

## Database Keys and Indexes in PostgreSQL

### Automatic increment variable

In PostgreSQL, if you want to create an automatic increment variable, you could simply use **"SERIAL"** command.

**"PRIMARY KEY"** is using for building index for the given field, which allow database quickly to search for it.

**"UNIQUE"** is using for logical key.

## AUTO\_INCREMENT

Often as we make multiple tables and need to JOIN them together we need an integer primary key for each row so we can efficiently add a reference to a row in some other table as a foreign key.

```
DROP TABLE users;  
  
CREATE TABLE users (  
    id SERIAL,  
    name VARCHAR(128),  
    email VARCHAR(128) UNIQUE,  
    PRIMARY KEY(id)  
);
```

### INDEX

Index is basically the shortcuts for every record, there are two types of indexes for common use, **Trees and hashes.**

## Indexes

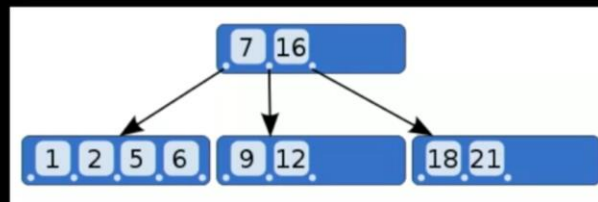
- As a table gets large (they always do), scanning all the data to find a single row becomes very costly
- When drchuck@gmail.com logs into Twitter, they must find my password amongst 500 million users
- There are techniques to greatly shorten the scan as long as you create data structures and maintain those structures - like shortcuts
- Hashes or Trees are the most common

## B-Trees

The basic idea about B-tree is, sort all data, and use tree index to represent different block of data range. When we look for specific data, we first look at where is the range contains that data.

# B-Trees

*A B-tree is a tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic amortized time. The B-tree is optimized for systems that read and write large blocks of data. It is commonly used in databases and file systems.*



<http://en.wikipedia.org/wiki/B-tree>

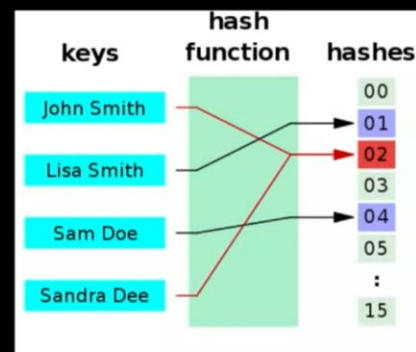
## Hashes

Hashes are calculation. In the following example, every letter are represented by numbers, and then do the calculation (MD5, SHA1 algorithm) to find the corresponded index.

**Hashes is only good for exact match** (e.g., Primary keys, unique columns), if we want to do prefix matching, do use it.

# Hashes

*A hash function is any algorithm or subroutine that maps large data sets to smaller data sets, called keys. For example, a single integer can serve as an index to an array (cf. associative array). The values returned by a hash function are called hash values, hash codes, hash sums, checksums, or simply hashes. Hash functions are mostly used to accelerate table lookup or data comparison tasks such as finding items in a database...*



## Summary

- SQL allows us to describe the shape of data to be stored and give many hints to the database engine as to how we will be accessing or using the data.
- SQL is a language that provides us operations to Create, Read, Update, and Delete (CRUD) our data in a database.