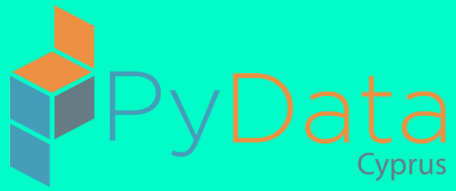
An Introduction to MySQL Optimisation

Andreas Loizou

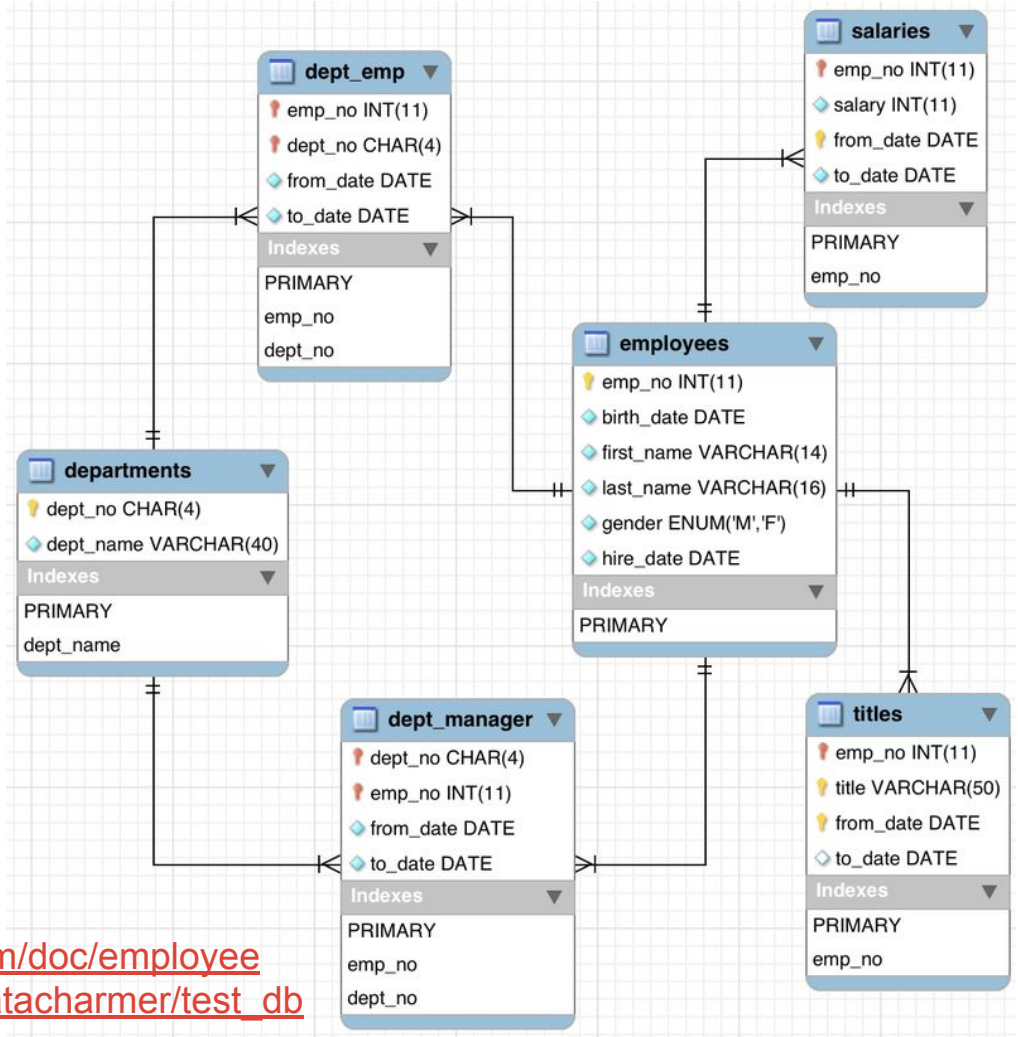


https://cy.linkedin.com/in/**loizouandreas**

Outline

- Optimize 🏃
- Optimize 🏃🏃
- Optimize 🏃🏃🏃
- As a Developer

Disclaimer



https://dev.mysql.com/doc/employee
https://github.com/datacharmer/test_db

OPTIMISE

“The most important factor in making a database application fast is its **basic design**” - MySQL documentation

Optimising data size

- Are the fields defined properly?
- Use the most efficient (smallest) data types possible
- Declare columns to be NOT NULL if possible. Faster indexes, less storage and elimination of overhead

```
CREATE TABLE employees (
    emp_no      INT             NOT NULL,
    birth_date  DATE            NOT NULL,
    first_name  VARCHAR(14)     NOT NULL,
    last_name   VARCHAR(16)     NOT NULL
    gender      ENUM ('M','F')  NOT NULL,
    hire_date   DATE            NOT NULL,
    PRIMARY KEY (emp_no)
);
```

QUERY OPTIMISATION

- ⬆️ ⬆️ ⬆️ Check if an index could help ⬆️ ⬆️ ⬆️
- Do not use “SELECT *”
- Find and fix repetitive calls and bottlenecks
- Use EXPLAIN

Indexes

- Speed access in the database
- Indexes are used to find rows with specific column values quickly.
- Efficient sorting
- Create only the indexes that you need to improve query performance.
    - Indexes are **good for retrieval, but slow down insert and update operations**.

https://dev.mysql.com/doc/refman/5.7/en/mysql-indexes.html


WARNING
Do not use indexes for everything.
“πᾶν μέτρον ἄριστον”

Without an index

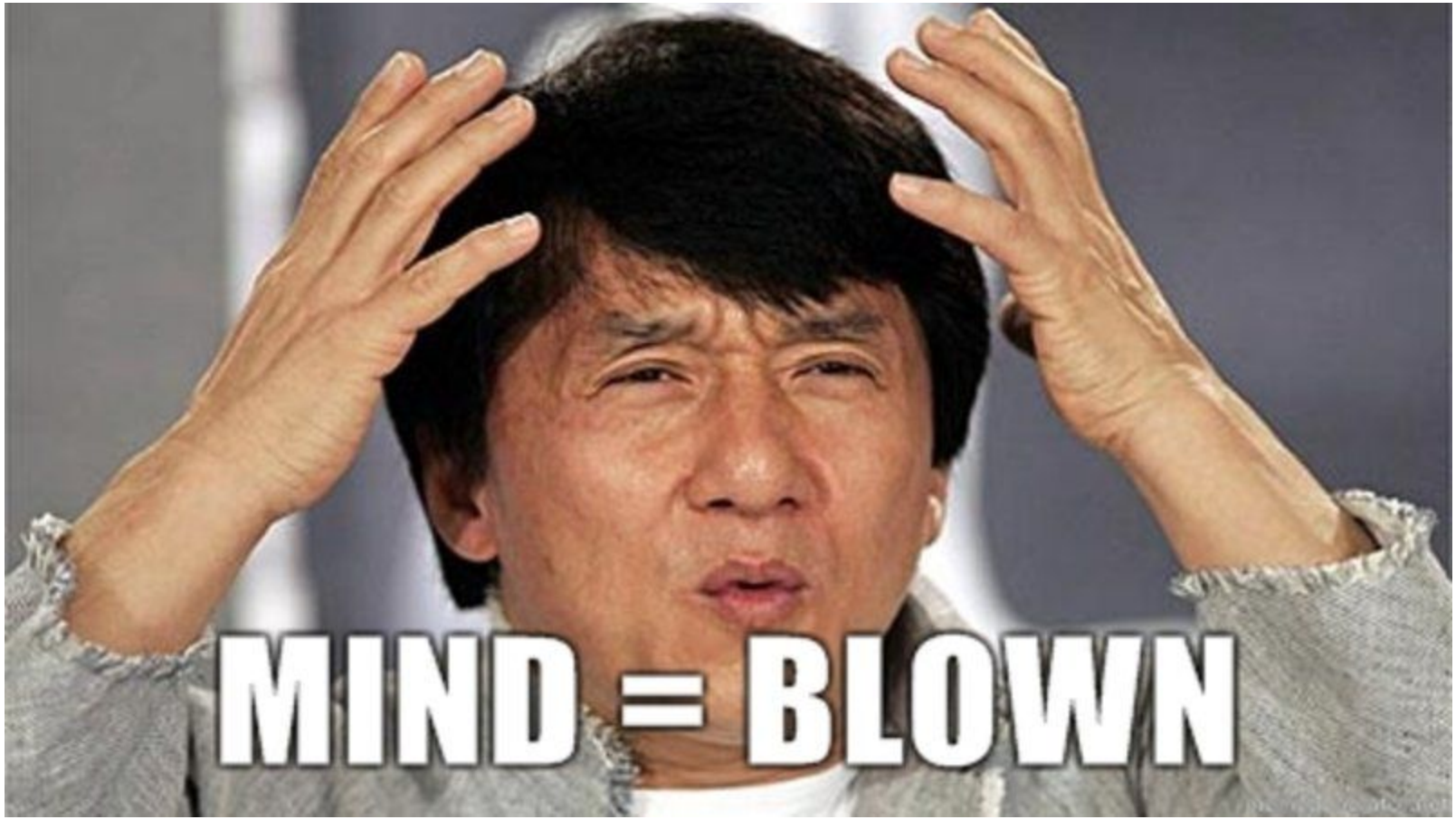
```
SELECT e.emp_no, e.first_name, e.last_name, s.*
FROM employees  as e left join salaries  as s on e.emp_no = s.emp_no
WHERE s.salary >= 100000
ORDER BY s.salary ASC;
-- 94709 row(s) returned
-- 3.390 sec
```

Index was added

```
SELECT e.emp_no, e.first_name, e.last_name, s.*
FROM employees  as e left join salaries  as s on e.emp_no = s.emp_no
WHERE s.salaries_btree_indexed >= 100000
ORDER BY s.salaries_btree_indexed ASC;
-- 94709 row(s) returned
-- 0.125 sec
```

27 TIMES FASTER

WOWWWWWWW


MIND = BLOWN

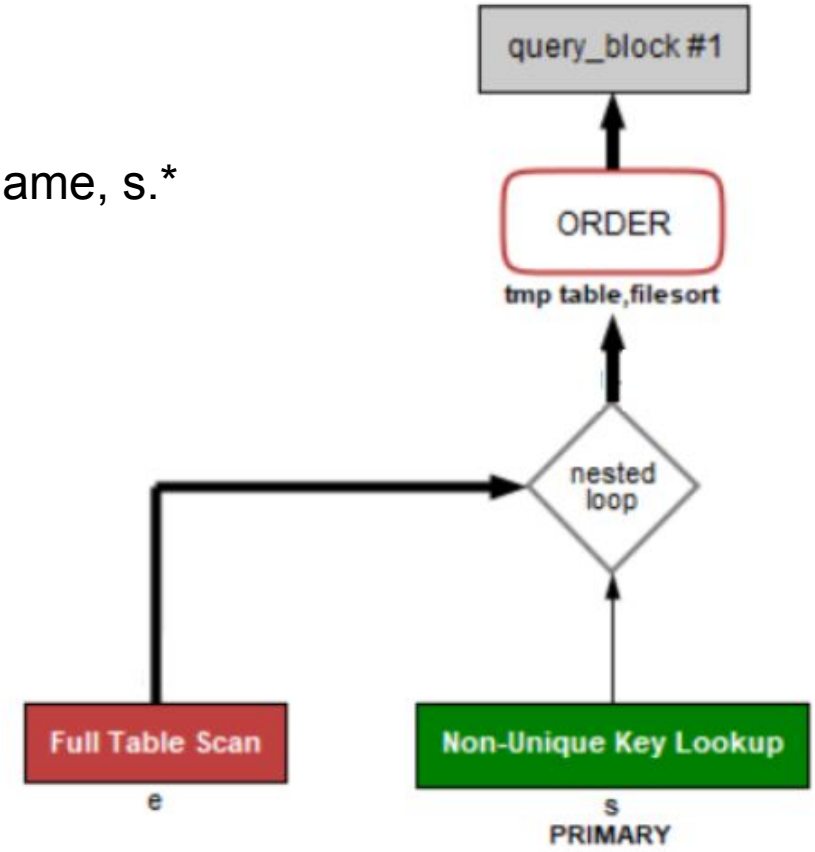
EXPLAIN TO UNDERSTAND THE QUERY EXECUTION PLAN

EXPLAIN is used to obtain a query execution plan (that is, an explanation of how MySQL would execute a query).

- Also available: Visual EXPLAIN at MySQL workbench
  - https://dev.mysql.com/doc/refman/8.0/en/explain-output.html

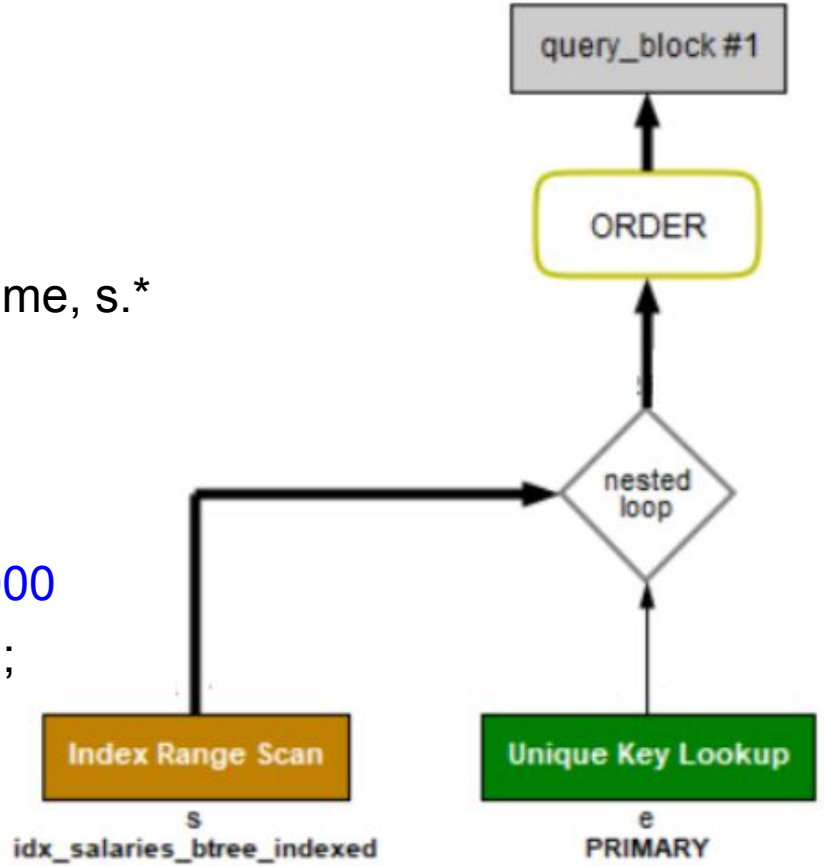
EXPLAIN A QUERY

```
EXPLAIN
SELECT e.emp_no, e.first_name, e.last_name, s.*
FROM employees  as e
left join salaries  as s
on e.emp_no = s.emp_no
WHERE s.salary >= 100000
ORDER BY s.salary ASC;
```



EXPLAIN A QUERY

```
EXPLAIN
SELECT e.emp_no, e.first_name, e.last_name, s.*
FROM employees  as e
left join salaries  as s
on e.emp_no = s.emp_no
WHERE s.salaries_btree_indexed >= 100000
ORDER BY s.salaries_btree_indexed ASC;
```



As a developer

CODING HORROR

programming and human factors

Google Custom Search

RESOURCES

26 Apr 2005

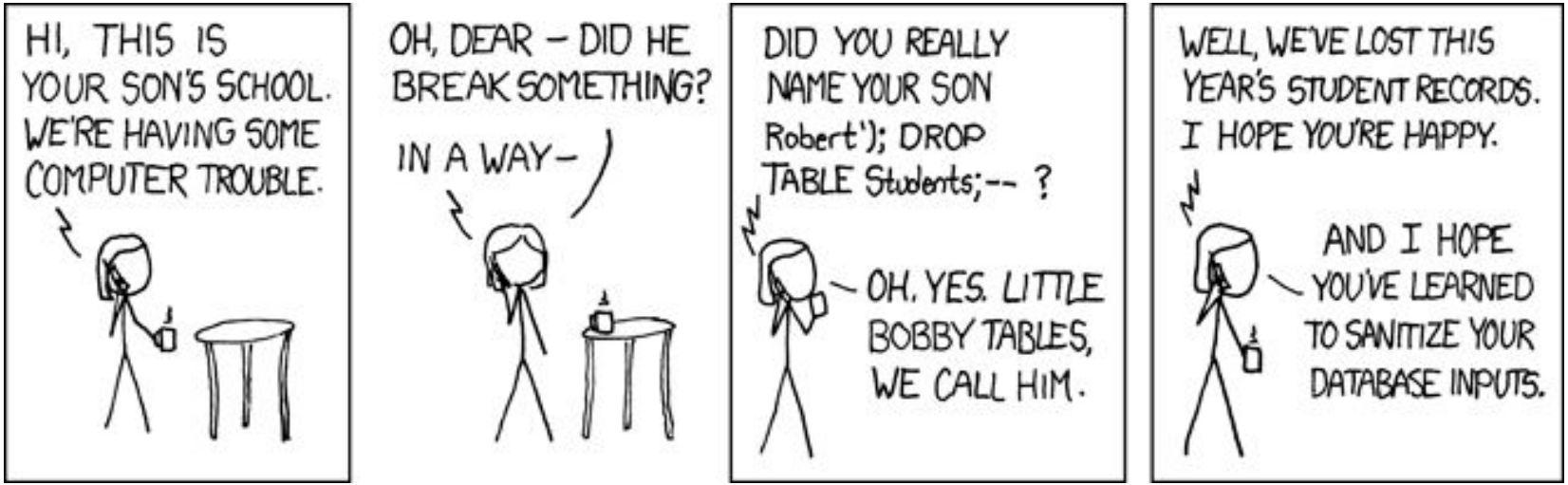
Give me parameterized SQL, or give me death

I have fairly strong feelings when it comes to the stored procedures versus dynamic SQL argument, but one thing is clear: you should never, ever use concatenated SQL strings in your applications. **Give me parameterized SQL, or give me death.** There are two good reasons you should never do this.

https://blog.codinghorror.com/give-me-parameterized-sql-or-give-me-death/

```
SELECT id, email, password, name
FROM users
WHERE email = 'x';
```

```
SELECT id, email, password, name
FROM users
WHERE email = 'x' OR full_name LIKE '%Admin%';
```



https://xkcd.com/327/

Inserting data - parameterised queries

```
cursor = cnx.cursor()

add_salary = (          "INSERT INTO salaries (emp_no, salary, from_date, to_date) "
                        "VALUES (%(emp_no)s, %(salary)s, %(from_date)s, %(to_date)s)")

# Insert salary information
data_salary = {
  'emp_no': emp_no,
  'salary': 50000,
  'from_date': date(2019, 1, 1),
  'to_date': date(9999, 1, 1),
}

cursor.execute(add_salary, data_salary)

cnx.commit()
cursor.close()
```

P. query

parameters

https://dev.mysql.com/doc/connector-python/en/connector-python-example-cursor-transaction.html

HAZARD: REAL (EX)-PRODUCTION CODE

```
sprintf(mySqlQueryPart, "(%d, '%s', '%s')",
user->id,
del_chars(user->name, "\\'%_\""),
del_chars(user->email, "\\'%_\"")
string.append(mySqlQueryPart);
```

Use Stored procedures

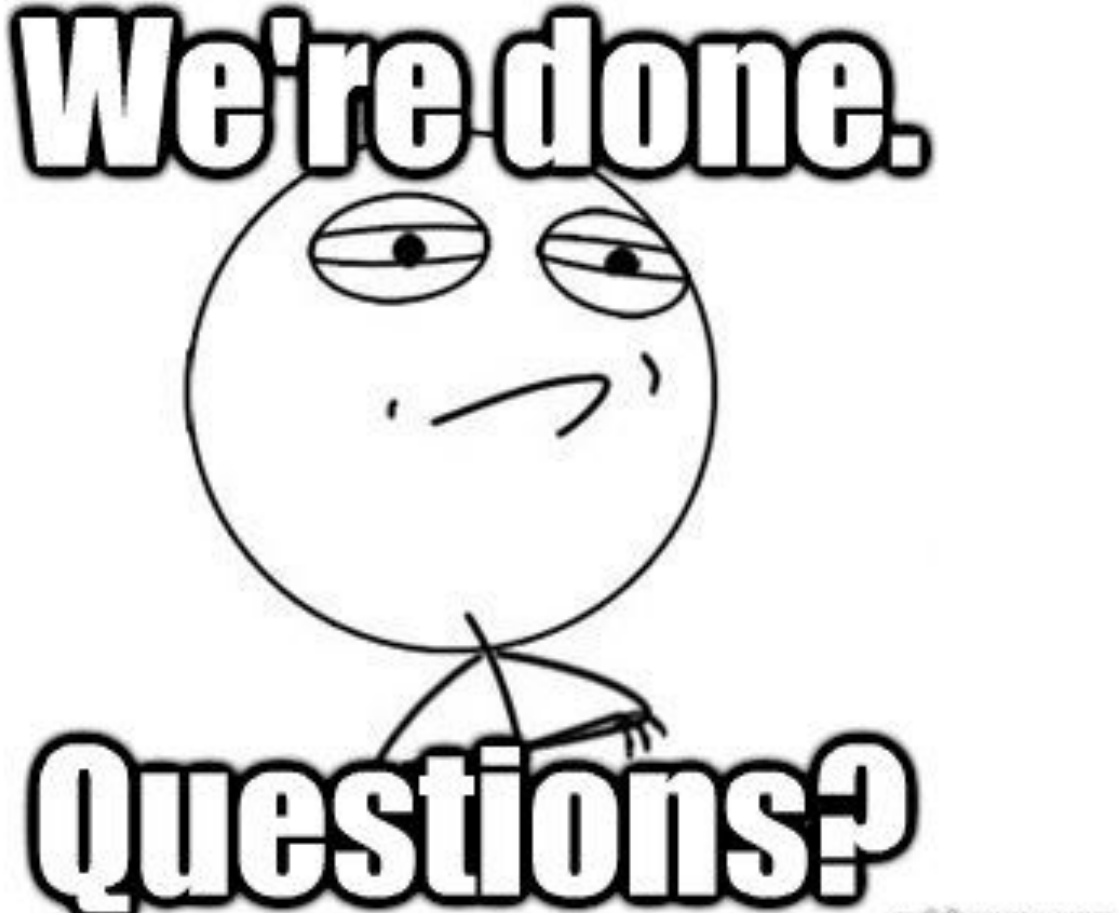

BIRTHDAY

Happy Birthday

```
DELIMITER $$
USE `employees`$$
CREATE PROCEDURE `get_employees_who_have_birthdate`()
BEGIN
set @today = CURDATE();
set @todaysMonth = month(@today);
set @todaysDay = day(@today);

select emp_no, first_name, last_name, birth_date
from employees.employees
where month(birth_date)=@todaysMonth and day(birth_date)=@todaysDay;
END$$
```

https://dev.mysql.com/doc/refman/8.0/en/optimization.html
https://dev.mysql.com/doc/refman/8.0/en/optimizing-innodb-queries.html
https://dev.mysql.com/doc/refman/8.0/en/optimizing-queries-myisam.html



https://cy.linkedin.com/in/**loizouandreas**

TECH TALKS
CYPRUS

https://cy.linkedin.com/in/**loizouandreas**