S02-L03 My personal best practices

1. Store code in a git repository

Always. Period.

Example: GitHub / michaljuhas / SQL-training-advanced https://github.com/michaljuhas/SQL-training-advanced

2. Do not use `select *`

Query: Select employees' contract sign date.

Not good (selecting all attributes from all tables):

```
SELECT
        *
FROM `sample_staff`.`employee`
INNER JOIN `contract` ON 1=1
        AND `contract`.`employee_id` = `employee`.`id`
WHERE 1=1
  AND `employee`.`deleted_flag` = 0
ORDER BY
  `employee`.`id`
LIMIT 1000
;
```

A slightly better version (only attributes form table `contract` selected):

```
SELECT
        `contract`.*
FROM `sample_staff`.`employee`
INNER JOIN `sample_staff`.`contract` ON 1=1
  ...
```

```
;
```

The best version (only attributes needed enumerated):

```
SELECT
        `employee`.`email`,
        `contract`.`sign_date` AS contract_sign_date /* added alias */
FROM `sample_staff`.`employee`
INNER JOIN `contract` ON 1=1
  ...
;
```

3. Separate attributes (columns) to rows

Query: Select employees' profile photos.

Not so good: A comma separated list of attributes in one line:

```
SELECT
        `employee`.`first_name`, `employee`.`last_name`, `employee`.`email`,
`employee`.`gender`, CONCAT(`photo`.`path`, `photo`.`filename`) AS
profile_photo
FROM `sample_staff`.`employee`
INNER JOIN `photo` ON `photo`.`employee_id` = `employee`.`id`
AND `photo`.`profile_photo_flag` = 1
        AND `photo`.`deleted_flag` = 0
WHERE 1=1
  AND `employee`.`deleted_flag` = 0
ORDER BY
  `employee`.`id`
LIMIT 1000
;
```

A better version (attributes split on multiple lines):

```
SELECT
```

```
        `employee`.`first_name`,
        `employee`.`last_name`,
        `employee`.`email`,
        `employee`.`gender`,
        CONCAT(`photo`.`path`, `photo`.`filename`) AS profile_photo
FROM `sample_staff`.`employee`
        ...
;
```

4. Set naming convention and don't allow exceptions

Consistency is the key.

- tables & columns (lower-case, snake-form)
- keys (suffix `_id`)
- date columns suffix `_date`
- datetime columns suffix `_datetime` or `_dt`
- indexes (prefix: `idx_`, `ak_`)
- table or column names always always always in singular
- flag indication (1=yes, 0=no, -1=unknown) always suffix `_flag` (`TINYINT NOT NULL DEFAULT -1`)
- separate integer ID's and varchar codes * suffix `_id` for `INTEGER(11)` * suffix `_code` for `VARCHAR(35)`

5. Be descriptive, don't use acronyms

At HotelQuickly, 3 years ago we started using acronyms such as:

- `cnt` = count
- `amt` = amount
- `catg` = category
- `ins` = insert
- and several others, but eventually very cumbersome to maintain.

Recommendation: be descriptive and *always use full words.* A simple rule like this can help a lot.

- `count` (i.e. `max_use_count`)
- `amount` (i.e. `voucher_amount`)
- `category` (i.e. `hotel_category_id`)
- `insert` (i.e. `insert_user_id`)

6. Use audit columns

- `insert_dt` - type `DATETIME` - time when the row was inserted (use `NOW()` at the time of insert)
- `insert_user_id` - type `INT (11)` - a user (if logged in) who inserted the row
- `insert_process_code` - type `VARCHAR (255)` - a process, function or class which inserted the row
- `update_dt` - type `TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP` - automatically changed
- `update_user_id` - type `INT (11)` - a user (if logged in) who modified the row
- `update_process_code` - type `VARCHAR (255)` - a process, function or class which inserted the row
- `deleted_flag` - type `TINYINT (4) NOT NULL DEFAULT 0` - use values `0` or `1`, nothing else

7. Batch delete & updates (example)

With 1+ mil. rows it will be slow (table locked, transactions piled up).

```
DELETE FROM salary
WHERE to_date IS NOT NULL
```

This will be faster, but you need to run it 100+ times in a loop cycle. Stored procedures are good for this.

```
DELETE FROM salary
WHERE to_date IS NOT NULL
LIMIT 10000
```

For example in PHP:

```
while (1) {
        mysql_query("DELETE FROM salary WHERE to_date IS NOT NULL LIMIT
10000");

        if (mysql_affected_rows() == 0) {
                        // done deleting
                        break;
                }

        // you can even pause for a few seconds
        sleep(5);
}
```

8. Reference the owner of the object

Always add a table name before column name.

Classic scenario - you start with a simple query:

```
SELECT
        id AS employee_id,
        first_name,
        last_name
FROM employee
WHERE 1=1
        AND deleted_flag = 0
LIMIT 100
;
```

Commit to git. Then decide to join another table (i.e. `contract`) and suddenly you need to rewrite half of the query because both `id` and `deleted_flag` would be in both tables... #cumbersome

9. Table names always singular

Imagine a list of tables in one schema:

- `employee`
- `contracts`
- `employee_contracts_rel`

And now try to write a query

```
SELECT
  .... /* fill in */
FROM employee
INNER JOIN contracts ON 1=1
  .... /* fill in */
WHERE 1=1
  AND employee.deleted_flag = 0
;
```

It can be even worse in NoSQL data storage...

10. WHERE 1=1 (and / or)

```
SELECT
  id AS employee_id
FROM employee
WHERE 1=1
  AND employee.deleted_flag = 0
  AND employee.birth_date >= '1960-01-01'
        AND employee.birth_date <= '1960-31-12'
ORDER BY
  employee.birth_date
LIMIT 1000
;
```

11. Old vs. new JOIN style

```
SELECT
  employee.id,
  employee.full_name,
  contract.start_date
FROM employee, contract, lst_contract_tp
WHERE 1=1
  AND employee.id = employee_contract_rel.employee_id
  AND lst_contract_tp.id = contract.contract_tp_id
  AND employee.deleted_flag = 0
  AND contract.deleted_flag = 0
```

12. Prefix database objects

- views with `v_`
- functions with `fc_`

13. Don’t use column rows in ORDER BY

```
SELECT
  employee.id AS employee_id
FROM employee
WHERE 1=1
  AND employee.deleted_flag = 0
  AND employee.birth_date >= '1960-01-01'
  AND employee.birth_date <= '1960-31-12'
ORDER BY
  1
LIMIT 1000
;
```

14. Use LIMIT 1 as much as possible

(example)

In your PHP code:

```
$todayDate = ... // Define
$sql = "SELECT birth_date FROM employee WHERE birth_date = {$todayDate}";

$result = $connection->query($sql);

if ($result->num_rows > 0) {
                echo "Yes, there's an employee with this birth date"
} else {
    echo "Nobody is celebrating";
}
```

Better would be to use `LIMIT 1` in case of a large dataset.

15. Use correct data type, it makes a difference (example: IP address)

From `varchar20` to `integer unsigned` (`145.54.123.90` => `2436266842`).

- INET_ATON(expr) http://dev.mysql.com/doc/refman/5.0/en/miscellaneous-functions.html#function_inet-aton
- INET_NTOA(expr) http://dev.mysql.com/doc/refman/5.0/en/miscellaneous-functions.html#function_inet-ntoa

```
TRUNCATE ip_address_int;

INSERT INTO ip_address_int (id, ip_address)
SELECT
        id,
        INET_ATON(ip_address_varchar20.ip_address)
FROM ip_address_varchar20
```

```
-- 1. Make sure to analyze tables first
ANALYZE TABLE ip_address_varchar20;
ANALYZE TABLE ip_address_int;

-- 2. Verify that the count of rows in each table is the same
select count(*) from ip_address_varchar20;
select count(*) from ip_address_int;

-- 3. Check the size of tables on disk (in MB)
SELECT
        table_name,
        (data_length + index_length) / power(1024, 2) AS tablesize_mb
FROM information_schema.tables
WHERE 1=1
        AND table_name IN ('ip_address_varchar20', 'ip_address_int')
;

-- Make sure you can get the same value
SELECT ip_address FROM ip_address_varchar20 WHERE id = 16;
SELECT INET_NTOA(ip_address) FROM ip_address_int WHERE id = 16;
```