Indexing...

The RDBMS PERFORMANCE BOOSTER

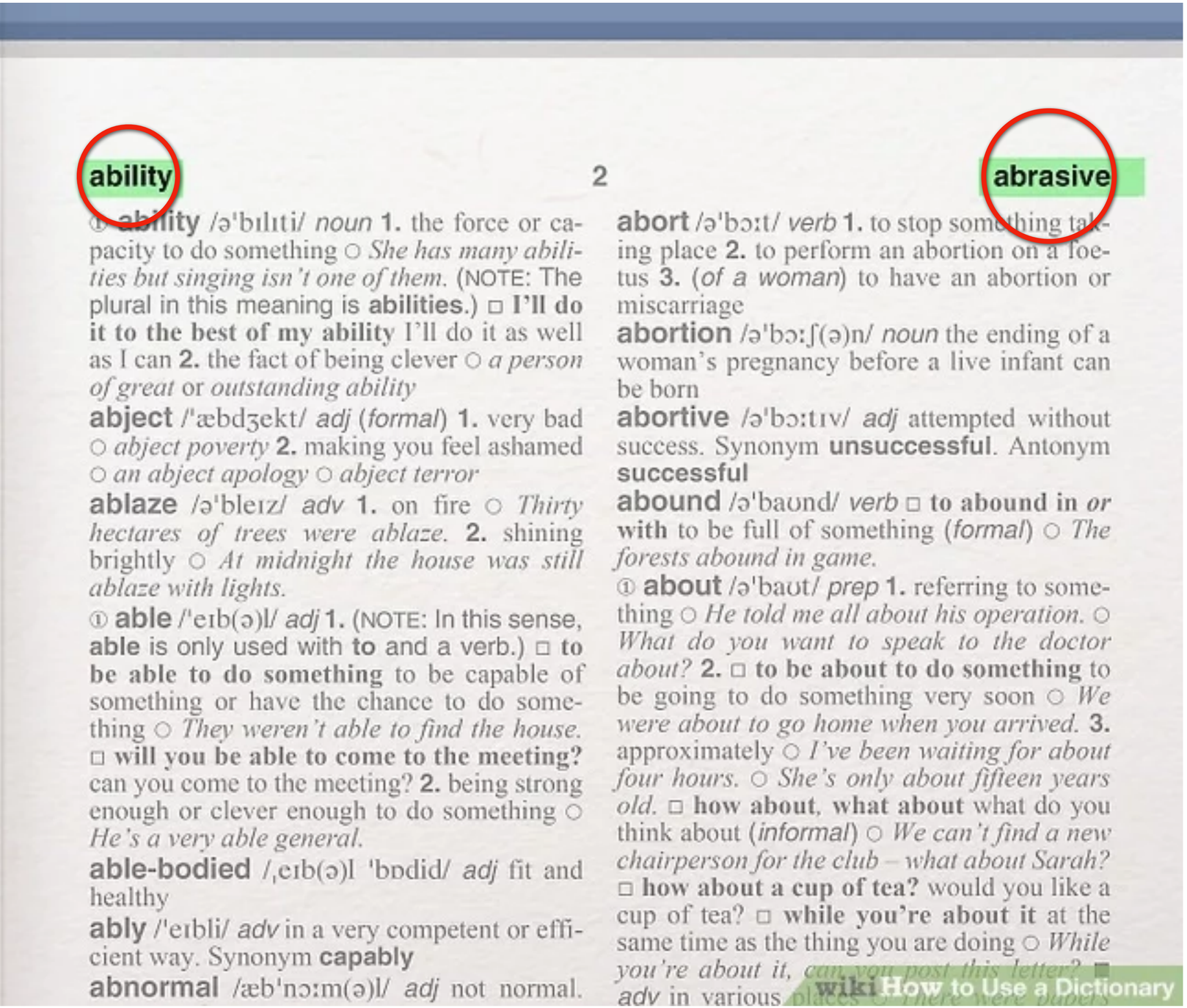
Database for Software Developers

Md. Emran Hasan
Anis Uddin Ahmad

INDEX

Book/Report Index

Telephone Index Book



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?

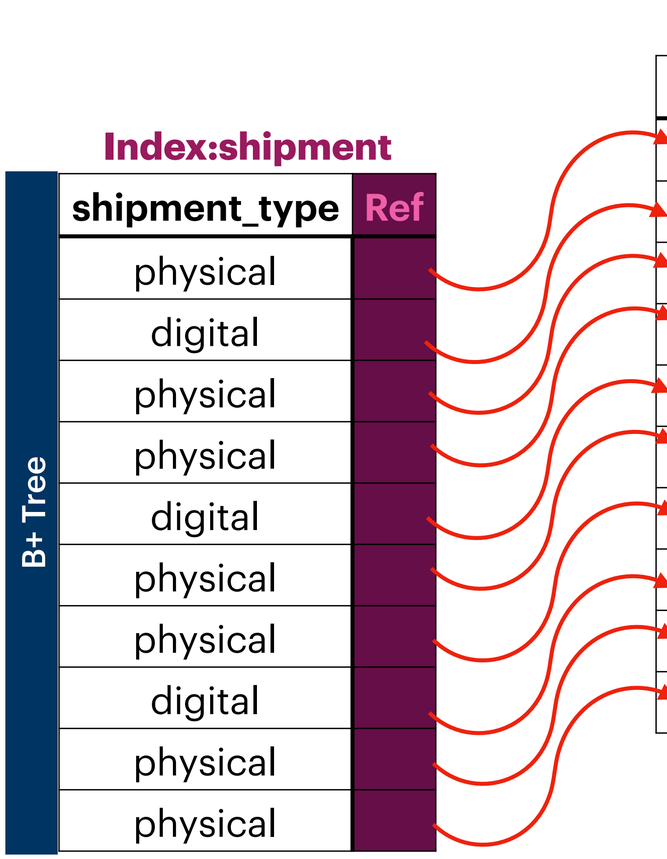


Table:products

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

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

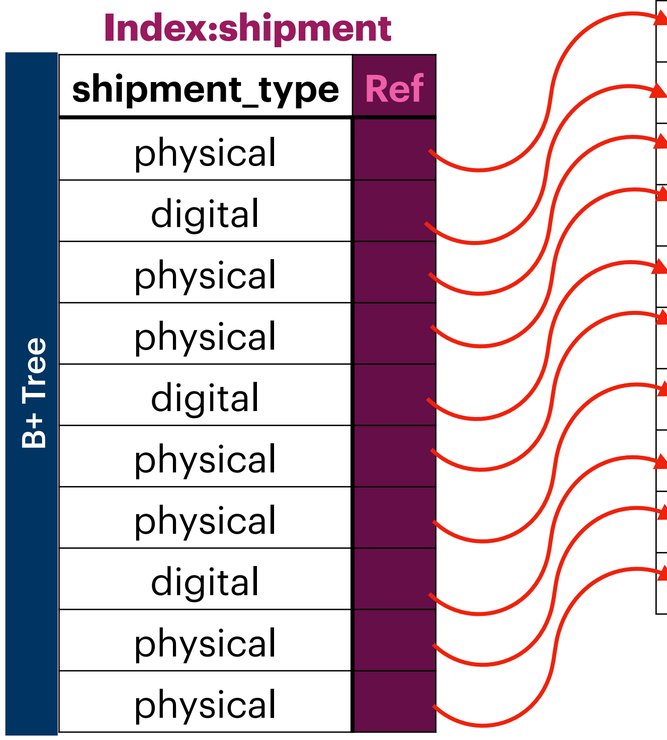


Table:products

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

Secondary KEY

Connected with PRIMARY KEY

B+ Tree

Index:shipment

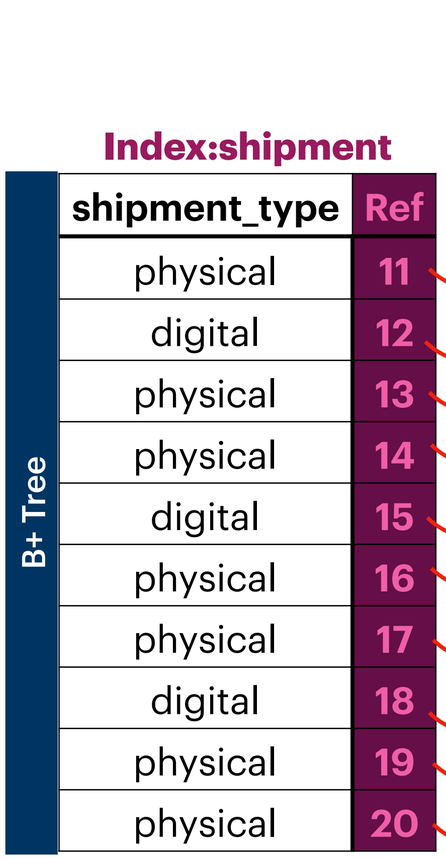
shipment_type	Ref
physical	11
digital	12
physical	13
physical	14
digital	15
physical	16
physical	17
digital	18
physical	19
physical	20

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

Secondary KEY

Connected with PRIMARY KEY



Index:shipment

shipment_type	Ref
physical	11
digital	12
physical	13
physical	14
digital	15
physical	16
physical	17
digital	18
physical	19
physical	20

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

How this logic works with PRIMARY KEY?

PRIMARY KEY

Data actually stored in this tree

Inde

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

Inde

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					igital
19	Super Camer					ysical
20	Eco Watch					ysical

!!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?

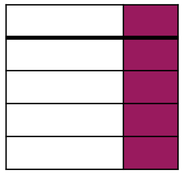
PRIMARY KEY

What should be the Data-Type?

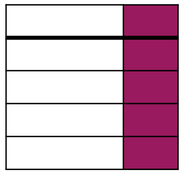
table

id	name	X	Y	Z

index-x



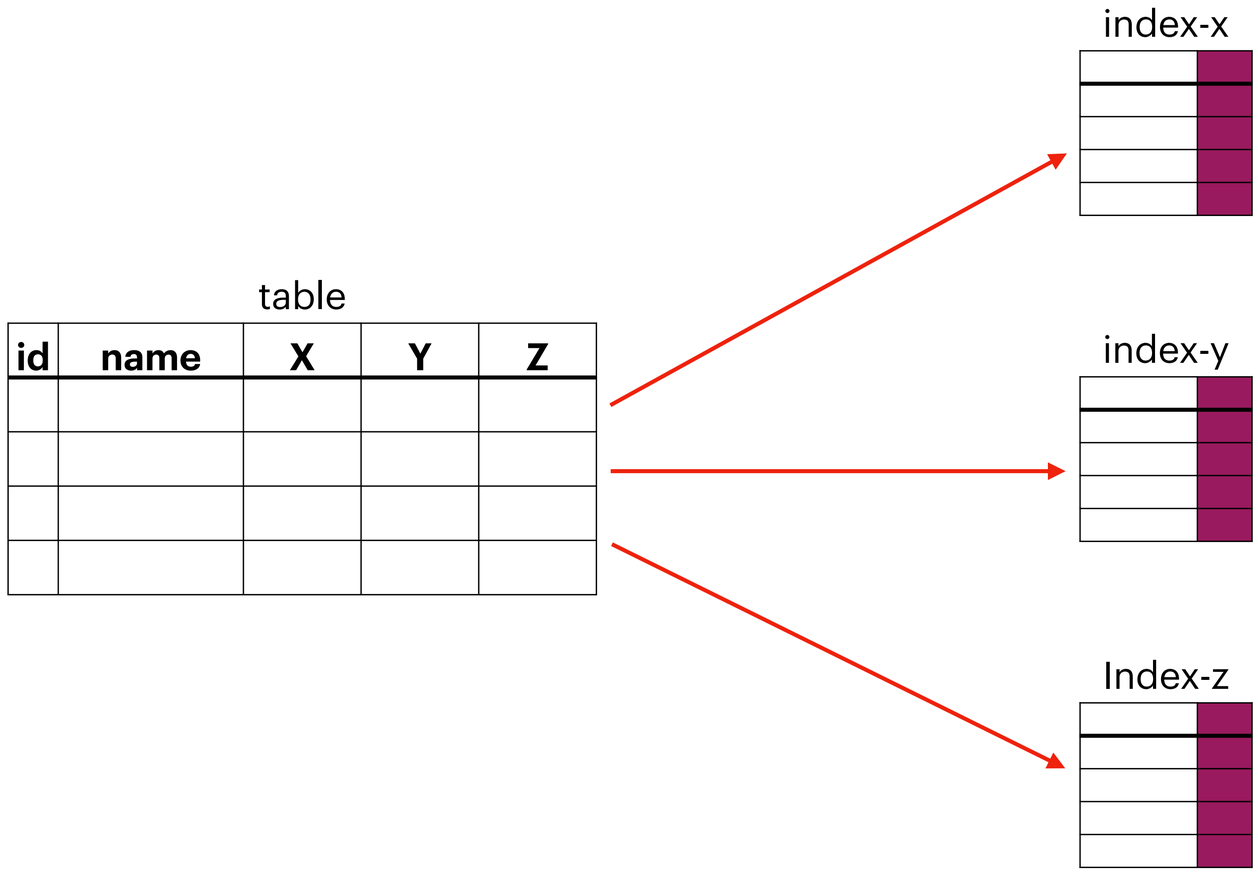
index-y



Index-z

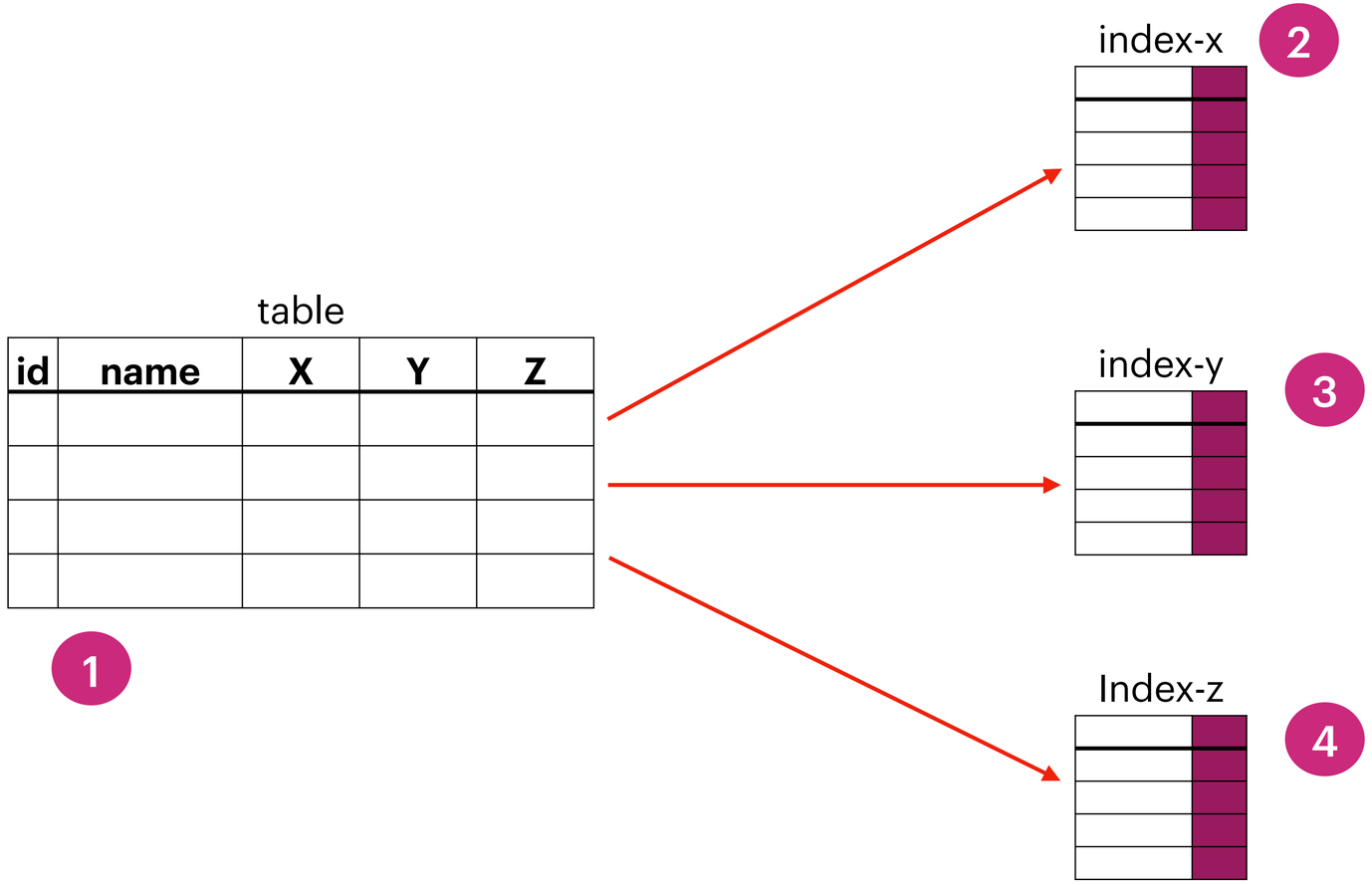
PRIMARY KEY

What should be the Data-Type?



PRIMARY KEY

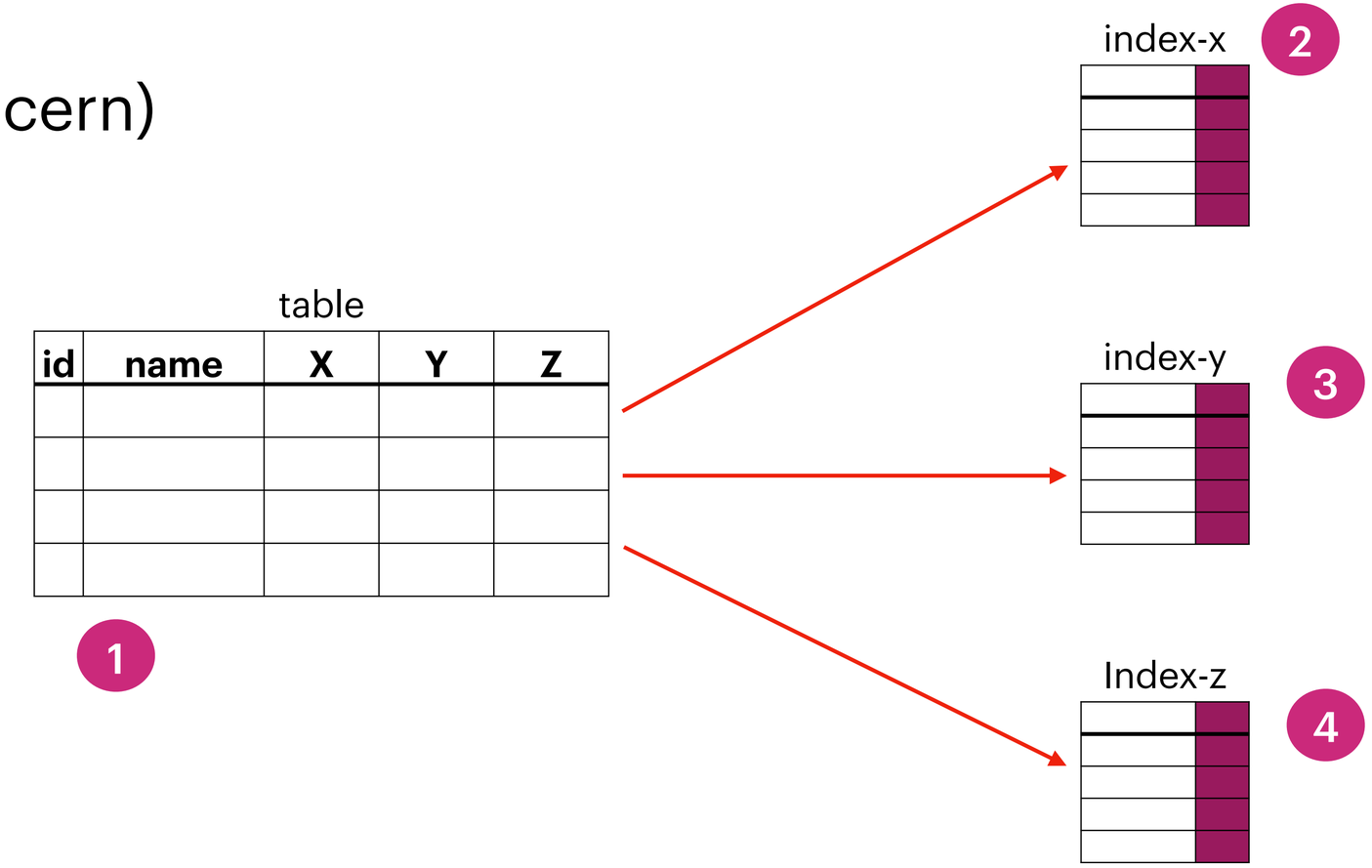
What should be the Data-Type?



PRIMARY KEY

What should be the Data-Type?

- Redundancy (Storage concern)
- Cost of Update
- Re-balancing B-Tree
- Obfuscation of ID



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

```sql
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?

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

Functional Index

What is this?

```sql
-- 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?

```sql
-- 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?

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

```sql
-- 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

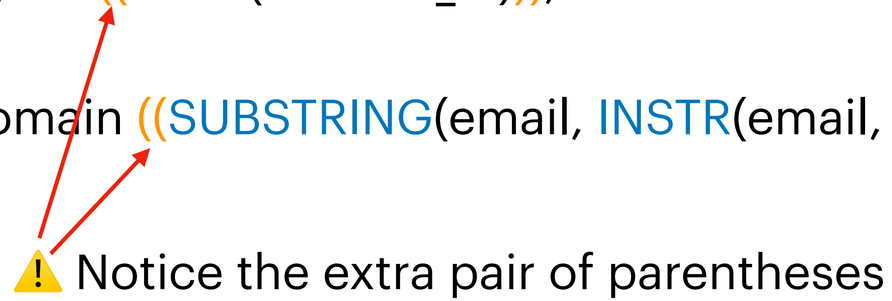
The Syntax and usages

```
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)));
```



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

```sql
-- 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;
```

```sql
-- 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 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

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

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

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

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

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

```sql
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)

```sql
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?