Structured Query Language (SQL)

CEE412 / CET522

Transportation Data Management and Visualization

WINTER 2020

Announcement

Teammates

Assignment 1

Assignment 2

SQL History

Why is the SQL pronounced SEQUEL?

- Dr. Edgar F. Codd’s paper (1970) on relational database theory.
- System/R Research Group of IBM.
- Prototyped DB2 and a support language for System/R’s multi-table and multi-user access called Structured English Query Language, or SEQUEL.
- Eventually, the language became known as SQL, the industry standard for relational databases.

SQL Introduction

SQL is a very-high-level language

- SQL assumes a table-like structure, which helps the programmer to avoid specifying a lot of data-manipulation details that would be necessary in conventional programming languages.
- SQL queries are “optimized” quite well, yielding efficient query executions.

SQL dialects

- SQL
- SQL-92 (SQL2)
- SQL-99 (SQL3)
- T-SQL (Transact-SQL) – Microsoft SQL Server

Case insensitivity

- SQL treats upper- and lower-case letters as the same letter.

Simple SQL Queries

The principal form of a query is:

```
SELECT attributes
  FROM tables
 WHERE conditions
```

What does SELECT do?

- **Projection** of data. This can be a list of existing fields or some expressions.

What does FROM do?

- Relations/tables. This can be existing relations, temporary tables, and subqueries.

What does WHERE do?

- **Selection** of data. Conditional expressions which restrict the rows returned by the query.

Simple SQL Queries

Movies

title	year	length	budget	rating	votes	mpaa	genres
'G' Men	1935	85	450000	7.2	281	NULL	Drama
'Manos' the Hands of Fate	1966	74	19000	1.6	7996	NULL	
'Til There Was You	1997	113	23000000	4.8	799	PG-13	Comedy, Romance
.com for Murder	2002	96	5000000	3.7	271	NULL	
10 Things I Hate About You	1999	97	16000000	6.7	19095	PG-13	Comedy, Romance
100 Mile Rule	2002	98	1100000	5.6	181	R	Comedy
…	…	…	…	…	…	…	…

```
SELECT title, year, length, rating
  FROM movies
 WHERE title = 'titanic'
```

title	year	length	rating
Titanic	1997	194	6.9

Simple SQL Queries

All of the following queries give you the same result:

```
SELECT title, year, length, rating
  FROM movies
 WHERE title = 'titanic'
```

```
select TITLE, YEAR, LENGTH, RATING
  from MOVIES
 where TITLE = 'TITANIC'
```

```
SELECT title, year, length, rating FROM movies WHERE title = 'titanic'
```

Simple SQL Queries

title	year	length	budget	rating	votes	mpaa	genres
'G' Men	1935	85	450000	7.2	281	NULL	Drama
'Manos' the Hands of Fate	1966	74	19000	1.6	7996	NULL	
'Til There Was You	1997	113	23000000	4.8	799	PG-13	Comedy, Romance
.com for Murder	2002	96	5000000	3.7	271	NULL	
10 Things I Hate About You	1999	97	16000000	6.7	19095	PG-13	Comedy, Romance
100 Mile Rule	2002	98	1100000	5.6	181	R	Comedy
…	…	…	…	…	…	…	…

```
SELECT *
  FROM movies
 WHERE title = 'titanic'
```

title	year	length	budget	rating	votes	mpaa	genres
Titanic	1997	194	200000000	6.9	90195	PG-13	Drama, Romance

Simple SQL Queries

Input schema:

Movies(title, year, length, budget, rating, votes, mpaa, genre)

```
SELECT title, year, length, rating
  FROM movies
 WHERE title = 'titanic'
```

Output schema:

Result(title, year, length, rating)

Projections in SQL

Rename columns in your result

```
SELECT title AS Name, year, length AS Duration, rating
  FROM movies
 WHERE title = 'titanic'
```

Name	year	Duration	rating
Titanic	1997	194	6.9

- Projection → choosing which columns the query shall return
- Selection → choosing which rows are to be returnd

Selections in SQL

What goes in the WHERE clause?

Common Boolean operators

Operator	Description	Operator	Description
=	Equal to	<=	Less than or equal to
<> or !=	Not equal to	BETWEEN *a* AND *b*	Between an inclusive range
>	Greater than	LIKE	Match a character pattern
<	Less than	IN (*a*, *b*, *c*)	Equal to one of multiple possible values
>=	Greater than or equal to	IS NULL *or* IS NOT NULL	Compare to null (missing data)

!= is not standard SQL operators, but is supported in most DBMSs

- For numbers, they have the usual meanings
- For characters and text: lexicographic ordering
- For dates and times: time1 < time2 means time1 is earlier than time2.

Selections in SQL

The LIKE operator

- Compare two strings on the basis of pattern match.
- Expression: s LIKE p, where s is a string and p is a pattern.
- p may contain two special symbols:
  - _ = any single character
  - % = any sequence of characters

Selections in SQL

```
SELECT title, year, length, rating
  FROM movies
 WHERE title LIKE 'star ____'
```

title	year	length	rating
Star Dust	1940	86	6.6
Star Maps	1997	86	6.1
Star Trak	1998	9	4.3
Star Wars	1977	125	8.8

```
SELECT title, year, length, rating
  FROM movies
 WHERE title LIKE '%star%'
```

title	year	length	rating
Last Starfighter, The	1984	101	6.2
Star Kid	1997	101	5
Star Trek III: The Search for Spock	1984	105	6.2
…	…	…	…

Selections in SQL

Logical operators

- AND: returns TRUE if both sides are TRUE
- OR: returns TRUE if either side is TRUE
- NOT: returns true if the following predicate is FALSE, and vice versa

Parenthesis can change the evaluation order

- TRUE OR TRUE AND FALSE
- (TRUE OR TRUE) AND FALSE
- FALSE AND FALSE OR TRUE
- FALSE AND (FALSE OR TRUE)

Operator precedence

Operators
* (Multiply), / (Division), % (Modulo)
=, >, <, >=, <=, <>, != (Comparison operators)
NOT
AND
BETWEEN, IN, LIKE, OR
= (Assignment)

Lower precedence level

Selections in SQL

Select movies that were released after 2000 with ratings higher than 8.5, or other movies rated higher than 9.

```
SELECT title, year, length, rating
  FROM movies
 WHERE rating > 9 OR year > 2000 AND rating > 8.5
```

title	year	length	rating
Looking Out	2002	15	9.4
Looking Up	1977	94	9.1
Lord of the Rings: The Fellowship of the Ring	2001	208	8.8
Lord of the Rings: The Return of the King	2003	251	9
Lord of the Rings: The Two Towers	2002	223	8.8
…	…	…	…

Ordering the Results

Use ORDER BY to sort your results.

- Example: find all movies with a budget higher than $10,000,000, and sort the movies by their rating.

```
SELECT title, year, length, budget, rating
  FROM movies
 WHERE budget > 10000000
 ORDER BY rating
```

title	year	length	budget	rating
From Justin to Kelly	2003	90	12000000	1.7
Son of the Mask	2005	94	74000000	1.9
Alone in the Dark	2005	96	20000000	2.1
Glitter	2001	104	22000000	2.1
…	…	…	…	…

Is this what we want?

Ordering the Results

- Ordering is ascending, unless you specify the DESC keyword
- You can order by fields that are not in the SELECT list.

```
SELECT title, year, length
  FROM movies
 WHERE budget > 10000000
 ORDER BY rating DESC
```

title	year	length
Shawshank Redemption	1994	142
Lord of the Rings: The Return of the King	2003	251
Godfather: Part II	1974	200
Schindler's List	1993	195
Lord of the Rings: The Two Towers	2002	223
Lord of the Rings: The Fellowship of the Ring	2001	208
Star Wars	1977	125
…	…	…

Ordering the Results

- Order by multiple fields.

```
SELECT title, year, length, rating
  FROM movies
 WHERE budget > 10000000
 ORDER BY year DESC, rating DESC
```

title	year	length	rating
Sin City	2005	124	8.3
Eternal Sunshine of the Spotless Mind	2004	108	8.6
Taegukgi hwinalrimyeo	2004	140	8.5
Incredibles, The	2004	121	8.3
Kill Bill: Vol. 2	2004	136	8.3
Million Dollar Baby	2004	132	8.3
Lord of the Rings: The Return of the King, The	2003	251	9
Kill Bill: Vol. 1	2003	111	8.3
Finding Nemo	2003	100	8.3
…	…	…	…

Eliminating Duplicates

Use DISTINCT keyword to remove duplicates in query results.

- Example: how many different movie genres do we have in the database?

```
SELECT DISTINCT genres
  FROM movies
```

genres
Animation, Documentary
Action, Animation
Action, Comedy, Documentary
Drama, Documentary, Romance
Action, Animation, Comedy, Romance
Action, Comedy, Drama, Romance
Drama, Short
Action, Comedy, Drama, Short
Action, Animation, Comedy
…

Compare to

```
SELECT genres
  FROM movies
```

genres
Drama
Comedy
Animation, Comedy, Short
Comedy, Drama
Animation, Comedy, Short
Drama
Drama
Drama
…

Joins in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is the connection between them?

Joins in SQL

Product (PName, Price, Category, Manufacturer)

Company (CName, StockPrice, Country)

For example, to find the names and prices for all products manufactured in Japan, we need to run the query:

```
SELECT pname, price
  FROM product, company
 WHERE manufacturer = cname AND country = 'Japan'
```

Join between Product and Company

Joins in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT pname, price
  FROM product, company
 WHERE manufacturer = cname
       AND country = 'Japan'
```

pname	price
SingleTouch	149.99
MultiTouch	203.99

Joins in SQL

Product (PName, Price, Category, Manufacturer)

Company (CName, StockPrice, Country)

Find all countries that produce some product in the ‘Gadgets’ category.

```
SELECT country
  FROM product, company
 WHERE manufacturer = cname AND category = 'Gadgets'
```

Joins in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT country
  FROM product, company
 WHERE manufacturer = cname AND category = 'Gadgets'
```

country
USA
USA

What is the problem? What’s the solution?

Joins in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT DISTINCT country
  FROM product, company
 WHERE manufacturer = cname AND category = 'Gadgets'
```

country
USA

Joins in SQL

Product (PName, Price, Category, Manufacturer)

Purchase (Invoice, Buyer, Seller, Store, Product)

Person(PerName, PhoneNumber, City)

Find names of people living in Seattle that bought some product in the ‘Gadgets’ category, and the names of the stores from where they bought such products.

```
SELECT DISTINCT pername, store
  FROM person, purchase, product
 WHERE pername=buyer AND product = pname
       AND city='Seattle' AND category='Gadgets'
```

Joins in SQL

Consider that we have two big tables:

Accident(ReportNum, Route, Milepost, **Date**, Severity)

Loopdata(LoopID, **Date**, Time, Speed, Volume)

What happens if we join these tables using only the common “Date” attribute?

- Many accidents each day, many detectors and times for each day.
- Result: Every accident will be matched with all loop data collected on the corresponding date.
- Huge useless response, possible memory overload.
- Need accident time and loop location to fully define the join.

Joins in SQL

Inner joins

- Cross join
- Theta join
- 👎 Natural join

Outer joins

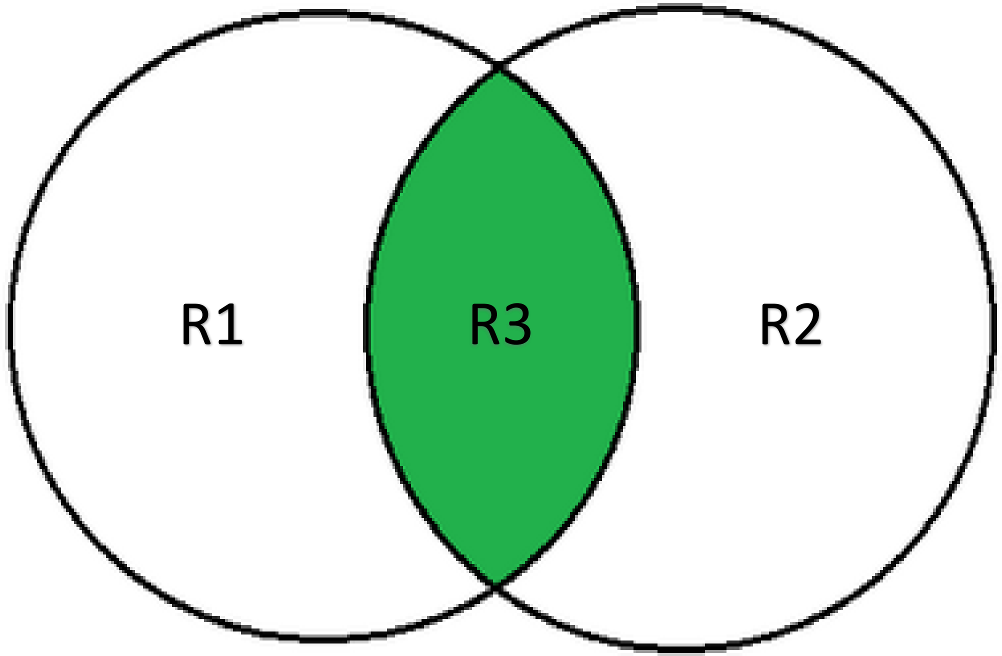
- Full outer join
- Right outer join
- Left outer join

```
SELECT *
  FROM product JOIN company
    ON manufacturer = cname
```

- Inner joins can be specified in either the FROM or WHERE clauses.
- Outer joins can be specified in the FROM clause only.

Inner Join

“Inner join” simply means that we only return rows in which the “ON” clause is true for both tables. That is, only return the rows for which there is a match in both tables.



Cross Join

Cross join is also known as Cartesian product:

In relational algebra R3 := R1 CROSS JOIN R2

- No join field.
- Pair each tuple $t_{1,i}$ of R1 with each tuple $t_{2,j}$ of R2.
- Concatenation $t_{1,i}t_{2,j}$ is a tuple of R3.
- Schema of R3 is the attributes of R1 and R2, in order.
- But beware attribute A of the same name in R1 and R2: use R1.A and R2.A.

Cross Join

Where is a cross join used?

First, consider that this is how an inner join works = Match up all possible rows, and then return only those that match the conditions in the WHERE clause.

If you include a table in the FROM clause without specifying a join condition in the WHERE clause, a cross join will result.

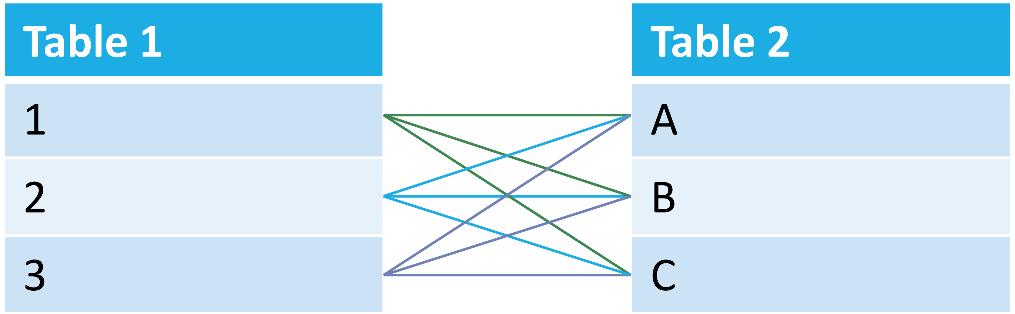


Table 1
1
2
3

Table 2
A
B
C

Cross Join

Example: consider you are managing the computer lab and the department has just purchased a list of software.

You want to make sure that all software are installed on all computers.

In this case, you could cross join the computer table with the software license table to create a task checklist.

Cross Join

Computers

ComputerID	Make	Model	Year	OperatingSystem
001	Dell	OptiPlex 9010	2012	Windows 10 64bit
002	Dell	OptiPlex 9030	2014	Windows 10 64bit

Software

SoftwareID	Name	Version	Developer	License Server
022	VISSIM	9	PTV	TLAB-2
023	AutoCAD	2016	Autodesk	TLAB-6

Result

- Can be useful for creating tables from existing data.
- Relatively infrequently used in practice (still need to know it).

ComputerID	SoftwareID	Completed
001	022	Yes
001	023	No
002	022	Yes
002	023	No

Theta Join

Theta Join is also known as conditional join

In relational algebra R3 := R1 JOIN$_C$ R2

- Take the product R1 X R2 (cross join)
- Then apply SELECT$_C$ to the result

In SQL Server, we use R1 JOIN R2 ON *C*

Condition *C* can be any boolean-valued condition.

- Expressed as*:* *A* theta *B*, where theta was =, <, etc.
- Hence the name "theta-join"

Theta Join

There is no functional difference between queries of the following forms.

- Use WHERE

```
SELECT *
  FROM product, company
 WHERE manufacturer = cname
```

- Use JOIN or INNER JOIN

```
SELECT *
  FROM product JOIN company
    ON manufacturer = cname
```

Natural Join

Like inner join, but automatically joins columns with identical names

- Does the same thing as an equijoin
- No need to specify the join condition

Not great practice to use this join type

- Invented to be a time saver
- Terrible time waster if anything goes wrong
- Not supported in SQL server

Disambiguating Attributes

Sometimes two relations can have the same attributes:

Student(Name, Address, StudyAt)
University(Name, Address)

```
SELECT DISTINCT name, address
  FROM student, university
 WHERE studyat = name
```

Which name?

Which Address?

```
SELECT DISTINCT student.name, university.address
  FROM student, university
 WHERE student.studyat = university.name
```

Outer Join

Suppose we do R OUTER JOIN S:

- A tuple of R that has no match tuple of S with which it joins is said to be dangling
- Similarly for a tuple of S

Outer join preserves dangling tuples by padding them with a special NULL symbol in the result

We can use FULL, LEFT, or RIGHT to specify the type of OUTER JOIN to conduct

Outer Join

R

A	B
1	2
4	5

S

A	C
1	3
6	7

```
SELECT *
  FROM R FULL OUTER JOIN S
    ON R.A = S.A
```

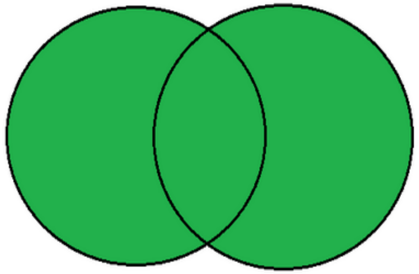
R.A	B	S.A	C
1	2	1	3
4	5	NULL	NULL
NULL	NULL	6	7

What happens if we do a LEFT OUTER JOIN?

Outer Join

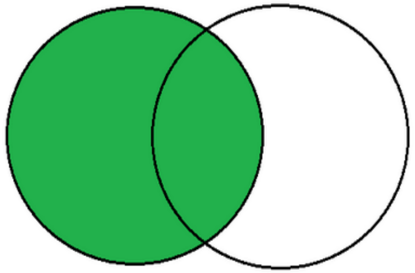
Full outer join

- Include all rows from both tables regardless of whether a match is found



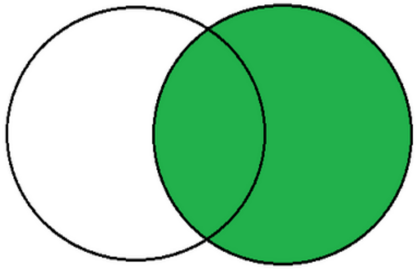
Left outer join

- Include all rows from table specified LEFT of the JOIN clause



Right outer join

- Include all rows from table specified RIGHT of the JOIN clause



Outer Join

These two queries are essentially identical, the only difference is the column order in the result.

```
SELECT *
  FROM injuries LEFT OUTER JOIN players
    ON injuries.name = players.name
```

```
SELECT *
  FROM players RIGHT OUTER JOIN injuries
    ON injuries.name = players.name
```

Join Examples

Candy

Name	Price	type	Calories
M&M	$1.29	Chocolate	190
Sourpatch	$1.09	Sour	180
Cognac	$3.29	Novelty	210

Person

Pername	Age	Predilection
James	4	M&M
Tina	29	Sourpatch
Seth	16	Gummybear

```sql
SELECT *
  FROM candy RIGHT OUTER JOIN person
    ON name = predilection
```

Name	Price	Type	Calories	Pername	Age	Predilection
M&M	$1.29	Chocolate	190	James	4	M&M
Sourpatch	$1.09	Sour	180	Tina	29	Sourpatch
NULL	NULL	NULL	NULL	Seth	16	Gummybear

Join Examples

Candy

Name	Price	type	Calories
M&M	$1.29	Chocolate	190
Sourpatch	$1.09	Sour	180
Cognac	$3.29	Novelty	210

Person

Pername	Age	Predilection
James	4	M&M
Tina	29	Sourpatch
Seth	16	Gummybear

```
SELECT *
  FROM candy INNER JOIN person
    ON name = predilection
```

Inner (Theta) Join Result

Name	Price	Type	Calories	Pername	Age	Predilection
M&M	$1.29	Chocolate	190	James	4	M&M
Sourpatch	$1.09	Sour	180	Tina	29	Sourpatch

Tuple Variables

Sometimes, a query needs to use two copies of the same relation.

Similarly to renaming columns, you can distinguish copies by following the relation name by the name of a tuple-variable, using the keyword AS.

Example: predict the traffic condition in the next 15 minutes.

- In my table, I need a set of columns to contain traffic conditions at a point of time, and another set of columns with traffic condition in the next 15 minutes.
- Select data from one single table, with different selection criteria for the two sets of columns.

Tuple Variables

Purchase(buyer, seller, store, product)

Find all stores that sell at least one product that the store ‘BestBuy’ also sells:

```
SELECT DISTINCT x.store
  FROM purchase AS x, purchase AS y
 WHERE x.product = y.product AND y.store = ‘bestbuy’
```

Data Definition in SQL

SQL consists of two components

- Data definition language (DDL)
- Data manipulation language (DML)

For data definition, SQL can also be used to define

- Data types (e.g., numbers, characters, date, and time)
- Database schema (create, alter, and delete tables)

Data Types in SQL

Exact numerics

- TINYINT
- SMALLINT
- INT
- BIGINT
- DECIMAL(p,s)

Approximate numerics

- FLOAT
- REAL

Character strings

- CHAR(n) – fixed length
- VARCHAR(n) – variable length
- TEXT

Unicode character strings

- NCHAR(n)
- NVARCHAR(n)
- NTEXT

What is the expression of 123.451 if it is defined as DECIMAL(6,2)?

0123.45

Data Types in SQL

Date and time

- TIME: hh:mm:ss
- DATE: YYYY-MM-DD
- DATETIME: YYYY-MM-DD hh:mm:ss
- DATETIME2: YYYY-MM-DD hh:mm:ss

Converting Data Types

Some useful functions for data conversion:

- Convert

```
SELECT CONVERT(DECIMAL(4,2), height)
FROM players
```

- Cast

```
SELECT CAST(height AS DECIMAL(4,2))
FROM players
```

- Datepart

```
Select DATEPART(MONTH, timestamp)
FROM loopdata
```

Creating Tables

Example: create a table based on the following relational schema

Person(name, ssn, age, city, gender, birthdate)

```
CREATE TABLE Person(
    name      VARCHAR(100),
    ssn       INT,
    age       SMALLINT,
    city      VARCHAR(30),
    gender    CHAR(1),
    birthdate DATE
)
```

Did I miss anything?

Define the Primary Key

The following two queries are equal:

```
CREATE TABLE Person(
    name      VARCHAR(100),
    ssn       INT PRIMARY KEY,
    age       SMALLINT,
    city      VARCHAR(30),
    gender    CHAR(1),
    birthdate DATE
)
```

```
CREATE TABLE Person(
    name      VARCHAR(100),
    ssn       INT,
    age       SMALLINT,
    city      VARCHAR(30),
    gender    CHAR(1),
    birthdate DATE,
    PRIMARY KEY (ssn)
)
```

Modifying Schemas

A table is empty right after it is created using the CREATE TABLE command.

You can use the keywords DROP TABLE to remove a table from your database:

```
DROP TABLE person
```

Both the table structure and the contents will be deleted.

Use with caution!

- It is unforgiving and unrecoverable.
- There is no warning when deleting tables in SQL Server.

Modifying Schemas

Modify the table structure

- Add a new column

```
ALTER TABLE person
  ADD phone CHAR(20)
```

- Remove a column

```
ALTER TABLE person
 DROP birthdate
```

Default Values

```
CREATE TABLE Person(
    name      VARCHAR(100),
    ssn       INT PRIMARY KEY,
    age       SMALLINT,
    city      VARCHAR(30) DEFAULT 'Seattle',
    gender    CHAR(1),
    birthdate DATE
)
```

The default of defaults: NULL

Insertions

General form:

INSERT INTO R(c_1,..., c_n) VALUES (v_1,..., v_n)

Column names

Values

Example: insert a new person into the table

```
INSERT INTO person(name, ssn, age, city, gender)
VALUES ('Kevin', 123456789, 28, 'Bellevue', 'M')
```

- Missing attribute → NULL.
- You can omit attribute names if you give values in order (must have a value for each attribute).

Truncate

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse)

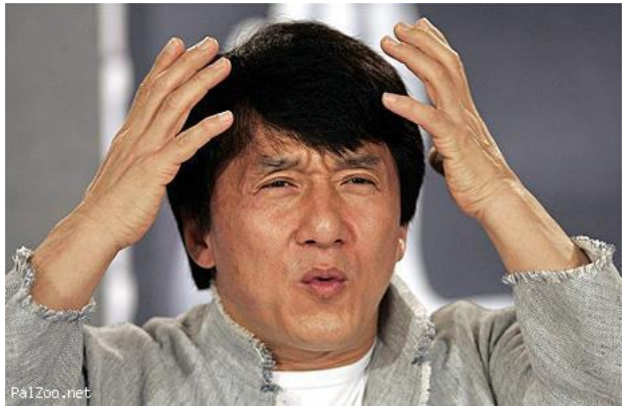
- The result of this operation quickly removes all data from a table

```
TRUNCATE TABLE table_name
```

DROP vs TRUNCATE

- Truncate is normally ultra-fast and its ideal for deleting data from a temporary table.
- Truncate preserves the structure of the table for future use, unlike drop table where the table is deleted with its full structure.
- Table or Database deletion using DROP statement **cannot** be rolled back, so it must be used wisely.

SQL Practice This Friday



Concepts (abstract)

Practice (concrete)

Like any new interface or language…

Practice is the best way to learn.