Analytics Vidhya
Learn everything about analytics

SQL for Data Science

Module 9: Introduction to Indexing

Learning Objectives of this module

- Introduction to Indexing
- How indexing works (basics)
- Relationships
- Types of Relationships
- Table Constraints - PRIMARY KEY, UNIQUNESS, FOREIGN KEY and Auto Increment

What is an Index

- Data structure that improves the speed of operations in a table.
- Indexes can be created using one or more columns
- Indexing a column improves search but increases the time for insert and update

Syntax

Create Index on a column

create index <index_name> on <table> (<Column to Index>)

Ex - create index name_1 on Contacts (Name)

Ex - create index name_composite on Contacts (Name, PhoneNumber)

How Indexing Works

- Some fairly complex data structures – like B trees are involved
- Here we try to see the big picture using a similar situation
- Why we need indexing?
- Similar situation – Library
  - New books come in
  - Some books are updated, some removed, some become outdated
  - Searching books should be easy
- Solution: Organize the books hierarchically
  - Subject
    - Author
      - Name
- Search time decreases, insertion and update increase

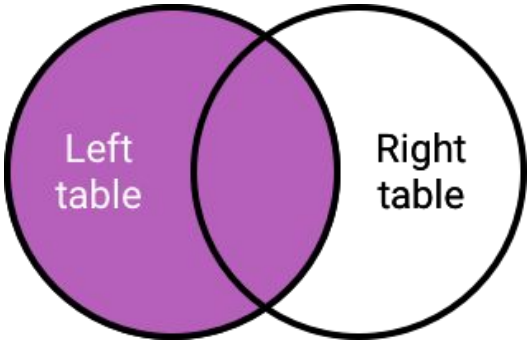
Types of Relationships

- One one relationship
  - One record in first table can only correspond to one record in second table
  - Example: One user can have only one Social Security Number or Aadhar Number in India which uniquely defines him/ her
  - Modeled using – PRIMARY KEY and UNIQUNESS constraint
- One to many relationship
  - One record in first table can correspond to multiple records in second table but not vice-versa
  - Example: One user can have multiple profiles on our job portal with different resumes, but a resume cant be of multiple users
  - Modeled using – Foreign Keys
- Many to many relationship
  - Multiple records in first table can correspond to multiple records in second table
  - Example: One user can apply on multiple jobs and one job can be applied on by multiple users.
  - Modeled using – Separate table with foreign keys from both tables

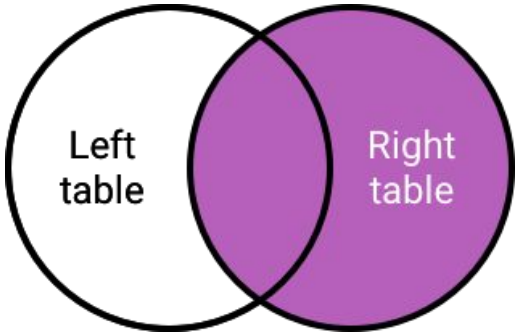
Table Constraints

- Primary Key
  - There can be only one primary key in a table
  - It cant be repeated in the table. Repeated insertion -> Error
- Foreign Key
  - Key of another table
  - Example: userid in applies table is foreign key of userid in UserDetails table
  - Only a user on the portal can apply (Constraint)
- Uniqueness
  - Any field can be constrained to be unique (cant repeat)
- Auto Increment
  - Field is like a serial number
  - Increments handled automatically by MySQL

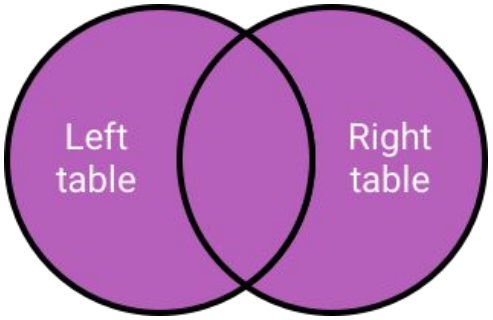
Types of Joins



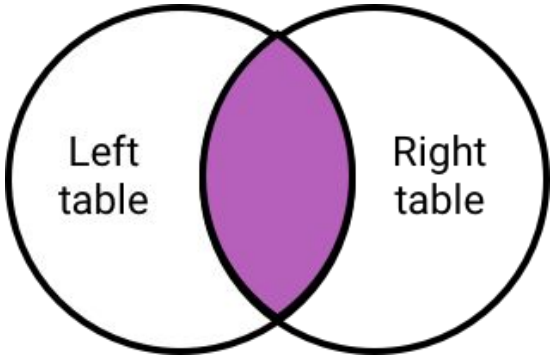
Left Join (90%)



Right Join (< 0.01 %)



Cartesian Join (< 0.01 %)

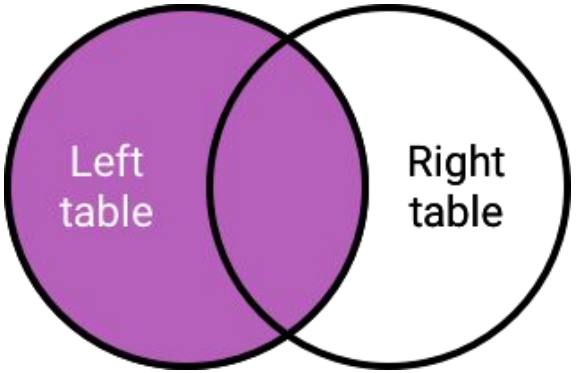


Inner Join (9.99%)

Types of Joins – The Left Join

Left Join

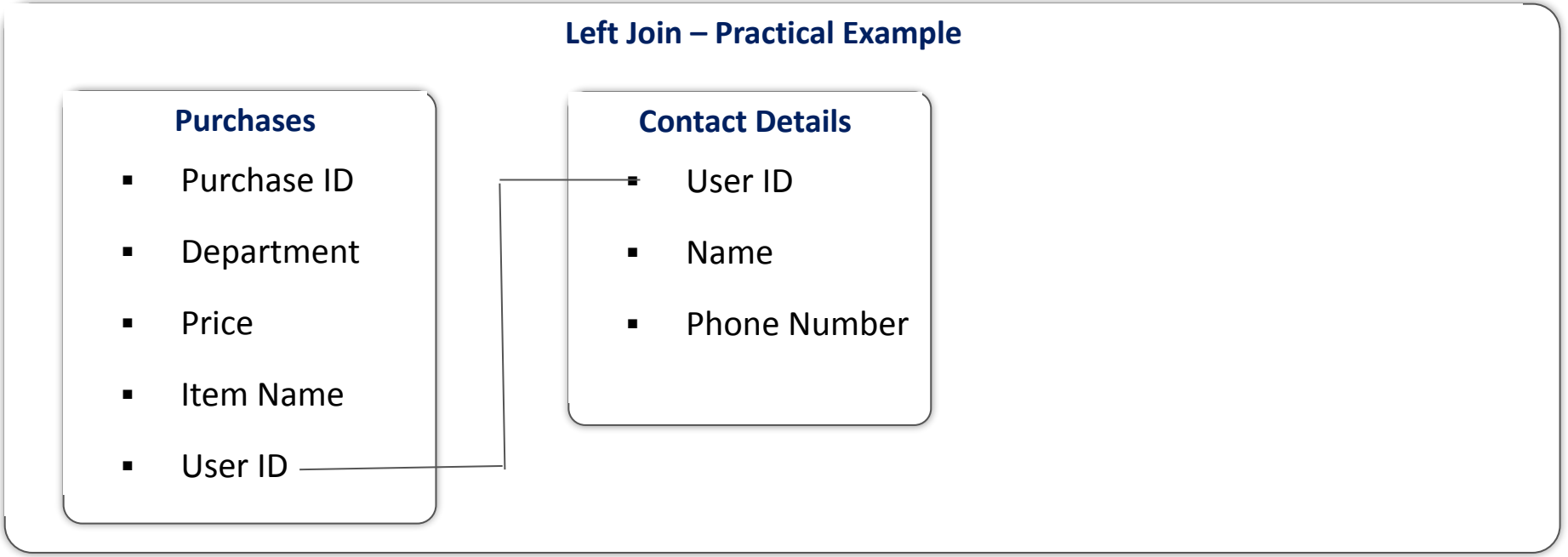
- Take all the records from the left table and get only the corresponding (matching) records from the right table
- ONLY records matching to the left table are considered
- Matching is done based on a particular column known as the key



Command

select <table 1> .a, <table 1> .b, <table 2> .c from <table 1> left join <table 2> on <table1>.<key1> = <table2>.<key2>

The Left Join – Practical Example

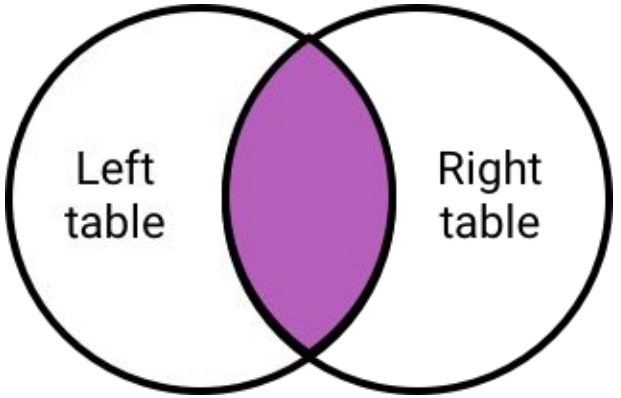


Query to find the Contacts of top revenue customers

```
select ContactDetails.Name, ContactDetails.PhoneNumber from
Purchases left join ContactDetails
on Purchases.UserID = ContactDetails.UserID
```

Inner Join

- The inner join returns ONLY the records that are matching in BOTH the tables
- Can do one inner join after other – there is no limit
- More the number of tables involved – Slower the query



Command

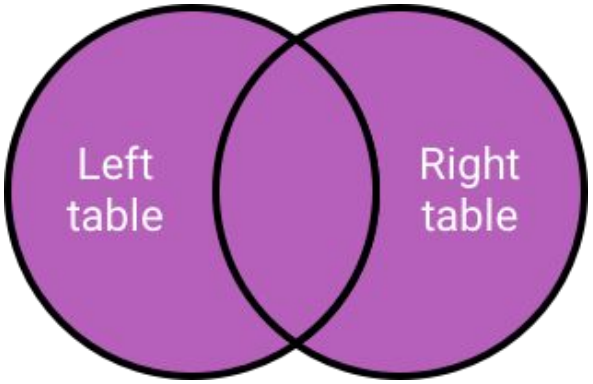
select a, b, c from <table 1> inner join <table 2> on <table1>.<key1> = <table2>.<key2>

Types of Joins – Cartesian Join or Cross Join

Analytics Vidhya
Learn everything about analytics

Cartesian Join or Cross Join

- Each row of the first table joins all the rows of second table
- It is computationally taxing
- If 1M records in one table and 1M in another – we need 10^12 computations
- Not frequently used

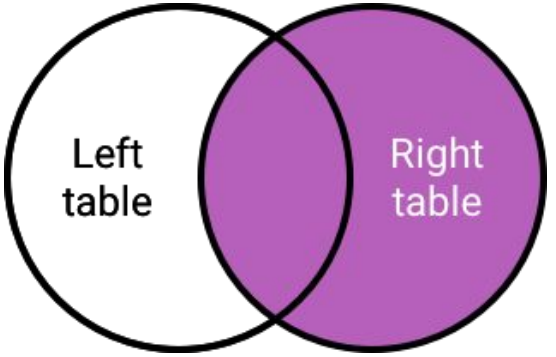


Command

select a, b, c from <table 1> cross join <table 2>

Right Join

- Take all the records from the right table and get only the corresponding (matching) records from the left table
- ONLY records matching to the right table are considered
- Unlimited number of tables can be joined one after another



Command

select a, b, c from <table 1> right join <table 2> on <table1>.<key1> = <table2>.<key2>

Self Join - Concept

Self Join

- Self Join is when we join a table to itself
- Why do we do so?
  - Creating pairwise lists
  - No other way left
  - Difficult to debug

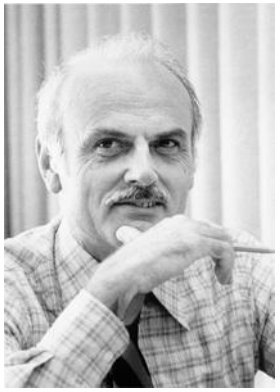
Command

select U.a, V.b from table as U <left/inner/..> join table as V on U.<key> = V.<key>

History of SQL

SQL started with IBM Researcher Edgar Codd's Research on Relational Databases

1972



Edgar Codd

- **Researcher at IBM Research Center**
- **Mathematician trained from Oxford**
- **Researching on Relational Databases**
- **Chamberlin and Boyce come up with SEQUEL (Structured English Query Language to interact with IBM System R database)**

1979

- **Trademark Issue with a Firm**
- **SEQUEL was changed to SQL**

Connecting to MySQL Server

Connecting to MySQL Server is pretty straightforward

Goto Terminal/ Command Prompt and type

```
anands-MacBook-Pro:~ analytics$ mysql -uroot -p
Enter password:
```

Data Definition Language

Commands used to

- Define the schema of database or its objects (like tables and indexes)
- Create and Modify the structure of database objects
- Examples:
  - CREATE
  - DROP
  - ALTER

Data Manipulation Language

Commands used to

- Manipulate and Select data in the database
- Examples:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

Data Control Language

Commands dealing with

- Rights, permissions and other controls of the database system
- Examples:
  - GRANT
  - REVOKE

Exploring databases

Here we explore some simple commands. Note that all commands end with ; or \G in MySQL

Show all databases

```
mysql> show databases;
```

Work with a particular database

```
mysql> use <database_name>;
```

Get help about commands

```
mysql> help;
```

Get topicwise help

```
mysql> help contents;
```

```
mysql> help Data Manipulation;
```

Creating Tables

Here we explore some simple commands. Note that all commands end with ; or \G in MySQL

Show all databases

```
mysql> show databases;
```

Work with a particular database

```
mysql> use <database_name>;
```

Get help about commands

```
mysql> help;
```

Get topicwise help

```
mysql> help contents;

mysql> help Data Manipulation;
```

Datatypes in MySQL

We have listed the most commonly used datatypes here. There are a lot more, to learn more: Refer to https://dev.mysql.com/doc/refman/8.0/en/data-types.html

Most Popular

- int(10)
- varchar(255)
- text
- TIMESTAMP
- ENUM ('Choice1', 'Choice2', …)

Not so common

- FLOAT
- DECIMAL
- BLOB
- TINYBLOB
- MEDIUMBLOB
- BIGINT
- SMALLINT
- TINYINT
- DATE
- TIME
- SET
- DOUBLE
- CHAR

NULL vs NOT NULL

Some fields we can keep optional – Others are Mandatory

Difference between NULL and NOT NULL Columns/ Fields

- A column which has NOT NULL constraint means it is mandatory to put some value for the column while inserting the row
- A column which has NULL constraint means its ok to give NULL value – a special value which means blank
- This is defined in the structure of the table