Mod 1.

What is RDBMS?

- RDBMS = Relational database management system
- A set of software tools that controls the data
  - Access, organization, and storage
- Examples are: MySQL, Oracle Database, IBM Db2, etc.

Basic SQL Commands

Create a table

Insert

Select

Update

Delete

Update

Delete

Using the UPDATE statement

UPDATE statement

UPDATE [TableName] SET [[ColumnName]=[Value]] <WHERE [Condition]>

Using the DELETE statement

Author_Id	LastName	FirstName	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@ibm.com	Transylvania	RO

DELETE FROM AUTHOR
WHERE AUTHOR_ID IN ('A2', 'A3')

Author_Id	LastName	FirstName	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN

Using the UPDATE statement

UPDATE AUTHOR SET LASTNAME='KATTA' FIRSTNAME='LAKSHMI'
WHERE AUTHOR_ID='A2'

Mod 2.

What you will learn

- Explain the advantage of the relational model
- Explain how the Entity name and attributes map to a relational database table
- Describe the difference between an entity and an attribute
- Identify some commonly used data types
- Describe the function of Primary Keys

ALTER TABLE

ALTER TABLE statements can be used to add or remove columns from a table, to modify the data type of columns, to add or remove keys, and to add or remove constraints.

ADD COLUMN syntax

```
ALTER TABLE table_name
ADD column_name data_type;
```

A variation of the syntax for adding column is:

```
ALTER TABLE table_name
ADD COLUMN column_name data_type;
```

By default, all the entries are initially assigned the value `NULL`. You can then use `UPDATE` statements to add the necessary column values.

For example, to add a **telephone_number** column to the **author** table in the **library** database, the statement will be written as:

```
ALTER TABLE author
ADD telephone_number BIGINT;
```

Modify column data type

```
ALTER TABLE table_name
MODIFY column_name data_type;
```

Sometimes, the data presented may be in a different format than required. In such a case, we need to modify the data_type of the column. For example, using a **numeric** data type for **telephone_number** means you cannot include **parentheses**, **plus signs**, or **dashes as part of the number**. For such entries, the appropriate choice of data_type is CHAR.

To modify the data type, the statement will be written as:

```
ALTER TABLE author
MODIFY telephone_number CHAR(20);
```

The entries can then be updated using UPDATE statements. An updated version of the "author" table is shown below.

author_id	lastname	firstname	email	city	country	telephone_number
1001	Thomas	John	johnt@...	New York	USA	555-1111
1002	James	Alice	alicej@...	Seattle	USA	555-1112
1003	Wells	Steve	stevew@...	Montreal	Canada	555-2222
				London	UK	555-3333

TRUNCATE Table

TRUNCATE TABLE statements are used to delete all of the rows in a table. The syntax of the statement is:

```
1  TRUNCATE TABLE table_name;
```

So, to truncate the "author" table, the statement will be written as:

```
1  TRUNCATE TABLE author;
```

The output would be as shown in the image below.

author_id	lastname	firstname	email	city	country

Note: *The TRUNCATE statement will delete the rows and not the table.*

Mod 3.

Retrieving rows - using a Range

```
db2 => select title, pages from Book
         WHERE pages >= 290 AND pages <= 300
```

Title	Pages
Database Fundamentals	300
Getting started with DB2 App Dev	298

2 record(s) selected.

```
db2 => select title, pages from Book
         WHERE pages between 290 and 300
```

Title	Pages
Database Fundamentals	300
Getting started with DB2 App Dev	298

Retrieving rows - using a Set of Values

```
db2 => select firstname, lastname, country from Author
         WHERE country='AU' OR country='BR'
```

Firstname	Lastname	Country
Xiqiang	Ji	AU
Juliano	Martins	BR

2 record(s) selected.

```
db2 => select firstname, lastname, country from Author
         WHERE country IN ('AU','BR')
```

Firstname	Lastname	Country
Xiqiang	Ji	AU
Juliano	Martins	BR

Functions, Multiple Tables, and Sub-queries

Aggregate or Column Functions

- INPUT: Collection of values (e.g. entire column)
- Output: Single value
- Examples: SUM(), MIN(), MAX(), AVG(), etc.

SCALAR and STRING FUNCTIONS

SCALAR: Perform operations on every input value
Examples: ROUND(), LENGTH(), UCASE, LCASE

ไม่เอาหลักๆ

Example 6: Round UP or DOWN every value in COST:

```
select
     ROUND(COST)
     from PETRESCUE
```

Example 6. Results:

```
1
450.00
667.00
100.00
50.00
76.00
```

SCALAR and STRING FUNCTIONS

SCALAR: Perform operations on every input value
Examples: ROUND(), LENGTH(), UCASE, LCASE

Example 7. Retrieve the length of each value in ANIMAL:

```
select
     LENGTH(ANIMAL)
     from PETRESCUE
```

Example 7. Results:

```
1
3
3
3
6
3
```

Date and Time

Date, Time Functions

Most databases contain special datatypes for dates and times

DATE: `YYYYMMDD`

TIME: `HHMMSS`

TIMESTAMP: `YYYYXXDDHHMMSSZZZZZZ`

Date / Time functions:

```
YEAR(), MONTH(), DAY(), DAYOFMONTH(), DAYOFWEEK(),
DAYOFYEAR(), WEEK(), HOUR(), MINUTE(), SECOND()
```

Date, Time Functions (continued)

Example 11: Extract the DAY portion from a date:

```
select DAY(RESCUEDATE) from PETRESCUE
where ANIMAL='Cat'
```

Example 11: Results:

ID	ANIMAL	QUANTITY	COST	RESCUEDATE
1	Cat	9	450.09	2018-05-29
7	Cat	1	44.44	2018-06-11

29
11

Date, Time Functions (continued)

Example 12: Get the number of rescues during the month of May:

```
select COUNT(*) from PETRESCUE
where MONTH(RESCUEDATE)='05'
```

Example 12: Results:

1

ID	ANIMAL	QUANTITY	COST	RESCUEDATE
1	Cat	9	450.09	2018-05-29
7	Cat	1	44.44	2018-06-11

Counting things in a specif

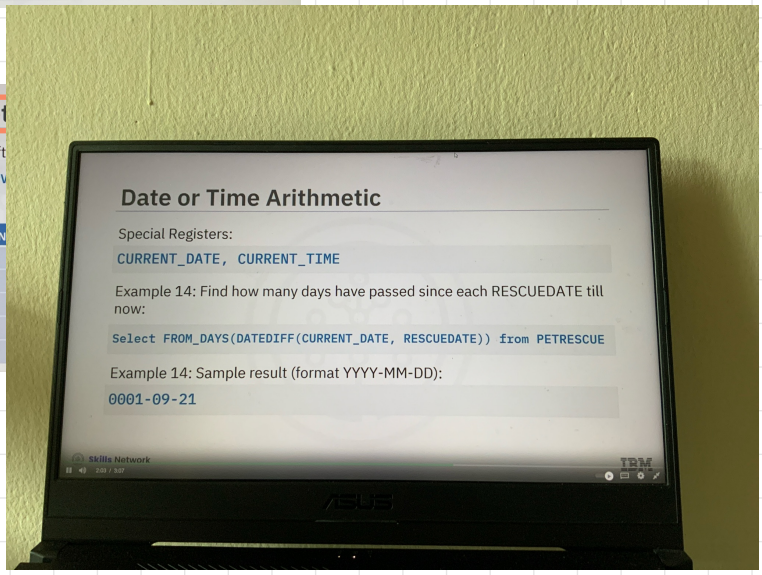
Date or Time Arithmet

Example 13: What date is it 3 days aft

```
Select DATE_ADD(RESCUEDATE, INTERV
```

Example 13: Results:

	ID	ANIMAL	QUAN
2018-06-01	1	Cat	
2018-06-04	2	Dog	
2018-06-07	3	Dog	
2018-06-07	4	Parrot	
2018-06-13	5	Dog	



Sub-queries and Nested Select

Sub-queries and Nested Selects

Sub-query: A query inside another query

```
select COLUMN1 from TABLE
  where COLUMN2 = (select MAX(COLUMN2) from TABLE)
```

Where clause

EMPLOYEES

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID	DEP_ID
E1001	John	Thomas	123456	1976-01-09	M	5631 Rice, OakPark,IL	100	100000	30001	2
E1002	Alice	James	123457	1972-07-31	F	980 Berry ln, Elgin,IL	200	80000	30002	5
E1003	Steve	Wells	123458	1980-08-10	M	291 Springs, Gary,IL	300	50000	30002	5

Sub-queries in list of columns

- Substitute column name with a sub-query
- Called Column Expressions

```
select EMP_ID, SALARY, AVG(SALARY) AS AVG_SALARY
       from employees ;
```

Error

From clause

```
select EMP_ID, SALARY,
       ( select AVG(SALARY) from employees )
                AS AVG_SALARY
       from employees ;
```

Sub-queries in FROM clause

- Substitute the TABLE name with a sub-query
- Called Derived Tables or Table Expressions
- Example:

```
select * from
       ( select EMP_ID, F_NAME, L_NAME, DEP_ID
                from employees) AS EMP4ALL ;
```

Sub-queries in FROM clause

Result:

EMP_ID	F_NAME	L_NAME	DEP_ID
E1002	Alice	James	5
E1003	Steve	Wells	5
E1004	Santosh	Kumar	5
E1005	Ahmed	Hussain	2
E1006	Nancy	Allen	2
E1007	Mary	Thomas	7
E1008	Bharath	Gupta	7
E1009	Andrea	Jones	7
E1010	Ann	Jacob	5

Multiple Table with Sub query

Where Clause

Ex1

Accessing multiple tables with sub-queries

Query:

```
select * from employees where DEP_ID IN
       ( select DEPT_ID_DEP from departments );
```

Result:

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID	DEP_ID
E1002	Alice	James	123457	7/31/1972	F	980 Berry ln, Elgin,IL	200	80000	30002	5
E1003	Steve	Wells	123458	8/10/1980	M	291 Springs, Gary,IL	300	50000	30002	5
E1004	Santosh	Kumar	123459	7/20/1985	M	511 Aurora Av, Aurora,IL	400	60000	30004	5
E1009	Andrea	Jones	123414	7/3/1						7
E1010	Ann	Jacob	123415	3/30/1982	F	Springs,Elgin,IL	220	70000	30004	5

table is used for filtering the result set of the outer query.

Ex2

Multiple tables with sub-queries

Query:

```
select * from employees where DEP_ID IN
       ( select DEPT_ID_DEP from departments where LOC_ID = 'L0002' );
```

Result:

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID
E1002	Alice	James	123457	7/31/1972	F	980 Berry ln, Elgin,IL	200	80000	30002
E1003	Steve	Wells	123458	8/10/1980	M	291 Springs, Gary,IL	300	50000	30002
E1004	Santosh	Kumar	123459	7/20/1985	M	511 Aurora Av, Aurora,IL	400	60000	30004

From Clause

Accessing multiple tables with Implicit Join

Specify 2 tables in the FROM clause:

```
select * from employees, departments;
```

The result is a full join (or Cartesian join):

- Every row in the first table is joined with every row in the second table
- The result set will have more rows than in both tables

Accessing multiple tables with Implicit Join

Use additional operands to limit the result set:

```
select * from employees, departments
       where employees.DEP_ID =
             departments.DEPT_ID_DEP;
```

Use shorter aliases for table names:

```
select * from employees E, departments D
       where E.DEP_ID = D.DEPT_ID_DEP;
```

Accessing multiple tables with Implicit Join

Query:

```
select * from employees E, departments D
       where E.DEP_ID = D.DEPT_ID_DEP;
```

Alias

Result:

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID	DEP_ID	DEPT_ID_DEP	DEP_NAME	MANAGER_ID	LOC_ID
E1002	Alice	James	123457	7/31/1972	F	980 Berry ln, Elgin,IL	200	80000	30002	5	5	Software Group	30002	L0002
E1003	Steve	Wells	123458	8/10/1980	M	291 Springs, Gary,IL	300	50000	30002	5	5	Software Group	30002	L0002
E1004	Santosh	Kumar	123459	7/20/1985	M	511 Aurora Av, Aurora,IL	400	60000	30002	5	5	Software Group	30002	L0002
E1007	Mary	Thomas	123412	5/5/1975	F	100 Rose Pl, Gary,IL	650	65000	30003	7	7	Design Team	30003	L0003
E1008	Bharath	Gupta	123413	5/6/1985	M	145 Berry Ln, Naperville,IL	660	65000	30003	7	7	Design Team	30003	L0003
E1009	Andrea	Jones	123414	7/9/1990	F	120 Fall Creek, Gary,IL	234	70000	30003	7	7	Design Team	30003	L0003

Accessing multiple tables with Implicit Join

To see the department name for each employee:

```
select EMP_ID, DEP_NAME
       from employees E, departments D
       where E.DEP_ID = D.DEPT_ID_DEP;
```

Column names in the select clause can be pre-fixed by aliases:

```
select E.EMP_ID, D.DEP_ID_DEP from
       employees E, departments D
       where E.DEP_ID = D.DEPT_ID_DEP
```

Accessing multiple tables with Implicit Join

Query:

```
select EMP_ID, DEP_NAME from employees E, departments D
      where E.DEP_ID = D.DEPT_ID_DEP;
```

Result:

EMP_ID	DEP_NAME
E1002	Software Group
E1003	Software Group
E1004	Software Group
E1007	Design Team
E1008	Design Team
E1009	Design Team
E1010	Software Group