE
Customer_Type_ID
Customer_Type
Customer_Description

REGION
RegionID
Customers_Region_Description

TERRITORIES
TerritoryID
Customer_Territory_Description
RegionID

PRODUCTS
ProductID
ProductName
SupplierID
CategoryID
Product Code
Product Country
BrandID
More…

BRAND
BrandID
CategoryID
Brand_Description

CATEGORIES
CategoryID
CategoryName
Description
Picture

SUPPLIERS
SupplierID
Supplier_Type_ID
Supplier_Name
Supplier_Location_Manager
Supplier_Region

SUPPLIER_TYPE
Supplier_Type_ID
Supplier_Type
Supplier_Description

PACKAGING
PackageID
Package_Description
Package_Type

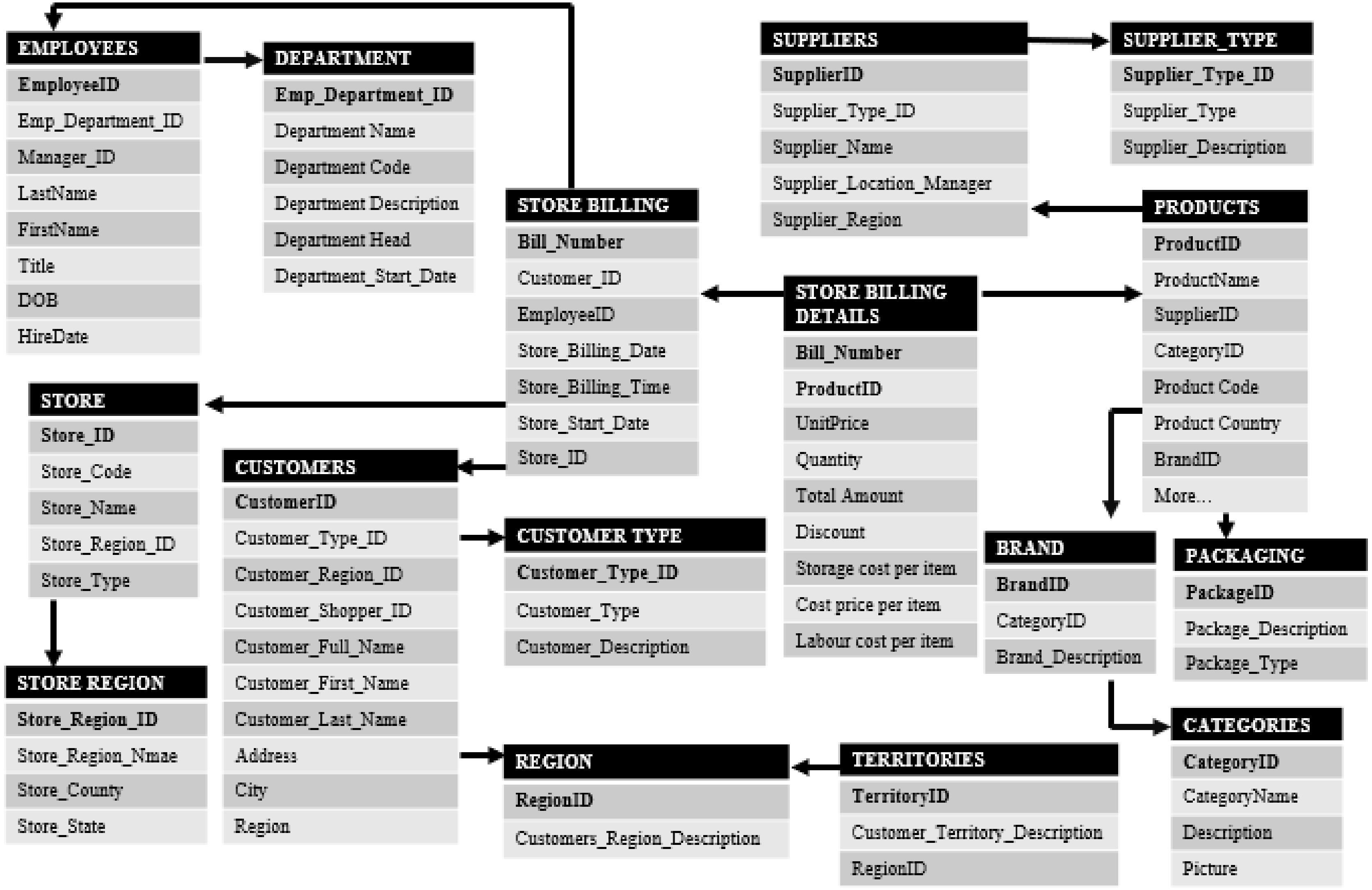
RELATIONAL DATABASE

Each Table/Entity is a **Relation** as it contains related data.

Suppose I need a list of Customers with their ID, Type and store billing details.

- In order to get the customer type, there needs to be a relation between Customers and Customer type which is established based on a common field which is ‘Customer_Type_ID’.
- Similarly, to get the store billing details, there should be a relation between Customers, Store billing and Store billing details based on a common field which is ‘CustomerID’ and ‘Bill_Number’ respectively.

A database which lets us look at any of the entities and allows us to derive more information through its relation with other entities is known as a **Relational Database**.



EMPLOYEES
EmployeeID
Emp_Department_ID
Manager_ID
LastName
FirstName
Title
DOB
HireDate

DEPARTMENT
Emp_Department_ID
Department Name
Department Code
Department Description
Department Head
Department_Start_Date

STORE BILLING
Bill_Number
Customer_ID
EmployeeID
Store_Billing_Date
Store_Billing_Time
Store_Start_Date
Store_ID

SUPPLIERS
SupplierID
Supplier_Type_ID
Supplier_Name
Supplier_Location_Manager
Supplier_Region

SUPPLIER_TYPE
Supplier_Type_ID
Supplier_Type
Supplier_Description

PRODUCTS
ProductID
ProductName
SupplierID
CategoryID
Product Code
Product Country
BrandID
More…

STORE BILLING DETAILS
Bill_Number
ProductID
UnitPrice
Quantity
Total Amount
Discount
Storage cost per item
Cost price per item
Labour cost per item

STORE
Store_ID
Store_Code
Store_Name
Store_Region_ID
Store_Type

CUSTOMERS
CustomerID
Customer_Type_ID
Customer_Region_ID
Customer_Shopper_ID
Customer_Full_Name
Customer_First_Name
Customer_Last_Name
Address
City
Region

CUSTOMER TYPE
Customer_Type_ID
Customer_Type
Customer_Description

BRAND
BrandID
CategoryID
Brand_Description

PACKAGING
PackageID
Package_Description
Package_Type

STORE REGION
Store_Region_ID
Store_Region_Nmae
Store_County
Store_State

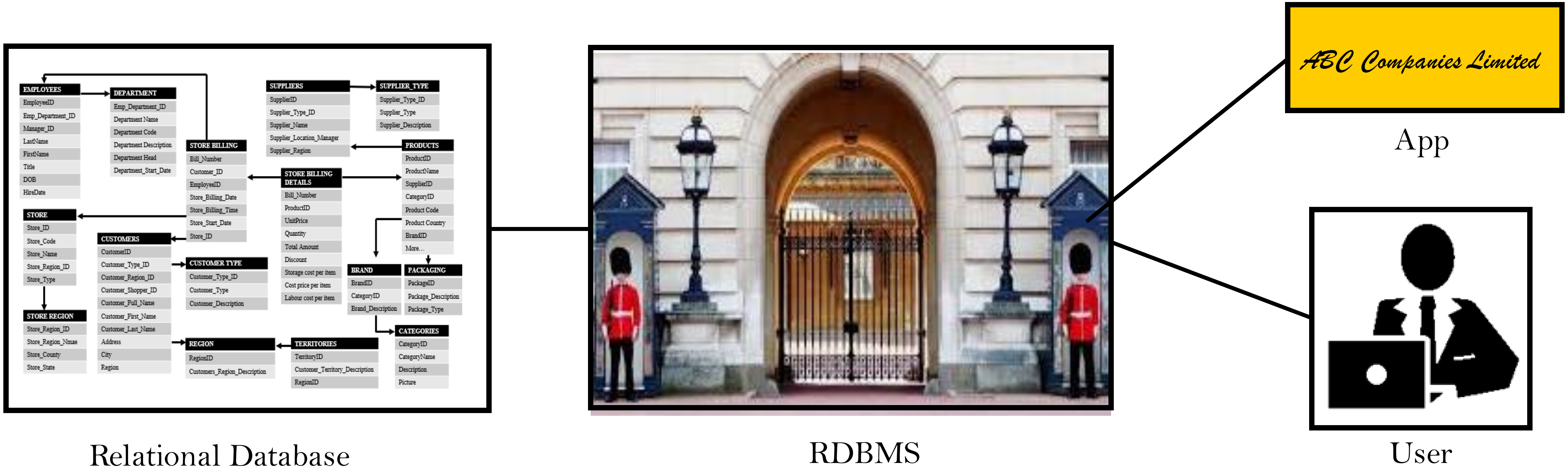
REGION
RegionID
Customers_Region_Description

TERRITORIES
TerritoryID
Customer_Territory_Description
RegionID

CATEGORIES
CategoryID
CategoryName
Description
Picture

Note: Fields in Bold are Primary Key

RELATIONAL DATABASE MANAGEMENT SYSTEM



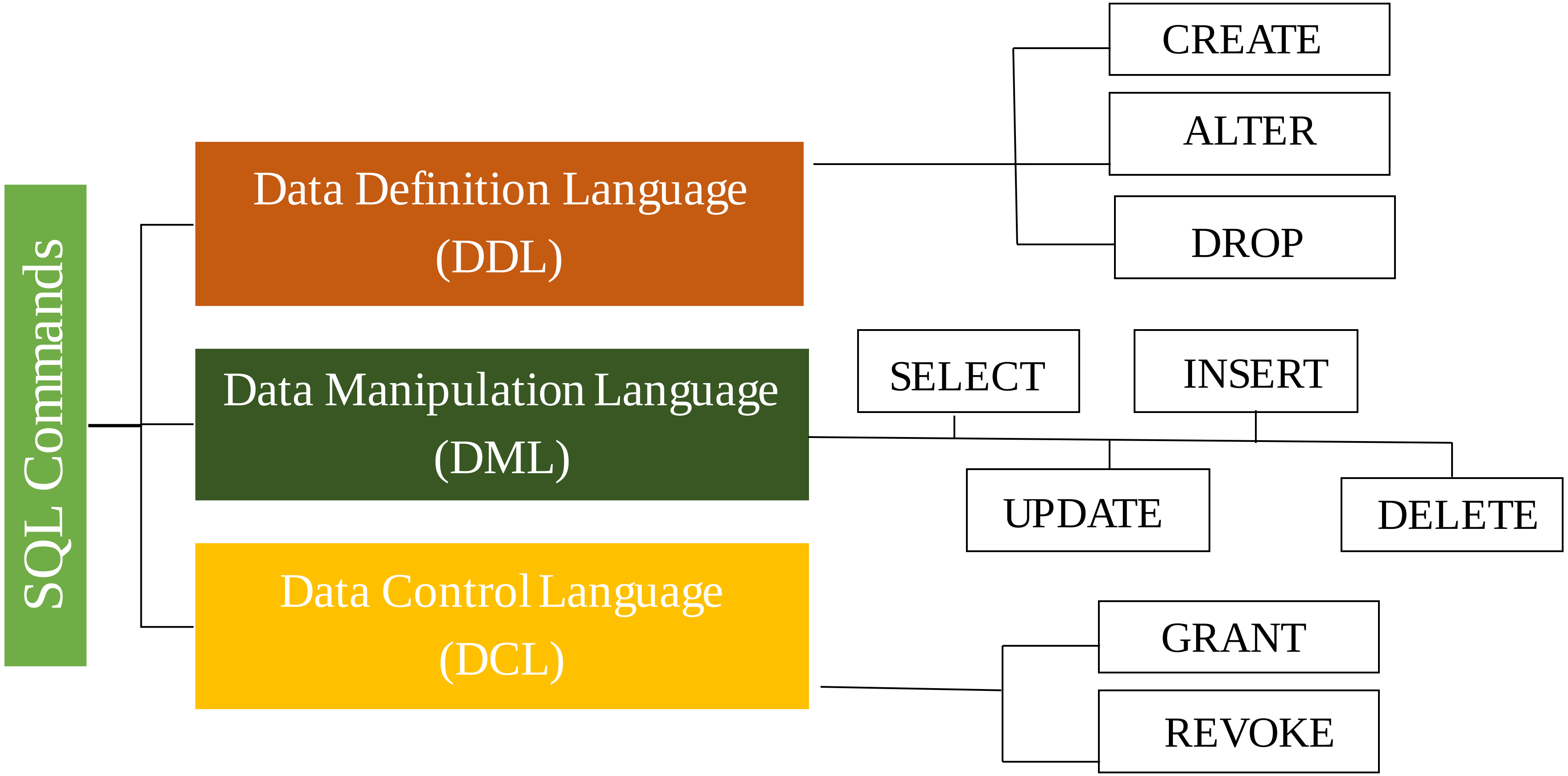
- RDBMS is a software that can be considered as a gatekeeper to relational database.
- It provides users and programmers with a systematic way to create, retrieve, update and delete data in a database.
- It ensures that data is consistently organized and remains easily accessible.

STRUCTURED QUERY LANGUAGE (SQL)

The language used to interact with a relational database is known as Structured Query Language (SQL).

The commands in SQL are classified in three groups as shown ☛

Suppose we want to create a database called ‘ABCretail’. We can use the following commands to create, see and select the database respectively.



SQL Commands

Data Definition Language (DDL)

CREATE

ALTER

DROP

Data Manipulation Language (DML)

SELECT

INSERT

UPDATE

DELETE

Data Control Language (DCL)

GRANT

REVOKE

Create database ABCretail; >>This will create a database called ‘ABCretail’

Use ABCretail; >>This will select ‘ABCretail’ as the database for data retrieval.

Drop database ABCretail; >> This will delete ‘ABCretail’ database.

SQL Commands in blue

DATA DEFINITION LANGUAGE

CREATE TABLE command creates an entity structure in the database by defining the following:

- **Entity Name:** SQL throws error for names with space character. Putting the name within square brackets [ ] works well for names with space character.

- **Field Name:** Putting the name within square brackets [ ] works well for names with space character.

- **Field data type:** The type of data that shall be going within the Column is defined.

- **Primary key:** It is a field which uniquely identidies each record. It can not have Null values. There can be only one primary key in a table which can be either on one field or more than one field.

Basic Syntax:

```
Create table table name (field1 datatype, field2 datatype, field3 datatype, .....fieldN datatype, PRIMARY KEY (one or
 more columns ));
```

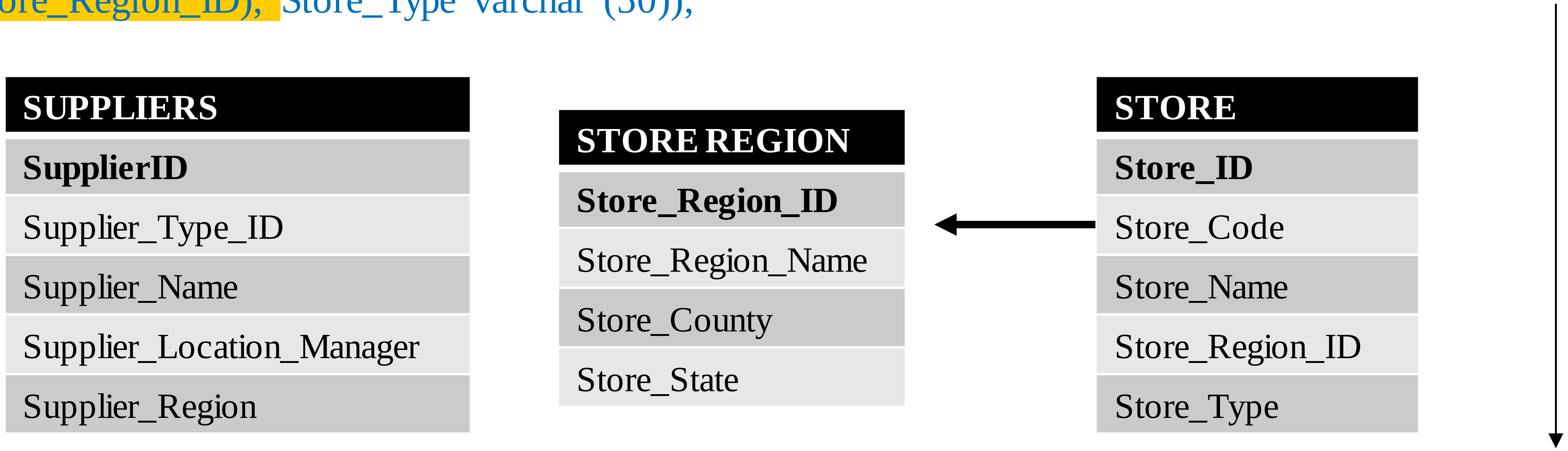
DATA DEFINITION LANGUAGE [CONT..]

Ex: Let us create the following entities:

Create table Suppliers (SupplierID int primary key not null, Supplier_Type_ID int, Supplier_Name varchar(100), Supplier_Location_Manager varchar(100), Supplier_Region varchar(75));

Create table [Store Region] (Store_Region_ID int primary key not null, Store_Region_Name varchar(100), Store_County varchar(50), Store_State varchar(100));

Create table Store (Store_ID int primary key not null, Store_Code int, Store_Name varchar (75), Store_Region_ID int references [Store Region] (Store_Region_ID), Store_Type varchar (50));



SUPPLIERS
SupplierID
Supplier_Type_ID
Supplier_Name
Supplier_Location_Manager
Supplier_Region

STORE REGION
Store_Region_ID
Store_Region_Name
Store_County
Store_State

STORE
Store_ID
Store_Code
Store_Name
Store_Region_ID
Store_Type

- The relation between Store Region and Store is based on a common field ‘Store_Region_ID’.
- It acts as a cross-reference between tables because it references the primary key of another table which in this case is Store Region, thereby establishing a link between them.
- Such a common field providing link between two tables is known as **Foreign key**.

DATA DEFINITION LANGUAGE [CONT..]

ALTER TABLE command is used to add, delete or modify fields in an existing entity.

- Let us add a field ‘Supplier_Contact_info’ to the Suppliers entity.

  Alter table Suppliers add Supplier_Contact_Info varchar(50);

- Drop field ‘Store_County’ from Store Region.

  Alter table [Store Region] drop Store_County;

- Let us modify the datatype of Store_ID from int (as defined in the previous slide) to smallint.

  Alter table Store modify column Store_ID smallint;

DROP TABLE command removes the table definition and all the data for that table. Once a table is deleted then all the information available in that table will also be lost forever.

- Suppose we want to drop Store Region

  Drop table [Store Region];

DATA MANIPULATION LANGUAGE

Consider the following entities populated with data as shown below:

SUPPLIERS

SupplierID	Supplier_Type_ID	Supplier_Name	Supplier_Location _Manager	Supplier_Region
3589	58	Zai Inc	David	Atlantic
5871	79	Johr services	Chris	Ontario
8746	34	Imi Labs	Bob	Quebec
4812	15	Yirtel Inc	Angela	Ontario
9578	46	Voda services	Sara	Quebec

*1

*2

STORE REGION

Store_Region_ID	Store_Region_Name	Store_County	Store_State
451	Atlantic	Halifax	Nova Scotia
879	Quebec	Montreal	Quebec
358	Ontario	Peterborough	Ontario
157	Atlantic	Charlottetown	Prince Edward Island
258	Ontario	Simcoe	Ontario

a

*3

STORE

Store_ID	Store_Code	Store_Name	Store_Region_ID	Store_Type
1547	15	Monezis	258	Speciality
2624	71	Campust	358	Super market
5455	89	Leastprice	879	Discount
3879	69	Bacabus	157	Super market
6894	37	Medicas	451	Drug store

*4

*5

DATA MANIPULATION LANGUAGE [CONT..]

SELECT command is used to retrieve data from database entities.

Select SupplierID, Supplier_Type_ID, Supplier_Name from Suppliers >> This retrieves all the records for fields SupplierID, Supplier_Type_ID and Supplier_Name from Suppliers entity.*[1]

Select * from Suppliers >> This gets all the records for all the fields in the entity. The result is the whole suppliers entity as shown in previous slide.

Select top 2 * from [Store Region] >> Returns a (Refer slide 13)

SELECT command with conditional statements using **WHERE, LIKE, AND, OR** clauses.

Select * from Suppliers where SupplierID = 8746 >> Returns *[2]

Select Store_Region_ID, Store_Region_Name from [Store Region] where Store_County = ‘Peterborough’ >> Returns *[3]

Select * from Store where Store_Name **like** ‘M%’ **and** Store_Code = 15 >> Looks for records where Store_Name starts with ‘M’ and Store_Code is 15 and returns *[4]

Select * from Store where Store_Type = ‘Super market’ **or** Store_Code = 71 >> Looks for records where Store_Name starts with ‘M’ or Store_Code is 15 and returns *[5]

*Note: * refers to slide 11*

DATA MANIPULATION LANGUAGE [CONT..]

SELECT command with conditional statements using **WHERE, GROUP BY, HAVING, ORDER BY** clauses.

As per Store Billing Details, Suppose we need a list of unique bill_number with their respective sum of [Total Amount] with the following conditions:

1) UnitPrice > 16.00
2) Sum of [Total Amount] > 110

in descending order of Bill_Number.

STORE BILLING DETAILS

Bill_Number	ProductID	UnitPrice	Quantity	Total Amount	Discount	Storage cost per item	Cost price per item	Labour cost per item
12345	11	20.99	5	104.95	3%	0.75	14.50	1.50
67891	25	15.49	2	30.98	2%	1.00	9.90	1.50
01112	48	121.89	1	121.89	5%	5.00	60.50	15.00
12345	67	78.45	2	156.9	5%	2.0	50.00	5.00
51617	83	35.55	3	106.65	2%	2.50	15.00	4.00
01112	57	90.99	1	90.99	5%	2.50	65.49	10.00

```
Select Bill_Number, sum([Total Amount]) from [Store Billing Details]
where UnitPrice > 16
group by Bill_Number
having sum([Total Amount]) > 110
order by Bill_Number Desc
```

Bill_Number	Sum(Total Amount)
12345	261.85
01112	212.88

DATA MANIPULATION LANGUAGE [CONT..]

INSERT INTO command is used to add new rows of data to an entity in the database.

Let us add 2 more rows of data to
Store Billing Details
as shown in previous slide

STORE BILLING DETAILS

Bill_Number	ProductID	UnitPrice	Quantity	Total Amount	Discount	Storage cost per item	Cost price per item	Labour cost per item
12345	11	20.99	5	104.95	3%	0.75	14.50	1.50
67891	25	15.49	2	30.98	2%	1.00	9.90	1.50
01112	48	121.89	1	121.89	5%	5.00	60.50	15.00
12345	67	78.45	2	156.9	5%	2.00	50.00	5.00
51617	83	35.55	3	106.65	2%	2.50	15.00	4.00
01112	57	90.99	1	90.99	5%	2.50	65.49	10.00
25987	35	52.49	3	157.47	5%	2.50	35.50	2.00
65874	88	35.55	3	106.65	2%	2.50	16.00	3.00

```
Insert into [Store Billing Details] (Bill_Number, PoductID, UnitPrice, Quantity, [Total Amount], Discount,
[Storage cost per item], [Cost price per item], [Labour cost per item]) values
(25987, 35, 52.49, 3, 157.47, ‘5%’, 2.50, 35.50, 2.00),
(65874, 88, 35.55, 3, 106.65, ‘2%’, 2.50, 16.00, 3.00)
```

DATA MANIPULATION LANGUAGE [CONT..]

UPDATE and **DELETE** commands are used to modify and delete the existing records in an entity respectively.

1) We want to change the discount percentage of the highlighted cell to 10%.

2) We want to keep only those records where quantity is less than 2.

3) Delete all records from table.

STORE BILLING DETAILS								
Bill_Number	ProductID	UnitPrice	Quantity	Total Amount	Discount	Storage cost per item	Cost price per item	Labour cost per item
12345	11	20.99	5	104.95	3%	0.75	14.50	1.50
67891	25	15.49	2	30.98	2%	1.00	9.90	1.50
01112	48	121.89	1	121.89	5%	5.00	60.50	15.00
12345	67	78.45	2	156.9	5%	2.00	50.00	5.00
51617	83	35.55	3	106.65	2%	2.50	15.00	4.00
01112	57	90.99	1	90.99	5%	2.50	65.49	10.00
25987	35	52.49	3	157.47	5%	2.50	35.50	2.00
65874	88	35.55	3	106.65	2%	2.50	16.00	3.00

1) Update [Store Billing Details] set Discount = ’10%’ where Bill_Number = 01112 and ProductID = 48

2) Delete from [Store Billing Details] where Quantity < 2

3) Delete from [Store Billing Details]

STORE BILLING DETAILS								
Bill_Number	ProductID	UnitPrice	Quantity	Total Amount	Discount	Storage cost per item	Cost price per item	Labour cost per item
01112	48	121.89	1	121.89	5%	5.00	60.50	15.00
01112	57	90.99	1	90.99	5%	2.50	65.49	10.00

DATA CONTROL LANGUAGE

Data control language commands are used by database administrator or owner of the database object to provide/remove privileges on a database object.

GRANT command is used by database administrator to provide access or privileges on the database objects to the users.

Grant select on Employees to user1 >> This command grants a SELECT permission on Employees entity to user1.

REVOKE command removes user access rights or privileges to the database objects.

Revoke select on Employees from user1 >> This command will revoke a SELECT privilege on employee entity from user1.

JOINS

In order to derive more information through entities relation with each other, we need to combine entities based on common fields using **JOIN** clause.

STORE REGION			
Store_Region_ID	Store_Region_Name	Store_County	Store_State
451	Atlantic	Halifax	Nova Scotia
879	Quebec	Montreal	Quebec
358	Ontario	Peterborough	Ontario
157	Atlantic	Charlottetown	Prince Edward Island
258	Ontario	Simcoe	Ontario

STORE				
Store_ID	Store_Code	Store_Name	Store_Region_ID	Store_Type
1547	15	Monezis	258	Speciality
2624	71	Campust	358	Super market
5455	89	Leastprice	879	Discount
3879	69	Bacabus	157	Super market
6894	37	Medicas	451	Drug store
4426	25	Medicas	554	Drug store
7413	46	Leastprice	631	Discount

From the above two entities, let's say, we want to know all the store Id's with their Name, type and state.

```
Select Store_ID, Store_Name, Store_Type, Store_State from [Store Region] join Store on [Store Region].Store_Region_ID = Store. Store_Region_ID
```

>>

Store_ID	Store_Name	Store_Type	Store_State
1547	Monezis	Speciality	Ontario
2624	Campust	Super market	Ontario
5455	Leastprice	Discount	Quebec
3879	Bacabus	Super market	Prince Edward Island
6894	Medicas	Drug store	Nova Scotia

This reflects the common field on which the join is based. In this case, Store_Region_ID is a common field where it is a primary key in Store Region and a foreign key in Store.

TYPES OF JOINS

Inner Join: Returns only those set of records that match in both the entities. Inner Join is same as join, as explained in the previous slide.

Left Join: Returns all rows from the left entity, even if there are no matches in the right entity.

Select Store_ID, Store_State from Store left join [Store Region] on Store.Store_Region_ID = [Store Region].Store_Region_ID

>>

Store_ID	Store_State
1547	Ontario
2624	Ontario
5455	Quebec
3879	Prince Edward Island
6894	Nova Scotia
4426	Null
7413	Null

Store_ID	Store_State
6894	Nova Scotia
5455	Quebec
2624	Ontario
3879	Prince Edward Island
1547	Ontario

Right Join: Returns all rows from the right entity, even if there are no matches in the left entity.

Select Store_ID, Store_State from Store right join [Store Region] on Store.Store_Region_ID = [Store Region].Store_Region_ID

Full Join: Returns all the records from both the entities. It is a combination of the result of both left join and right join.

Select Store_ID, Store_State from Store full join [Store Region] on Store.Store_Region_ID = [Store Region].Store_Region_ID

WHY DO WE USE SQL WHEN WE HAVE EXCEL?

- Considering the huge database of ABC Companies Limited (refer slide 4), with some entities like store billing going up to millions of records, Excel won’t last a few hours before it comes unusable.

- Let’s say we want to know the number of units of a particular product broken down by store and how many of those customers are end users. In a SQL database, that’s a fairly easy query to write and we will retrieve data in seconds.

  Doing the same in excel might be arduous and time consuming.

- SQL *separates analysis from data*. When using SQL, data is stored separately from analysis.

  That saves us from emailing large excel files. Instead send small text files having the instructions for analysis.

  Also saves from having to manage file versions or risk corrupting the data.

APPENDIX

DATA TYPES

Exact Numeric Data Types

DATA TYPE	FROM	TO
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38+1	10^38-1
numeric	-10^38+1	10^38-1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

APPENDIX

DATA TYPES

Character Strings Data Types

DATA TYPE	Description
char	Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
varchar	Maximum of 8,000 characters.(Variable length non-Unicode data)
varchar(max)	Maximum length of 231 characters. Variable length non-Unicode data (SQL Server 2005 only).
text	Variable length non-Unicode data with a maximum length of 2,147,483,647 characters.

Unicode Character Strings Data Types

DATA TYPE	Description
nchar	Maximum length of 4,000 characters.(Fixed length Unicode)
nvarchar	Maximum of 4,000 characters.(Variable length Unicode)
nvarchar(max)	Maximum length of 231 characters. Variable length Unicode (SQL Server 2005 only).
ntext	Maximum length of 1,073,741,823 characters. (Variable length Unicode)

All the commands (Highlighted in blue) work for MS SQL Server.

Thank You

Created by Pavani Ganti