



SQL Server Developer

Module 1: Introduction to SQL and RDBMS

Instructor: Jesse Teixeira

Where can we find help?

- Books

- **Fundamentals of database systems – 6th edition**
- **Head First SQL - Lynn Beighley**

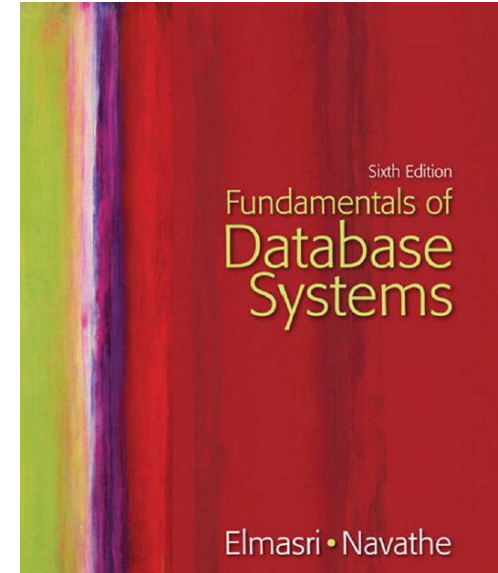
- Internet

- W3Schools

<http://www.w3schools.com/sql>

- Structured Query Language @ hit:

<http://home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf>



What will we see here?

- Introduction to SQL and its history
- SQL Process to execute tasks
- DDL – Data definition Language
- DML – Data manipulation Language
- DCL – Data Control Language
- DQL – Data Query Language
- What is RDBMS
- What is table, field, record row, column and Constraint
- Data integrity
- Database Normalization





Introduction to **S**tructured **Q**uery **L**anguage

What it is

- Structured Query Language is a computer language for storing, manipulating and retrieving data stored in a relational database.

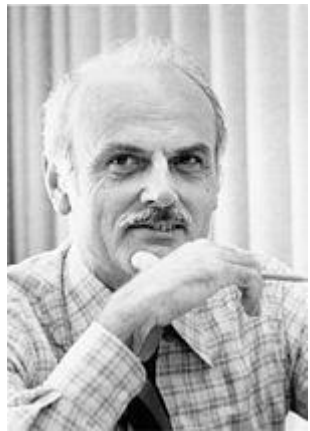
Why use it

- Allows users to :
 - Access data in relational database management systems.
 - Describe the data.
 - Define and manipulate the data in the database.
 - Create and drop databases and tables.
 - Create views, stored procedures and functions in a database.
 - Set permissions on tables, procedures and views
 - ...



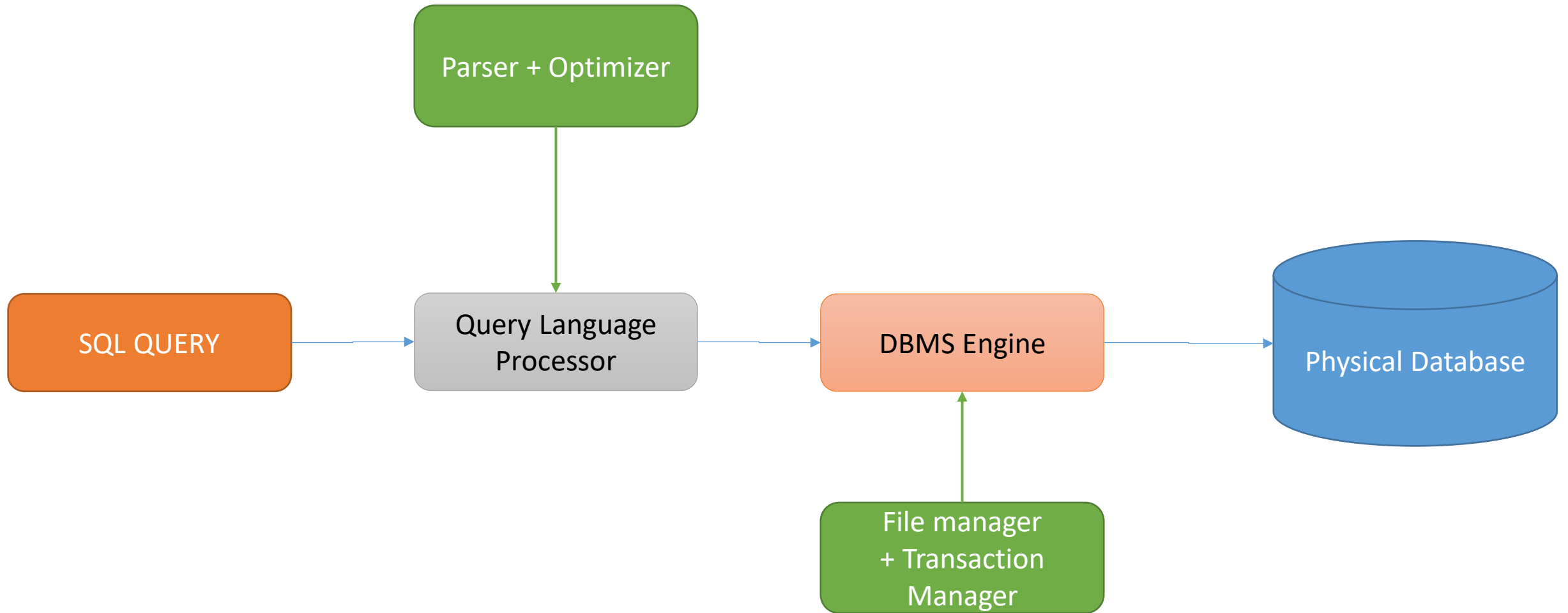
History

- **1970**
 - Dr. E. F. "Ted" of IBM is known as the father of relational databases. He described a relational model for databases.
- **1974**
 - Structured Query Language appeared.
- **1978**
 - IBM worked to develop Codd's ideas and released a product named System/R.
- **1986**
 - IBM developed the first prototype of relational database and standardized by ANSI.
 - The first relational database was released by Relational Software - later becoming Oracle.



Edgar Frank Codd
1923 – 2003

SQL Process to execute tasks



Summary of SQL Commands

- There is a set of standard SQL Commands to interact with relational databases
 - CREATE
 - SELECT
 - INSERT
 - UPDATE
 - DELETE
 - DROP
 - ...

These commands can be classified based on their nature

DDL : Data Definition language

- Create

- Creates a new database, view, table or other database object

- Alter

- Modify an existing database object

- Drop

- Delete an entire table, view or other database object

DML: Data Manipulation language

- Insert
 - Insert new records on the table
- Update
 - Update existing records on the table
- Delete
 - Delete objects (records) from a table

* All the commands above make use of “conditions”. We will study it during this course.

DCL: Data Control language

- **Grant**
 - Gives privileges for a user or group
- **Revoke**
 - Removes privileges of an user or group

DQL: Data Query language

- **Select**
 - Retrieves information from tables based on a set of conditions

The CRUD functions

- Despite all the definitions, the 4 major functions implemented in a database application can be defined as:

- Create (insert command)

NEW RECORDS

- Read or Retrieve (select command)

- Update (update command)

- Delete (delete command)

EXISTING RECORDS

Table

id	full_name	address	gender	birth_date	course	registration_date
11177898022	Bill Gates	879 4th Street	m	1965-11-22	Windows concepts	2014-03-20
09878965621	Edgar Frank Codd	Isle of Portland, England	m	1923-08-19	SQL advanced	2015-06-19
05255986933	Jesse Teixeira Silva	1327 maple Street - Vancouver	m	1983-06-08	SQL	2015-07-08
12345678901	Marie Curie	Warsaw, Poland	f	1867-11-07	NULL	NULL
98997867833	Martin Pert	234 Haro Street	m	1965-12-11	Java 1	2015-03-08

Field

id	full_name	address	gender	birth_date	course	registration_date
11177898022	Bill Gates	879 4th Street	m	1965-11-22	Windows concepts	2014-03-20
09878965621	Edgar Frank Codd	Isle of Portland, England	m	1923-08-19	SQL advanced	2015-06-19
05255986933	Jesse Teixeira Silva	1327 maple Street - Vancouver	m	1983-06-08	SQL	2015-07-08
12345678901	Marie Curie	Warsaw, Poland	f	1867-11-07	NULL	NULL
98997867833	Martin Pert	234 Haro Street	m	1965-12-11	Java 1	2015-03-08

Column

id	full_name	address	gender	birth_date	course	registration_date
11177898022	Bill Gates	879 4th Street	m	1965-11-22	Windows concepts	2014-03-20
09878965621	Edgar Frank Codd	Isle of Portland, England	m	1923-08-19	SQL advanced	2015-06-19
05255986933	Jesse Teixeira Silva	1327 maple Street - Vancouver	m	1983-06-08	SQL	2015-07-08
12345678901	Marie Curie	Warsaw, Poland	f	1867-11-07	NULL	NULL
98997867833	Martin Pert	234 Haro Street	m	1965-12-11	Java 1	2015-03-08

Row

id	full_name	address	gender	birth_date	course	registration_date
11177898022	Bill Gates	879 4th Street	m	1965-11-22	Windows concepts	2014-03-20
09878965621	Edgar Frank Codd	Isle of Portland, England	m	1923-08-19	SQL advanced	2015-06-19
05255986933	Jesse Teixeira Silva	1327 maple Street - Vancouver	m	1983-06-08	SQL	2015-07-08
12345678901	Marie Curie	Warsaw, Poland	f	1867-11-07	NULL	NULL
98997867833	Martin Pert	234 Haro Street	m	1965-12-11	Java 1	2015-03-08

It will be clearer when we start to make our own queries ☺

Null Values

id	full_name	address	gender	birth_date	course	registration_date
11177898022	Bill Gates	879 4th Street	m	1965-11-22	Windows concepts	2014-03-20
09878965621	Edgar Frank Codd	Isle of Portland, England	m	1923-08-19	SQL advanced	2015-06-19
05255986933	Jesse Teixeira Silva	1327 maple Street - Vancouver	m	1983-06-08	SQL	2015-07-08
12345678901	Marie Curie	Warsaw, Poland	f	1867-11-07	NULL	NULL
98997867833	Martin Pert	234 Haro Street	m	1965-12-11	Java 1	2015-03-08

YOU CAN DECIDE WHEN YOUR FIELD WILL ACCEPT OR NOT NULL VALUES!
We will learn more about it when we talk about Constraints

Constraints

- Constraints are the rules enforced on table data columns used to limit (restrict) the type of data that can go into a table and ensures the accuracy and reliability of the data in the database.
 - Column level constraints
 - Applied only to one column
 - Table level constraints
 - Applied to the whole table

Constraints

- Examples of SQL Constraints

NOT NULL

Ensures that a column cannot have NULL value.



Can you think about a reason for not accepting null values in a field?

```
CREATE TABLE [dbo].[students](  
  id int NOT NULL ,  
  full_name varchar(50) NOT NULL UNIQUE,  
  address varchar(100) NULL,  
  gender char(1) NOT NULL,  
  birth_date date NOT NULL,  
  course int references [dbo].[Courses](id),  
  registration_date date NULL,  
  PRIMARY KEY (ID)
```

Constraints


- Examples of SQL Constraints

DEFAULT

Provides a default value for a column when none is specified.

Give some examples where would be useful defining “default” values.

```
CREATE TABLE [dbo].[students](  
  id int NOT NULL ,  
  full_name varchar(50) NOT NULL UNIQUE,  
  address varchar(100) NULL,  
  gender char(1) NOT NULL,  
  birth_date date NOT NULL,  
  course int references [dbo].[Courses](id),  
  registration_date date default '2015-01-01',  
  PRIMARY KEY (ID)  
)
```



Keep in mind: not specifying a value is different from setting a field as null. We will learn that difference by doing it 😊

Constraints


- Examples of SQL Constraints

PRIMARY Key

Uniquely identifies each rows/records in a database table. The value must be unique

What is your primary key? And the primary key for your car, class, house, etc.?

```
CREATE TABLE [dbo].[students](  
  id int NOT NULL ,  
  full_name varchar(50) NOT NULL UNIQUE,  
  address varchar(100) NULL,  
  gender char(1) NOT NULL,  
  birth_date date NOT NULL,  
  course int references [dbo].[Courses](id),  
  registration_date date default '2015-01-01',  
  PRIMARY KEY (ID)  
)
```



Constraints

- Examples of SQL Constraints

FOREIGN Key

Uniquely identifies a row/record in any other database table.

```
CREATE TABLE [dbo].[students](  
  id int NOT NULL ,  
  full_name varchar(50) NOT NULL UNIQUE,  
  address varchar(100) NULL,  
  gender char(1) NOT NULL,  
  birth_date date NOT NULL,  
  course int references [dbo].[Courses](id),  
  registration_date date default '2015-01-01',  
  PRIMARY KEY (ID)  
)
```



id	description
1	Java 1
2	Java 2
3	Java 3
7	PHP
5	Project Management
6	Quality assurance
4	SQL Concepts

Constraints

- Examples of SQL Constraints

UNIQUE

Ensures that all values in a column are different (not replicate).

Give some examples of values (information) that are unique in REAL LIFE and how it would impact our tables.



```
CREATE TABLE [dbo].[students](  
  id int NOT NULL ,  
  full_name varchar(50) NOT NULL UNIQUE,  
  address varchar(100) NULL,  
  gender char(1) NOT NULL,  
  birth_date date NOT NULL,  
  course int references [dbo].[Courses](id),  
  registration_date date default '2015-01-01',  
  PRIMARY KEY (ID)  
)
```

In this example, should full_name really be unique? Elaborate on it.

Constraints

- Examples of SQL Constraints

INDEX

We Use to create and retrieve data from the database very quickly.
Very helpful in huge databases (lot of information and columns)

```
CREATE INDEX idIndex ON students (full_name)
```


Constraints

- Examples of SQL Constraints

CHECK Constraint

Ensures that all values in a column satisfy certain conditions.

```
CREATE TABLE [dbo].[students](  
  id int NOT NULL ,  
  full_name varchar(50) NOT NULL UNIQUE,  
  address varchar(100) NULL,  
  gender char(1) NOT NULL,  
  birth_date date NOT NULL CHECK (birth_date <= '1997-05-05'),  
  course int references [dbo].[Courses](id),  
  registration_date date default '2015-01-01',  
  PRIMARY KEY (ID)  
)
```





SQL DATA TYPES

Exact numerics

DATA TYPE	FROM	TO
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	$-10^{38} + 1$	$10^{38} - 1$
numeric	$-10^{38} + 1$	$10^{38} - 1$
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Approximate numerics

DATA TYPE	FROM	TO
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38

Date and Time

DATA TYPE	FROM	TO
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079

DATA TYPE	DESCRIPTION
date	Stores a date without the time
time	Stores a time without the date part.

Character String

DATA TYPE	DESCRIPTION
char	Max of 8,000 characters.
varchar	Max of 8,000 characters.
varchar(max)	Maximum length of 231characters
text	Max of 2,147,483,647 characters.

Unicode Character String

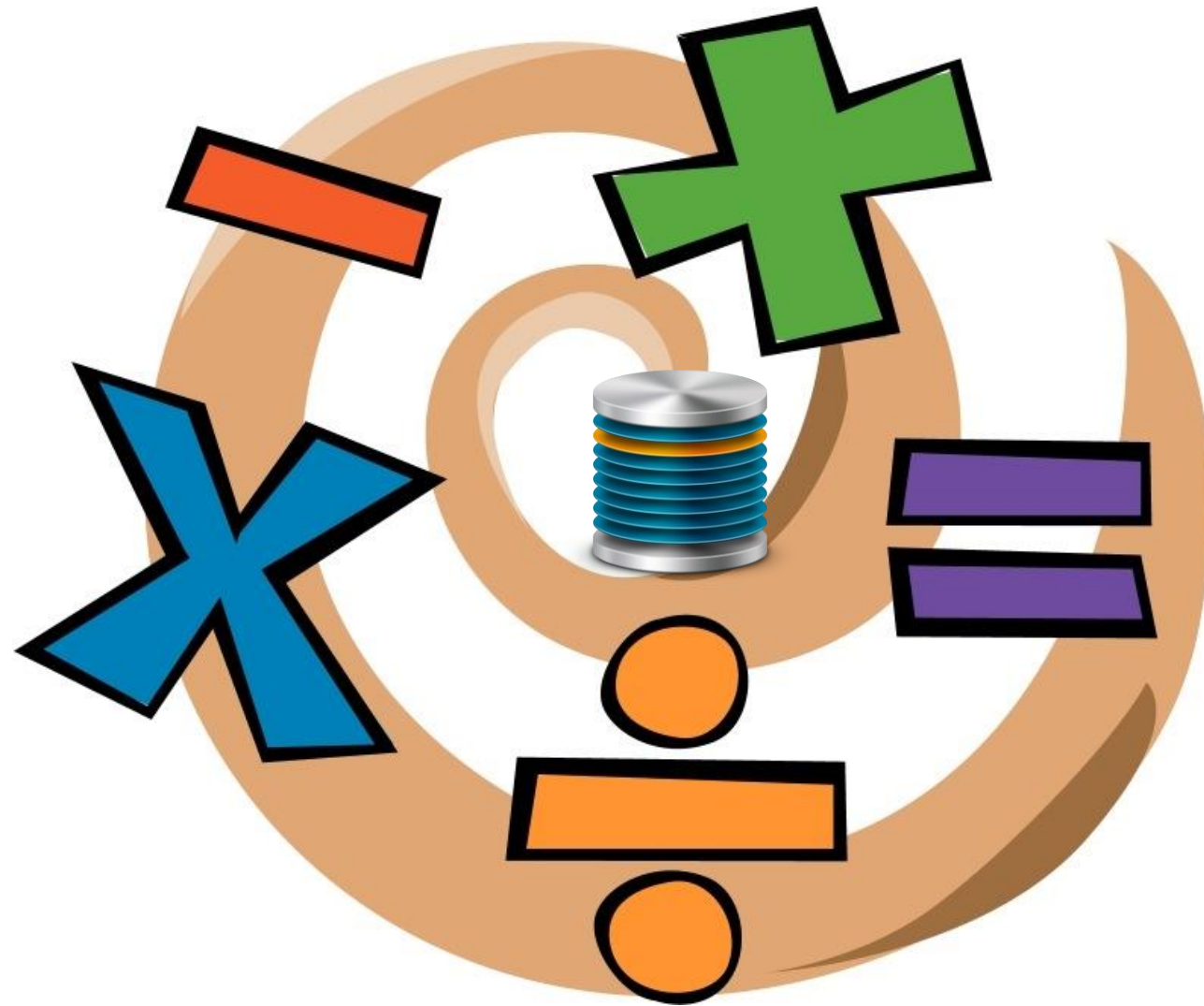
DATA TYPE	DESCRIPTION
nchar	Max of 4,000 characters
nvarchar	Max of 4,000 characters.
nvarchar(max)	Max of 231characters (SQL Server 2005 only)
Ntext	Max of 1,073,741,823 characters. (Variable length Unicode)

Binary

DATA TYPE	DESCRIPTION
Binary	Max of 8,000 bytes
Varbinary	Max of 8,000 bytes.
varbinary(max)	Max of 231 bytes
Image	Max of 2,147,483,647 bytes.

Miscellaneous Types

DATA TYPE	DESCRIPTION
sql_variant	Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
timestamp	Stores a database-wide unique number that gets updated every time a row gets updated
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	For SQL server 2005. Stores XML data. You can store xml instances in a column or a variable
cursor	Reference to a cursor object
table	Stores a result set for later processing



SQL OPERATORS

Arithmetic operators

- +

- Sum the given values

- -

- Subtracts the given values

- *

- Multiplies the given values

- /

- Divides the left operand by the right hand operand.

- %

- Divides the left operand by the right hand operand and gives us the remainder.

Comparison operators

- =
 - Returns true if the given values are equals. Otherwise returns false.
- != or <>
 - Returns true if the given values are different. Otherwise returns false
- >
 - Returns true if the left operand is greater than the right operand.
- <
 - Returns true if the left operand is less than the right operand.

Comparison operators

- `>=`

- Returns true if the left operand is greater than or equal the right operand.

- `<=`

- Returns true if the left operand is less than or equals the right operand.

- `!<`

- Returns true if the left operand is not less than the right operand.

- `!>`

- Returns true if the left operand is not greater than the right operand.

Logical operators

ALL

Compares a given value against all the values in a value set.

AND

Allows the existence of multiple conditions on a *where* clause

ANY

Compares a given value to any applicable value in the list according to the condition.

BETWEEN

Given the minimum and maximum values, returns all the values between the ranges.

EXISTS

Verify if a row in a specific table exists given a criteria.

Logical operators

IN

Used to compare a value to a list of literal values that have been specified. Returns true if the list contains the value.

LIKE

Used to compare a value to similar values using wildcard operators. We will see more about it on module 2.

NOT

Used to deny the meaning of the logical operator in which it is used. For example NOT in will return true only if the value does not exist in the given list.

OR

Similar with the AND, it is used to combine multiple conditions in a where clause.

IS NULL

Returns true if a given value is null. Remember that null values are different from blanks space or 0 (zero).

UNIQUE

Searches every row of a specified table for uniqueness (no duplicates).



Data Integrity

Overview

- Data integrity refers to the overall completeness, accuracy and consistency of data
- This can be indicated by the absence of alteration between two instances or between two updates of a data record, meaning data is intact and unchanged.
- Data integrity is usually imposed during the database design phase through the use of standard procedures and rules and can be maintained through the use of various error checking methods and validation procedures

These are the categories of the data integrity in RDBMS:

Entity Integrity

There are no duplicate row (registry) in a specific table.

Domain Integrity

Enforces valid entries for a given column by restricting the type, the format, or the range of values (remember the constraints).

Referential Integrity

Rows that are used by other records can't be deleted.

User-Defined Integrity

Enforces some specific business rules that do not fall into entity, domain, or referential integrity. For example default values or checks.



Database Normalization

Database Normalization

- Database normalization is the process of efficiently **organize** data in a database.
There are two reasons for the normalization process:
 - Eliminating redundant data, for example, storing the same data in more than one table.
 - Ensuring data dependencies make sense.

- There is a set of normal forms that you can use to organize the database structure. We will aim here in the 3 main ones
 - 1NF - First Normal Form
 - 2NF - Second Normal Form
 - 3NF - Third Normal Form

1NF - First Normal Form

It is the most basic set of rules for an organized database and:

- Defines the data items required as they will be the columns in a table.
- Avoids repeating groups of data.
- Guarantee that there will be always a primary key.

1NF - First Normal Form

First Rule : Define the data items.

Example : Look at the data to be stored, organize it into columns, define what type of data each column contains and finally put the **related columns** into their own table.

Column name	Table	Type of data
full_name		
Street_name		
Gender		
birth_date		
City		
Zip code		
number		
age		

Divide the information on the left into two tables: **customer** and **address**. Define which type of data is adequate for the field.

1NF - First Normal Form

Second Rule : Avoid repeating groups of data.

Always try to break the table into others tables and join them using a key if it is possible!

ID	NAME	AGE	ADDRESS	ORDERS
1	Jesse	32	1325 maple street	Smartphone
1	Jesse	32	1325 maple street	TV
1	Jesse	32	1325 maple street	Lap Top
1	Jesse	32	1325 maple street	Mouse
1	Jesse	32	1325 maple street	Memory
1	Jesse	32	1325 maple street	Drone xy

This is what we need to avoid. Can you see how much repeated information we have here? Can we improve this table?

1NF - First Normal Form

Second Rule : Avoid repeating groups of data.

We can do this by analyzing and applying the first rule and then create a proper table and its columns for each information... if you look at the original table and follow the rules, you can solve the problem!

ID	NAME	AGE	ADDRESS
1	Jesse	32	1325 maple street

ID	CUSTOMER_ID	ORDERS
1	1	Ipad
2	1	Drone xy
3	1	Mouse Wireless
4	1	TV 43

1NF - First Normal Form

Third Rule : Create a primary key for each table we have already created.

That's is a very simple and straight forward rule.
Basically when you apply the first ones you are forced to create the primary keys for its relationships to work 😊

2NF – Second Normal Form

1NF + there must be no partial dependencies of any of the columns on the primary key.

- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.

From **Relational Database Normalization Basics** at <http://www.databasedev.co.uk/> we have the following example:

Consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

2NF – Second Normal Form

In the example below we need to use our knowledge to put this table in accordance with the second normal form.

STUD_ID	STUD_NAME	COURSE_ID	COURSE_DETAIL	COURSE_DATE
1	jesse	1	Java	2015-02-13 00:00:00.000
1	jesse	2	SQL	2015-08-07 00:00:00.000
1	jesse	3	QC	2015-12-12 00:00:00.000
1	jesse	4	PHP	2015-04-15 00:00:00.000

Tip: Look at the possibility of creating more than 2 tables!

3NF – Third Normal Form

2NF + All non primary fields are dependent on the primary key.

- Remove transitive dependencies
 - Doing that we reduce the amount of duplicated data, simplify the database maintenance and keep the data integrity, just for start!

Imagine a customer table with the following fields. Notice that the field **ZIP** and **STREET** have a transitive dependency. It's against the Third normal form, so we need to fix it.

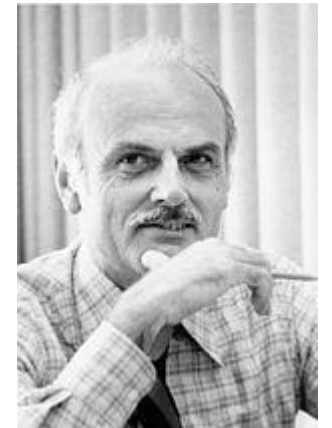
CUST_ID	CUST_NAME	BIRTH_DATE	STREET	CITY	STATE	ZIP	EMAIL_ID
---------	-----------	------------	--------	------	-------	-----	----------

What is RDBMS

- **R**elational **D**atabase **M**anagement **S**ystem is the base for the SQL and for all modern database systems
 - It is a DBMS (database management system) based on the relational model introduced by “Ted” Codd



Access



Edgar Frank Codd
1923 – 2003

PRACTICING

HomeWork



- Our course is based on Relational databases, but do we know the differences between it and a transactional database?
 - Do a Research and highlight the points you think are important to discuss with your colleagues next class!
- We discussed here about 3 normal Forms. What are the other ones? Is it important to know how they work?
- Design a set of tables that can store information about a student, his personal information, where he lives, his courses, his grades, his tuition fees and the total time of his courses.
 - Try to design it following the normalization rules
 - You need only to describe the columns, its data types relations (no code necessary)
 - Be **creative** and don't be afraid of errors!
 - We will fix it later 😊