Indexes

Suan Lee

Indexes

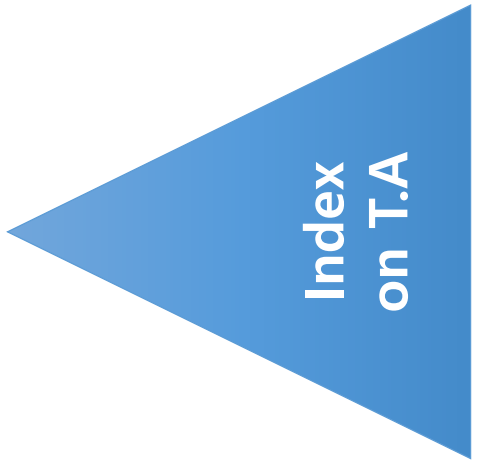
- Primary mechanism to get improved performance on a database
- Persistent data structure, stored in database
- Many interesting implementation issues
  - But we are focusing on user/application perspective

Functionality

T

	A	B	C
1	Cat	2	...
2	Dog	5	...
3	Cow	1	...
4	Dog	9	...
5	Cat	2	...
6	Cat	8	...
7	Cow	6	...
	...	...	...

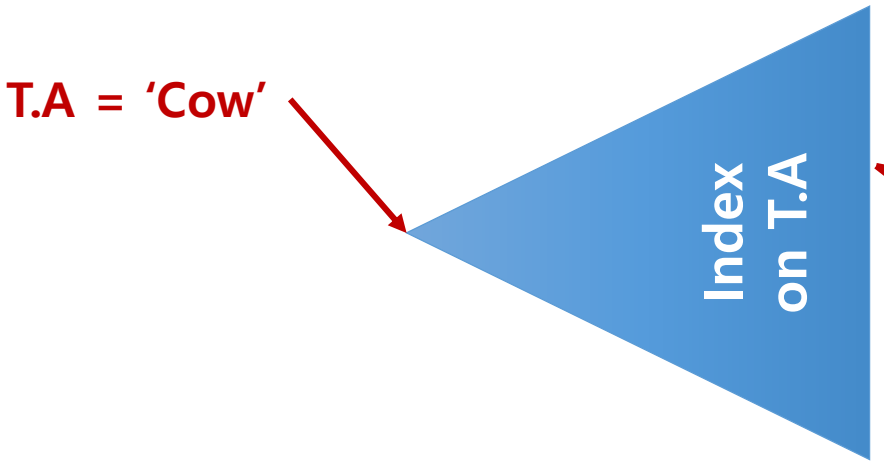
Functionality



T

	A	B	C
1	Cat	2	...
2	Dog	5	...
3	Cow	1	...
4	Dog	9	...
5	Cat	2	...
6	Cat	8	...
7	Cow	6	...
	...	...	...

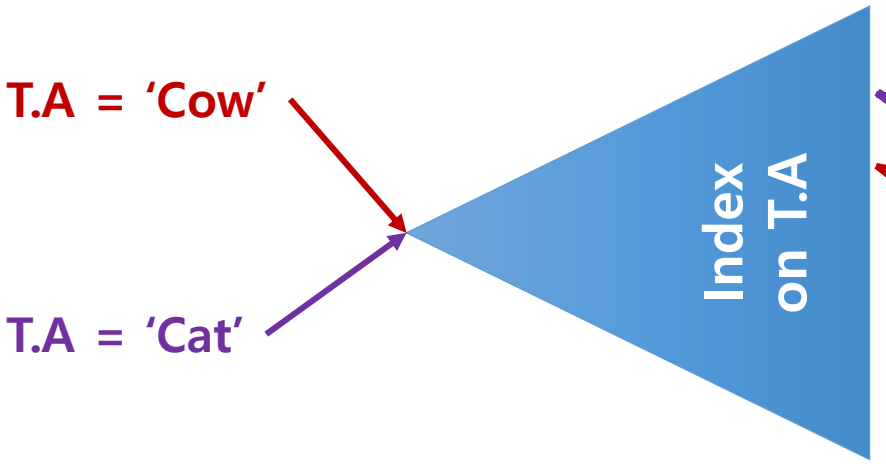
Functionality



T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

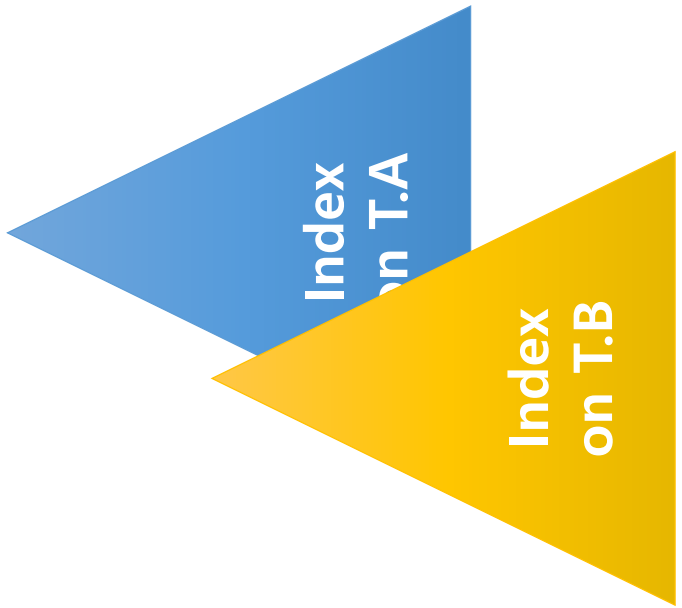
Functionality



T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

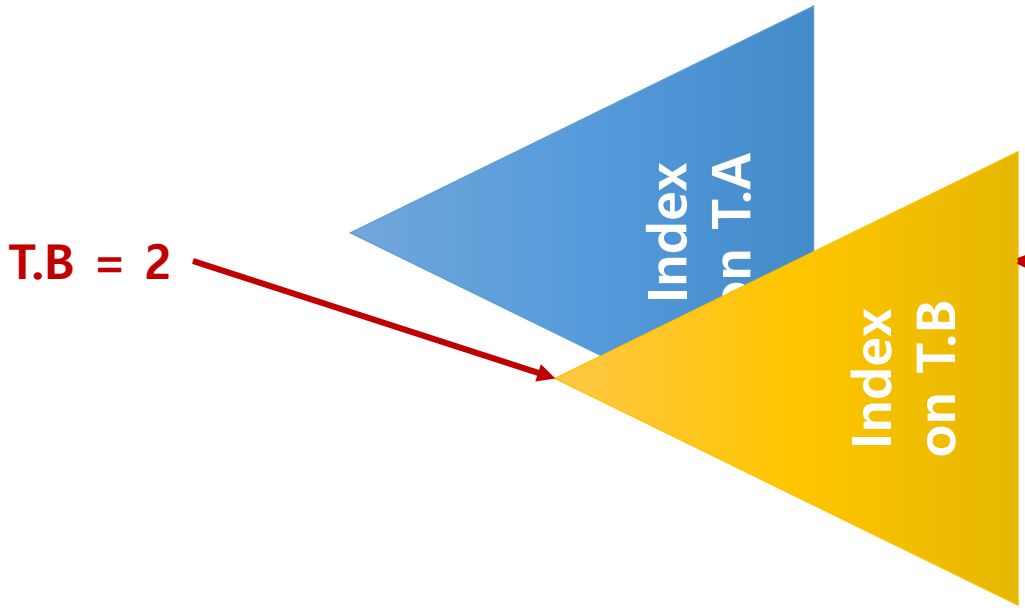
Functionality



T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

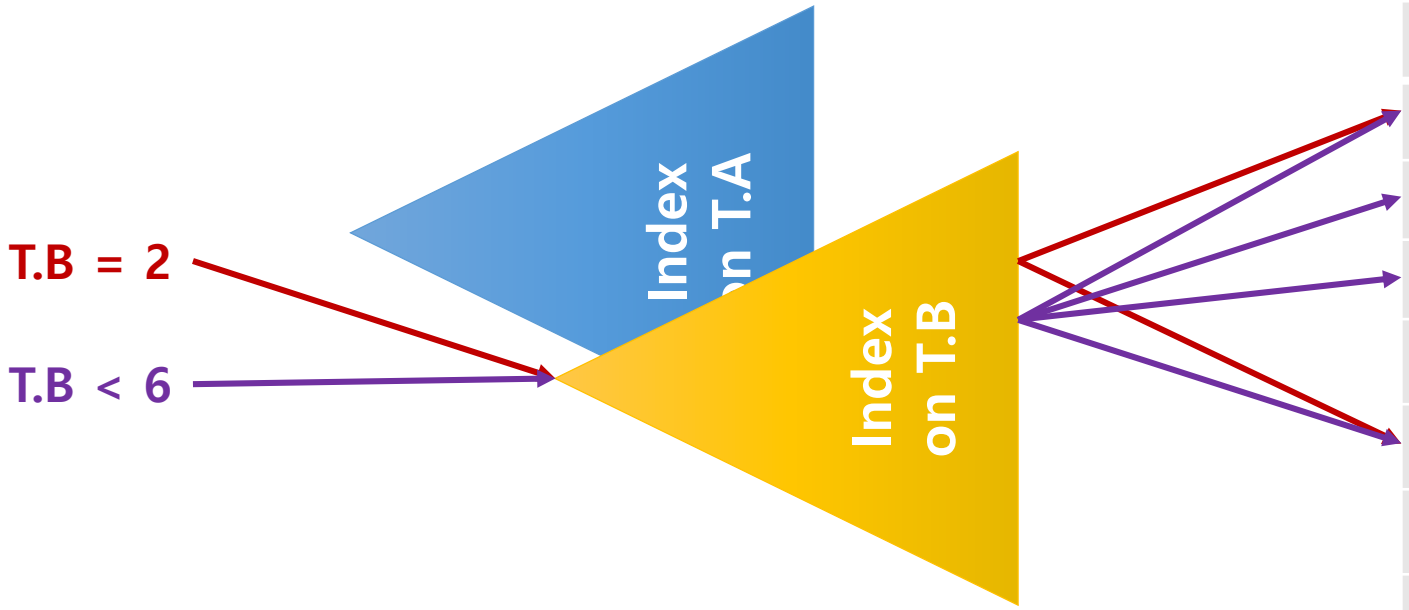
Functionality



T

	A	B	C
1	Cat	2	...
2	Dog	5	...
3	Cow	1	...
4	Dog	9	...
5	Cat	2	...
6	Cat	8	...
7	Cow	6	...
	...	...	...

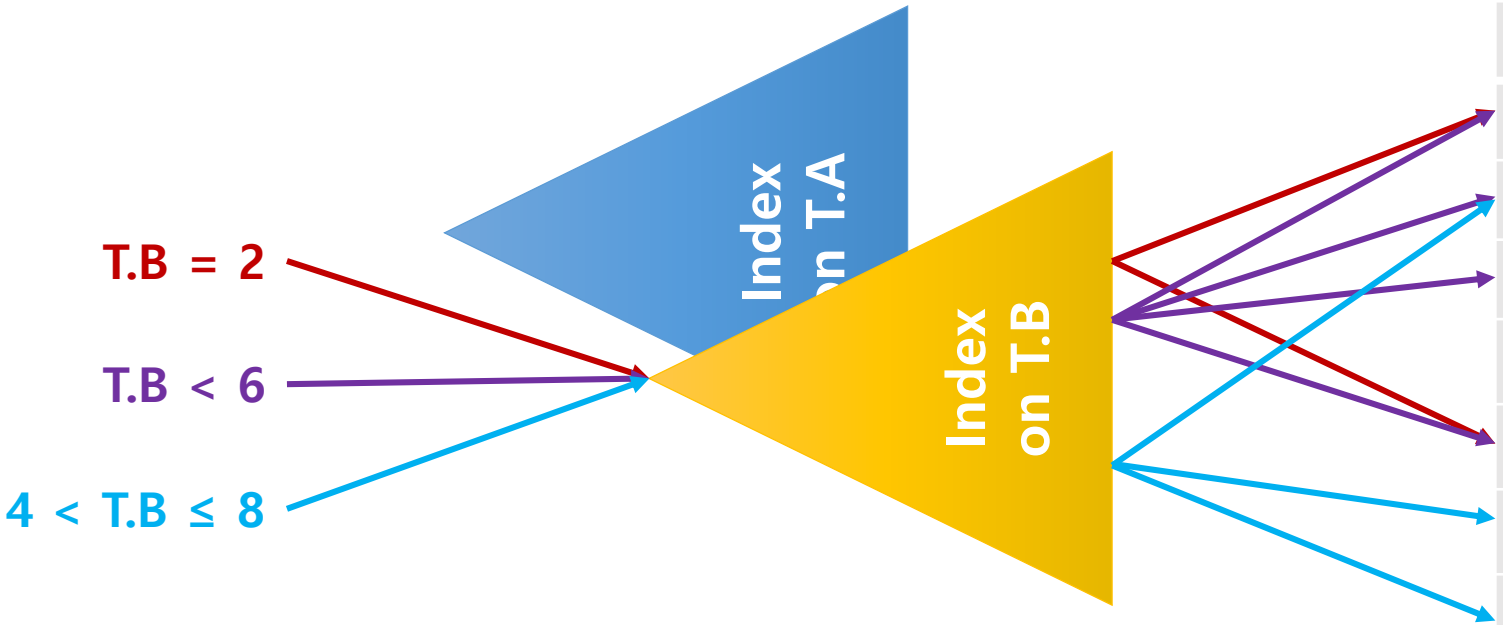
Functionality



T

	A	B	C
1	Cat	2	...
2	Dog	5	...
3	Cow	1	...
4	Dog	9	...
5	Cat	2	...
6	Cat	8	...
7	Cow	6	...
	...	...	...

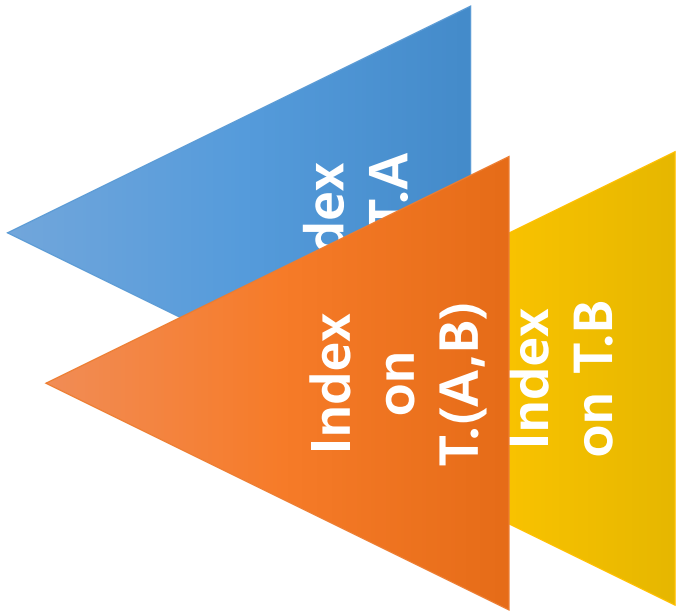
Functionality



T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

Functionality

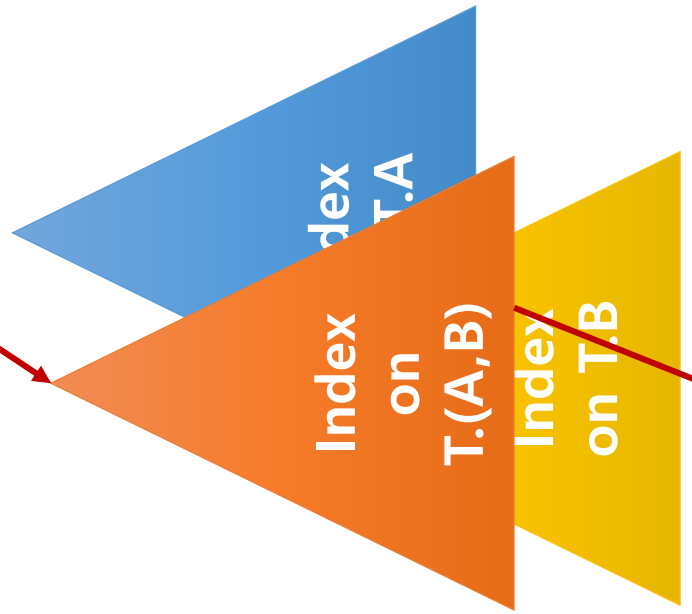


T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

Functionality

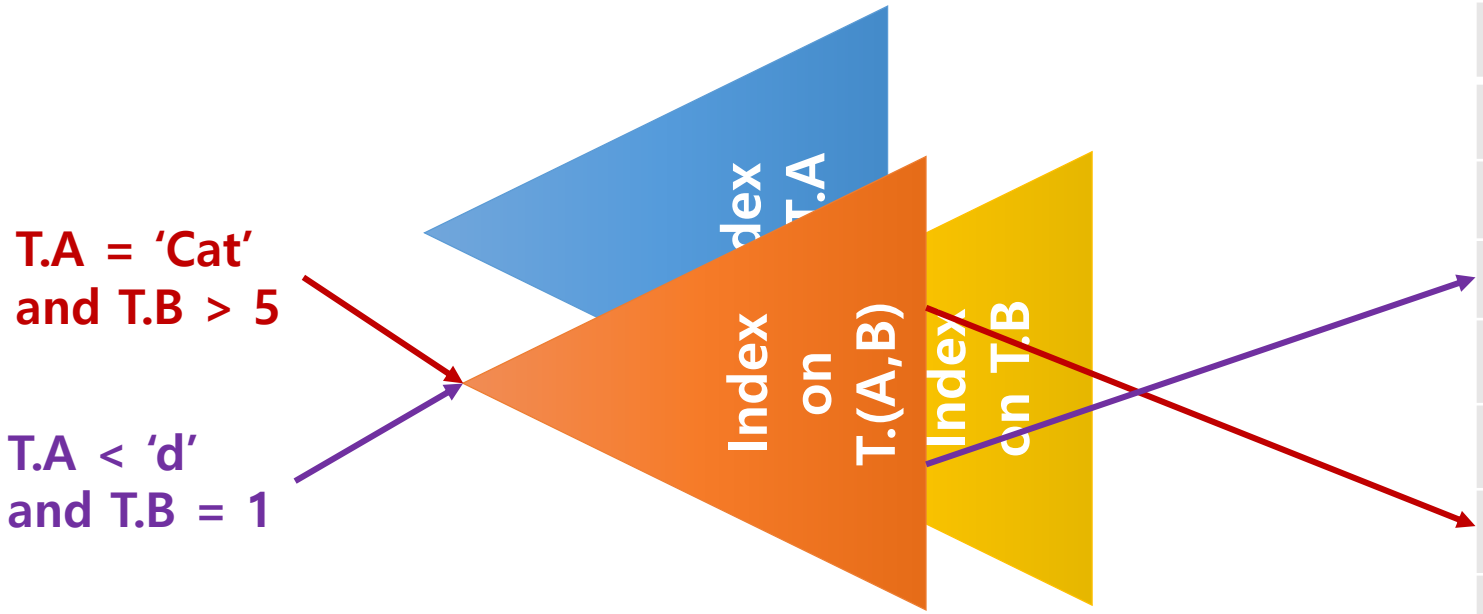
T.A = ‘Cat’
and T.B > 5



T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

Functionality



T

	A	B	C
1	Cat	2	…
2	Dog	5	…
3	Cow	1	…
4	Dog	9	…
5	Cat	2	…
6	Cat	8	…
7	Cow	6	…
	…	…	…

Utility

- Index = difference between full table scans and immediate location of tuples
  - Orders of magnitude performance difference

Utility

- Index = difference between full table scans and immediate location of tuples
  - Orders of magnitude performance difference
- Underlying data structures
  - Balanced trees (B trees, B+ trees)
  - Hash tables

SQL

```
SELECT      sName
FROM        Student
WHERE       sID = 18942
```

- Many DBMS's build indexes automatically on **PRIMARY KEY** (and sometimes **UNIQUE**) attributes

SQL

```
SELECT      sID
FROM        Student
WHERE       sName = ‘Mary’
AND         GPA > 3.9
```

SQL

```
SELECT      sName, cName
FROM        Student, Apply
WHERE       Student.sID =
            Apply.sID
```

Downsides of Indexes

1. Extra space – marginal
2. Index Creation - medium
3. Index maintenance – can offset benefits

Picking which indexes to create

- Benefit of an index depends on:
  - Size of table (and possibly layout)
  - Data distributions
  - Query vs. update load

“Physical design advisors”

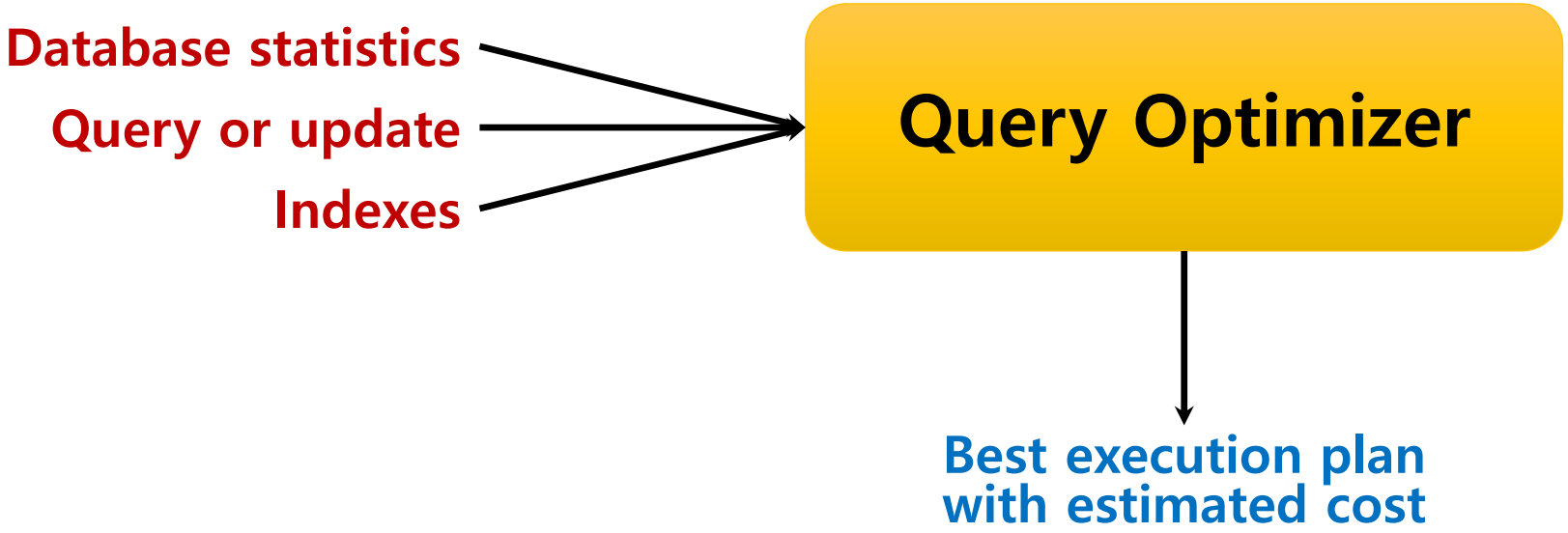
Input: database (statistics) and workload

Output: recommended indexes

“Physical design advisors”

Input: database (statistics) and workload

Output: recommended indexes



SQL Syntax

```
CREATE INDEX IndexName ON T(A)
```

```
CREATE INDEX IndexName ON T(A1, A2, …, An)
```

```
CREATE UNIQUE INDEX IndexName ON T(A)
```

```
DROP INDEX IndexName
```

Indexes

- Primary mechanism to get improved performance on a database
- Persistent data structure, stored in database
- Many interesting implementation issues
  - But we are focusing on user/application perspective