



coding-bootcamps.com

Introduction to SQL Programming

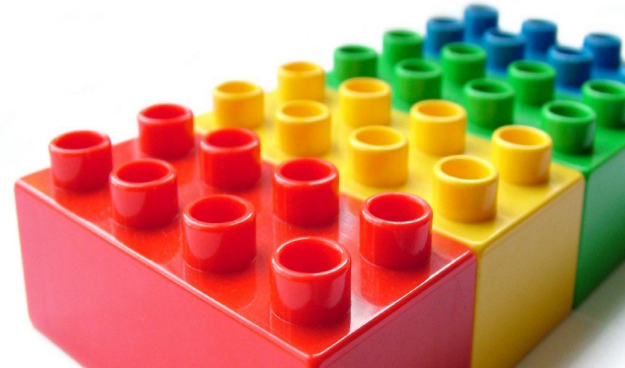


Coding Bootcamps

By Jim Sullivan from [Coding Bootcamps](#)

Prerequisite

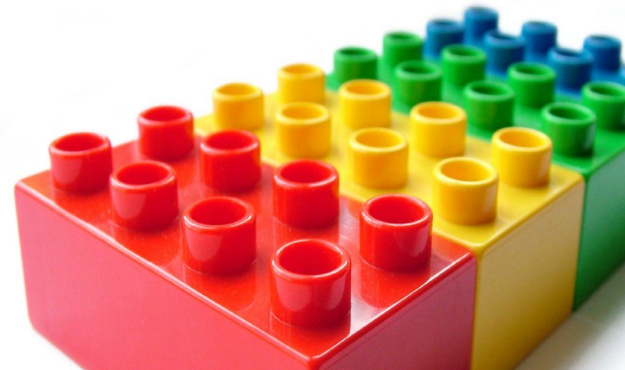
Introduction to Database Design



Recap

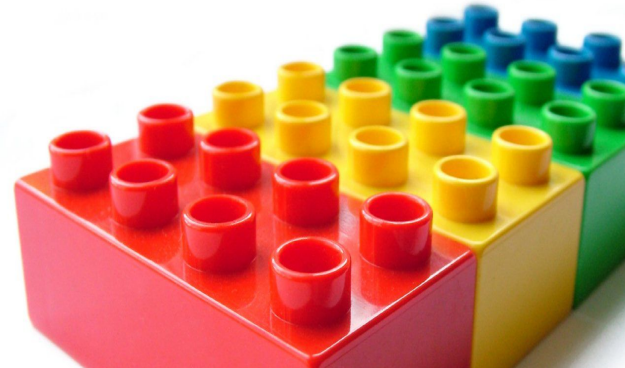
In our previous course, we covered the following topics:

- 1- Before the Advent of Database Systems
- 2- Fundamental Concepts
- 3- Characteristics and Benefits of a Database
- 4- Types of Data Models
- 5- Data Modeling



Recap...

- 6- Classification of Database Management Systems
- 7- The Relational Data Model
- 8- The Entity Relationship Data Model
- 9- Integrity Rules and Constraints
- 10- ER Modeling



Functional Dependencies

Session 1

Session 1

- Functional Dependencies
- Rules of Functional Dependencies
- Inference Rules
- Dependency Diagram



Functional Dependency

- A *functional dependency* (FD) is a relationship between two attributes, typically between the PK and other non-key attributes within a table. For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y

Roles of Functional Dependency

Inference Rules

Armstrong's axioms are a set of inference rules used to infer all the functional dependencies on a relational database.

Union

This rule suggests that if two tables are separate, and the PK is the same, you may want to consider putting them together.

Roles of Functional Dependency

Decomposition

Decomposition is the reverse of the Union rule. If you have a table that appears to contain two entities that are determined by the same PK, consider breaking them up into two tables.

Dependency Diagram

A dependency diagram illustrates the various dependencies that might exist in a *non-normalized table*.

Normalization

Session 2

Session 2

- What Is Normalization?
- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)



Normalization should be part of the database design process. However, it is difficult to separate the normalization process from the ER modeling process so the two techniques should be used concurrently.

Use an Entity Relation Diagram (ERD) to provide the big picture, or macro view, of an organization's data requirements and operations. This is created through an iterative process that involves identifying relevant entities, their attributes and their relationships.

*Normalization procedure focuses on characteristics of specific entities and represents the **micro** view of entities within the ERD.*

Normalization is the branch of relational theory that provides design insights. It is the process of determining how much redundancy exists in a table. The goals of normalization are to:

- Be able to characterize the level of redundancy in a relational schema
- Provide mechanisms for transforming schemas in order to remove redundancy

Normalization forms

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)

Normalization and Database Design

During the normalization process of database design, make sure that proposed entities meet required normal form before table structures are created. Many real-world databases have been improperly designed or burdened with anomalies if improperly modified during the course of time. You may be asked to redesign and modify existing databases. This can be a large undertaking if the tables are not properly normalized.

Database Development Process

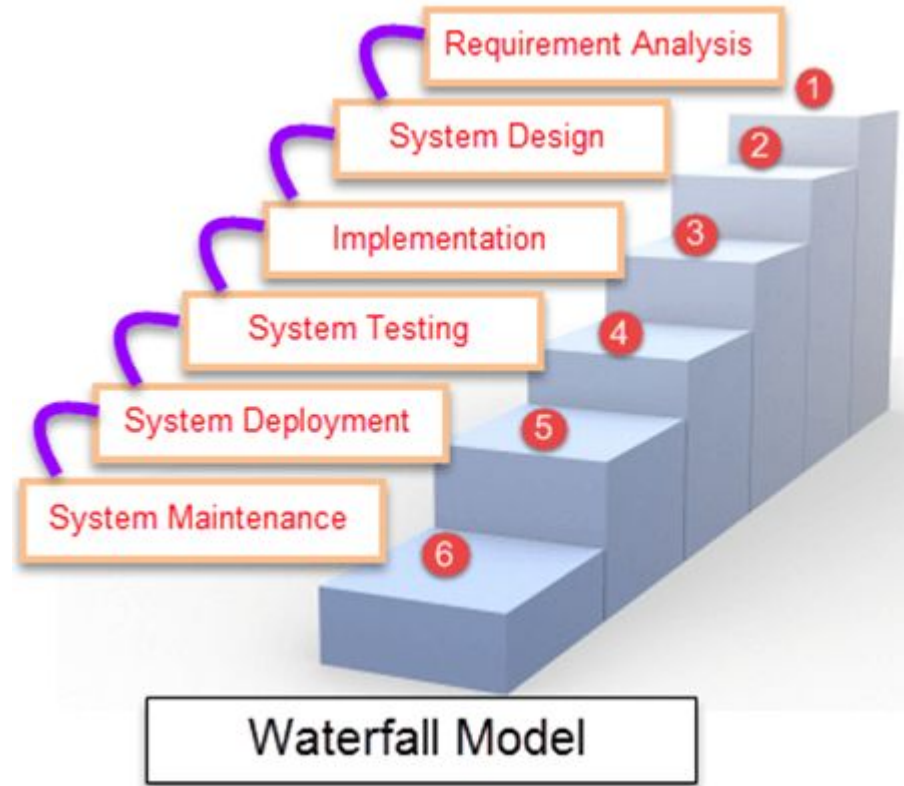
Session 3

Session 3

- Software Development Life Cycle – Waterfall
- Database Life Cycle
- Requirements Gathering & Analysis
- Logical Design
- Implementation & Populating the Database



Software Development Life Cycle – Waterfall



Database Life Cycle

We can use the waterfall cycle as the basis for a model of database development that incorporates three assumptions:

- 1- We can separate the development of a database – that is, specification and creation of a schema to define data in a database – from the user processes that make use of the database.

Database Life Cycle...

2- We can use the three-schema architecture as a basis for distinguishing the activities associated with a schema.

3- We can represent the constraints to enforce the semantics of the data once within a database, rather than within every user process that uses the data.

Database Life Cycle Steps

1. Requirements Gathering
2. Analysis
3. Logical Design
4. Implementation
5. Realizing the Design
6. Populating the Database

Guidelines for Developing an ER Diagram

1. Document all entities discovered during the information-gathering stage.
2. Document all attributes that belong to each entity. Select candidate and primary keys. Ensure that all non-key attributes for each entity are full-functionally dependent on the primary key.
3. Develop an initial ER diagram and review it with appropriate personnel. (Remember that this is an iterative process.)
4. Create new entities (tables) for multivalued attributes and repeating groups. Incorporate these new entities (tables) in the ER diagram. Review with appropriate personnel.
5. Verify ER modeling by normalizing tables.

Database Users

Session 4

Session 4

- End Users



End Users

- Application user
- Sophisticated user
- Application Programmers
- Database Administrators (DBA)

SQL Structured Query Language

Session 5

Session 5

- Create Database
- Create tables
- Data Type
- Optional Column Constraints
- Table Constraints
- User Defined Types



Create Database

CREATE DATABASE ...

Create tables

```
CREATE TABLE <tablename>
```

```
(
```

```
  ColumnName, Datatype, Optional Column Constraint,
```

```
  ColumnName, Datatype, Optional Column Constraint,
```

```
  Optional table Constraints
```

```
);
```

Data Types

- **Int** –Integer (whole number) data from -2^{31} (-2,147,483,648) through $2^{31} - 1$ (2,147,483,647)
- **Smallint** –Integer data from 2^{15} (-32,768) through $2^{15} - 1$ (32,767)
- **Tinyint**–Integer data from 0 through 255
- **Decimal** –Fixed precision and scale numeric data from $-10^{38} - 1$ through 10^{38}
- **Numeric** –A synonym for decimal
- **Timestamp** –A database-wide unique number
- **Uniqueidentifier** –A globally unique identifier (GUID)

For more data types, read your course materials

Optional Column Constraints

The Optional Column Constraints are NULL, NOT NULL, UNIQUE, PRIMARY KEY and DEFAULT, used to initialize a value for a new record. The column constraint NULL indicates that null values are allowed, which means that a row can be created without a value for this column. The column constraint NOT NULL indicates that a value must be supplied when a new row is created.

Table Constraints

IDENTITY constraint

UNIQUE constraint

The UNIQUE constraint prevents duplicate values from being entered into a column.

- Both PK and UNIQUE constraints are used to enforce entity integrity.
- Multiple UNIQUE constraints can be defined for a table.
- When a UNIQUE constraint is added to an existing table, the existing data is always validated.
- A UNIQUE constraint can be placed on columns that accept nulls. Only one row can be NULL.
- A UNIQUE constraint automatically creates a unique index on the selected column.

Table Constraints...

FOREIGN KEY constraint

- The FOREIGN KEY (FK) constraint defines a column, or combination of columns, whose values match the PRIMARY KEY (PK) of another table.
- Values in an FK are automatically updated when the PK values in the associated table are updated/changed.
- FK constraints must reference PK or the UNIQUE constraint of another table.
- The number of columns for FK must be same as PK or UNIQUE constraint.
- If the WITH NOCHECK option is used, the FK constraint will not validate existing data in a table.
- No index is created on the columns that participate in an FK constraint.

Table Constraints...

CHECK constraint

The CHECK constraint restricts values that can be entered into a table.

- It can contain search conditions similar to a WHERE clause.
- It can reference columns in the same table.
- The data validation rule for a CHECK constraint must evaluate to a boolean expression.
- It can be defined for a column that has a rule bound to it.

Table Constraints...

DEFAULT constraint

The DEFAULT constraint is used to supply a value that is automatically added for a column if the user does not supply one.

- A column can have only one DEFAULT.
- The DEFAULT constraint cannot be used on columns with a timestamp data type or identity property.
- DEFAULT constraints are automatically bound to a column when they are created.

User Defined Types

User defined types are always based on system-supplied data type. They can enforce data integrity and they allow nulls.

ALTER TABLE

You can use ALTER TABLE statements to add and drop constraints.

ALTER TABLE allows columns to be removed.

When a constraint is added, all existing data are verified for violations.

DROP TABLE

The DROP TABLE will remove a table from the database. Make sure you have the correct database selected.

SQL Data Manipulation Language

Session 6

Session 6

- SELECT SQL statements
- INSERT SQL statements
- UPDATE SQL statements
- DELETE SQL statements
- SQL Built-in Functions
- Joining Tables



The SQL data manipulation language (DML) is used to query and modify database data. In this Session, we will describe how to use the SELECT, INSERT, UPDATE, and DELETE SQL DML command statements, defined below.

- *SELECT* – to query data in the database
- *INSERT* – to insert data into a table
- *UPDATE* – to update data in a table
- *DELETE* – to delete data from a table

In the SQL DML statement:

- Each clause in a statement should begin on a new line.
- The beginning of each clause should line up with the beginning of other clauses.
- If a clause has several parts, they should appear on separate lines and be indented under the start of the clause to show the relationship.
- Upper case letters are used to represent reserved words.
- Lower case letters are used to represent user-defined words.

SELECT SQL statements

```
SELECT DISTINCT item(s)  
FROM table(s)  
WHERE predicate  
GROUP BY field(s)  
ORDER BY fields
```

SELECT SQL statements

- SELECT statement alone and with WHERE criteria
- Using wildcards in the LIKE clause
- SELECT statement with ORDER BY clause
- SELECT statement with GROUP BY clause
- Restricting rows with HAVING

INSERT SQL statements

The *INSERT statement* adds rows to a table. In addition,

- INSERT specifies the table or view that data will be inserted into.
- Column_list lists columns that will be affected by the INSERT.
- If a column is omitted, each value must be provided.
- If you are including columns, they can be listed in any order.
- VALUES specifies the data that you want to insert into the table. VALUES is required.

INSERT SQL statements

```
INSERT [INTO] Table_name | view name [column_list]  
DEFAULT VALUES | values_list | select statement
```

INSERT SQL statements

When inserting rows with the INSERT statement, these rules apply:

- Inserting an empty string (' ') into a varchar or text column inserts a single space.
- All char columns are right-padded to the defined length.
- All trailing spaces are removed from data inserted into varchar columns, except in strings that contain only spaces. These strings are truncated to a single space.
- If an INSERT statement violates a constraint, default or rule, or if it is the wrong data type, the statement fails and SQL Server displays an error message.

INSERT SQL statements

When you specify values for only some of the columns in the column_list, one of three things can happen to the columns that have no values:

1. A default value is entered if the column has a DEFAULT constraint, if a default is bound to the column, or if a default is bound to the underlying user-defined data type.
2. NULL is entered if the column allows NULLs and no default value exists for the column.
3. An error message is displayed and the row is rejected if the column is defined as NOT NULL and no default exists.

UPDATE statement

The *UPDATE statement* changes data in existing rows either by adding new data or modifying existing data.

Including subqueries in an UPDATE statement

DELETE statement

The *DELETE statement* removes rows from a record set. DELETE names the table or view that holds the rows that will be deleted and only one table or row may be listed at a time. WHERE is a standard WHERE clause that limits the deletion to select records.

```
DELETE [FROM] {table_name | view_name }  
[WHERE clause]
```

DELETE statement

The rules for the DELETE statement are:

- If you omit a WHERE clause, all rows in the table are removed (except for indexes, the table, constraints).
- DELETE cannot be used with a view that has a FROM clause naming more than one table. (Delete can affect only one base table at a time.)

Built-in Functions

There are many built-in functions in SQL Server such as:

- **Aggregate**: returns summary values
- **Conversion**: transforms one data type to another
- **Date**: displays information about dates and times
- **Mathematical**: performs operations on numeric data
- **String**: performs operations on character strings, binary data or expressions
- **System**: returns a special piece of information from the database
- **Text and image**: performs operations on text and image data

Built-in Functions

FUNCTION	DESCRIPTION
AVG	Returns the average of all the values, or only the DISTINCT values, in the expression.
COUNT	Returns the number of non-null values in the expression. When DISTINCT is specified, COUNT finds the number of unique non-null values.
COUNT(*)	Returns the number of rows. COUNT(*) takes no parameters and cannot be used with DISTINCT.
MAX	Returns the maximum value in the expression. MAX can be used with numeric, character and datetime columns, but not with bit columns. With character columns, MAX finds the highest value in the collating sequence. MAX ignores any null values.
MIN	Returns the minimum value in the expression. MIN can be used with numeric, character and datetime columns, but not with bit columns. With character columns, MIN finds the value that is lowest in the sort sequence. MIN ignores any null values.
SUM	Returns the sum of all the values, or only the DISTINCT values, in the expression. SUM can be used with numeric columns only.

Joining Tables

Joining two or more tables is the process of comparing the data in specified columns and using the comparison results to form a new table from the rows that qualify. A join statement:

- Specifies a column from each table
- Compares the values in those columns row by row
- Combines rows with qualifying values into a new row

Joining Tables

- Inner join
- Left outer join
- Right outer join
- Full outer join
- Cross join

Recap

What we have learned so far?

Next Session

- Exercises and projects



Private Coaching Sessions

Private tutoring sessions for system administrator
management- Weekly and monthly plans

Database design and SQL coding- Private tutoring sessions

Next Classes

- Introduction to Linux OS
- Learn PHP Programming
- Web Development with PHP & MySQL
- Learn Node.JS, Express.JS
and MongoDB



coding-bootcamps.com

Thank you



Coding
Bootcamps