A Survey for Database Smells

We are conducting this survey to understand the importance and usefulness of various database schema smells. The survey is expected to take 10-15 minutes. By taking this survey, you are helping us understand developers' perspective on database smells. At the same time, we hope you will learn some schema practices to avoid and thus improve your developer skills.

Please note that this survey is anonymous and we will publish the results of the survey collectively without revealing any personal details.

Thank you!

Page 1

---

1. Please specify your software development experience in years. *
   A. 0
   B. 1
   C. 2
   D. 3
   E. 4
   F. 5
   G. 6-10
   H. 10-20
   I. >20

2. How many database-based applications you have written (either individually or as part of a team)?
   A. 0
   B. 1
   C. 2
   D. 3
   E. 4
   F. 5-10
   G. >10

Page 2: Database Smells

Database smells are the characteristics of database code (either DDL (Data Definition Language) or DML (Data Manipulation Language) SQL statements), database system, or

stored data that indicate a violation of the recommended best practices and potentially affect the quality of the software system in a negative way.

We categorize database smells in three categories

1. Schema smells (arising due to poor schema),
2. SQL statement (e.g. query) smells
3. Data smells (arising from poor data handling).

In this survey, we focus only on database schema smells.

3. What is your opinion about the following statement?
   "Awareness and knowledge about database smells is very crucial for every developer working with database-based applications to keep the maintainability and performance of the application high."
   A. Strongly disagree
   B. Disagree
   C. Neither agree nor disagree
   D. Agree
   E. Strongly agree

Instructions for questions 4-16:

For questions 4-16, we describe 13 "practices" concerning database schema smells. Kindly read the description of each practice and rate them based on the following two criteria:
Importance *(i.e., the degree of the practice's association with software quality issues), and*
Usefulness *(i.e., the degree of accuracy of the practice in predicting software quality issues).*
Here, by software quality issues, we mainly refer *maintainability* and *performance*.

Some additional information about the database smells can be found online (removed the link now to maintain anonymity).

Each presented practice could be a smell, a recommended practice, or neither. For instance, one may think that eliminating constraints from a *create table* statement is an important performance optimization tool, rather than a smell. Therefore, we present these to you and ask your opinion about them whether they are schema smells, recommended practices, neither, or both based on the context.

Compound attribute
Each attribute value must be stored and retrieved atomically. However, in this practice, a column is used to store a non-atomic attribute. For instance, storing comma-separated lists for an attribute to avoid creating an intersection table for a many-to-many relationship or storing a JSON file which is not used atomically.

Example:
CREATE TABLE Books (

```
  book_id SERIAL PRIMARY KEY,
  book_title VARCHAR (1000),
  authors VARCHAR (1000), -- comma-separated list
--, , ,
};
```

4. Based on the description given above and using the criteria provided on the top of the page, how would you classify **compound attribute**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Adjacency list

In this practice, an attribute in a table refers another row in the same table i.e., a table has a recursive relationship to model hierarchical structure.

Example:

```
CREATE TABLE Comments (
  comment_id SERIAL PRIMARY KEY,
  parent_id BIGINT UNSIGNED,
-- ,  ,  ,
  FOREIGN KEY (parent_id) REFERENCES Comments (comment_id),
--, , ,
};
```

5. Based on the description given above and using the criteria provided on the top of the page, how would you classify **adjacency list**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Superfluous key

In this practice, an unnecessary superfluous pseudo key is defined in a table where other attribute(s) in the table may serve as a primary key.

Example:

```
CREATE TABLE BugsProducts (
  id SERIAL PRIMARY KEY,
  bug_id BIGINT UNSIGNED NOT NULL,
  product_id BIGINT UNSIGNED NOT NULL,
  UNIQUE KEY (bug_id, product_id),
-- ,  ,  ,
};
```

6. Based on the description given above and using the criteria provided on the top of the page, how would you classify **superfluous key**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Missing constraints

Referential integrity is an essential property of relational databases. However, in this practice, constraints for a foreign key are missing from a schema definition.

Example:

```
CREATE TABLE BugsProducts (
  id SERIAL PRIMARY KEY,
  bug_id BIGINT UNSIGNED NOT NULL,
  product_id BIGINT UNSIGNED NOT NULL,
  UNIQUE KEY (bug_id, product_id),
-- No foreign key constraints for bug_id and product_id
};
```

7. Based on the description given above and using the criteria provided on the top of the page, how would you classify **missing constraints**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Metadata as data

In this practice, metadata is stored as data in the form of EAV (Entity-Attribute-Value) pattern.

Example:

```
CREATE TABLE IssueAttributes (
  issue_id    BIGINT UNSIGNED NOT NULL,
  attr_name   VARCHAR(100) NOT NULL,
  attr_value  VARCHAR(100),
 -- constraints on attributes
);
```

8. Based on the description given above and using the criteria provided on the top of the page, how would you classify **metadata as data**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context

E. Don’t know

Polymorphic association
Relational database schema does not allow us to declare polymorphic association i.e., declare a foreign key that references multiple parent tables. However, many times developers define an additional column in a table as a tag to realize a polymorphic association. In such cases, a table uses a multi-purpose foreign key.

Example

```
CREATE TABLE Comments (
  comment_id SERIAL PRIMARY KEY,
  issue_type VARCHAR(20),     -- "Bugs" or "FeatureRequests"
  issue_id BIGINT UNSIGNED NOT NULL,  -- used along with issue_type to implement
polymorphic association
-- , , ,
  FOREIGN KEY (author) REFERENCES Accounts(account_id)
);
```

In the above case, the developer wants to define a foreign key something like this: "FOREIGN KEY (issue_id) REFERENCES Bugs (bug_id) or FeatureRequests (feature_id)". However, such a declaration is not possible, so developer added *issue_type* to know whether she will refer to *Bugs* or *FeatureRequests* table with *issue_id*.

9. Based on the description given above and using the criteria provided on the top of the page, how would you classify **polymorphic association**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don’t know

Multicolumn attribute
In this practice, multiple serial columns are created for an attribute.

Example

```
CREATE TABLE Bugs (
    bug_id     SERIAL PRIMARY KEY,
    description VARCHAR(1000),
    tag1       VARCHAR(20),
    tag2       VARCHAR(20),
    tag3       VARCHAR(20)
);
```

10. Based on the description given above and using the criteria provided on the top of the page, how would you classify **multicolumn attribute**?

A. Database smell
B. Recommended practice

C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Clone tables
In this practice, a table is split horizontally in multiple tables using some criterion (for example, year) to achieve scalability.

Example
```
CREATE TABLE Bugs_2008 ( . . . );
CREATE TABLE Bugs_2009 ( . . . );
CREATE TABLE Bugs_2010 ( . . . );
```

11. Based on the description given above and using the criteria provided on the top of the page, how would you classify **clone tables**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Values in attribute definition
In this practice, specific values are defined in an attribute definition to restrict possible values of the attribute.

Example
```
CREATE TABLE Bugs (
    -- other columns
status VARCHAR(20) CHECK (status IN ('NEW', 'IN PROGRESS', 'FIXED')) );
```

12. Based on the description given above and using the criteria provided on the top of the page, how would you classify **values in attribute definition**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

Index abuse
In this practice, the indexes are used poorly. It has the following variants:

- Missing indexes
- Insufficient indexes (indexes must be prepared at least for primary and foreign keys)
- Unused indexes

13. Based on the description given above and using the criteria provided on the top of the page, how would you classify **index abuse**?

A. Database smell

B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don’t know

God table
In this practice, a table contains excessive number of attributes.

Example (table with 23 attributes)
CREATE TABLE bet_matches(id TEXT PRIMARY KEY, tx0_index INTEGER, tx0_hash TEXT, tx0_address TEXT, tx1_index INTEGER, tx1_hash TEXT, tx1_address TEXT, tx0_bet_type INTEGER, tx1_bet_type INTEGER, feed_address TEXT, initial_value INTEGER, deadline INTEGER, target_value REAL, leverage INTEGER, forward_quantity INTEGER, backward_quantity INTEGER, tx0_block_index INTEGER, tx1_block_index INTEGER, block_index INTEGER, tx0_expiration INTEGER, tx1_expiration INTEGER, match_expire_index INTEGER, fee_fraction_int INTEGER, status TEXT, FOREIGN KEY (tx0_index, tx0_hash, tx0_block_index) REFERENCES transactions(tx_index, tx_hash, block_index), FOREIGN KEY (tx1_index, tx1_hash, tx1_block_index) REFERENCES transactions(tx_index, tx_hash, block_index));

14. Based on the description given above and using the criteria provided on the top of the page, how would you classify **god table**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don’t know

Meaningless name
In this practice, a table or an attribute name is cryptic or meaningless.
Rationale: Meaningless or cryptic names hamper readability of the database's schema.
Example

```
CREATE TABLE master (
  id TEXT PRIMARY KEY,
  tx0 INTEGER,
  tx1 TEXT,
  -- other attributes
);
```

15. Based on the description given above and using the criteria provided on the top of the page, how would you classify **meaningless name**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don’t know

Overloaded attribute names
In this practice, two or more attributes are defined with identical names but as distinct data types in different tables.

Example
```
CREATE TABLE `spell_override` (
`comment` LONGTEXT NOT NULL,
-- other columns
);
CREATE TABLE `creature_text` (
`comment` varchar(255) DEFAULT, -- overloaded attribute
-- other columns
);
```

16. Based on the description given above and using the criteria provided on the top of the page, how would you classify **overloaded attribute names**?

A. Database smell
B. Recommended practice
C. Neither a smell nor a recommended practice
D. Both a smell and a recommended practice depending on the context
E. Don't know

17. Kindly list other database schema smells that you consider relevant but are missing from this questionnaire. Provide a name, description, and if possible a reference for each smell. (Optional)

_____
_____
_____

18. Kindly provide your email address if you would like to receive compiled results of the survey. (Optional)

_____
_____

19. Do you have any comments, reservations, feelings, or objections regarding the smells in the presented list? Feedback to improve the survey is also welcome. (Optional)

_____
_____