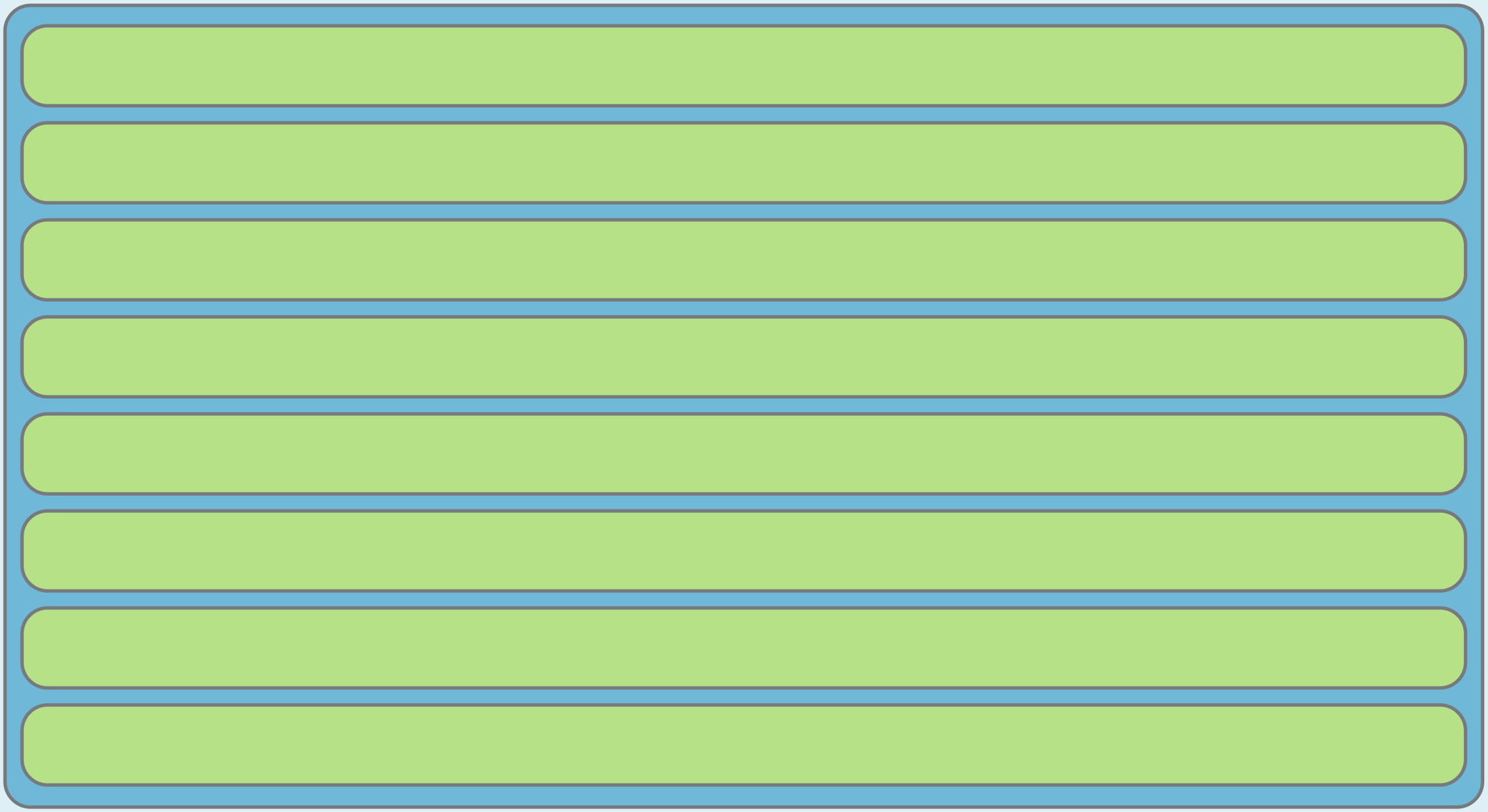
fourkitchens

We make **big** websites

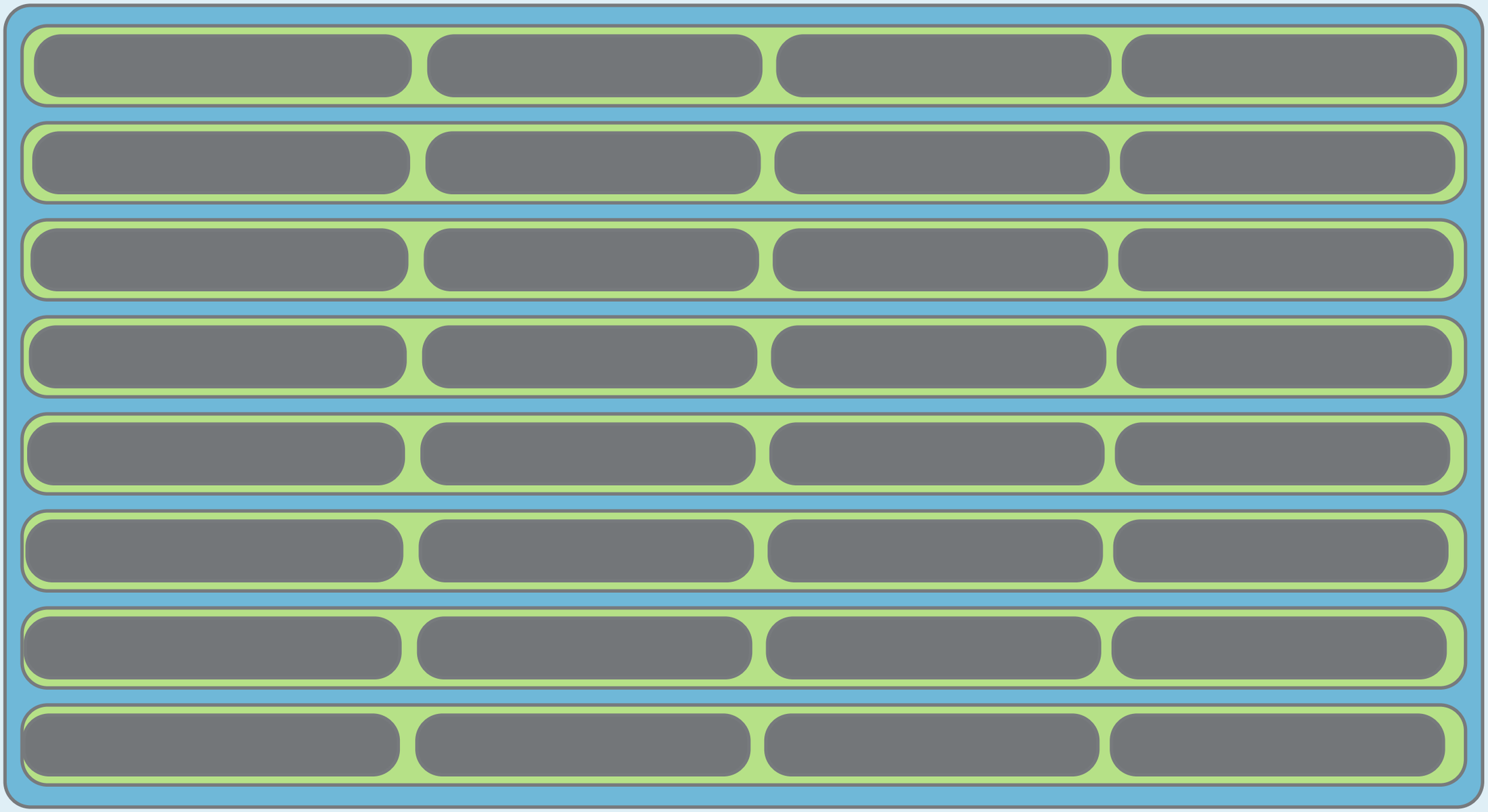
Indexes and DNA

Strategies to scale sites with lots of users and content

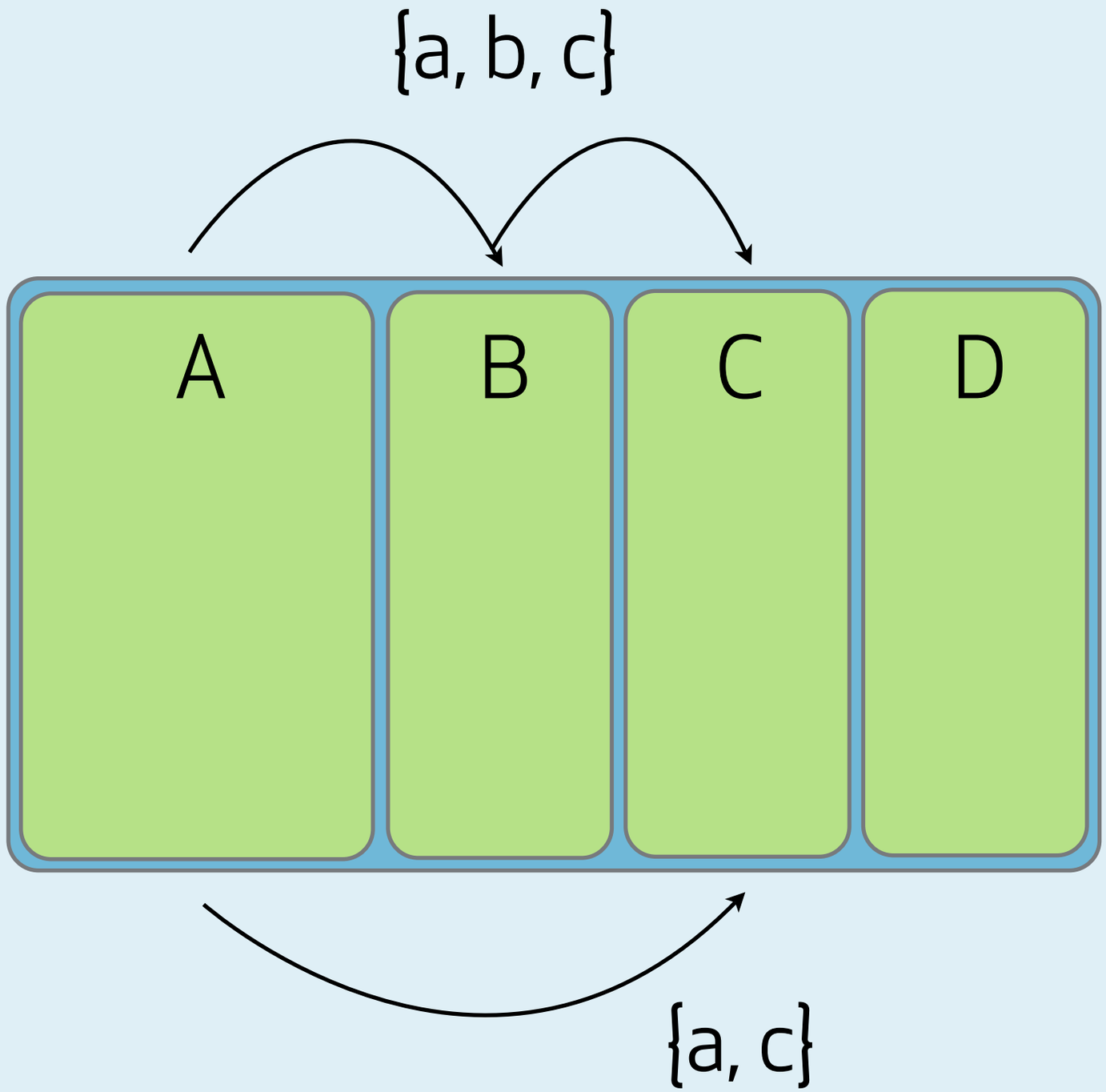
The anatomy of table rows



The anatomy of table columns



Defining an index

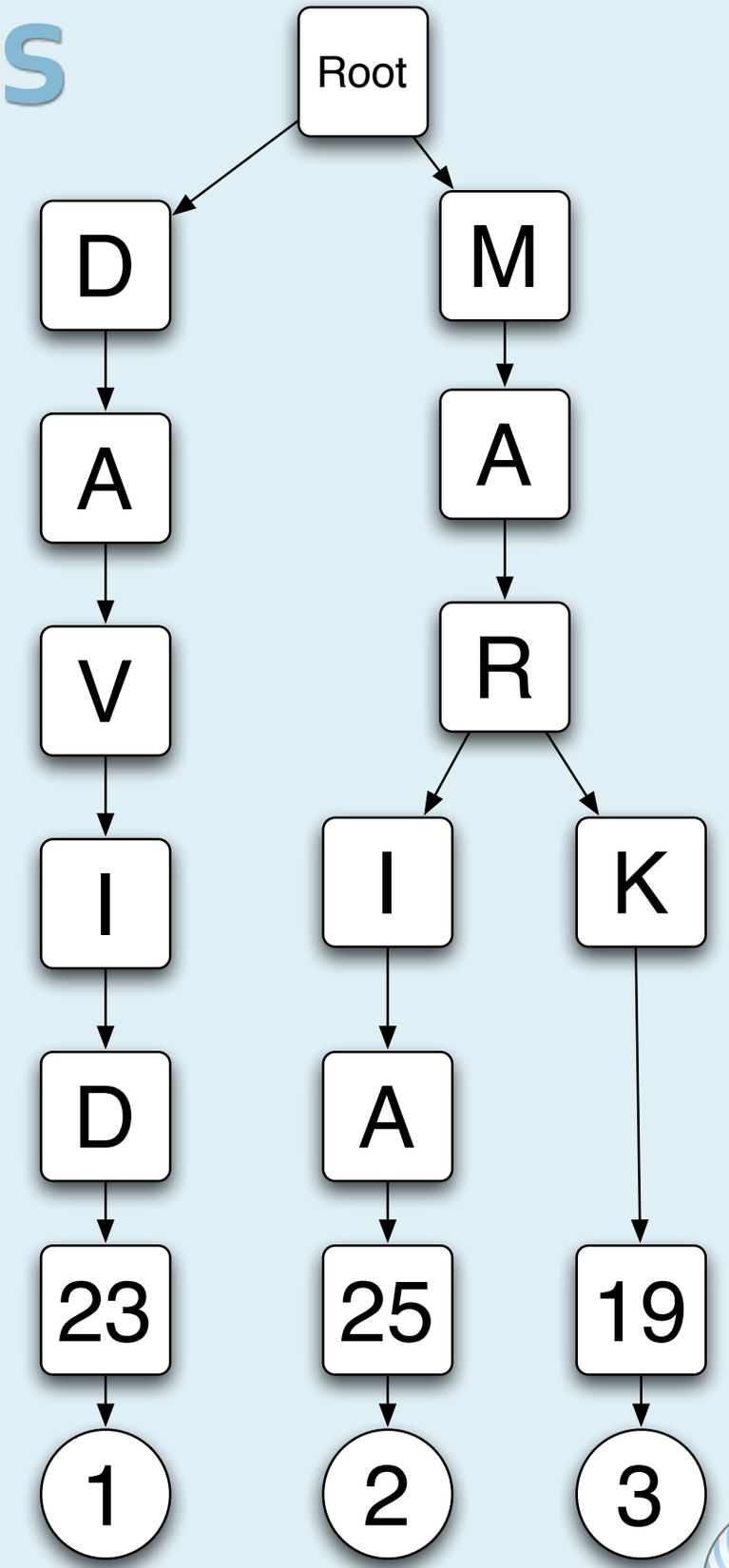


The anatomy of indexes

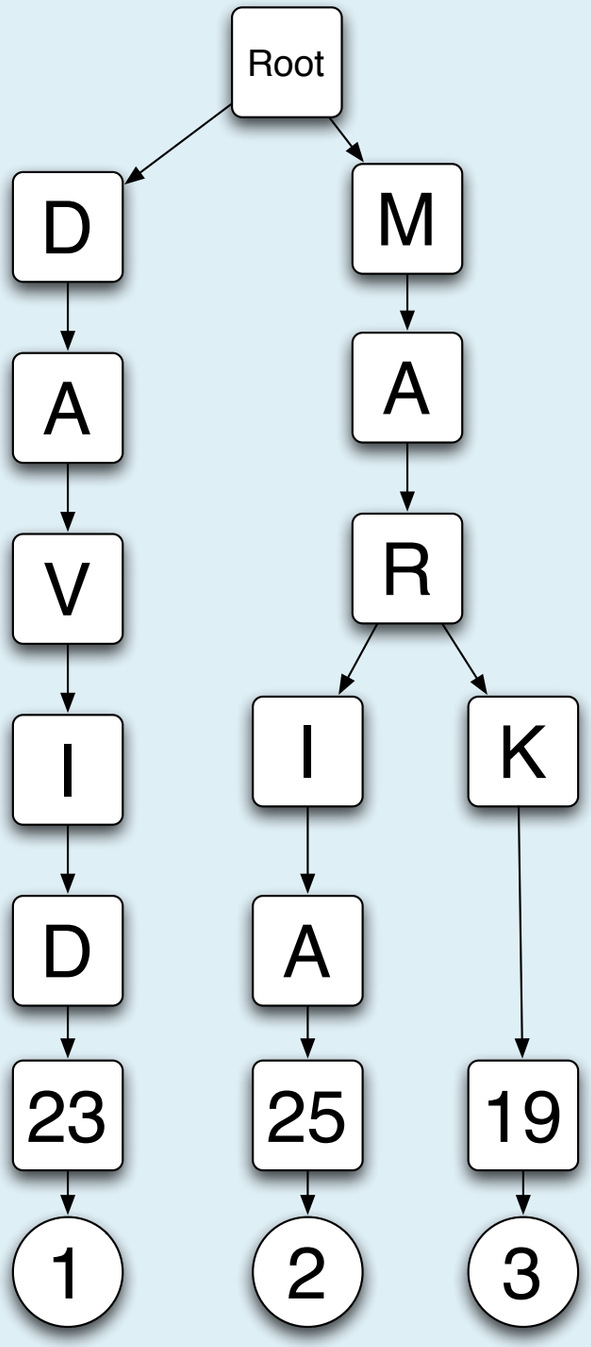
id	*name*	*age*	*evil*
1	David	23	1
2	Maria	25	0
3	Mark	19	0

```
CREATE TABLE IF NOT EXISTS `people` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `name` varchar(32) NOT NULL,
  `age` smallint(5) unsigned NOT NULL,
  `evil` tinyint(1) unsigned NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `name` (`name`,`age`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8
AUTO_INCREMENT=4 ;

INSERT INTO `people` (`id`, `name`, `age`, `evil`)
VALUES
(1, 'David', 23, 0),
(2, 'Maria', 25, 0),
(3, 'Mark', 19, 0);
```



What does this index do?



- Allows the following queries to run efficiently
  - SELECT age FROM {people}
  - SELECT * FROM {people} WHERE name LIKE “Mar%”
  - SELECT * FROM {people} ORDER BY name, age
  - SELECT * FROM {people} WHERE name = “Maria” ORDER BY age

How MySQL uses the indexes

```
mysql> EXPLAIN SELECT * FROM people;
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | people | ALL  | NULL          | NULL | NULL    | NULL |    3 |       |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT age FROM people;
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
| id | select_type | table  | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | people | index | NULL          | name | 100     | NULL |    3 | Using index |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM people WHERE name LIKE "Mar%";
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
| id | select_type | table  | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | people | range | name          | name | 98      | NULL |    1 | Using where |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
1 row in set (0.01 sec)
```

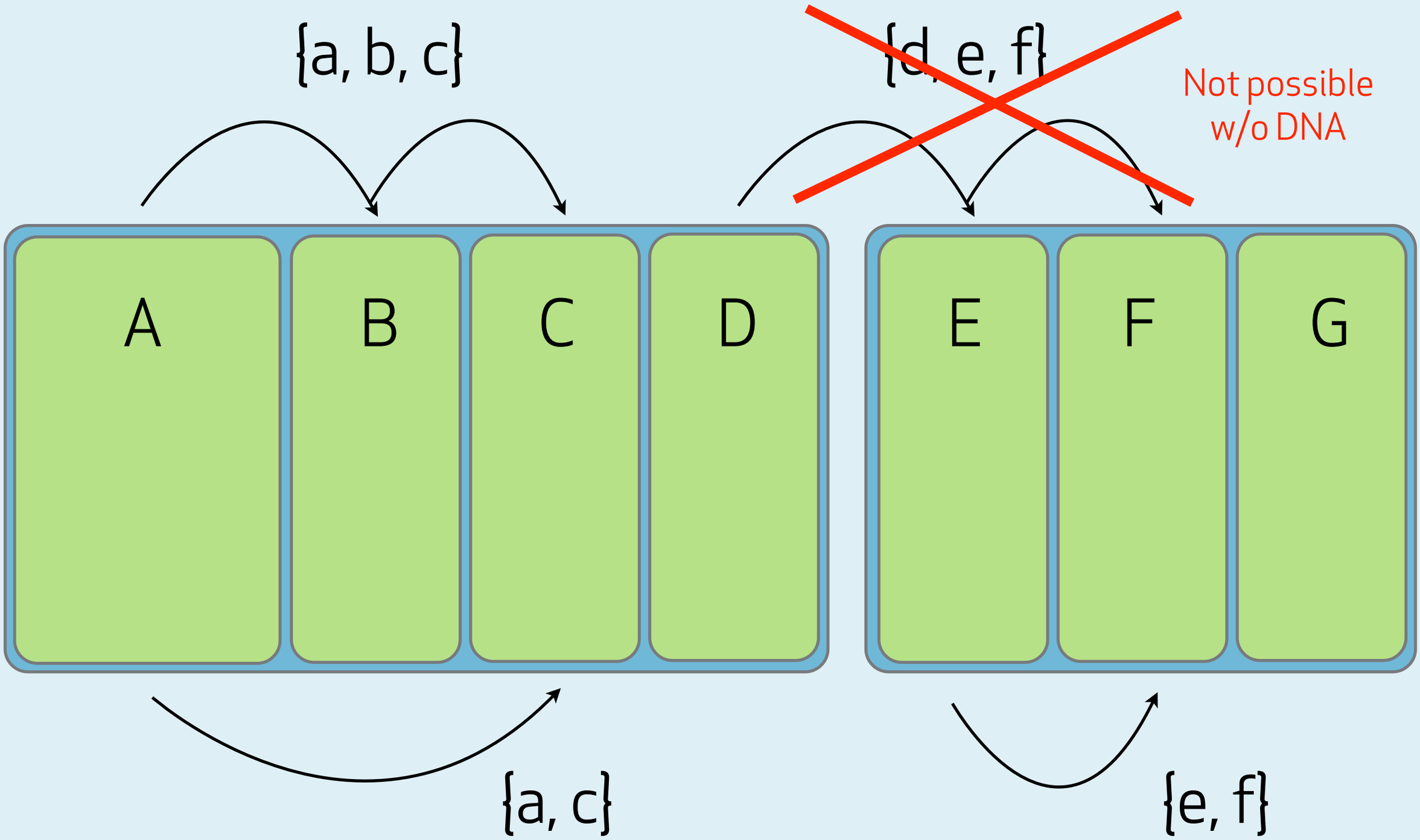
How MySQL uses the indexes

```
mysql> EXPLAIN SELECT * FROM people ORDER BY name, age;
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | people | index | NULL          | name | 100     | NULL |    3 |       |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM people ORDER BY name, age;
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | people | index | NULL          | name | 100     | NULL |    3 |       |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM people WHERE age = 23;
+----+-------------+--------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | people | ALL  | NULL          | NULL | NULL    | NULL |    3 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Indexing limitations



Pulling together data

node

nid	title	status	changed

node_comment_statistics

nid	last_ comment_ timestamp

"I want all published nodes, ordered by the later of the node's changed timestamp and the last comment timestamp."

This is amazingly inefficient.

How MySQL handles the query

Looks at indexes

Realizes criteria span two tables

Picks a table to start from

Applies criteria to that table using the index

Builds a temporary table

Applies remaining criteria to the temporary table

Sorts the temporary table

Returns results

We make **big** websites

fourkitchens

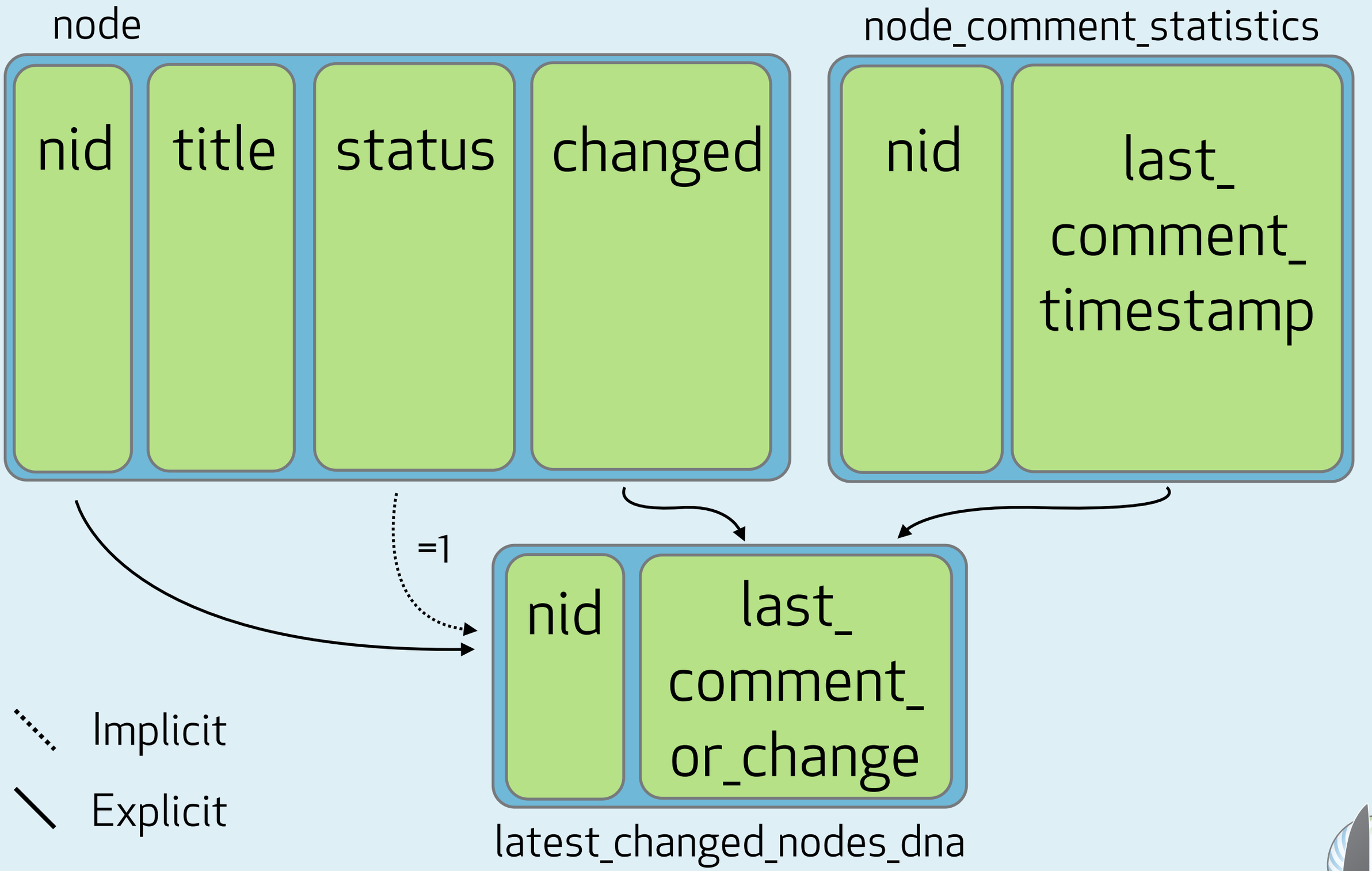
Enter DNA

Using the Denormalization API to create great indexes

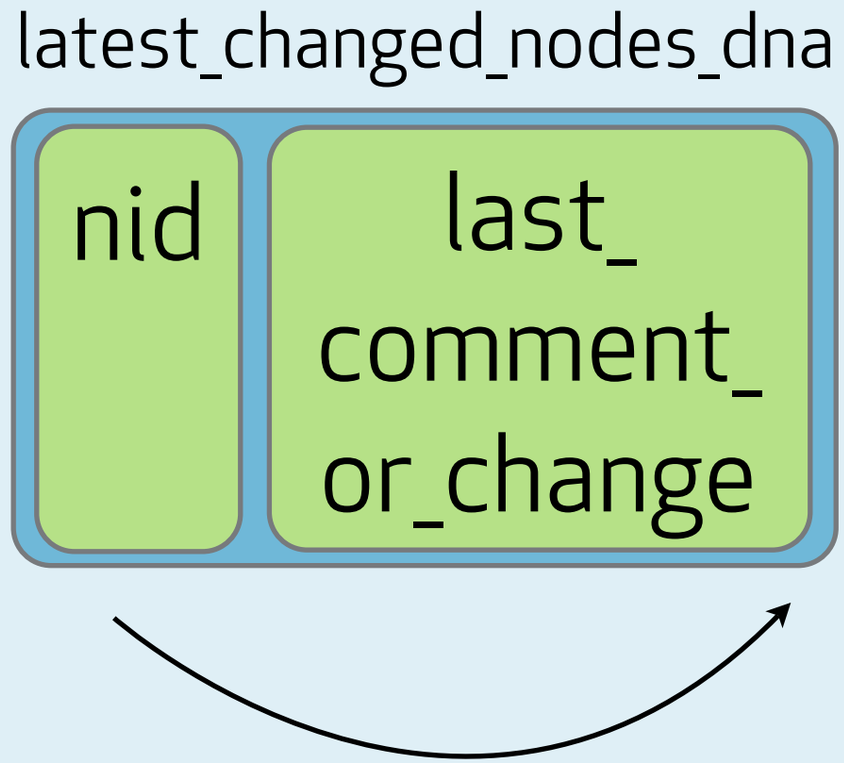
What is DNA?

DNA (The Denormalization API) is a module to consolidate information related to nodes from multiple tables into a single table.

Consolidating node data



Indexing the consolidated table



{last_comment_or_change, nid}

How MySQL handles this query

Looks at indexes

Applies criteria to the table using the index

Returns results