

# Indexing...

PERFORMANCE

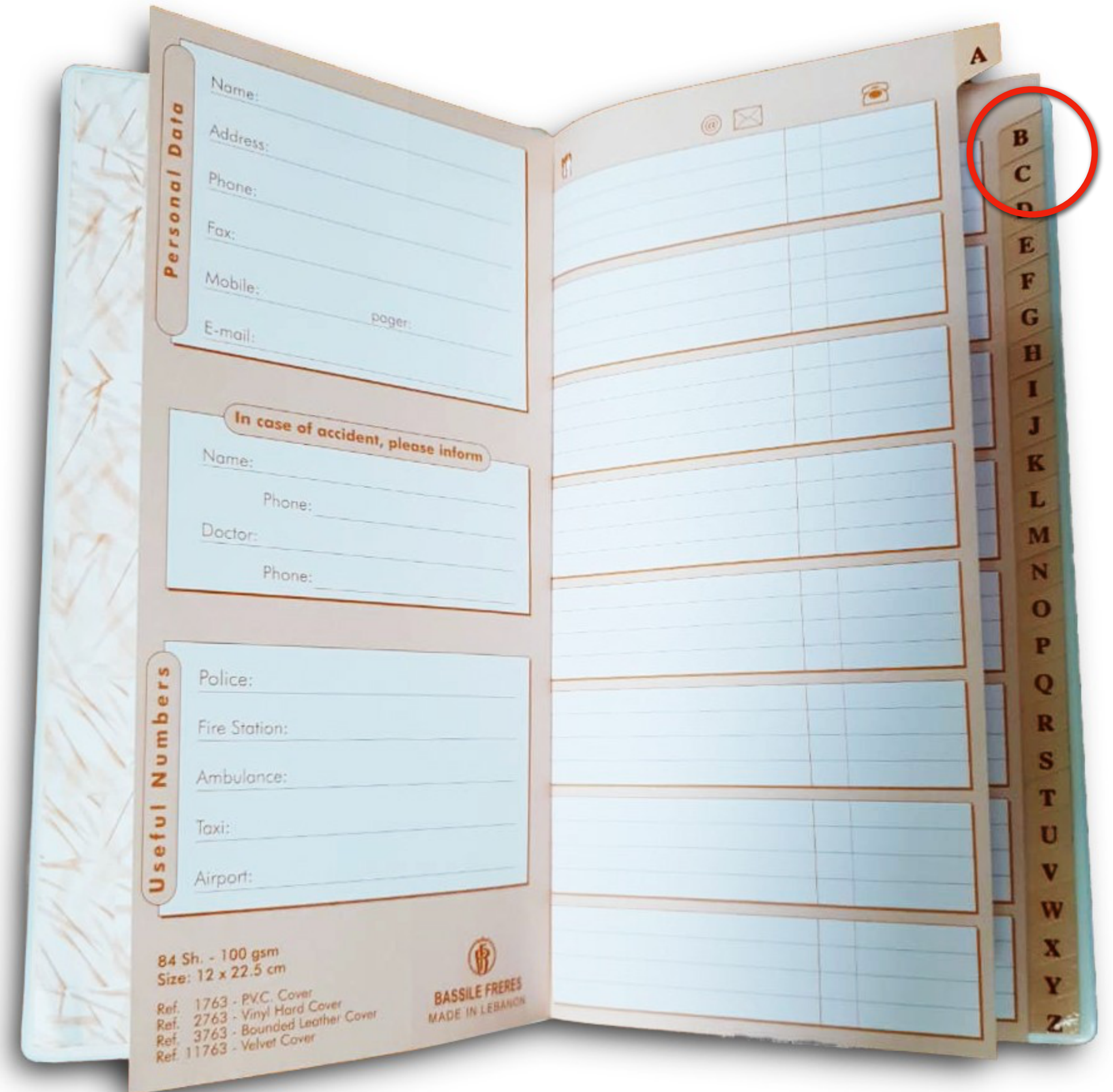
The RDBMS **BOOSTER** 

DATABASE FOR SOFTWARE DEVELOPERS

Md. Emran Hasan  
Anis Uddin Ahmad

INDEX		
SR NO.	CONTENTS	PAGE NO.
1	CHAPTER 1 INTRODUCTION	1
1.1	Abstract	1
1.2	Company Introduction	1
1.3	Project Profile	2
1.4	Project Overview	3
1.5	Administrative & User Modules	3
1.6	Features	4
1.7	Objective	5
2	CHAPTER 2 REQUIREMENT & SPECIFICATION	6
2.1	User Interface Requirement	6
2.2	Data Base Requirements	6
2.3	Proposed System	7
2.4	System Analysis	8
2.5	Tools/Platform	8
3	CHAPTER 3 COMPLETE STRUCTURE OF PROJECT	10
3.1	Entity-Relationship (E-R) Diagram	10
3.2	Data Flow Diagram (DFD)	11
3.3	Searching Process Diagram	19

Book/Report Index



Telephone Index Book



## ability

① **ability** /ə'bilɪti/ *noun* 1. the force or capacity to do something ○ *She has many abilities but singing isn't one of them.* (NOTE: The plural in this meaning is **abilities**.) □ **I'll do it to the best of my ability** I'll do it as well as I can 2. the fact of being clever ○ *a person of great or outstanding ability*

**abject** /'æbdʒekt/ *adj (formal)* 1. very bad ○ *abject poverty* 2. making you feel ashamed ○ *an abject apology* ○ *abject terror*

**ablaze** /ə'bleɪz/ *adv* 1. on fire ○ *Thirty hectares of trees were ablaze.* 2. shining brightly ○ *At midnight the house was still ablaze with lights.*

① **able** /'eɪb(ə)l/ *adj* 1. (NOTE: In this sense, **able** is only used with **to** and a verb.) □ **to be able to do something** to be capable of something or have the chance to do something ○ *They weren't able to find the house.* □ **will you be able to come to the meeting?** can you come to the meeting? 2. being strong enough or clever enough to do something ○ *He's a very able general.*

**able-bodied** /'eɪb(ə)l 'bɒdɪd/ *adj* fit and healthy

**ably** /'eɪbli/ *adv* in a very competent or efficient way. Synonym **capably**

**abnormal** /æb'nɔ:m(ə)l/ *adj* not normal.

2

## abrasive

**abort** /ə'bo:t/ *verb* 1. to stop something taking place 2. to perform an abortion on a foetus 3. (of a woman) to have an abortion or miscarriage

**abortion** /ə'bo:tʃ(ə)n/ *noun* the ending of a woman's pregnancy before a live infant can be born

**abortive** /ə'bo:tɪv/ *adj* attempted without success. Synonym **unsuccessful**. Antonym **successful**

**abound** /ə'baʊnd/ *verb* □ **to abound in or with** to be full of something (formal) ○ *The forests abound in game.*

① **about** /ə'baʊt/ *prep* 1. referring to something ○ *He told me all about his operation.* ○ *What do you want to speak to the doctor about?* 2. □ **to be about to do something** to be going to do something very soon ○ *We were about to go home when you arrived.* 3. approximately ○ *I've been waiting for about four hours.* ○ *She's only about fifteen years old.* □ **how about, what about** what do you think about (informal) ○ *We can't find a new chairperson for the club – what about Sarah?* □ **how about a cup of tea?** would you like a cup of tea? □ **while you're about it** at the same time as the thing you are doing ○ *While you're about it, can you post this letter?* □ **adv** in various places ○ *There were papers*

## Dictionary

- Natural, Alphabetical order
- Hint of word range



# What is an Index?

A database index is like an index in a book.  
It helps MySQL find data faster.

# Index in Database

## How it works?

Table:products

id	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
19	Super Camera	Lorem ipsum...	11	206.38	98	physical
20	Eco Watch	Lorem ipsum...	10	158.72	30	physical

# Index in Database

## How it works?

Table:products						Index:shipmen
id	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
19	Super Camera	Lorem ipsum...	11	206.38	98	physical
20	Eco Watch	Lorem ipsum...	10	158.72	30	physical

# Index in Database

## How it works?

Index:shipment

shipment_type
physical
digital
physical
physical
digital
physical
physical
digital
physical
physical

Table:products

id	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
19	Super Camera	Lorem ipsum...	11	206.38	98	physical
20	Eco Watch	Lorem ipsum...	10	158.72	30	physical

# Index in Database

## How it works?

Index:shipment	
shipment_type	Ref
physical	
digital	
physical	
physical	
digital	
physical	
physical	
digital	
physical	
physical	

Table:products						
id	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
19	Super Camera	Lorem ipsum...	11	206.38	98	physical
20	Eco Watch	Lorem ipsum...	10	158.72	30	physical



# Index in Database

## How it works?

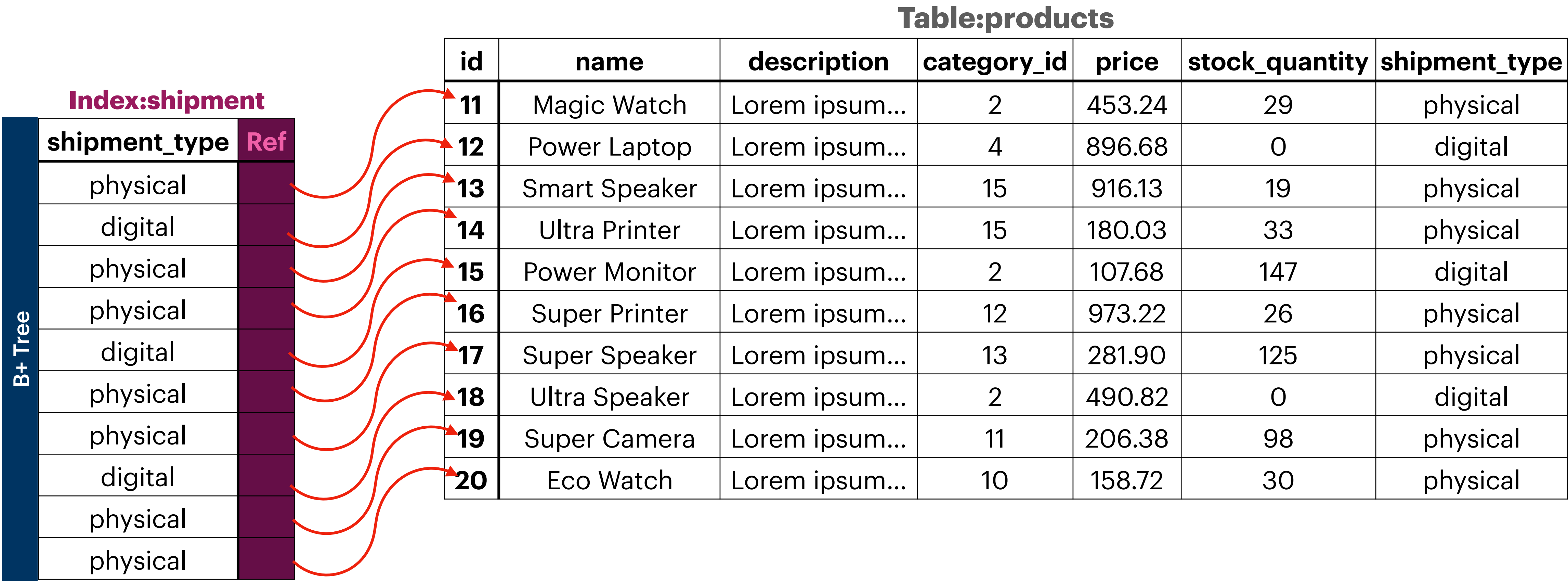
Index:shipment		
B+ Tree	shipment_type	Ref
	physical	
	digital	
	physical	
	physical	
	digital	
	physical	
	physical	
	digital	
	physical	
	physical	

Table:products

id	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
19	Super Camera	Lorem ipsum...	11	206.38	98	physical
20	Eco Watch	Lorem ipsum...	10	158.72	30	physical

# Index in Database

## How it works?





# Why Use Indexes?

- ☑ Significantly speeds up data retrieval operations.
- ☑ Improve performance of Joining, searching and analytical operations.
- ☑ Essential for enhancing performance on large databases.

# Types of Index

**PRIMARY INDEX**

**SECONDARY INDEX**



# Types of Index

**PRIMARY INDEX**

**SECONDARY INDEX**

# Types of Index

**PRIMARY INDEX**

**SECONDARY INDEX**

**UNIQUE INDEX**



# Types of Index

**PRIMARY INDEX**

**SECONDARY INDEX**

**UNIQUE INDEX**

**Non-UNIQUE INDEX**

# Types of Index

**PRIMARY INDEX**

**SECONDARY INDEX**

**UNIQUE INDEX**

**Non-UNIQUE INDEX**

**FUNCTIONAL INDEX**



# Types of Index

**PRIMARY INDEX**

**SECONDARY INDEX**

**UNIQUE INDEX**

**Non-UNIQUE INDEX**

**FUNCTIONAL INDEX**

**FULLTEXT INDEX**

# PRIMARY KEY

## *Always...*

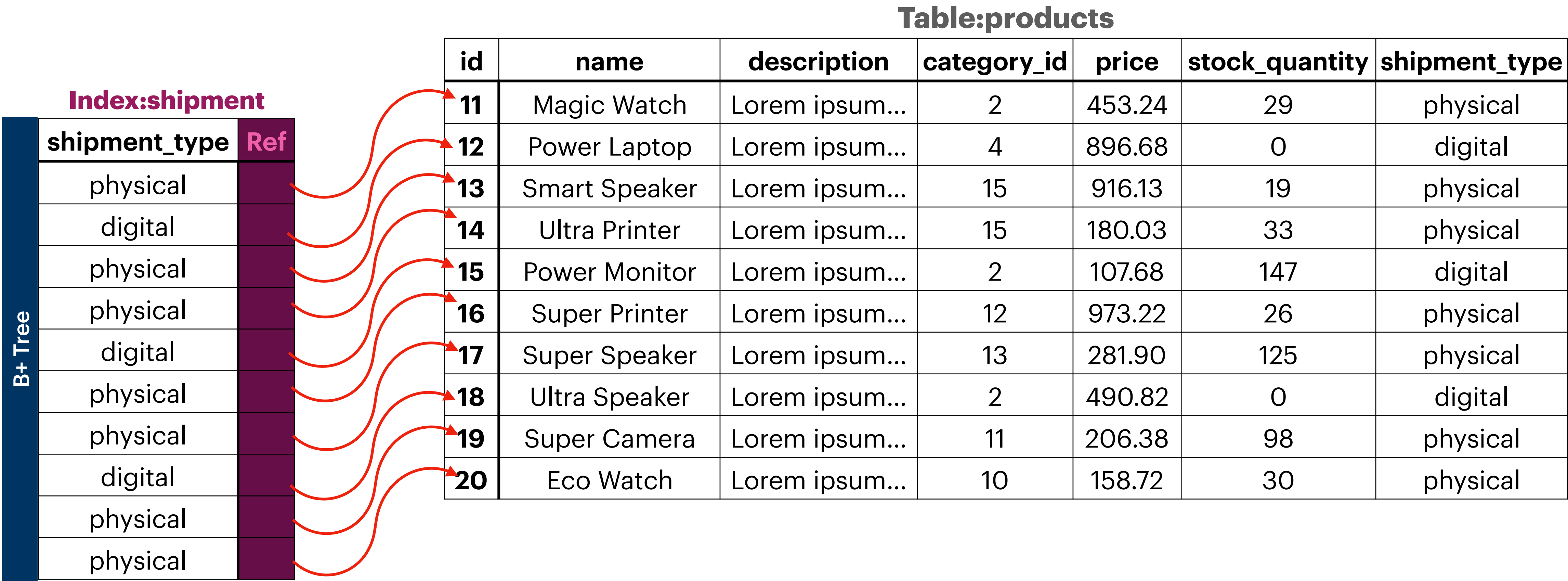
- ☑ A Unique index (always)
- ☑ NOT NULL (always)
- ☑ A table will have it
- ☑ One per table
- ☑ Data actually stored in this tree

# Secondary KEY

- ☑ An Index that is not the primary key
- ☑ Any number of Secondary key may exists per table
- ☑ Always Refers back to the original record (by PK)



# Secondary KEY



# Secondary KEY

Connected with PRIMARY KEY

Table:products											
Index:shipment											
B+ Tree	shipment_type	Ref	id	name	description	category_id	price	stock_quantity	shipment_type		
	physical	11	11	Magic Watch	Lorem ipsum...	2	453.24	29	physical		
	digital	12	12	Power Laptop	Lorem ipsum...	4	896.68	0	digital		
	physical	13	13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical		
	physical	14	14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical		
	physical	13	15	Power Monitor	Lorem ipsum...	2	107.68	147	digital		
	physical	14	16	Super Printer	Lorem ipsum...	12	973.22	26	physical		
	digital	15	17	Super Speaker	Lorem ipsum...	13	281.90	125	physical		
	physical	16	18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital		
	physical	17	19	Super Camera	Lorem ipsum...	11	206.38	98	physical		
	digital	18	20	Eco Watch	Lorem ipsum...	10	158.72	30	physical		
	physical	19									
	physical	20									

# Secondary KEY

Connected with PRIMARY KEY

Index:shipment			Table:products						
B+ Tree	shipment_type	Ref	id	name	description	category_id	price	stock_quantity	shipment_type
	physical	11	11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
	digital	12	12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
	physical	13	13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
	physical	14	14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
	digital	15	15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
	physical	16	16	Super Printer	Lorem ipsum...	12	973.22	26	physical
	digital	15	17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
	physical	16	18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
	physical	17	19	Super Camera	Lorem ipsum...	11	206.38	98	physical
	digital	18	20	Eco Watch	Lorem ipsum...	10	158.72	30	physical
	physical	19							
	physical	20							

How this logic works with PRIMARY KEY?

# PRIMARY KEY

Data actually stored in this tree

Index		Table:products				
Ref	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speaker	Lorem ipsum...	2	490.82	0	digital
19	Super Camera	Lorem ipsum...	11	206.38	98	physical
20	Eco Watch	Lorem ipsum...	10	158.72	30	physical



# PRIMARY KEY

Data actually stored in this tree

Index		Table:products				
Ref	name	description	category_id	price	stock_quantity	shipment_type
11	Magic Watch	Lorem ipsum...	2	453.24	29	physical
12	Power Laptop	Lorem ipsum...	4	896.68	0	digital
13	Smart Speaker	Lorem ipsum...	15	916.13	19	physical
14	Ultra Printer	Lorem ipsum...	15	180.03	33	physical
15	Power Monitor	Lorem ipsum...	2	107.68	147	digital
16	Super Printer	Lorem ipsum...	12	973.22	26	physical
17	Super Speaker	Lorem ipsum...	13	281.90	125	physical
18	Ultra Speake	Lorem ipsum...	2	488.88	0	digital
19	Super Camer	Lorem ipsum...	12	100.00	0	physical
20	Eco Watch	Lorem ipsum...	2	100.00	0	physical

!!SURPRISE!!

THE TABLE ITSELF IS AN INDEX

# PRIMARY KEY

What should be the Data-Type?

- ☑ CHAR/VARCHAR column?
- ☑ UUID / ULID?
- ☑ INT / BIGINT
- ☑ INT / BIGINT with UNSIGNED?

# PRIMARY KEY

## What should be the Data-Type?

table

id	name	X	Y	Z

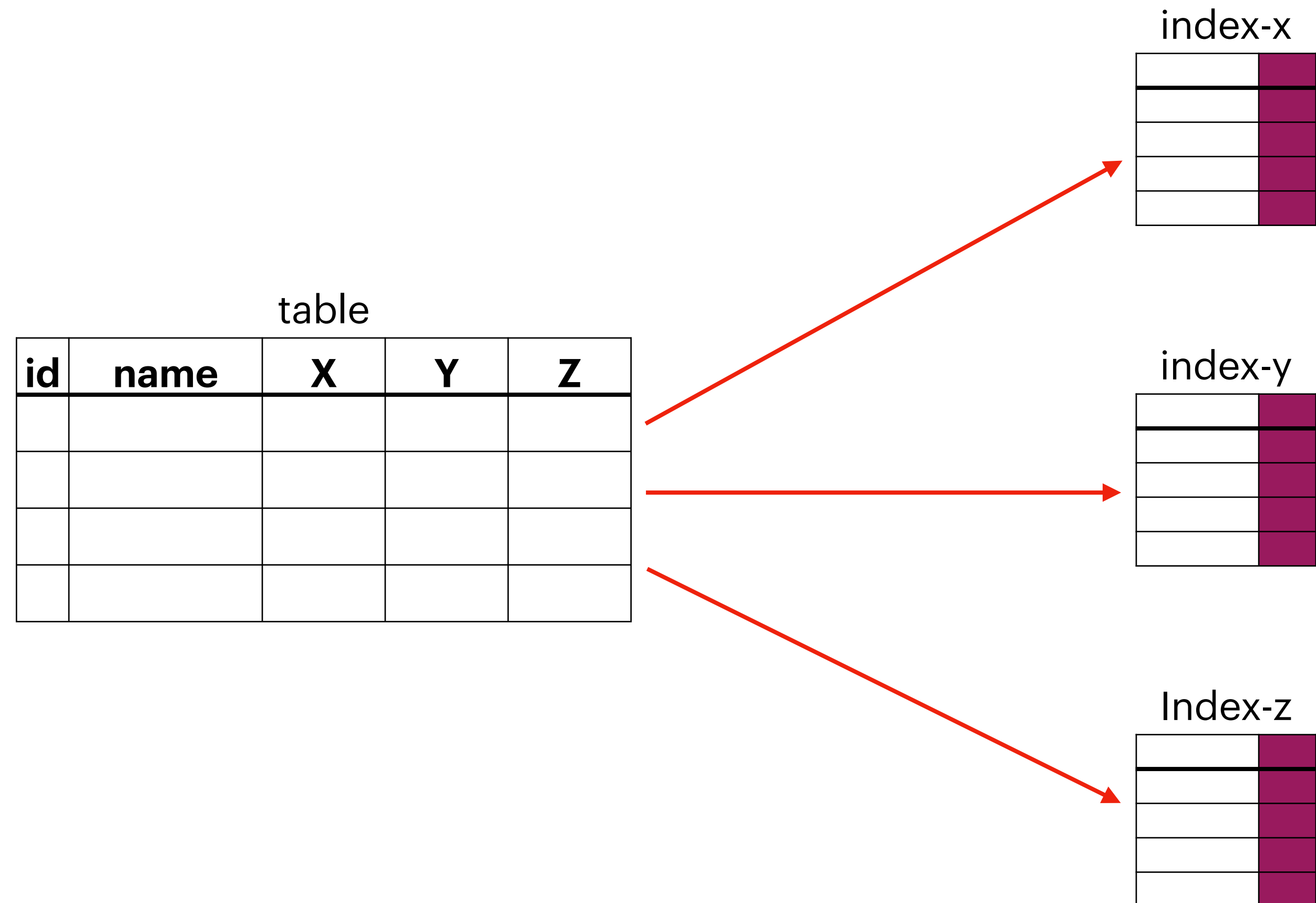
index-x


index-y


Index-z


# PRIMARY KEY

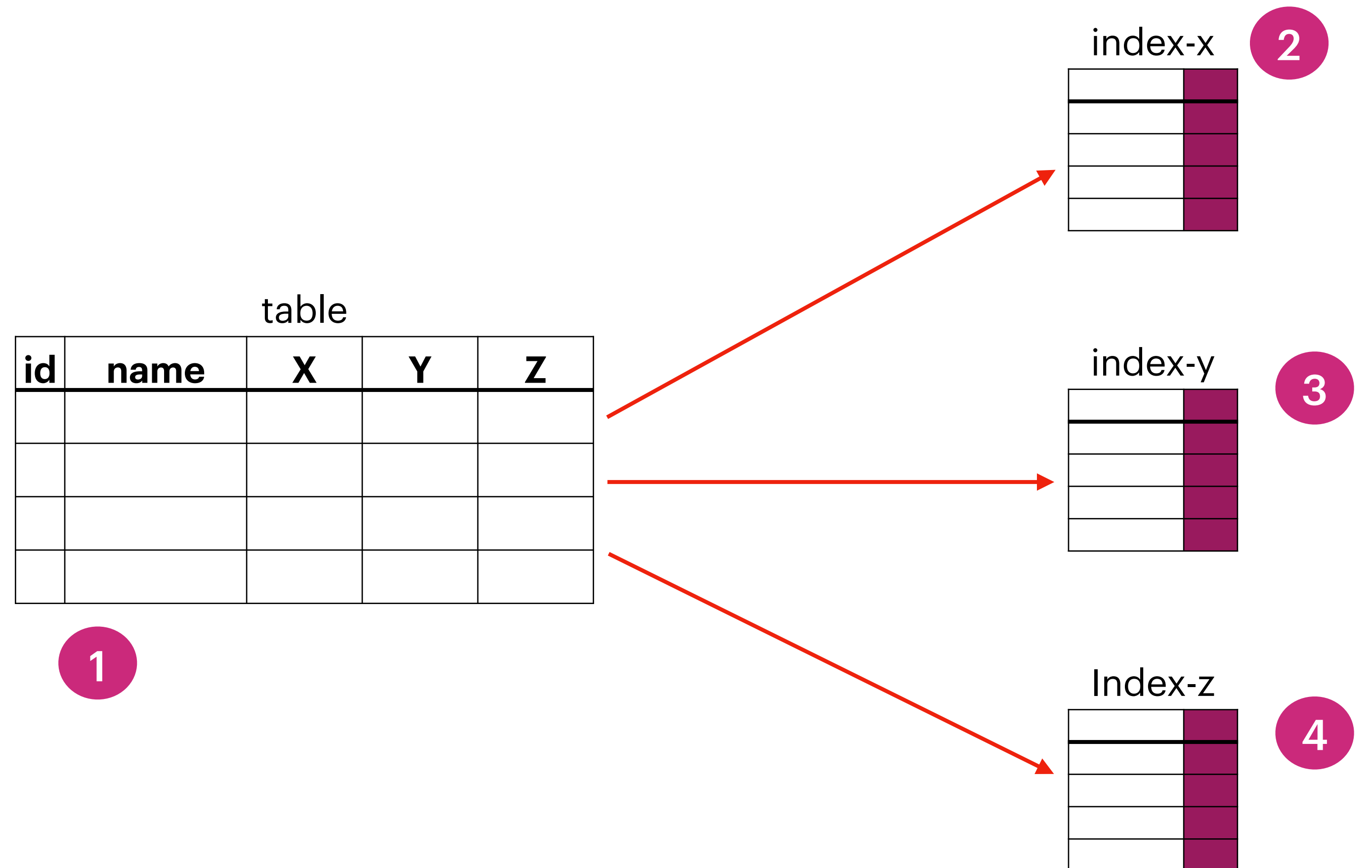
What should be the Data-Type?





# PRIMARY KEY

What should be the Data-Type?



# PRIMARY KEY

What should be the Data-Type?

► Redundancy (Storage concern)

► Cost of Update

► Re-balancing B-Tree

► Obfuscation of ID

table

id	name	X	Y	Z

1

index-x

2


index-y

3


Index-z

4


# PRIMARY KEY

What should be the Data-Type?

- ☑ CHAR/VARCHAR column?
- ☑ UUID / ULID?
- ☑ INT / BIGINT
- ☑ INT / BIGINT with UNSIGNED?



BIGINT UNSIGNED AUTO\_INCREMENT



ULID - ONLY if you have an unavoidable reason

# Planning Indexes

Where to add them?

- ☑ Observe the data-access pattern (Retrieve Queries)
- ☑ Consider all queries being run and their respective access patterns.
- ☑ Consider the entire query - includes sorting, grouping, and joining.



# Planning Indexes

It's a game of - Check and Balance



# Planning Indexes

It's a game of - Check and Balance

New Index to support query / Change query to make use of index



# Planning Indexes

It's a game of - Check and Balance

New Index to support query / Change query to make use of index

Minimum Indexes / Too Much Indexes



# Using Indexes

## Let's make and try them

— Make Index

```
ALTER TABLE products ADD INDEX (name);
```

```
ALTER TABLE products ADD INDEX (stock_quantity);
```

```
ALTER TABLE products ADD INDEX (created_at);
```

```
ALTER TABLE products ADD INDEX (shipment_type);
```

— List Indexes

```
SHOW indexes FROM products;
```

— Remove Indexes

```
DROP INDEX `index_name` ON products;
```

# Using Indexes

## Let's make and try them

```
SELECT * FROM products WHERE name = 'Smart Printer'; -- Yes (Equality)
```

```
SELECT * FROM products WHERE stock_quantity > 100; -- Yes (Range)
```

```
SELECT * FROM products WHERE stock_quantity BETWEEN 100 AND 200; -- Yes (Range)
```

```
SELECT * FROM products WHERE ORDER BY created_at LIMIT 10; -- Yes (Sorting)
```

```
SELECT * FROM products ORDER BY created_at LIMIT 10 OFFSET 10000; -- No (inefficient because of huge offset)
```

```
SELECT shipment_type, COUNT(*) FROM products GROUP BY shipment_type; -- Yes (Grouping)
```

```
SELECT * FROM products WHERE YEAR(created_at) > '2022' LIMIT 10; -- No (Using function)
```

```
SELECT p.name, c.`name` FROM products p
```

```
JOIN categories c ON p.category_id = c.id
```

```
WHERE p.name = 'Smart Printer'
```

```
AND stock_quantity > 50; -- Yes (with Join)
```



# Partial/Prefix Indexes

Let's make and try them

```
ALTER TABLE products ADD INDEX (name(4));
```

```
ALTER TABLE users ADD INDEX (email(6));
```

- ☑ Shorter index size (takes less space)
- ☑ Less cardinality

# Using Index with Wildcard

Let's make and try them

```
SELECT * FROM products WHERE name LIKE 'ultra%';
```

```
SELECT * FROM products WHERE name LIKE 'ult%printer';
```

```
SELECT * FROM products WHERE name LIKE '%printer';
```

- ☑ Index can be used with wildcard queries
- ☑ But, only characters before the first wildcard (red marked) can use index

# Functional Index

What is this?

```
-- Get the users registered in 2022  
SELECT id, name, email  
FROM users  
WHERE YEAR(`created_at`) = 2022;
```

# Functional Index

What is this?

-- Get the users registered in 2022

```
SELECT id, name, email
```

```
FROM users
```

```
WHERE YEAR(`created_at`) = 2022;
```

-- Let's make it faster with an index

```
ALTER TABLE users ADD INDEX(created_at);
```

# Functional Index

## What is this?

-- Get the users registered in 2022

```
SELECT id, name, email
```

```
FROM users
```

```
WHERE YEAR(`created_at`) = 2022;
```

-- Let's make it faster with an index

```
ALTER TABLE users ADD INDEX(created_at);
```

⚠ MySQL is not considering this index!



# Functional Index

## What is this?

```
-- Get the users registered in 2022  
SELECT id, name, email  
FROM users  
WHERE YEAR(`created_at`) = 2022;
```

```
-- Let's make it faster with an index  
ALTER TABLE users ADD INDEX(created_at);
```

⚠ MySQL is not considering this index!



# Functional Index

What is this?

- ✗ Indexes created on **fields**  
cannot be used for comparing **function output** .

# Functional Index

## What is this?

- ✗ Indexes created on **fields** cannot be used for comparing **function output** .
- ✓ We can make index on **function output** instead.

# Functional Index

## The Syntax and usages

```
ALTER TABLE users ADD INDEX joining_year ((YEAR(created_at)));
```

# Functional Index

## The Syntax and usages

```
ALTER TABLE users ADD INDEX joining_year ((YEAR(created_at)));
```

```
ALTER TABLE users ADD INDEX email_domain ((SUBSTRING(email, INSTR(email, '@') + 1)));
```

# Functional Index

## The Syntax and usages

```
ALTER TABLE users ADD INDEX joining_year ((YEAR(created_at)));
```

```
ALTER TABLE users ADD INDEX email_domain ((SUBSTRING(email, INSTR(email, '@') + 1)));
```



Notice the extra pair of parentheses



# Functional Index

## The Syntax and usages

```
ALTER TABLE users ADD INDEX joining_year ((YEAR(created_at)));
```

```
ALTER TABLE users ADD INDEX email_domain ((SUBSTRING(email, INSTR(email, '@') + 1)));
```

- ☑ Keep the comparison **exactly same** as the function used in making index
- ☑ Can be used as a part of a Composite Index
- ☑ Internally works using a **Generated column**

# Composite Index

## Let's make and try them

-- We have a query with multiple conditions

```
SELECT name, price, stock_quantity, shipment_type
FROM products
WHERE shipment_type = 'physical'
      AND `name` LIKE 'Ultra%'
      AND stock_quantity > 0;
```

-- We can make an Index including all relevant columns

```
ALTER TABLE products ADD INDEX search_q (shipment_type, name, stock_quantity);
```

# Composite Index

## General rules of using composite index

- ☑ Composite Index can be utilized for one or multiple columns
- ☑ But, can access the index only in Left-To-Right index-defining order
- ☑ Can't skip or jump index columns
- ☑ Index can be used up to the first range condition (then skip subsequent conditions, if any)

# Composite Index

## General rules of using composite index

```
ALTER TABLE products ADD INDEX multicol_idx (col_a, col_b, col_c);
```

```
WHERE col_a = 'X' AND col_b = 100 AND col_c = 'Y';
```

```
WHERE col_a = 'X';
```

```
WHERE col_a = 'X' AND col_b = 100;
```

```
WHERE col_a = 'X' AND col_c = 'Y';
```

```
WHERE col_a = 'X' AND col_b > 100 AND col_c = 'Y';
```

# Composite Index

## General rules of using composite index

```
ALTER TABLE products ADD INDEX multicol_idx (col_a, col_b, col_c);
```

```
WHERE col_a = 'X' AND col_b = 100 AND col_c = 'Y';
```

```
WHERE col_a = 'X';
```

```
WHERE col_a = 'X' AND col_b = 100;
```

```
WHERE col_a = 'X' AND col_c = 'Y';
```

```
WHERE col_a = 'X' AND col_b > 100 AND col_c = 'Y';
```

# FULLTEXT Index

## Why do we need them?

-- Let's find films with the words 'Victory' and 'Drama' and ...

```
SELECT * FROM film_text WHERE title LIKE '%Victory%'
      OR description LIKE '%Victory%'
      OR title LIKE '%Drama%'
      OR description LIKE '%Drama%'
      OR ...;
```

- Too much conditions
- Not efficient at all (cannot use index)



# FULLTEXT Index

## How to use them (basic)

```
ALTER TABLE film_text ADD FULLTEXT INDEX search_film (title, description);
```

```
SELECT *
```

```
FROM film_text
```

```
WHERE MATCH(title, description) AGAINST('Any number of keywords');
```

# FULLTEXT Index

## How to use them (basic)

```
SELECT *  
FROM film_text  
WHERE MATCH(title, description) AGAINST('Any number of keywords');
```

- ☑ Supports *InnoDB* or *MyISAM* storage engine
- ☑ Can be used with only CHAR, VARCHAR and TEXT columns
- ☑ Words with 3/4 characters and stop-words are ignored (by default)

**Questions?**