MySQL Database Unit 07 – Database Normalization

Chapter 10 – How to Design a Database

Data Structure Design:

* Each table represents one object, or entity in the real-world system
* Within each table
* **Each column** stores one item of information / attribute
* **Each Row** stored one occurrence / instance of the entity
* Steps:
* 1: Identify all the data elements that need to be stores in the database
* 2: Break complex elements down into smaller components whenever that makes sense
* 3: Identify the tables that will make up the system & determine which data elements are assigned as columns in each table
* 4: Define the relationships between the tables by identifying the primary and foreign keys
* 5: **Normalize** the database to reduce data redundancy
* 6: Identify the Indexes that are needed for each table
* Entity Relationship Modeling Technique:
* Used for modeling a database system after a real-world system
* EER Diagram ??

Identify the Data Elements: **( 1 )**

* Techniques:
* Analyzing the existing system if there is one
* Documents used by the existing system

-invoices

* Evaluating comparable systems
* Interview anyone who will be using the system

Subdivide the Data Elements: **( 2 )**

* If a data element contains 2+ components 🡪 consider dividing the element into those components
* This way, there’s no need to parse the element each time it’s used
* Can easily rebuild the element when necessary by concatenating individual components
* Divide the data elements into their smallest useful values
* Divide Name 🡪 First, Last
* Divide Address 🡪 Street # & Name, City, State, Zip

Identify Tables & Assign Columns: **( 3 )**

* Identify the **main entities** for the system 🡪 **Tables**
* Usually achieved while identifying data elements in step 1
* List the possible **elements** associated w/ each entity 🡪 **Columns**
* When association isn’t obvious – can list an element under 2+ entities
* Repeated elements can be removed during normalization if needed
* Document Design Decisions: (CASE Tool)
* Italics – added data elements
* Strikethrough – removed data elements
* Asterisk – data elements included in 2+ tables

Identify Relationships: **( 4 )**

* Primary Key:
* Can use an existing column when possible
* Ensure values are unique, rarely change; short & easy to enter correctly
* Standard 🡪 create an ID column that’s incremented by 1 for each new row
* Composite Key:
* Used when 2+ columns are needed to identify each row
* When a single column might not be unique by itself
* Foreign Key:
* Added when identifying the relationship between tables / between primary & foreign
* Foreign Key column must have same data type as the Primary Key column its related to
* One-To-Many:
* Add a Foreign Key to the table on the “many” side
* Many-To-Many:
* Create a linking table to relate the 2 tables 🡪 doesn’t need a Primary Key
* Each table in the many-to-many will have a one-to-many w/ the linking table
* One-To-One:
* Data could be stored in a single table
* Columns w/ large amounts of data can be stored in a separate table
* Good for infrequently used columns; storing in separate table improves efficiency
* Both tables have same Primary Key; relate tables by their Primary Keys
* When column is needed, combine tables using a join

Enforce Relationships: **( 4 continued )**

* DBMS doesn’t always enforce the relationships indicated by the Primary & Foreign Keys
* In this case – deleting, inserting, or updating violate referential integrity – orphaned rows
* Enforce Referential Integrity:
* Use **Declarative Referential Integrity 🡪 define Foreign Key Constraints**
* Define how referential integrity is enforced:
* Return an error
* Delete the related rows in the Foreign Key table
* Set the Foreign Key values in the related rows to null

Normalization: **( 5 )**

* Process used to separate the data in the data structure into related tables
* To normalize a structure 🡪 apply the normal forms in sequence
* Unnormalized Structure:
* Each column can contain info about 2+ entities
* Can contain repeating columns, columns w/ repeating values, data repeated in 2+ rows
* Normalized Structure:
* Each table contains info about 1 entity
* Each piece of info is stored in exactly one place

Identify Indexed Columns: **( 6)**

* **Index:** copy of selected columns 🡪 allows DBMS to locate info / rows directly & quickly
* Has pointers that direct the system to a specific row
* MySQL automatically creates indexes for the Primary & Foreign Keys in each table
* Create an index when:
* The column is used frequently in search conditions or joins
* The column is updated frequently
* The column contains a large number of distinct value
* Composite index: includes 2+ columns
* \*Indexes must be updated each time you add, delete, or update a row – only use when needed

Data Structure Normalization:

* 3rd Form eliminates **Transitive Dependencies:** 1 column depends on another, which depends on a third
* 4th Form eliminates Multivalued Dependencies
* The Important Normal Forms: (There are 7 Total)
* 1NF:
* 2NF:
* 3NF:

First Normal Form:

* Eliminate repeating Column Values
* Cannot contain repeating columns that represent a set of values
* In 1NF, tables often have repeating Row Values

Second Normal Form:

* Applies only to tables w/ Composite Primary Keys
* Every non-key column must depend on the entire Primary Key
* If they don’t 🡪 indicates that the table contains info for 1+ entity
* Move columns that don’t depend on the entire Primary Key to another table & establish relationship between the 2 tables
* This form removes redundant row data

Third Normal Form:

* Every non-key column must depend only on the Primary Key
* If they don’t 🡪 indicates that the column is assigned to the wrong table OR it can be computed from other columns in the table (derived data)

Notes from:

<https://www.slideshare.net/jagaarj/database-design-normalization>

<http://download.nust.na/pub6/mysql/tech-resources/articles/intro-to-normalization.html>

SPREADSHEET SYNDROME:

* The tendency for the developer to lump every possible piece of information into as few tables as possible, often into a single table
* A schema doing this is subject to data redundancies, data anomalies, & various inefficiencies
* The solution for Spreadsheet Syndrome is **Database Normalization**

DATABASE NORMALIZATION:

* The process by which we efficiently organize data to achieve these goals:
* Eliminate redundancy
* Ensure data is stored in the correct table
* Eliminating the need for restructuring the database when data is added
* Levels of a Normal Form:
* In order to achieve one level of normal form, each previous level must be met
* Third Normal Form is sufficient for most typical applications
* First Normal Form (1NF):

\*Getting rid of repeat data / redundancy

* There are no repeating or duplicate fields
* Each cell contains only a single value
* Each record (row) is unique – meaning its identified by a primary key
* Second Normal Form (2NF):
* All non-key fields depend on all components of the primary key
* \*Each column depends on the entire primary key

-This is guaranteed when the primary key is a single field

\*The primary key can be made up of one or more attributes of the table – called Prime Attributes

\*Multiple Prime Attributes form a Composite Key – a Primary Key w/ 1+ field

* Third Normal Form (3NF):

\*Find all the non-prime attributes in the table & ensure they depend the key, whole key & nothing but the key

* No non-key field depends upon another
* All non-key fields depend only on the primary key