Database project

Database for a Dog Shelter Management

Costin Radu Ionuț

Universitatea Tehnică din Cluj-Napoca
Centru Universitar Nord Baia Mare
Specializarea Calculatoare

Abstract

This article presents how to create a database project that administrates a dog shelter and explains its implementation together with the mode of operation. The article explains how to make a project composed of 2 databases complying with normal forms, each with multiple tables containing primary, foreign, and unique keys; autoincrement functions; not null; current_timestamp(); default change; joins, views, procedures, etc.

The paper is addressed to anyone who is interested in databases or is curious about reading this presentation, as its main focus is to present the project in a simple way for better comprehension.
Keywords: Database, Table, SQL, Primary Key, Foreign Key, Unique Key, Dates, Alter Table, Constrains

1 Introduction

At first, just like Neufeld and Cornog [NC86] deduced, computers were only used for arithmetic calculations. But because the quickness of computation has accelerated and the programming languages have evolved, a substantial volume of data needs to be processed in a small quantum of time.
A database stores all of the vital data needed to create and maintain an application, which is why it is so significant and helpful.

1.1 Necessary tools

To finish the database, we should have the accompanying projects installed:

- **XAMPP**: Mearaj et al. [MMK18] said that the Apache web server, MySQL database (really MariaDB), PHP, and Perl (as command-line executables and Apache modules) are all included with XAMPP, which is a software distribution that runs on Windows, Mac OS X, and Linux. PHP and MySQL may be integrated without any setup.
- **BDeaver**: Anbumani et al. [AGK$^+$21] suggest that developers, database administrators, analysts, and anybody else who needs to deal with databases may use this free multi-platform database tool. MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, and more common databases are supported.

2 Related work

It is understood from Florescu and Kossmann's [FK09] work that a database is a collection of data specially organized for rapid search and retrieval by a computer and is structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations.

3 Normalization of the database

Before implementing the project, we need to ensure through the relational entity-relationship model that the database is normalized, respecting the 4 normal forms:

1. **1NF**: As Hillyer [Hil03] pointed out, the data must be unique and atomic. Each record ought to be depicted so that it will, in general, be perceived unambiguously by a primary key;
2. **2NF**: As shown in Kent's [Ken83] paper, the elements of a table must be functionally dependent on each part of the primary key and must adhere to 1FN;
3. **3NF**: All non-primary fields must be functionally dependent on the primary fields and must meet the requirements of 1FN and 2FN;
4. **BCNF**: From Arenas and Libkin's [AL04] work, it can be observed that for each functional dependency between A and B, A should be a super key and the table should already be in 3FN, 2FN, and 1FN.

4 Project implementation

4.1 Creating the database, tables, setting constraints and inserting data

- To build a database, select the host server where the project will be implemented by starting MySQL from XAMPP, then open BDeaver and choose the option **"Create New Database"** from the open menu by right-clicking and giving the chosen name.
- Next, Beaulieu [Bea09] points out that to insert the tables together with the corresponding columns, select **"SQL editor"—"Open SQL Console"** and insert them into the database using the SQL command:

```
CREATE TABLE TableName (
    column1 data_type,
    ...,
    columnN data_type);
```

- According to Houkjr et al. [HTW06], what values a column can contain is decided by the data type. Therefore, it is important to specify the correct data type, otherwise many errors like the Invalid Data Type Error may occur. The three main data types in SQL are:
  1. **String**: CHAR(size), VARCHAR(size), ENUM(val1, val2, val3, ...);
  2. **Numeric**: BOOLEAN, INT(size), FLOAT(p),
  3. **Date and time**: DATE, DATETIME, TIMESTAMP, TIME, YEAR.
     Portrayed by Gdalyahu et al. [GWW01], in a database, images can be inserted too, using the longblob data type.
- Notwithstanding, the SQL command applied is "DROP" if a table should be erased. For example, `DROP TABLE TableName;`

- Each table requires the presence of a single primary key, about which Shah et al. [SSK+11] confirm that it has typically set the constraint auto increment. A primary key unambiguously identifies each file in a table, can't include NULL values, and ought to be unique.
  The syntax to add a primary key:

  ```
  ALTER TABLE TableName
  ADD PRIMARY KEY (ColumnName) NOT NULL AUTO_INCREMENT;
  ```

- It can be observed from Sharma and Dave's publication [SD12] that the unique keys are set when we do not want duplicate data in the table (such as phone numbers). The primary key is always unique by definition. However, a table could have diverse unique keys rather than the primary key.
  What comes up next is how the unique key is set:

  ```
  ALTER TABLE TableName
  ADD UNIQUE (ColumnName);
  ```

- Some columns want to have special constraints, so for the default data to be the one we pick out to be, the subsequent syntax executed is:

  ```
  ALTER TABLE TableName
  ALTER columnName SET DEFAULT 'value';
  ```

- In some cases, it's essential to register the moment of data insertion; this is always done mechanically by the function **CURRENT_TIMESTAMP();**
- Anley [Anl02] affirms that after completing the column insertion into the table, the data is pushed into the column from the SQL editor according to the following pattern:

  ```
  INSERT INTO TableName (column1, ..., colummnN)
  VALUES (data1, ..., dataN);.
  ```

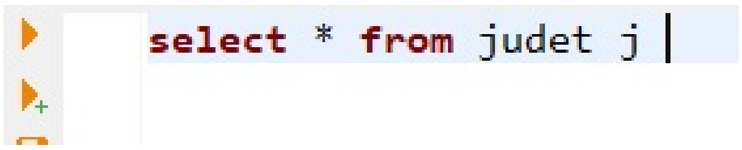
  The aim was to enter at least 20 records into each table.
- The next step within the implementation method is to attach the tables employing a foreign key, which Blaha [Bla05] declares is a constraint that refers to the primary key in another table and is used to stop actions that might break the links between tables. This ought to be conceivable by exploiting the syntax:

  ```
  ALTER TABLE TableName1
  ADD CONSTRAINT FK_ForeignKeyName
  FOREIGN KEY (ForeignKeyColumn) REFERENCES
  TableName2(PrimaryKeyTable2);
  ```

4.2 Joins, views, built-in functions, operators

Creating joins

It is deduced from Slutz's [Slu98] studies that the SQL SELECT statement is used to select data from a database: SELECT ColumnName FROM TableName;



```
select * from judet j
```

Figure 1: Select Example

Still, there are occasions when further processing is needed.
According to Gadkari et al. [GNM14], the join clause in SQL consolidates records from at least two tables dependent on a common section.

1. **Inner join** keyword selects statistics that have matching values in each table.

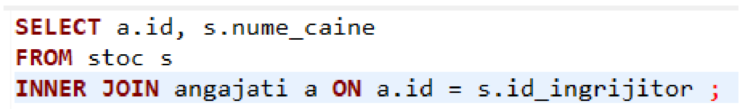
   SELECT ColumnName(s)
   FROM TableName1
   INNER JOIN TableName2
   ON TableName1.ColumnName = TableName2.ColumnName;

2. **Left join** keyword returns the records from the left table (TableName1) and also the matching records from the proper table (TableName2).

   SELECT ColumnName(s)
   FROM TableName1
   LEFT JOIN TableName3
   ON TableName1.ColumnName = TableName1.ColumnName;

3. **Right join** keyword returns all records from the left table (TableName1) and also the matching records from the proper table (TableName2).

   SELECT ColumnName(s)
   FROM TableName1
   RIGHT JOIN TableName3
   ON TableName1.ColumnName = TableName1.ColumnName;



```
SELECT a.id, s.nume_caine
FROM stoc s
INNER JOIN angajati a ON a.id = s.id_ingrijitor ;
```

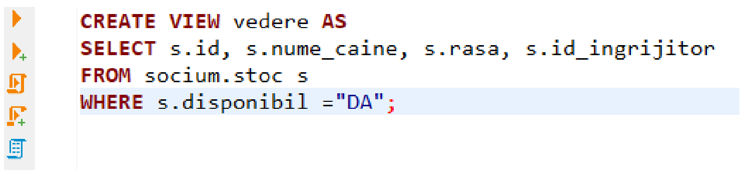
Figure 2: A Simple Example of an Inner Join

Creating views

A view, according to Sequeda et al. [SDM09], is a virtual table in SQL that depends on the outcome set of a SQL articulation and involves lines and columns, equivalent to a customary table. The fields are the bones of at least one genuine table from the information base.
The SQL structure for making a view is:

```
CREATE VIEW ViewName AS
SELECT ColumnName, ColumnName2, ...
FROM TableName
WHERE Condition;
```

The WHERE clause is harnessed to filter records.
Furthermore, studies like the ones of Egenhofer [Ege94] tell us about the LIKE operator, which is used in a WHERE clause to search for a specified pattern in a column.
The two trump cards regularly utilized related to the LIKE operator are the percent sign (%) and the highlight sign (_);



```
CREATE VIEW vedere AS
SELECT s.id, s.nume_caine, s.rasa, s.id_ingrijitor
FROM socium.stoc s
WHERE s.disponibil ="DA";
```

Figure 3: Example of View

It is essential to realize that the columns picked should have various names.

Useful built-in functions

According to Lu et al. [LCW93], SQL has several intrinsic functions for performing calculations on data. Some of the functions are:

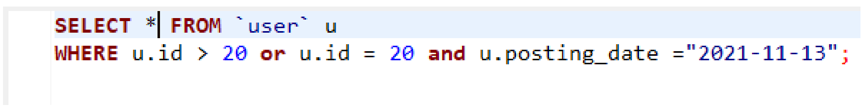
- MIN() - Reveal the lowest value;
- MAX() - Returns the biggest value;
- COUNT() - Returns a quantity of rows;
- AVG() - Returns the common value;
- SUM() - Shows the sum;
- FIRST() - Returns the first insert;
- LAST() - Returns the latest value;
- LEN() - Returns the duration of a textual content field.

Operators

Blunschi et al. [BJK+12] present in their work some of these comparison operators:

- Comparison operators
  - =, <=>, >=, >, <=, < <>, !=;

  - BETWEEN ... AND ... - Whether a value is within a range of values;
  - GREATEST() - Return the argument with the largest size;
  - IN() - Whether a value is within a set of values;
  - IS - Test a value against a boolean;
  - IS NOT - tests the NULL value;
  - NOT IN() - Whether a value is not within a bloc of values;
  - ISNULL() - Determines whether or not the argument is NULL.
- Arithmetic operators
  - +, Add;
  - -, Subtract;
  - *, Multiply;
  - /, Divide;
  - %, Modulo.
- Logical operators
  - All, TRUE if all of the subquery values meet the condition;
  - AND, TRUE if all the conditions separated by AND is TRUE;
  - ANY, TRUE if any of the subquery values meet the condition;
  - OR, TRUE if any of the conditions separated by OR is TRUE;
  - SOME, TRUE if any of the subquery values meet the condition.
- Bitwise operators
  - &, Bitwise AND;
  - |, Bitwise OR;
  - ^, Bitwise exclusive OR;
- Compound operators
  - +=, Add equals;
  - -=, Subtract equals;
  - *=, Multiply equals;
  - /=, Divide equals;
  - %=, Modulo equals.



```
SELECT * FROM `user` u
WHERE u.id > 20 or u.id = 20 and u.posting_date ="2021-11-13";
```

Figure 4: Simple example of using operators

4.3 Stored Procedures

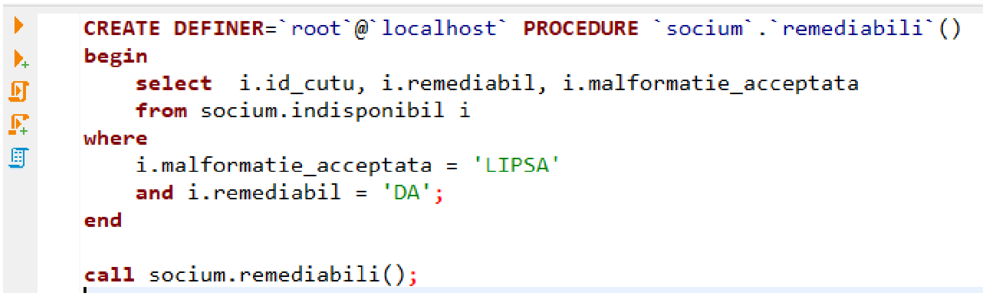
Wei et al.'s [WMK06] say that a stored procedure is a code that can be saved and reused. Along these lines, assuming you have a SQL query, protect it as a stored procedure, and thereafter, just call it to run it.

CREATE DEFINER='DB_user'@'DBhost' PROCEDURE 'DBName'.'ProcedureName'()
begin

SELECT ColumnName1, ColumnName2,...ColumnNameN
FROM TableName
WHERE
ColumnName1 *Condition*
AND ColumnName2 *Condition*,
...
AND ColumnNameN *Condition*;
end

To execute a stored procedure, it utilizes the "CALL" command.

CALL DBName.ProcedureName();



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `socium`.`remediabili`()
begin
    select  i.id_cutu, i.remediabil, i.malformatie_acceptata
    from socium.indisponibil i
where
    i.malformatie_acceptata = 'LIPSA'
    and i.remediabil = 'DA';
end

call socium.remediabili();
```

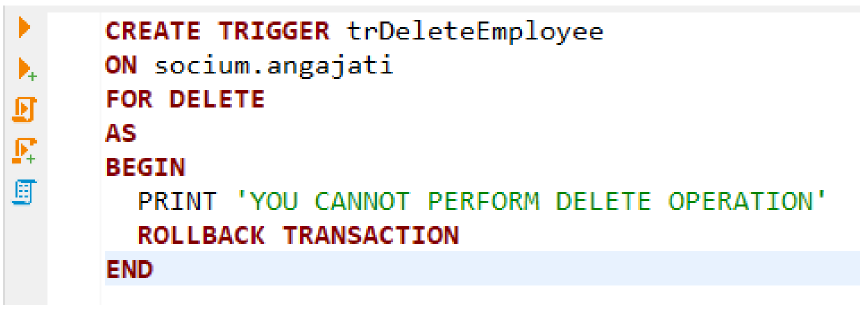
Figure 5: Procedure Example

4.4 Triggers

Cochrane et al. [CPM96] said that a SQL Server trigger is a bit of procedural code, like a saved manner that's best achieved when a given occasion happens. In other words, if you need to execute some preprocessing or postprocessing logic prior to or after an Insert, Update, or Delete in a table, you should definitely use Triggers in SQL Server.

CREATE/ALTER TRIGGER TriggerName
ON TableName/ViewName
[WITH <Options>]
FOR | AFTER | INSTEAD OF { [INSERT], [UPDATE] , [DELETE] }
AS
BEGIN
<BODDY>
END

Next, it presents a trigger that prevents an employee from being deleted.



```
CREATE TRIGGER trDeleteEmployee
ON socium.angajati
FOR DELETE
AS
BEGIN
  PRINT 'YOU CANNOT PERFORM DELETE OPERATION'
  ROLLBACK TRANSACTION
END
```

Figure 6: Trigger Example

5 Functionality

5.1 First database: Socium

One of the sections of the first database, "Socium" is to monitor the user and his actions. A user can post messages (if he is not logged in, contact details are required, otherwise they are taken from the ID) or place orders for the adoption of a dog from stock.

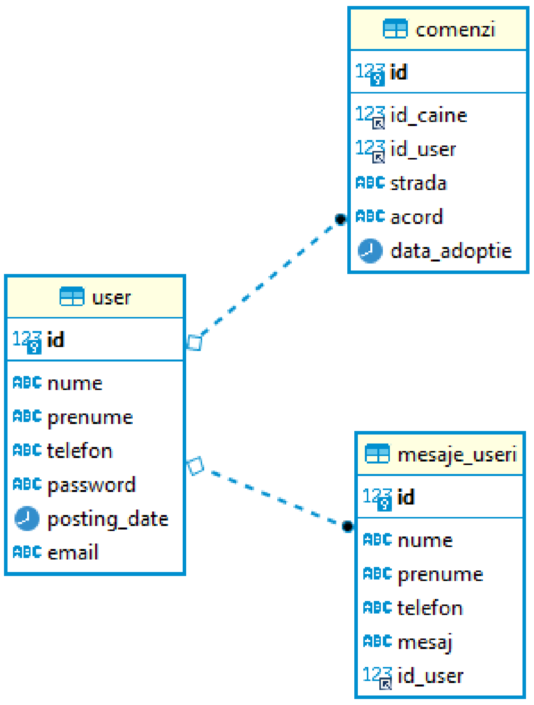


Figure 7: User Management

It additionally includes a component to monitor employees and canines.
Employee management is done from the "angajati" table. Each employee cares for a dog if the "job" column contains the value 1, or delivers orders if it has the value 0.

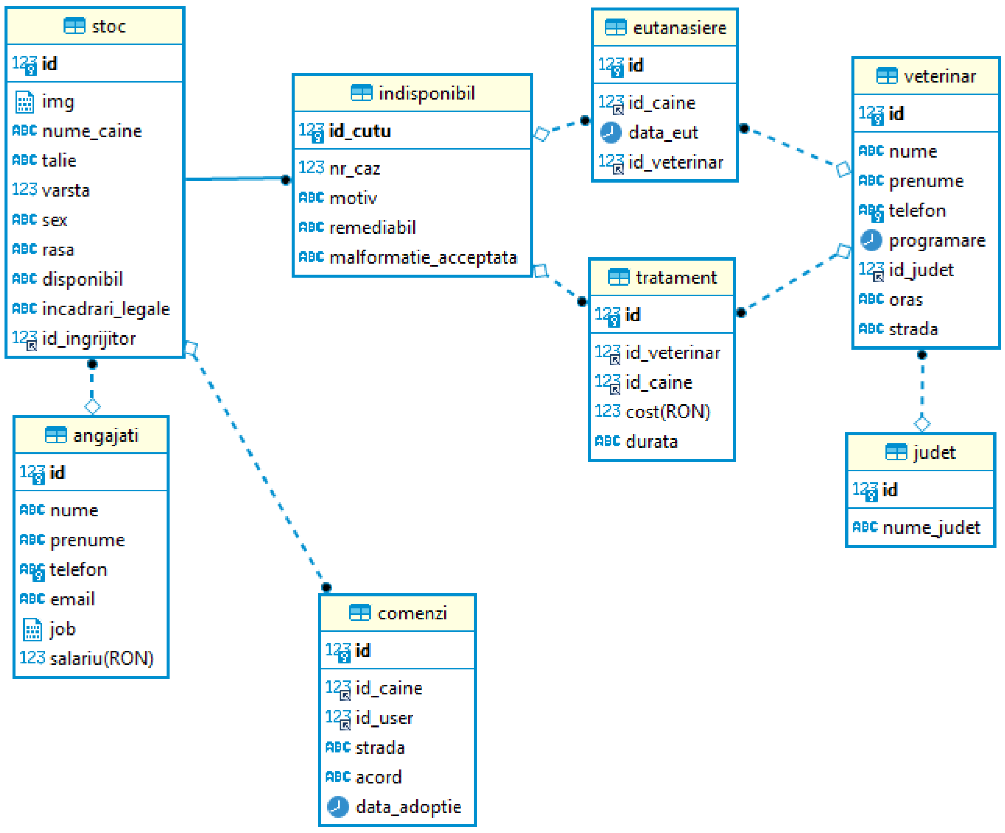


Figure 8: Employee and canine management

The data of each dog is stored in the "stoc" table. If the canines are accessible, they will show up in the adoption segment of the site; if they are not available, they appear in the "indisponibil" table, where there are 2 possibilities:

1. The irremediable case: except for the accepted malformations, the irretrievable case will be sent to euthanasia. If it is an accepted malformation, it is sent back to adoption, specifying that.
2. The remediable case: is sent to treatment, where treatment data is stored. Both treatment and euthanasia are done by a veterinarian in a particular county at a particular address.

If a dog is selected by a user for adoption, the "disponibil" column receives the "comanda" value, and the data required for adoption is taken over from the rest of the database and added to the table called "comanda".
In addition, we have the "admin" table, which has the sole role of helping to log the admin on the site.

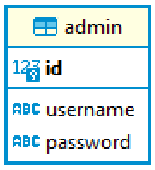


Figure 9: The "admin" table

5.2 Second database: Legislatie

In the "legislatie" database, the legal classification of dogs is kept.

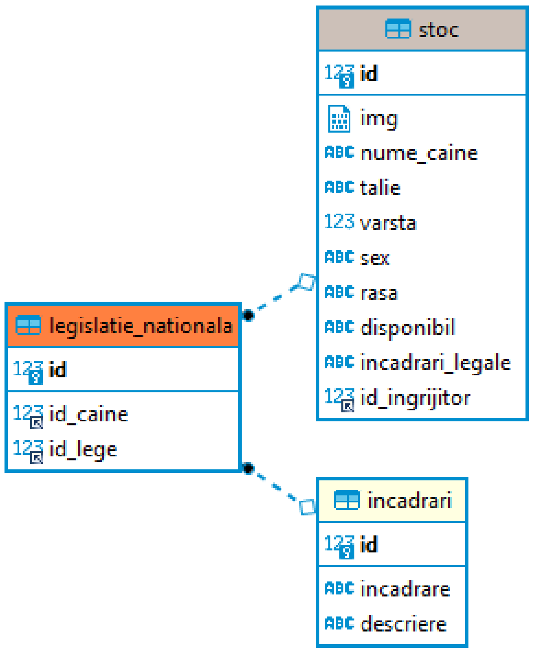


Figure 10: Legally Framed

We have the table "incadrari", which specifies which categories of dogs exist: magnifying glass, attack, aggressive and non-aggressive, and then the table "legislatie_nationala", which classifies dogs from stock in a category using a foreign key.

Conclusion

As depicted in Hammer and Michael's work [HML81], a database is an easy-to-manage data collection that helps you manage an application, optimize work productivity, and

even improve user experiences. Therefore, establishing the database could be a beneficial tool for any organization, website, store, or even a multitude of other uses.
This paper presents a method for making a database alongside the clarifications expected to finish it and gives data to perform different actions on it. The project was conducted in DBeaver, which runs on the XAMPP server on a laptop with an Intel Core i5-9300H CPU running at 2.40 GHz with sixteen GB of RAM.

References

[AGK^{+}21] V Anbumani, V Geetha, V Praveen Kumar, D Sabaree, and K Sivanantham. Development of cloud-based agriculture marketing system with intellectual weigh machine. In *IOP Conference Series: Materials Science and Engineering*, volume 1055, page 012016. IOP Publishing, 2021.

[AL04] Marcelo Arenas and Leonid Libkin. A normal form for xml documents. *ACM Transactions on Database Systems (TODS)*, 29(1):195–232, 2004.

[Anl02] Chris Anley. Advanced sql injection in sql server applications. 2002.

[Bea09] Alan Beaulieu. *Learning SQL: master SQL fundamentals.* " O'Reilly Media, Inc.", 2009.

[BJK^{+}12] Lukas Blunschi, Claudio Jossen, Donald Kossman, Magdalini Mori, and Kurt Stockinger. Soda: Generating sql for business users. *arXiv preprint arXiv:1207.0134*, 2012.

[Bla05] Michael Blaha. Referential integrity is important for databases. *Modelsoft Consulting Corp*, 2005.

[CPM96] Roberta Cochrane, Hamid Pirahesh, and Nelson Mattos. Integrating triggers and declarative constraints in sql database systems. In *VLDB*, volume 96, pages 3–6. Citeseer, 1996.

[Ege94] Max J. Egenhofer. Spatial sql: A query and presentation language. *IEEE Transactions on knowledge and data engineering*, 6(1):86–95, 1994.

[FK09] Daniela Florescu and Donald Kossmann. Rethinking cost and performance of database systems. *ACM Sigmod Record*, 38(1):43–48, 2009.

[GNM14] Ajinkya Gadkari, VB Nikam, and BB Meshram. Implementing joins over hbase on cloud platform. In *2014 IEEE International Conference on Computer and Information Technology*, pages 547–554. IEEE, 2014.

[GWW01] Yoram Gdalyahu, Daphna Weinshall, and Michael Werman. Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, 2001.

[Hil03] Mike Hillyer. An introduction to database normalization. *MySQL AB*, 2003.

[HML81] Michael Hammer and Dennis Mc Leod. Database description with sdm: A semantic database model. *ACM Transactions on Database Systems (TODS)*, 6(3):351–386, 1981.

[HTW06] Kenneth Houkjær, Kristian Torp, and Rico Wind. Simple and realistic data generation. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1243–1246, 2006.

[Ken83] William Kent. A simple guide to five normal forms in relational database theory. *Communications of the ACM*, 26(2):120–125, 1983.

[LCW93] Hongjun Lu, Hock Chuan Chan, and Kwok Kee Wei. A survey on usage of sql. *ACM SIGMOD Record*, 22(4):60–65, 1993.

[MMK18] Insha Mearaj, Piyush Maheshwari, and Maninder Jeet Kaur. Data conversion from traditional relational database to mongodb using xampp and nosql. In *2018 Fifth HCT Information Technology Trends (ITT)*, pages 94–98. IEEE, 2018.

[NC86] M Lynne Neufeld and Martha Cornog. Database history: From dinosaurs to compact discs. *Journal of the American society for information science*, 37(4):183–190, 1986.

[SD12] Vatika Sharma and Meenu Dave. Sql and nosql databases. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8), 2012.

[SDM09] Juan F Sequeda, Rudy Depena, and Daniel P Miranker. Ultrawrap: Using sql views for rdb2rdf. In *8th International Semantic Web Conference (ISWC2009), Washington DC, USA*, 2009.

[Slu98] Donald R Slutz. Massive stochastic testing of sql. In *VLDB*, volume 98, pages 618–622. Citeseer, 1998.

[SSK$^+$11] Shetal Shah, S Sudarshan, Suhas Kajbaje, Sandeep Patidar, Bhanu Pratap Gupta, and Devang Vira. Generating test data for killing sql mutants: A constraint-based approach. In *2011 IEEE 27th International Conference on Data Engineering*, pages 1175–1186. IEEE, 2011.

[WMK06] Kei Wei, Muthusrinivasan Muthuprasanna, and Suraj Kothari. Preventing sql injection attacks in stored procedures. In *Australian Software Engineering Conference (ASWEC'06)*, pages 8–pp. IEEE, 2006.