Database Design Debts

MASHEL ALBARAK
UNIVERSITY OF BIRMINGHAM & KING SAUD UNIVERSITY

DR. RAMI BAHSOON
UNIVERSITY OF BIRMINGHAM
Overview

- Technical debt/ database debt
- Motivation
- Research objective and approach
- Database design debt definition
- Taxonomy for database design debt
- Case Study
- Future Work
Technical Debt
Technical Debt

Customer’s view

Developer’s view
Technical Debt

Code ➔

Design
Architecture
Requirements
Test
Documentation
.....
Database Design

Information Systems (ISs) continually evolve in response to changes in users’ requirements, environment and emergent needs for improved information provision and service qualities.

Databases are core for the functioning of these systems and most systems tend to adapt existing legacy databases, extending on their underlying design ➔ Schema evolution is a common practice.

Studies have reveals situations, where non-optimal database design decisions and flaws can carry some benefits ➔ Debt.
Motivation

Databases are Core of many ISs

Tightly coupled with data maintained

Hurt data quality+ hurt the entire system
Research objective and approach

Identify technical debts that relate to designing and evolving relational databases schemas.

1. Define the concept
2. Develop a taxonomy that classifies various types of database design debt.
Database design debt can be attributed to immature or suboptimal database design decisions that lags behind the optimal/desirable ones. These decisions can have negative impact on the schema, its maintenance and evolution.
Database Design Debt Taxonomy

Database Design

- Conceptual
- Logical
- Physical
Conceptual Design

conducted after requirements analysis phase.

It is a high-level data model, which translates the requirements to a conceptual schema.

Conceptual design debt is not purely technical and can be attributed to social aspects in engineering the system.
Logical Design

The first step for implementing the database.

Transforming the conceptual schema into the implementation schema or the “relational” database schema that will be used in the database system.

Debt in this phase may occur during:

- Normalization
- Orthogonal design
- Defining the referential Integrity constraints
- Using the anti-Patterns:
  - Referencing the same table
  - Entity attribute value tables
  - Polymorphic association
Normalization

A process of testing a schema more than once to ensure that it satisfies a certain normal form.

Why?
- Enforce data integrity as it reduces data redundancy.
- Eliminate updating and deleting anomalies and facilitate maintenance.

Poorly normalized database can carry a debt

Big amount of data duplication; it can put the entire burden on the applications code developer to modify the data.

From a business perspective, the cost of bad normalization will affect the quality of the stored data.
Physical Design

Logical design concerns of “What” to store, physical design is concerned with “How” to store it.

During this final phase of database design, developers specify files organization, indexes, storage and other physical features for the database.

Developers need to first analyze the application, in terms of the transactions, queries, expected update operations frequencies..etc., to make the appropriate decisions for a better physical design.

Debt in this phase may occur during:

- Indexing
- De-normalization
- Creating Views
- Partitioning
- Using lookup tables
- Hardware/ Software choices.
Indexing

The main reason behind index creation is to **quickly locate rows** searched based on the indexed attribute(s). In addition, unique indexes guarantee uniquely identifiable records in the database.

- indexes allocate space on the disc.
- it will increase update and insert overhead
- decrease the performance of the database.
**Case Study**

**MediaWiki** web application that was first introduced in 2001 and has 26 schema versions that evolved throughout the years.

Logical design debts:

**Referential integrity constraint debt:**
- No foreign keys from release 1 ➔ 4 of the database schema because the database was stored in MYISAM engine
- since release 5 they moved to InnoDB engine that supports referential integrity constraints, but there was no implementation of foreign keys on the schema file.
Case Study

Logical design debts:

**Normalization debt:**
- 6th release of the schema. The violation resembles in the addition of two new columns, `ipb_range_start` and `ipb_range_end`, to ipblocks table. Since `ipb_range_end` depends on the `ipb_range_start`, those columns should be moved to a separate table to satisfy second normal form.

| `ipb_range_start` | `ipb_range_end` |
Case Study

Logical design debts:

**Referencing the same table debt:**

- In the 10th release, a new column, `rev_parent_id`, was added to revision table to reference `rev_id` in the same table creating a tree structure. The tree can extend in depth depending on the numbers of revisions, which will increase the complexity of querying specific revision at a specific level of the tree.

<table>
<thead>
<tr>
<th>rev_id</th>
<th>rev_parent_id</th>
</tr>
</thead>
</table>
Case Study

Logical design debts:

**Orthogonal design debt:**

- The 18th release, table `user_former_groups` was added to the schema to store groups that the user has once belonged to. The user may again belong to these groups. The new table is a clone of the `User_groups` table. Since some of the user current groups may be stored in both tables, which indicates an overlapping between those tables, it will create redundancy. Moreover, developers will have to make more effort to keep them both synchronized.
# Quality attributes affected by database design debts

<table>
<thead>
<tr>
<th>Design Principle</th>
<th>Affected quality attributes</th>
</tr>
</thead>
</table>
| Normalization                             | 1. Correctness  
2. Maintainability  
3. Performance efficiency            |
| Indexes                                   | 1. Performance efficiency  
2. Maintainability             |
| Partitioning                              | 1. Performance efficiency  
2. Reliability  
3. Security             |
| Views                                     | 1. Performance  
2. Security                |
| Hardware and software choices             | 1. Performance efficiency  
2. Compatibility  
3. Maintainability  
4. Scalability  
5. Reliability |
Conclusion and future work

Even if logical design principles are fulfilled successfully, it can carry a debt that can be reflected in performance efficiency and other qualities of the system. This can be one of the reasons behind developers “Intentionally” carrying a positive debt to capture some benefits that will outweigh the debt on the system.

In the future we intend to elaborate on the developed taxonomy to consider how we can quantify the principal and interest of the debts.
Email: Mesh55@gmail.com