

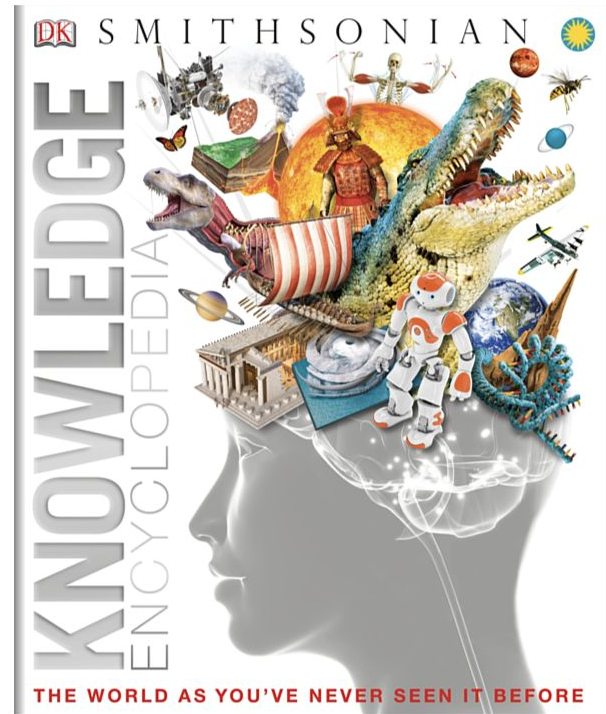
Indexes

Why use indexes?

Encyclopedias use index to refer keywords to pages with related content

Scanning for term without index requires scanning entire book

This is not an efficient task!



Why use indexes?

Encyclopedia uses index to refer keywords to pages with related content

Scanning is **$O(n)$** complexity (linear), where **n** = number of pages

If the encyclopedia has 345 pages, and the index for "snakes" shows 5 pages, we've reduced **n** from 345 to 5!

smell 109, 150, 165, **174-175**

see also noses

smoking, tobacco 179

smooth muscle 144

snakes **108**, 109, 125,
129, 131

snow 59, 62

Sobek 257

social change 299, 303,
324-235

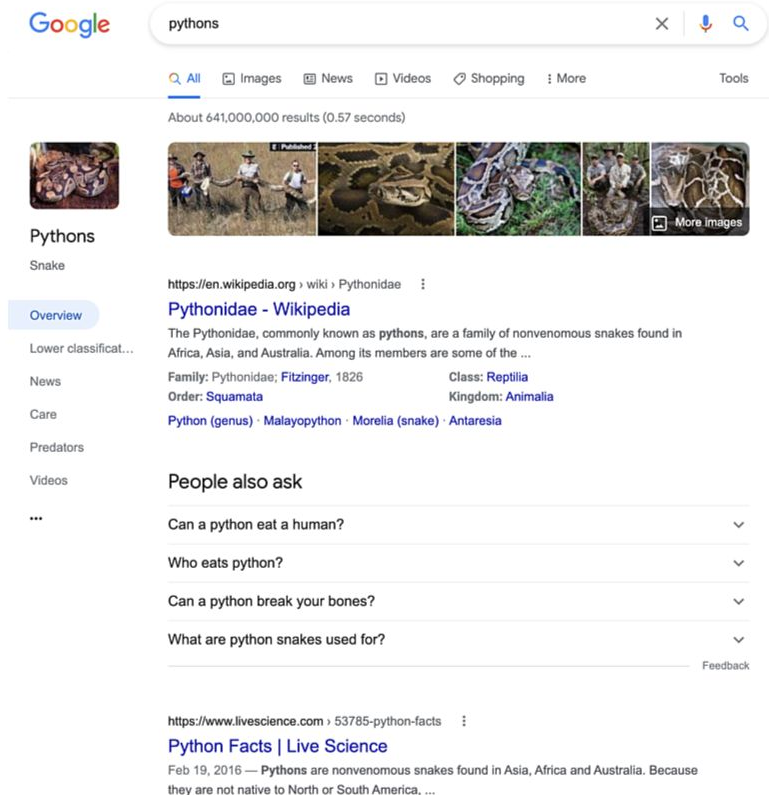
social classes 266

social networks 243, 326

Google does not scan entire internet for every search request

Instead, builds index ahead of time

Index points keywords to relevant webpages



The screenshot shows a Google search for "pythons". The search bar at the top contains the word "pythons". Below the search bar, there are tabs for "All", "Images", "News", "Videos", "Shopping", and "More". The "All" tab is selected. The search results show "About 641,000,000 results (0.57 seconds)".

On the left side, there is a vertical navigation menu with the following items: "Pythons", "Snake", "Overview", "Lower classificat...", "News", "Care", "Predators", "Videos", and "...". The "Overview" item is highlighted.

The main content area displays the search results. At the top, there is a row of five small images showing various pythons. Below the images, there is a link to "https://en.wikipedia.org/wiki/Pythonidae" with the text "Pythonidae - Wikipedia". The Wikipedia snippet reads: "The Pythonidae, commonly known as **pythons**, are a family of nonvenomous snakes found in Africa, Asia, and Australia. Among its members are some of the ...". Below the snippet, there is a table with two columns: "Family: Pythonidae: Fitzinger, 1826" and "Class: Reptilia". The "Order: Squamata" and "Kingdom: Animalia" are also listed. At the bottom of the snippet, there are links for "Python (genus)", "Malayopython", "Morelia (snake)", and "Antaresia".

Below the Wikipedia result, there is a section titled "People also ask" with four questions and their corresponding answers, each with a dropdown arrow to the right:

- Can a python eat a human?
- Who eats python?
- Can a python break your bones?
- What are python snakes used for?

At the bottom of the page, there is a link to "https://www.livescience.com/53785-python-facts" with the text "Python Facts | Live Science". The snippet reads: "Feb 19, 2016 — **Pythons** are nonvenomous snakes found in Asia, Africa and Australia. Because they are not native to North or South America, ...".

Indexing in Postgres

Improves query time for specific columns

Each index entry contains column value that can be thought of as "search term" (e.g. "snakes")

Also contains reference to its corresponding row (usually by primary key)

Indexing in Postgres

magic_markers table

id	color
1	blue
2	green
3	blue
4	red
5	blue
6	red
7	yellow

Indexing in Postgres

Index on magic_markers color column

Column Value	Record ID
blue	3
blue	1
blue	5
green	2
red	4
red	6
yellow	7

Index example use case

```
CREATE TABLE people (  
    id SERIAL PRIMARY KEY,  
    full_name TEXT NOT NULL,  
    birthdate DATE DEFAULT now()  
);
```

Assume 100,000,000 rows

Index example use case

```
SELECT full_name FROM people WHERE birthdate = '1912-06-23';  
SELECT full_name FROM people WHERE birthdate = '1903-12-28';  
SELECT full_name FROM people WHERE birthdate = '1916-04-30';
```



Scans 300,000,000 rows (no index)

```
CREATE INDEX index_name ON table_name (column_name);
```

Postgres starts building index structure

While building index, table is in read-only mode (SELECT only)

Once index is built, read-only mode is turned off

Postgres updates index as table values change

Index example use case

```
CREATE TABLE people (  
    id SERIAL PRIMARY KEY,  
    full_name TEXT NOT NULL,  
    birthdate DATE DEFAULT now()  
);
```

```
CREATE INDEX people_birthdate_index ON people (birthdate);
```

```
SELECT full_name FROM people WHERE birthdate = '1912-06-23';  
SELECT full_name FROM people WHERE birthdate = '1903-12-28';  
SELECT full_name FROM people WHERE birthdate = '1916-04-30';
```

Q: Why do indexes have same size as table but faster lookup time?

A: Data structures optimized for lookup

Index on magic_markers color column

Column Value	Record ID
blue	3
blue	1
blue	5
green	2
red	4
red	6
yellow	7

magic_markers table

[illegible]

B-Tree Index	Hash Index
Ordered	Not ordered
Used for matching with $<$, $<=$, $>=$, $>$, or $=$	Used only for matching with $=$
$O(\log n)$ lookup	$O(1)$ lookup
$O(\log n)$ insertion	$O(1)$ insertion
$O(\log n)$ deletion	$O(1)$ deletion

Q: How can we index on birthdate to optimize this query?

```
SELECT full_name FROM people WHERE birthdate = '1916-04-30';
```

B-Tree Index	Hash Index
Ordered	Not ordered
Used for matching with $<$, $<=$, $>=$, $>$, or $=$	Used only for matching with $=$
$O(\log n)$ lookup	$O(1)$ lookup
$O(\log n)$ insertion	$O(1)$ insertion
$O(\log n)$ deletion	$O(1)$ deletion

Q: How can we index on birthdate to optimize this query?

```
SELECT full_name FROM people WHERE birthdate = '1916-04-30';
```

A: Either, but Hash index is preferred

B-Tree Index	Hash Index
Ordered	Not ordered
Used for matching with $<$, $<=$, $>=$, $>$, or $=$	Used only for matching with $=$
$O(\log n)$ lookup	$O(1)$ lookup
$O(\log n)$ insertion	$O(1)$ insertion
$O(\log n)$ deletion	$O(1)$ deletion

Q: How can we index on birthdate to optimize this query?

```
SELECT full_name FROM people WHERE birthdate > '1916-04-30';
```

A: Must use B-Tree index

Indexes: Final tips

Common way to enhance database performance

Tradeoffs:

1. Indexes take up a lot of space
2. Indexes must stay up to date

Rule of thumb: Only index columns that are queried frequently

PostgreSQL automatically creates an index for each table's primary key