CSE2DBF – CSE4DBF

Relational Database Language - SQL

12/04/2021

Reading:
Elmasri and Navathe, "Fundamentals of Database Systems, Chapters 1 & 2", Pearson, 2016.
***Ebook**: https://ebookcentral-proquest-com.ez.library.latrobe.edu.au/lib/latrobe/detail.action?docID=5573709*

Introduction

SQL (Structured Query Language)

The standard relational database language

SQL 1

The first standard for SQL was defined by the *American Standards Institute* (ANSI) in 1986 and subsequently adopted by the International Standards Organisation (ISO) in 1987.

SQL2

The revised version of the processor (also called SQL 92). This is the standard that has been adopted as the formal standard language for defining and manipulating relational database.

SQL 3

Further extension with additional features such as user-defined data types, triggers, user-defined functions and other Object Oriented features.

Introduction

SQL has 2 major components:

A **Data Definition Language (DDL)** for defining the database structure

A **Data Manipulation Language (DML)** for retrieving and updating data

SQL

Data Definition Language

1. CREATE Table Command
2. Table Specifications
3. Data Types
4. Integrity Constraints
5. DROP Command

SQL – 1. CREATE TABLE Command

- A **CREATE TABLE command** is used to define a new table by specifying the table and its attributes and constraints.

- Create table command:

```
CREATE TABLE <table_name>
    (<field1> <data_type> [<constraint>] [<check>],
     <field2> <data_type> [<constraint>] [<check>],
     ..,
     PRIMARY KEY (<unique_field>),
     [FOREIGN KEY (<foreign_field>)
            REFERENCES <associated_table> (<foreign_field>)]
    );
```

SQL – 1. CREATE Command

Example:

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

DEPARTMENT

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

SQL – 1. CREATE Command

Create table specifications:

```
CREATE  TABLE  DEPARTMENT(
  deptNumber   VARCHAR2(5) not null,
  deptName     VARCHAR2(30) not null,
  location     VARCHAR2(15),
  mailNumber   NUMBER CHECK (mailNumber > 10),
  PRIMARY KEY (deptNumber));

CREATE  TABLE  EMPLOYEE(
  employeeNo  VARCHAR2(5) not null,
  firstName   VARCHAR2(20),
  lastName    VARCHAR2(20),
  deptNumber  VARCHAR2(5),
  salary      NUMBER,
  PRIMARY KEY (employeeNo),
  FOREIGN KEY (deptNumber) REFERENCES DEPARTMENT(deptNumber));
```

```
CREATE  TABLE  DEPARTMENT(
  deptNumber   VARCHAR2(5) not null, PRIMARY KEY
  deptName     VARCHAR2(30) not null,
  location     VARCHAR2(15),
  mailNumber   NUMBER CHECK (mailNumber > 10));
```

SQL – 1. CREATE Command

Example: In the case of weak entities, multivalued attributes, M:M relations and n-ary relations part of all of the primary key will also form a foreign-key relationship:

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

DEPENDENT

Employee No.	Dependent Name	Date of Birth
E1	Joshua Smith	12-Jun-2008
E3	Jay Ramanthan	04-Jan-2006
E1	Jemima Smith	08-Sep-2010

SQL – 1. CREATE Command

Create table specifications:

```
CREATE  TABLE  DEPENDENT(
  employeeNo              VARCHAR2(5) not null,
  dependentName           VARCHAR2(70) not null,
  dateOfBirth             DATE,
  PRIMARY  KEY (employeeNo, dependentName),
  FOREIGN KEY (employeeNo) REFERENCES EMPLOYEE(employeeNo));
```

SQL – 2. Table Specification

- Table and field names:
  - Have 1-30 characters
  - Have to be formed of alphanumeric characters and the special characters ($ _ #)
  - Must begin with an alphanumeric character
  - Cannot be a reserved word

SQL – 3. Data Types

- Data type specifies type of data stored in a field
  - Error checking
  - Efficient use of storage space

- Data types in Oracle:
  - Character data types: 4 sub types
  - Number data types: 3 sub types
  - Date data type

SQL – 3. Data Types

- **Character data types**

VARCHAR2:

- Variable-length character strings with maximum of 4,000 characters
- Must specify maximum width allowed
- No trailing blank spaces are added
- Example declaration: `student_name VARCHAR2(30)`

CHAR:

- Fixed-length character data with maximum size 255 characters
- Must specify maximum width allowed
- Adds trailing blank spaces to pad width
- Example declaration: `s_gender CHAR(1)`

NCHAR:

- Supports 16-digit binary character codes
- Used for alternate alphabets (e.g. Japanese Kanji)

LONG:

- Stores up to 2 GB of variable-length character data
- Each table can have only one LONG field

SQL – 3. Data Types

- **Number data types**
  - General declaration format for **NUMBER**

```
variablename NUMBER <(precision,scale)>
```

  - Number type (integer, fixed point, floating point) specified by precision and scale
    - **Precision**: Total number of digits on either side of the decimal point
    - **Scale**: Number of digits to right of decimal point

SQL – 3. Data Types

Integer Numbers

- Whole number with no digits to right of decimal point
- Precision is maximum width
- Scale is omitted
- Example declaration: `s_age NUMBER(2)`

Fixed Point Numbers

- Contains a specific number of decimal places
- Precision is maximum width
- Scale is number of decimal places
- Sample declaration: `item_price NUMBER(5,2)`

Floating Point Numbers

- Contains a variable number of decimal places
- Precision and scale are omitted
- Sample declaration: `s_GPA NUMBER`

SQL – 3. Data Types

- **Date data type**
  - **DATE** stores dates from 1/1/4712 BC to 12/31/4712 AD
  - Default date format: **`DD-MON-YYYY`**, example: 05-MAR-2016

    Example declaration: `s_dob DATE`
  - DATE data type also stores time values

    Default time format: **`HH:MI:SS A.M`**.
  - If no time value is given when a date is inserted, default value is 12:00:00 A.M.
  - If no date value is given when a time is inserted, default date is first day of current month
  - Example s_dob field: `01-APR-2020 12:00:00 A.M.`

SQL – 4. Integrity Constraints

- **Used to define primary and foreign keys**
- Primary Key constraints:
  - Defining a primary key:

```
PRIMARY KEY (<unique_field>);
```

  - Defining a composite primary key:

```
PRIMARY KEY (<field1, field2>);
```

- Foreign Key constraints:
  - Defining a foreign key:

```
FOREIGN KEY (<foreign_field>) REFERENCES
<associated_table> (<foreign_field>)
```

SQL – 4. Integrity Constraints

- **Value Constraints**:
  - Restricts data values that can be inserted into a field
  - Types
    - Check condition: `CHECK <values>`
    - Example:

```
s_gender CHAR(1) CHECK ((s_gender = 'M') OR
                        (s_gender = 'F'))
```

    - **NOT NULL**: Specifies that a field cannot be NULL

      Example:

```
s_name VARCHAR2(30) NOT NULL
```

SQL – 5. Drop Commands

The **DROP** Command

- To remove a table that is no longer needed in the database.

  Example:

```
DROP TABLE <TABLE_NAME> [RESTRICT|CASCADE]
```

  - If **RESTRICT** is specified then the table is dropped only if it is not referenced by any other table.
  - If **CASCADE** is specified then all references/dependents will also be dropped.

```
SQL> DROP TABLE DEPARTMENT CASCADE CONSTRAINTS;

Table dropped.

SQL> SELECT * FROM EMPLOYEE;

EMPLO FIRSTNAME            LASTNAME            DEPTN     SALARY
----- -------------------- -------------------- ----- ----------
E1    Mandy                Smith                D1         50000
E2    Daniel               Hodges               D2         45000
E3    Shaskia              Ramanthan            D2         58000
E4    Graham               Burke                D1         64000
E5    Annie                Nguyen               D1         60000
```

SQL – 5. Drop Commands

The **DROP** Command:

- Drop puts the table in the recycle bin.

- To see what is in the recycle bin use SHOW.

  Example:

```
SHOW RECYCLEBIN;
```

- The table can then be retrieved from the recycle bin.

  Example:

```
FLASHBACK TABLE <TABLE_NAME> TO BEFORE DROP;
```

- To remove the table from the database completely use PURGE.

  Example:

```
PURGE TABLE <TABLE_NAME>;
```

SQL

Data Definition Language (ctd)

Other CREATE Commands

- **CREATE SCHEMA OR CREATE DATABASE**
  - CREATE SCHEMA [schema_name]
  - CREATE DATABASE [database_ name]
- **CREATE INDEX**
  - CREATE [UNIQUE] INDEX [index_name] ON [attribute_name]

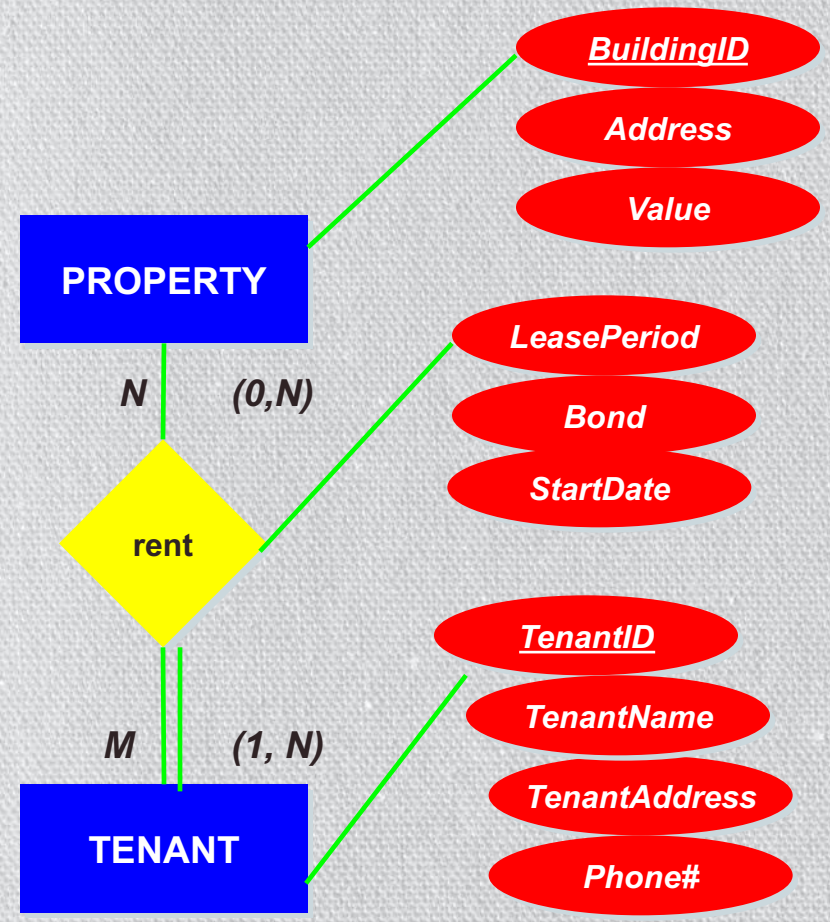
SQL

Data Definition Language (ctd)

Other DROP Commands

- **DROP SCHEMA**
  - DROP SCHEMA [name]
- **DROP INDEX**
  - DROP INDEX [index_name]

DDL Exercise



Recall this adapted part of ER Diagram from Rental Property Case Study in Lecture 2. You need to implement the tables using DDL.

Make sure you use appropriate data types and identify the keys. In addition, follow these requirements:

- Tenant's phone number must be recorded.
- If recorded, the value of "bond" must be written with 2 number of digits for the scale;
- Lease Period must be one of the following (half year, one year, two years);

DDL Exercise

PROPERTY (BuildingID, Address, Value)
TENANT(TenantID, TenantName, TenantAddress, Phone#)
RENT(*BuildingID, TenantID,* StartDate, LeasedPeriod, Bond)

```
CREATE  TABLE  PROPERTY(
 BuildingID          VARCHAR2(5) not null,
 Address             VARCHAR2(100) not null,
 Value               NUMBER,
 PRIMARY  KEY (BuildingID));

CREATE  TABLE  TENANT(
 TenantID            VARCHAR2(5) not null,
 TenantName          VARCHAR2(20) not null,
 TenantAddress       VARCHAR2(40),
 Phone#              VARCHAR2(10) not null,
 PRIMARY  KEY (TenantID));
```

DDL Exercise

PROPERTY (BuildingID, Address, Value)
TENANT(TenantID, TenantName, TenantAddress, Phone#)
RENT(*BuildingID, TenantID,* StartDate, LeasedPeriod, Bond)

```
CREATE  TABLE  RENT(
 BuildingID          VARCHAR2(5) not null,
 TenantID            VARCHAR2(5) not null,
 StartDate           DATE not null,
 LeasedPeriod        VARCHAR2(20) CHECK
            (LeasedPeriod IN ('half year' , 'one year' , 'two years')),
 Bond                NUMBER (6,2),
 PRIMARY  KEY (BuildingID, TenantID, StartDate),
 FOREIGN KEY (BuildingID) REFERENCES PROPERTY(BuildingID),
 FOREIGN KEY (TenantID) REFERENCES TENANT(TenantID));
```

SQL

Data Manipulation Language Statements

INSERT: to insert data into a table.

SELECT: to query data in the database.

UPDATE: to update data in a table.

DELETE: to delete data from a table.

SQL

Modify Commands - Insertion

There are three SQL modify commands that can be used to modify the database (Update and Delete will be discussed on the next few slides) :

- **INSERT**

```
INSERT INTO <table_name>
VALUES (column1 value, column2 value, …);

INSERT INTO <table_name> (field1, field2, …)
VALUES (column1 value, column2 value, …);
```

Example

```
INSERT INTO DEPARTMENT
VALUES ('D5', 'Biology', 'Bundoora',33);

INSERT INTO DEPARTMENT(DeptNumber, DeptName)
VALUES ('D6', 'Chemistry');
```

SQL

Data Manipulation Language (ctd)

Example:

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

DEPARTMENT

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

SQL

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

Basic SQL SELECT Queries

```
SELECT <attribute list>
FROM <table list>
[WHERE <conditions>]
```

```
SELECT firstName, lastName
FROM Employee
WHERE employeeNo = 'E1';
```

FIRSTNAME	LASTNAME
Mandy	Smith

The Relational Algebra Expression:

Π firstName, lastName (σ employeeNo = "E1" (EMPLOYEE))

PROJECTION SELECTION

SQL

Operators

=	equal to
<	less than
>	greater than
<=	less than equal to
>=	greater than equal to
<>	not equal to
LIKE	% used as wildcard
	eg. LIKE ‘%PRE%’
IN	test for in an enumerated list.

SQL

Compound Comparison

```
SELECT      deptNumber
FROM        EMPLOYEE
WHERE       lastName = 'Smith'
OR          lastName = 'Hodges';
```

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

```
Dept Num
--------
D1
D2
```

SQL

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

Set Membership Search (IN / NOT IN)

```
SELECT deptNumber, mailNumber
FROM DEPARTMENT
WHERE deptName IN ('Computer Science', 'Physics');
```

Dept Num	Mail Number
D1	39
D3	37

Pattern Match Search (LIKE / NOT LIKE)

```
SELECT employeeNo, deptNumber
FROM EMPLOYEE
WHERE firstName LIKE '%an%';
```

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Employee	Dept Num
E1	D1
E2	D2

SQL

Pattern Match Search (ctd)

Alternatives:

LIKE 'X%' : first character must be 'X', the rest can be anything.

LIKE '%Y' : any sequence of characters, of length at least 1 , with the last character a 'Y'.

NOT LIKE '%Z' : the last character cannot be a 'Z'.

SQL

Duplicate Removal

SQL does not remove duplicates unless explicitly asked to do so. (Removal of duplicates is computationally expensive.)

```
SELECT DISTINCT deptNumber
FROM EMPLOYEE;
```

```
SQL> SELECT deptnumber FROM EMPLOYEE;

Dept Num
--------
D1
D2
D2
D1
D1

SQL> SELECT DISTINCT deptnumber FROM EMPLOYEE;

Dept Num
--------
D1
D2
```

SQL

Sorting Output from Queries

SQL Output is not sorted by default.

We can use '**Order by**' statement to display an ordered selection.

```
SELECT employeeNo, lastName
FROM EMPLOYEE
ORDER BY lastName;
```

Employee	LASTNAME
E4	Burke
E2	Hodges
E5	Nguyen
E3	Ramanthan
E1	Smith

SQL

Grouping Output from Queries

Example – no grouping

```
SELECT count(*)
FROM EMPLOYEE;
```

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Without group by COUNT(*) returns the number of tuples in the table:

```
  COUNT(*)
----------
         5
```

COUNT(*) = 5

SQL

Grouping Output from Queries

Example – group by

```
SELECT deptNumber, count(*)
FROM EMPLOYEE
GROUP BY deptNumber;
```

```
Dept Num    COUNT(*)
--------  ----------
D1                 3
D2                 2
```

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Count WITHIN the grouping

Dept Number	Count(*)
D1	3
D2	2

SQL

Grouping Output from Queries

To restrict a **GROUP BY** use **HAVING**, because **WHERE** clause only applies to single rows.

```
SELECT deptNumber, count(*)
FROM EMPLOYEE
GROUP BY deptNumber
HAVING count(*)>2;
```

Dept Num	COUNT(*)
D1	3

SQL

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

HAVING and WHERE Condition

```
SELECT deptNumber, count(*)
FROM EMPLOYEE
GROUP BY deptNumber
HAVING count(*) > 2;
```

```
SELECT deptNumber, count(*)
FROM EMPLOYEE
WHERE salary > 50000
GROUP BY deptNumber
HAVING count(*) >= 1;
```

Dept Num	COUNT(*)
D1	3

Dept Num	COUNT(*)
D1	2
D2	1

SQL

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

SQL Aggregate Function

COUNT: returns the number of tuples, which meet the specified condition.

```
SELECT COUNT(*) AS hi_sal
FROM EMPLOYEE
WHERE salary > 50000;
```

```
   HI_SAL
---------
         3
```

SUM: returns the sum of the values in a specified column (numeric column).

```
SELECT COUNT (*) AS hi_sal, SUM (salary) as
       total_hi_sal
FROM EMPLOYEE
WHERE salary > 50000;
```

```
   HI_SAL TOTAL_HI_SAL
--------- ------------
         3       182000
```

SQL

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

SQL Aggregate Function (ctd)

MIN: returns the minimum value in a specified column.

MAX: returns the maximum value in a specified column.

AVG: returns the average of the values in a specified column.

Example:

```
SELECT MIN(salary) AS min_sal,
       MAX(salary) AS max_sal,
       AVG(salary) AS avg_sal
FROM EMPLOYEE;
```

```
   MIN_SAL     MAX_SAL     AVG_SAL
---------- ---------- ----------
     45000       64000       55400
```

SQL

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

SQL Aggregate Function (ctd)

```
SELECT deptNumber, MIN(salary) AS min_sal,
       MAX(salary) AS max_sal, AVG(salary) AS avg_sal
       FROM EMPLOYEE
       GROUP BY deptNumber;
```

Dept Num	MIN_SAL	MAX_SAL	AVG_SAL
D1	50000	64000	58000
D2	45000	58000	51500

SQL

Oracle SQL PLUS
DATETIME Functions

EMPLOYEE

Employee No.	First Name	Last Name	*Dept Number*	Salary	HiringDate
E1	Mandy	Smith	D1	50000	05-Jan-2018
E2	Daniel	Hodges	D2	45000	10-Feb-2018
E3	Shaskia	Ramanthan	D2	58000	16-Feb-2018
E4	Graham	Burke	D1	64000	01-Mar-2018
E5	Annie	Nguyen	D1	60000	01-Apr-2018

TO_CHAR(): converts a DateTime object (or a number) to a string

TO_CHAR has a number of parameters that affect the format of the returned string.

For example:

- **TO_CHAR(SYSDATE, 'DD-MM-YYYY')** returns today's date in the format 04-05-2020 (assuming today's date is May 4th 2020)
- **TO_CHAR(SYSDATE, 'YYYY')** just returns the year: 2020
- **TO_CHAR(SYSDATE, 'HH:MI')** just returns the time component: 14:25

There are many ways a Date object can be formatted using TO_CHAR() – please refer to the Oracle Documentation.

SQL

Oracle SQL PLUS DATETIME Functions

TO_DATE(): converts a string to a DateTime object

TO_DATE takes the string and the converted format as arguments

For example:

TO_DATE('April 04, 2017', 'MONTH DD, YYYY')

If you don't specify a format you can use the default:

TO_DATE('04-APR-2017')

i.e. default: DD-MON-YYYY

If you don't specify a time component the time is set to 12:00:00 A.M. by default

SQL

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Subqueries or Nested Queries

A complete SQL statement embedded within another SELECT statement.

The result of the inner SELECT statement is used in the outer statement to help determine the contents of the final result.

Subquery with equality:

```
SELECT firstName,lastName
FROM EMPLOYEE
WHERE deptNumber =
        (SELECT deptNumber
         FROM DEPARTMENT
         WHERE mailNumber = 39);
```

```
FIRSTNAME            LASTNAME
-------------------- --------------------
Mandy                Smith
Graham               Burke
Annie                Nguyen
```

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

SQL

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

Nested Subquery (use of IN):

```
SELECT firstName, lastName
FROM EMPLOYEE
WHERE deptNumber IN
        (SELECT  deptNumber
         FROM  DEPARTMENT
         WHERE   location = 'Bundoora');
```

FIRSTNAME	LASTNAME
Mandy	Smith
Graham	Burke
Annie	Nguyen

Subquery with aggregate function

```
SELECT firstName, lastName,
   salary-(SELECT avg(salary) FROM EMPLOYEE) AS sal_diff
FROM EMPLOYEE
WHERE salary >
   (SELECT avg(salary)
    FROM EMPLOYEE);
```

FIRSTNAME	LASTNAME	SAL_DIFF
Shaskia	Ramanthan	2600
Graham	Burke	8600
Annie	Nguyen	4600

SQL

Subquery in the FROM clause

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

```
SELECT E.firstName, E.lastName, (E.salary-T.avgsal) as
  Sal_Diff
FROM EMPLOYEE E ,
     (SELECT avg(salary) as avgsal FROM EMPLOYEE) T
WHERE E.salary > T.avgsal;
```

FIRSTNAME	LASTNAME	SAL_DIFF
Shaskia	Ramanthan	2600
Graham	Burke	8600
Annie	Nguyen	4600

SQL

MULTI TABLE QUERIES

Simple Join

```
SELECT E.firstName, E.lastName, D.deptName
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.deptNumber = D.deptNumber;
```

A **prefix** which represents the table name is used in front of each attribute. In join queries where more than one table is involved, a prefix for each attribute is recommended to avoid ambiguity.

You must include a ***join clause for every link*** between 2 tables. If you have N tables in the FROM clause, you must have (N - 1) join clauses.

FIRSTNAME	LASTNAME	DEPTNAME
Mandy	Smith	Computer Science
Daniel	Hodges	Information Science
Shaskia	Ramanthan	Information Science
Graham	Burke	Computer Science
Annie	Nguyen	Computer Science

SQL

MULTI TABLE QUERIES

Joining more than two tables

FIRSTNAME	LASTNAME	PROJTITLE
Mandy	Smith	Project B
Mandy	Smith	Project A
Daniel	Hodges	Project C
Shaskia	Ramanthan	Project A
Graham	Burke	Project A
Annie	Nguyen	Project B

```
SELECT E.firstName, E.lastName, P.projTitle
FROM EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE E.employeeNo = W.employeeNo
AND W.projNo = P.projNo;
```

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000

WORKS_ON

Employee No.	ProjNo
E1	1
E4	1
E5	2
E2	3
E3	1
E1	2

PROJECT

ProjNo	Project Title
1	Project A
2	Project B
3	Project C

SQL

MULTI TABLE QUERIES

Outer Join

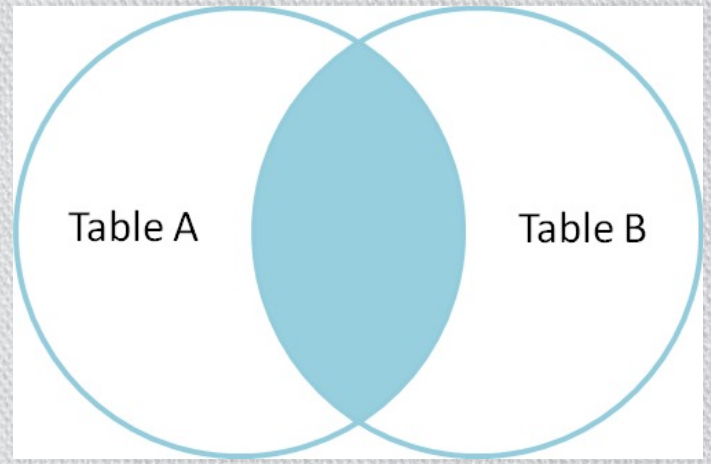
Left Outer Joins: retains rows of the first (left) table that are unmatched with rows from the second (right) table.

Right Outer Joins: retains rows of the second (right) table that are unmatched with rows from the first (left) table.

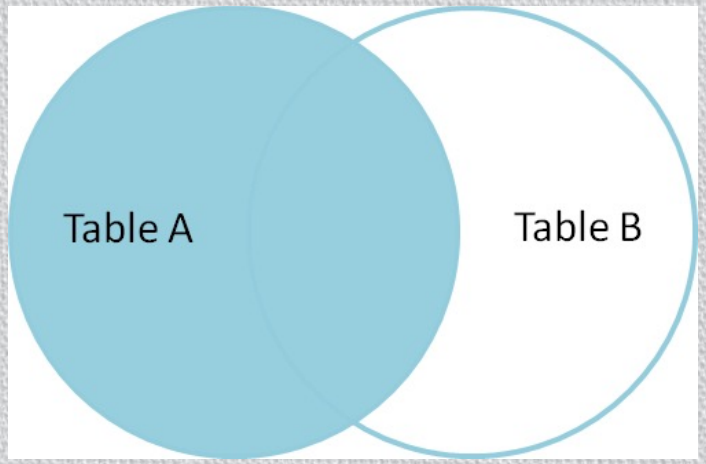
Full Outer Joins: retains rows that are unmatched in both the tables.

In all the above outer joins, the displayed unmatched columns are filled with **NULLS**.

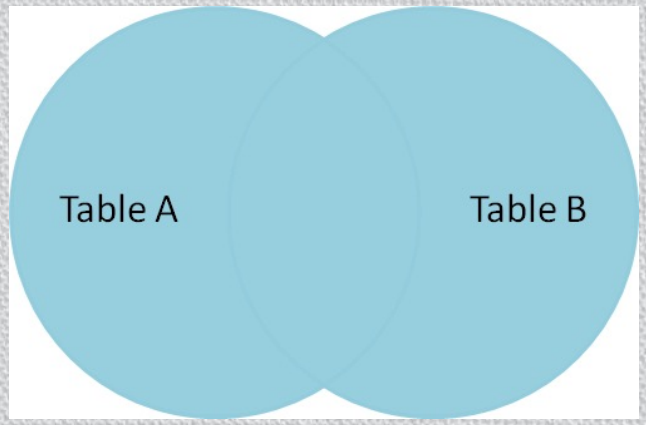
Joins - Visualized



Inner Join



Left Outer Join



Full Outer Join

SQL

```
SELECT E.firstName, E.lastName, D.deptName
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON D.deptNumber = E.deptNumber;
```

MULTI TABLE QUERIES

Outer Join

Example (Left Outer Join):

```
SELECT E.firstName, E.lastName, D.deptName
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.deptNumber = E.deptNumber(+);
```

FIRSTNAME	LASTNAME	DEPTNAME
Mandy	Smith	Computer Science
Daniel	Hodges	Information Science
Shaskia	Ramanthan	Information Science
Graham	Burke	Computer Science
Annie	Nguyen	Computer Science
		Physics
		Chemistry

SQL

MULTI TABLE QUERIES

Full Outer Join

Example (Full Outer Join):

```
SELECT E.firstName, E.lastName, D.deptName
FROM DEPARTMENT D
FULL OUTER JOIN EMPLOYEE E
ON D.deptNumber = E.deptNumber;
```

SQL

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

DEPARTMENT

Dept Number	Dept Name	Location	Mail Number
D1	Computer Science	Bundoora	39
D2	Information Science	Bendigo	30
D3	Physics	Bundoora	37
D4	Chemistry	Bendigo	35

Queries using EXISTS or NOT EXISTS

Designed for use only with sub-queries

EXISTS return true if there exists at least one row in the result table returned by the sub-query, it is false if the sub-query returns an empty result table.

Example:

```
SELECT firstName, lastName
FROM EMPLOYEE E
WHERE EXISTS
      (SELECT *
       FROM DEPARTMENT D
        WHERE D.deptNumber = E.deptNumber
       AND D.location = 'Bendigo');
```

```
FIRSTNAME            LASTNAME
-------------------- --------------------
Shaskia              Ramanthan
Daniel               Hodges
```

SQL

Combining Result Table (UNION, INTERSECT, DIFFERENCE)

Example:

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

DEPENDENT

Employee No.	First Name	Last Name	Date of Birth
E1	Joshua	Smith	12-Jun-2008
E3	Jay	Ramanthan	04-Jan-2006
E1	Jemima	Smith	08-Sep-2010

SQL

NOTE: when applying these operators, ORACLE returns the result as set in Relational Algebra. In another word, duplicate will be removed.

Combining Result Table (UNION, INTERSECT, DIFFERENCE)

Example:

```
SELECT employeeNo, firstName,
lastName
FROM EMPLOYEE
UNION
SELECT employeeNo, firstName,
lastName
FROM DEPENDENT;
```

```
EMPLO FIRSTNAME            LASTNAME
----- -------------------- ------------
E1    Jemima               Smith
E1    Joshua               Smith
E1    Mandy                Smith
E2    Daniel               Hodges
E3    Jay                  Ramanthan
E3    Shaskia              Ramanthan
E4    Graham               Burke
E5    Annie                Nguyen

8 rows selected.
```

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

```
SELECT employeeNo
FROM EMPLOYEE
INTERSECT
SELECT employeeNo
FROM DEPENDENT;
```

```
Employee
--------
E1
E3
```

DEPENDENT

Employee No.	First Name	Last Name	Date of Birth
E1	Joshua	Smith	12-Jun-2008
E3	Jay	Ramanthan	04-Jan-2006
E1	Jemima	Smith	08-Sep-2010

SQL

NOTE: when applying these operators, ORACLE returns the result as set in Relational Algebra. In another word, duplicate will be removed.

Combining Result Table (UNION, INTERSECT, DIFFERENCE)

```
SELECT lastName
FROM EMPLOYEE
UNION
SELECT lastName
FROM DEPENDENT;
```

```
SELECT lastName
FROM EMPLOYEE
UNION ALL
SELECT lastName
FROM DEPENDENT;
```

EMPLOYEE

Employee No.	First Name	Last Name	Dept Number	Salary
E1	Mandy	Smith	D1	50000
E2	Daniel	Hodges	D2	45000
E3	Shaskia	Ramanthan	D2	58000
E4	Graham	Burke	D1	64000
E5	Annie	Nguyen	D1	60000

DEPENDENT

Employee No.	First Name	Last Name	Date of Birth
E1	Joshua	Smith	12-Jun-2008
E3	Jay	Ramanthan	04-Jan-2006
E1	Jemima	Smith	08-Sep-2010

SQL

Modify Commands (ctd)

- **UPDATE**

```
UPDATE <table_name>
SET <column_name> = <new_value>
[WHERE <search_condition>];
```

- **Records can be updated in only one table at a time with a single UPDATE command.**
- Search conditions are listed in the WHERE clause to make the command update specific records

  Format: **`WHERE <expression> <operator> <expression>`**

  Operators:

  `Equal (=), Greater than, Less than (>, <),`

  `Greater than or Equal to (>=),`

  `Less than or Equal to (<=), Not equal (< >)`

SQL

Modify Commands

- **Example**

```
UPDATE EMPLOYEE
SET firstName = 'Johanna'
WHERE employeeNo = 'E1';
```

SQL

Modify Commands

- **DELETE**

```
DELETE FROM <table_name>
[WHERE <search_condition>];
```

  - Can delete multiple records if search condition specifies multiple records.
  - If search condition is omitted, all table records are deleted.

Example

```
DELETE FROM Employee
WHERE firstName = 'Shaskia';
```

SQL

VIEWS (Virtual Tables) in SQL

- A VIEW is a single table that is derived from other existing tables. These other tables can be ordinary tables or another previously defined view.
- A VIEW does not exist in physical form, as opposed to an ordinary base table whose records are actually stored in the database. This is why it is called a virtual table.
- We usually create a view for specifying a table that we need to reference frequently. This table may be built by joining a number of different tables.

- **VIEW general syntax:**

```
CREATE VIEW Name [(attribute names)]
AS
SELECT statements
```

SQL

VIEWS in SQL

```
SQL> SELECT *
  2  FROM DEPT_INFO;

DEPTNAME                       TOTALEMPLOYEE TOTALSALARY
------------------------------ ------------- -----------
Information Science                        2      103000
Computer Science                           3      174000
     .
```

VIEW example:

```
CREATE VIEW DEPARTMENT_STAFF
AS
SELECT E.firstName, E.lastName, D.deptName
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.deptNumber = D.deptNumber;
```

Class Exercise: Complete the following VIEW definition:

```
CREATE VIEW DEPT_INFO(DeptName, TotalEmployee, TotalSalary)
AS
SELECT D.DeptName, COUNT(*), SUM(E.salary)
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.DeptNumber = E.DeptNumber
GROUP BY D.DeptName;
```

SQL

Query by Example (QBE)

- QBE is a user-friendly relational query language.
- Query is defined by filling in templates
- QBE, which was developed by IBM in the 1970's is now provided by a number of commercial DBMS, such as Microsoft Access.

For Example, the query given on a previous slide (Multi Tables Queries – Simple Join) can be represented by QBE as follows:

Field	FirstName	LastName	Department name
Table	EMPLOYEE	EMPLOYEE	DEPARTMENT
Show	✓	✓	✓

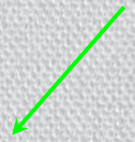
Selected fields from DEPARTMENT table

Selected fields from EMPLOYEE table

SQL

The following example shows a QBE where the criteria is specified.

Field	FirstName	LastName	Department name
Table	EMPLOYEE	EMPLOYEE	DEPARTMENT
Show	✓	✓	✓
Criteria			"Physics"
Or:			"Computer Science"



Criteria on different rows represents OR operator

Next Lecture

Stored Procedure

Reading:
Elmasri and Navathe, "Fundamentals of Database Systems, Chapters 1 & 2", Pearson, 2016.
Ebook: *https://ebookcentral-proquest-com.ez.library.latrobe.edu.au/lib/latrobe/detail.action?docID=5573709*