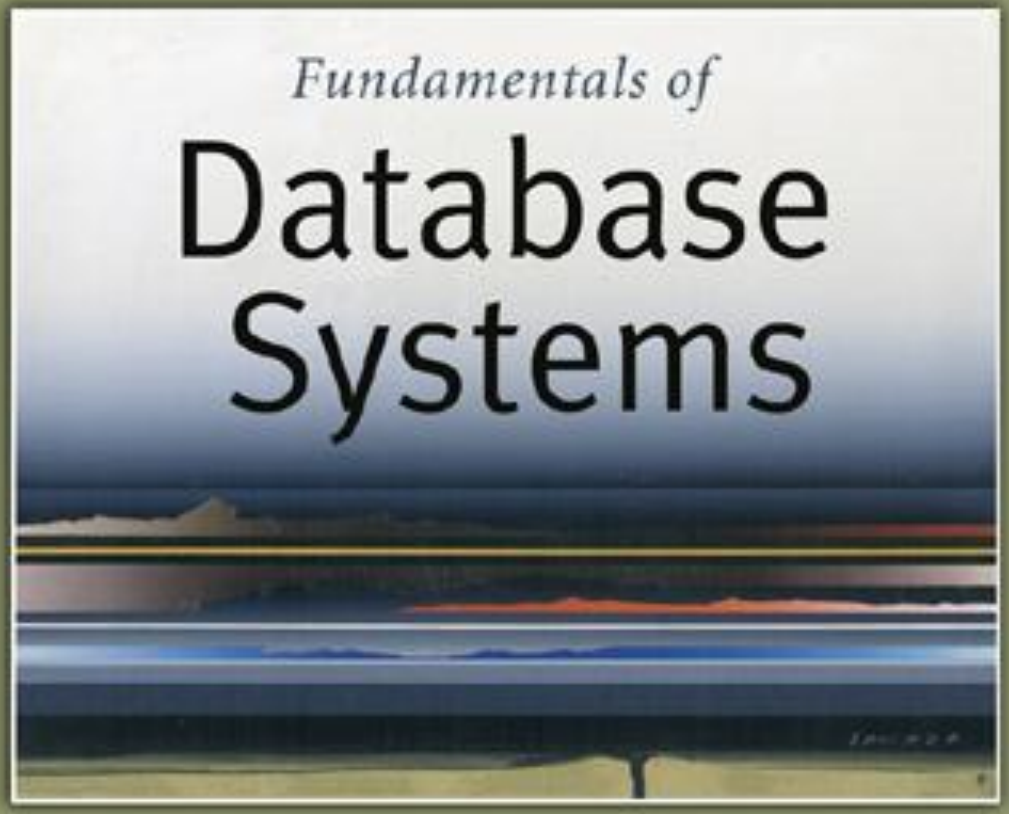


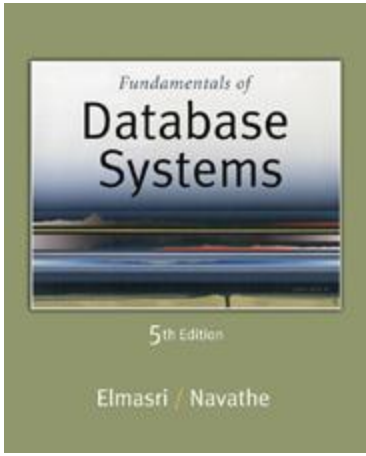
5th Edition

Elmasri / Navathe

Chapter 4

Structured Query Language- SQL

- By Jyoti Tryambake



SQL History

- IBM Sequel Language developed as a part of System R Project at the IBM San Jose Research Laboratory
- Renamed as Structured Query Language (SQL)
- ANSI and ISO standard SQL:
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL: 1999
  - SQL: 2003,2005,2008, 2012, 2014, 2016, 2017

SQL Facilities

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

Data Definition Language (DDL)

- Create and destroy databases and objects
  - creating a table or view
  - Altering/expanding definition of table
  - Creating/dropping an index
- Integrity constraints can be defined at the time of creation or later
- Primarily used by database admin during setup and removal phases of database object
- Commands are – create, update, drop

Data Definition Language (DDL)

- CREATE command
  - Create and manage independent database
  - For example, to maintain a database of customer contacts for your sales department and a personnel database for your Human Resource department
  - Command - creates an empty database named "Employees" on your DBMS
    - **CREATE DATABASE Employees;**

Data Definition Language (DDL)

- CREATE command
  - Next step is to create tables that will contain data
  - The CREATE TABLE command specifies
    - a new base relation by giving it a name,
    - specifying each of its attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n) etc.)
    - A constraint may be specified on an attribute

Data Definition Language (DDL)

- CREATE command
  - The command:

```
CREATE TABLE personal_info
(first_name varchar(20) NOT NULL,
last_name varchar(20) NOT NULL,
employee_id int NOT NULL)
```

Creates a table titled "personal_info" in the current database

Data Definition Language (DDL)

- CREATE command
  - Also used for specifying the primary key attributes, and referential integrity constraints (foreign keys)
  - Key attributes can be specified via the PRIMARY KEY, FOREIGN KEY, REFERENCES and UNIQUE phrases

Data Definition Language (DDL)

- CREATE command
  - To specify CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE   dept_info
(DNAME VARCHAR(10)  NOT NULL,
DNUMBER INTEGER NOT NULL,
EMPLOYEE_ID int,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (EMPLOYEE_ID) REFERENCES
personal_info
ON DELETE SET DEFAULT ON UPDATE CASCADE  );
```

Constraints

There are 5 different referential actions: CASCADE, RESTRICT, NO ACTION, SET NULL, SET DEFAULT

CASCADE

- ON DELETE CASCADE means that if the parent record is deleted, any child records are also deleted.
- ON UPDATE CASCADE means that if the parent primary key is changed, the child value will also change to reflect that.
- ON UPDATE CASCADE ON DELETE CASCADE means that if you UPDATE **OR** DELETE the parent, the change is cascaded to the child.

Constraints

RESTRICT

- RESTRICT means that any attempt to delete and/or update the parent will fail throwing an error.
- This is the default behavior in the event that a referential action is not explicitly specified.
- For an ON DELETE or ON UPDATE that is not specified, the default action is always RESTRICT`.

Constraints (cont..)

NO ACTION

- NO ACTION: equivalent to RESTRICT.
- The MySQL Server rejects the delete or update operation for the parent table if there is a related foreign key value in the referenced table.

SET NULL

- SQL allows NULLs attribute values, a NOT NULL constraint may be specified if NULL is not permitted for a particular attribute
- SET NULL - Delete or update the row from the parent table, and set the foreign key column or columns in the child table to NULL.

Constraints (cont..)

SET DEFAULT

- A default value for an attribute could be set and it will be included in new tuple if an explicit value is not provided for that attribute
- SET DEFAULT. allows the developer to specify a value to which to set the foreign key column(s) on an UPDATE or a DELETE.

CHECK

- Restrict attribute or domain values using CHECK clause following an attribute or domain definitions.

```
Dnumber INT NOT NULL CHECK (Dnumber > 0 AND
Dnumber < 21);
```

Example

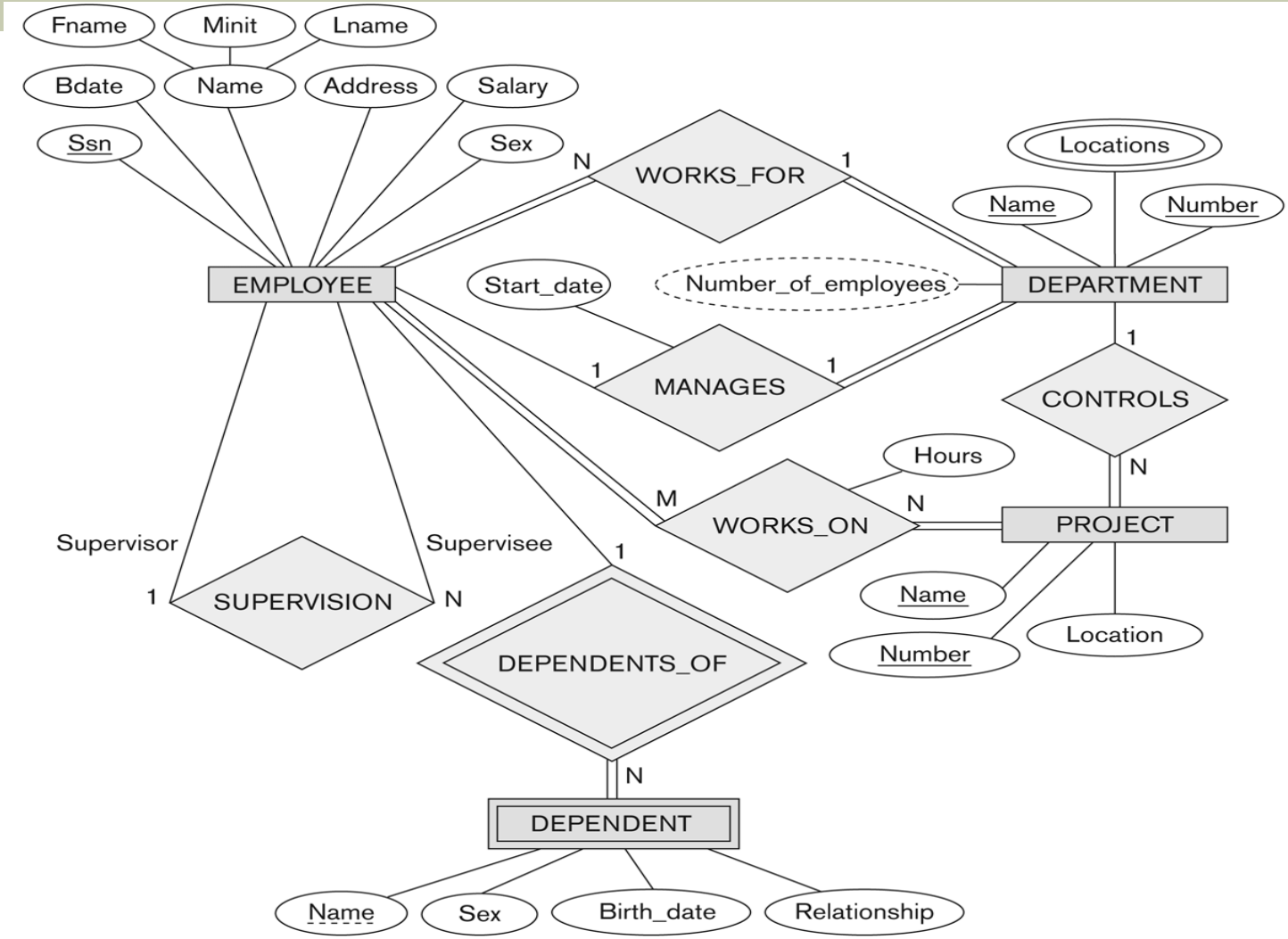


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Examples

- Consider the following relational database schema corresponding to a COMPANY database

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Specifying Key and Referential Integrity Constraints in SQL

CREATE TABLE EMPLOYEE

(. . . ,

Dno INT **NOT NULL DEFAULT** 1,

CONSTRAINT EMPPK ← **Constraint name**

PRIMARY KEY (Ssn),

CONSTRAINT EMPSUPERFK

FOREIGN KEY (Super_ssn) **REFERENCES** EMPLOYEE(Ssn)

ON DELETE SET NULL **ON UPDATE** CASCADE,

CONSTRAINT EMPDEPTFK

FOREIGN KEY(Dno) **REFERENCES** DEPARTMENT(Dnumber)

ON DELETE SET DEFAULT **ON UPDATE** CASCADE);

Data Definition Language (DDL)

- ALTER command
  - Modify information - to make changes to the structure of a table without deleting and recreating it
  - For example, add a new attribute to the personal_info table -- an employee's salary
    - ALTER TABLE personal_info
      ADD salary money null
  - The "money" argument specifies that an employee's salary will be stored using a dollars and cents format
    - Example 2;

      ALTER TABLE personal_info ADD JOB VARCHAR(12);

Alter Command (cont.)

Various forms of Alter command with syntax:

To add a new column:

ALTER TABLE table_name ADD column_name *datatype*;

To delete a column:

ALTER TABLE table_name DROP COLUMN column_name;

To modify a column:

ALTER TABLE table_name MODIFY column_name *datatype*;

Alter Command (cont.)

Various forms of Alter command with syntax:

To rename table:

ALTER TABLE table_name RENAME TO new_table_name;

To rename the column:

ALTER TABLE table_name RENAME COLUMN

old_Column_name to new_Column_name;

Ex.

ALTER TABLE STUDENT ADD Address varchar2 (100);
ALTER TABLE STUDENT DROP COLUMN AGE;

Data Definition Language (DDL)

- DROP command
  - To permanently remove the table
    - DROP TABLE personal_info
  - To remove the entire database
    - DROP DATABASE employees

Data Manipulation Language (DML)

- Retrieving and updating information from more than two tables
- Commands are – select, update, delete, insert
  - **INSERT** command :
    - The INSERT command in SQL is used to add records to an existing table
    - Syntax:

      INSERT INTO table-name (column-names)

      VALUES (values)

Insert examples

Ex.

Problem:

Add a record for a new customer

```
INSERT INTO Customer (FirstName, LastName, City, Country, Phone)
VALUES ('Craig', 'Smith', 'New York', 'USA', 1-01-993 2800)
```

Problem:

Add a new customer named Anita Coats to the database

```
INSERT INTO Customer (FirstName, LastName)
VALUES ('Anita', 'Coats')
```

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

Data Manipulation Language (DML)

- **SELECT** command :
  - Allows database users to retrieve the specific information they desire from an operational database
  - Example, the command shown below retrieves all of the information contained within the personal_info table

```
SELECT * FROM personal_info
```

    - the asterisk (*) is used as a wildcard in SQL - "Select everything from the personal_info table.“

Data Manipulation Language (DML)

- **SELECT** command :
  - Users limit the attributes that are retrieved from the database
  - For example, the Human Resources department may require a list of the last names of all employees in the company
    - SELECT last_name

      FROM personal_info

Data Manipulation Language (DML)

- **SELECT** command :
  - The WHERE clause can be used to limit the records that are retrieved to those that meet specified criteria
  - The following command retrieves all of the data contained within personal_info for records that have a salary value greater than $50,000:

```
SELECT *
FROM   personal_info
WHERE salary > $50,000
```

Data Manipulation Language (DML)

- **UPDATE** command :
  - To modify information contained within a table, either in bulk or individually
  - Syntax:
    - UPDATE table-name SET column-name = value, column-name = value, ...
  - For example, Each year, company gives all employees a 3% cost-of-living increase in their salary
    - UPDATE personal_info
      SET salary = salary * 1.03
    - UPDATE personal_info
      SET salary = salary + $5000
      WHERE employee_id = 2

Data Manipulation Language (DML)

- **DELETE** command :
  - The DELETE command with a WHERE clause can be used to remove specific record from the personal_info table:
  - Syntax:

    DELETE from table-name WHERE condition
  - Example;
    - DELETE FROM personal_info
      WHERE employee_id = 2
    - DELETE all the records from the CUSTOMERS table

      DELETE FROM CUSTOMERS;

Examples

- Consider the following relational database schema corresponding to a COMPANY database

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

- Example of a simple query on *one* relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

- **Q0:**

```
SELECT   BDATE, ADDRESS
FROM     EMPLOYEE
WHERE    FNAME='John' AND
MINIT='B'
AND      LNAME='Smith'
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

- Query 1: Retrieve the name and address of all employees who work for the 'Research' department

```
Q1: SELECT    FNAME, LNAME,
ADDRESS
FROM          EMPLOYEE, DEPARTMENT
WHERE         DNAME='Research' AND
DNUMBER=DNO
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

- Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate

```
Q2:     SELECT PNUMBER, DNUM, LNAME, BDATE,
        ADDRESS
        FROM  PROJECT, DEPARTMENT,
        EMPLOYEE
        WHERE DNUM=DNUMBER AND
        MGRSSN=SSN  AND
        PLOCATION='Stafford'
```

- Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate

Note:

- The join condition DNUM=DNUMBER relates a project to its controlling department
- The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department
- A *missing WHERE-clause* indicates no condition; hence, *all tuples* of the relations in the FROM-clause are selected

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*

  - Examples:

```
Q:  SELECT  * FROM   EMPLOYEE
            WHERE    DNO=5

Q:  SELECT * FROM    EMPLOYEE, DEPARTMENT
            WHERE    DNAME='Research' AND
                     DNO=DNUMBER
```

AND , OR, NOT clause

Problem: Get customer named Thomas Hardy

- SELECT Id, FirstName, LastName, City, Country

FROM Customer

WHERE FirstName = 'Thomas' AND LastName = 'Hardy'

Problem: List all customers from Spain or France

- SELECT Id, FirstName, LastName, City, Country

FROM Customer

WHERE Country = 'Spain' OR Country = 'France‘

Problem: List all customers that are not from the USA

- SELECT Id, FirstName, LastName, City, Country

FROM Customer

WHERE NOT Country = 'USA'

Order by

- ORDER BY allows sorting by one or more columns.
- Records can be returned in ascending or descending order. **The default sort order is ascending.**
- The general syntax is:

  SELECT column-names FROM table-name

  WHERE condition

  ORDER BY column-names ASC|DESC

Problem: List all suppliers in alphabetical order

```
SELECT CompanyName, ContactName, City, Country
FROM Supplier
ORDER BY CompanyName
```

Order by

Problem: List all customers in descending order

SELECT * FROM CUSTOMERS ORDER BY NAME DESC;

Set Operations

Union

- combine the results of two or more Select statements
- it will eliminate duplicate rows from its result set
- number of columns and datatype must be same in both the tables.

Union all

This operation is similar to Union. But it also shows the duplicate rows.

Intersect

- combine two SELECT statements, but it only returns the records which are common from both SELECT statements.
- In case of **Intersect** the number of columns and data-type must be same.

Set Operations

Minus

- combines result of two Select statements and return only those result which belongs to first set of result

Set Operations

Example on Union:
Query: select * from First **UNION** select * from second

Second	
ID	**Name**
2	adam
3	Chester

First	
ID	**Name**
1	abhi
2	adam

Result:

ID	NAME
1	abhi
2	adam
3	Chester

Set Operations

Example on Union all:
Query: select * from First **UNION ALL** select * from second

ID	Name
2	adam
3	Chester

ID	Name
1	abhi
2	adam

Result:

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

Set Operations

Example on Intersect:
Query: select * from First **INTERSECT** select * from second

ID	Name
1	abhi
2	adam

ID	Name
2	adam
3	Chester

Result:

ID	NAME
2	adam

Set Operations

Example on Minus:

Query: select * from First **MINUS** select * from second

The above query will return only those rows which are unique in ‘First’

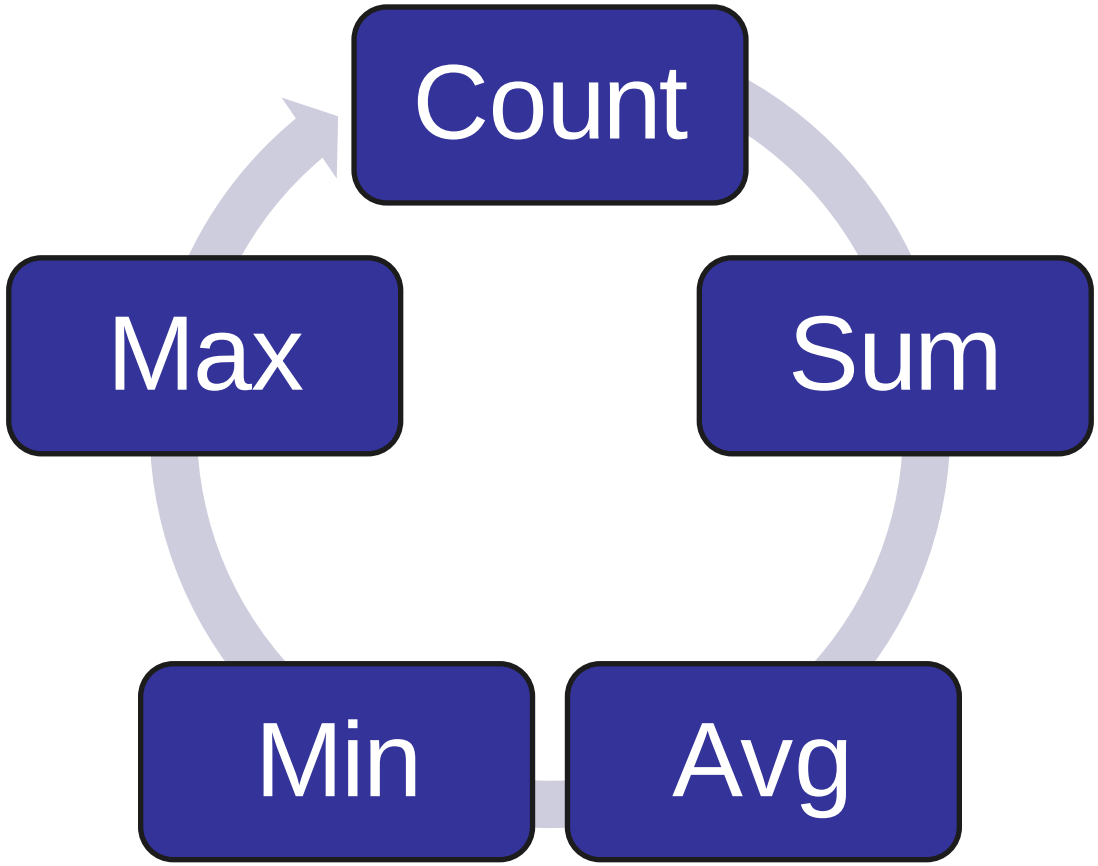
ID	Name
1	abhi
2	adam

ID	Name
2	adam
3	Chester

Result:

ID	NAME
1	abhi

Aggregate Functions



Aggregate Functions

- SELECT COUNT returns a count of the number of data values.
- SELECT SUM returns the sum of the data values.
- SELECT AVG returns the average of the data values.

Problem: Find the number of customers

- SELECT COUNT(Id)

FROM Customer

Count
91

Aggregate Functions

Problem: Compute the total amount sold in 2013

- SELECT SUM(TotalAmount)

FROM [Order]

WHERE YEAR(OrderDate) = 2013

Sum
658388.75

Problem: Compute the average size of all orders

- SELECT AVG(TotalAmount)

FROM [Order]

Average
1631.877819

Aggregate Functions

- SELECT MIN returns the minimum value for a column.
- SELECT MAX returns the maximum value for a column.

Problem: Find the cheapest product

```
- SELECT MIN(UnitPrice)
FROM Product
```

UnitPrice
2.50

Problem: Find the largest order placed in 2014

```
- SELECT MAX(TotalAmount)
FROM [Order]
WHERE YEAR(OrderDate) = 2014
```

TotalAmount
17250.00

Distinct

- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used
- For example, the result of Q1 may have duplicate SALARY values whereas Q2 does not have any duplicate values

```
Q1:   SELECT SALARY
      FROM EMPLOYEE

Q2:   SELECT DISTINCT SALARY
      FROM EMPLOYEE
```

Distinct

- DISTINCT can be used with aggregates: COUNT, AVG, MAX, etc.
- DISTINCT operates on a single column. DISTINCT for multiple columns is not supported.

Distinct examples

- **Problem**: List all supplier countries in alphabetical order.

```
SELECT DISTINCT Country
FROM Supplier
ORDER BY COUNTRY
```

- **Problem**: List the number of supplier countries

```
SELECT COUNT (DISTINCT Country)
FROM Supplier
```

Between

- WHERE BETWEEN returns values that fall within a given range.
- WHERE BETWEEN is a shorthand for >= AND <=.
- BETWEEN operator is inclusive: begin and end values are included.

The general syntax is:

SELECT column-names

FROM table-name

WHERE column-name BETWEEN value1 AND value2

Between

Problem: List all products between $10 and $20

Query:

```
SELECT Id, ProductName, UnitPrice
FROM Product
WHERE UnitPrice BETWEEN 10 AND 20
ORDER BY UnitPrice
```

Id	ProductName	UnitPrice
3	Aniseed Syrup	10.00
46	Spegesild	12.00
31	Gorgonzola Telino	12.50

In

- WHERE IN returns values that matches values in a list or subquery.
- WHERE IN is a shorthand for multiple OR conditions.

The general syntax is:

SELECT column-names

FROM table-name

WHERE column-name IN (values)

In

Problem: List all suppliers from the USA, UK, OR Japan

Query:

```
SELECT Id, CompanyName, City, Country
FROM Supplier
WHERE Country IN ('USA', 'UK', 'Japan')
```

Id	CompanyName	City	Country
1	Exotic Liquids	London	UK
2	New Orleans Cajun Delights	New Orleans	USA
3	Grandma Kelly's Homestead	Ann Arbor	USA
4	Tokyo Traders	Tokyo	Japan

Like

- WHERE LIKE determines if a character string matches a pattern.
- Use WHERE LIKE when only a fragment of a text value is known.
- WHERE LIKE supports two wildcard match options: % and _.

The general syntax is:

SELECT column-names

FROM table-name

WHERE column-name LIKE value

Optional Wildcard characters allowed in 'value' are % (percent) and _ (underscore).

A % matches any string with zero or more characters.

An _ matches any single character.

Like

- **Problem**: List all products with names that start with 'Ca‘
- **Query**:

```
SELECT Id, ProductName, UnitPrice, Package
FROM Product
WHERE ProductName LIKE 'Ca%'
```

Id	ProductName	UnitPrice	Package
18	Carnarvon Tigers	62.50	16 kg pkg.
60	Camembert Pierrot	34.00	15-300 g rounds

Like

- **Problem**: List all products that start with 'Cha' or 'Chan' and have one more character.
- Query:

```
SELECT Id, ProductName, UnitPrice, Package
FROM Product
WHERE ProductName LIKE 'Cha_' OR ProductName LIKE 'Chan_'
```

Id	ProductName	UnitPrice	Package
1	Chai	18.00	10 boxes x 20 bags
2	Chang	19.00	24 - 12 oz bottles

Alias

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of the query.

Syntax:

For Column

SELECT *column_name* AS *alias_name*

FROM *table_name;*

Alias

Examples:

1. Alias for columns

```
SELECT CustomerID as ID, CustomerName AS Customer
FROM Customers;
```

```
SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' +
Country AS Address
FROM Customers;
```

Alias

Syntax:

For Table

SELECT *column_name(s)*

FROM *table_name* AS *alias_name;*

Examples:

2. Alias for tables

```
SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName="Around the
Horn" AND c.CustomerID=o.CustomerID;
```

Alias

Problem: List total customers in each country.

Display results with easy to understand column headers.

Query:

3. Alias for resultant table

```
SELECT COUNT(C.Id) AS TotalCustomers, C.Country AS Nation
FROM Customer C
GROUP BY C.Country
```

TotalCustomers	Nation
3	Argentina
2	Austria
2	Belgium

Alias

Problem: List details of customers who have placed orders (consider two tables- customer and order)

Query:

```
SELECT C.ID, C.NAME, C.AGE, O.AMOUNT FROM
CUSTOMERS AS C, ORDERS AS O WHERE C.ID =
O.CUSTOMER_ID;
```

NULL values

- **NULL** is the term used to represent a missing value.
- a NULL value is different than a zero value or a field that contains spaces.
- **IS NULL** or **IS NOT NULL** operators to check for a NULL value.
- Example:

```
SELECT ID, NAME, AGE, ADDRESS, SALARY FROM CUSTOMERS
WHERE SALARY IS NOT NULL;

SELECT ID, NAME, AGE, ADDRESS, SALARY FROM CUSTOMERS
WHERE SALARY IS NULL;
```

Group by

- The GROUP BY clause groups records into summary rows.
- GROUP BY returns one record for each group.
- GROUP BY also involves aggregates: COUNT, MAX, SUM, AVG, etc.
- GROUP BY can group one or more columns.

  The general syntax is:

  SELECT column-names

  FROM table-name

  WHERE condition

  GROUP BY column-names

Group by

Problem: List the number of customers in each country

- SELECT COUNT(Id), Country

FROM Customer

GROUP BY Country

Count	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada

Group by

employee_number	last_name	first_name	salary	dept_id
1001	Smith	John	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Brad	71000	501
1004	Horvath	Jack	42000	501

Problem: Calculate total salary offered by each department

Query:-

```
SELECT dept_id, SUM(salary) AS total_salaries FROM employees GROUP BY dept_id;
```

Result:	
dept_id	total_salaries
500	119500
501	113000

Group by

product_id	product_name	category_id
1	Pear	50
2	Banana	50
3	Orange	50
4	Apple	50
5	Bread	75
6	Sliced Ham	25
7	Kleenex	NULL

Query:-

```
SELECT category_id, COUNT(*) AS total_products FROM products
WHERE category_id IS NOT NULL GROUP BY category_id ORDER
BY category_id;
```

category_id	total_products
25	1
50	4
75	1

Group by

employee_number	last_name	first_name	salary	dept_id
1001	Smith	John	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Brad	71000	501
1004	Horvath	Jack	42000	501

Problem: Find min salary in each department

Query:

```
SELECT dept_id, MIN(salary) AS lowest_salary
FROM employees GROUP BY dept_id;
```

dept_id	lowest_salary
500	57500
501	42000

Having

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- WHERE and HAVING can be in the same query.
- Syntax:

```
SELECT column-names
FROM table-name
WHERE condition
GROUP BY column-names
HAVING condition
```

Having

Problem: List the number of customers in each country. Only include countries with more than 10 customers.

Query:

```
SELECT COUNT(Id), Country
FROM Customer
GROUP BY Country
HAVING COUNT(Id) > 10
```

Result

Count	Country
11	France
11	Germany
13	USA

Having

Problem: Return only those records from department where the minimum salary is greater than 35000

Query:

```
SELECT department, MIN(salary) AS "Lowest
salary"
FROM employees
GROUP BY department
HAVING MIN(salary) > 35000;
```

Sample Queries

1. Display all data from *Employees* table for all employees who was hired before January 1st, 1992
2. Display the employee number, first name, job id and department number for all employees whose department number is not equal to 20, 60 and 80 (*Employees* table).
3. Display the last name, phone number, salary and manager number, for all employees whose manager number equals 100, 102 or 103 (*Employees* table).
4. Display the first name and salary for all employees whose first name ends with an *e* (*Employees* table).

Sample Queries

Solution

1. SELECT *

FROM employees

WHERE hire_date < '01-JAN-1992‘

2. SELECT employee_id , first_name , job_id, department_id

FROM employees

WHERE department_id NOT IN (20 , 60 , 80)

3. SELECT last_name , phone_number , salary , manager_id

FROM employees

WHERE manager_id IN (103 , 102 , 100)

4. SELECT first_name , salary

FROM employees

WHERE first_name LIKE '%e'

Sample Queries

5. Display the last name and department number for all employees where the second letter in their last name is *i* (*Employees* table).

6. Average salary per department

-Display the department number and average salary for each department.

-Modify your query to display the results only for departments 50 or 80.

7. Display the department number, and the average salary for each department, for all departments whose number is in the range of 20 and 80, and their average salary is greater than 9000.

8. Customers and internet packages (*Customers* & *Packages* tables) –

Write a query to display first name, last name, package number and internet speed for all customers whose package number equals 22 or 27. Order the query in ascending order by last name.

Sample Queries

Solution

5. SELECT last_name , department_id
FROM employees
WHERE last_name LIKE '_i%‘

6. SELECT department_id , AVG(salary)
FROM employees
GROUP BY department_id

SELECT department_id , AVG(salary)
FROM employees
WHERE department_id IN (50, 80)
GROUP BY department_id

7. SELECT AVG(salary) , department_id
FROM employees
WHERE department_id BETWEEN 20 AND 80
GROUP BY department_id
HAVING AVG(salary) > 9000

8. SELECT cust.last_name , cust.first_name , cust.pack_id , pack.speed
FROM customers cust JOIN packages pack
ON cust.pack_id = pack.pack_id
WHERE cust.pack_id IN (27, 22)
ORDER BY cust.last_name

SQL Facilities (cont..)

- Data Control Language (DCL)
  - Database security control including privileges and revoke privileges
  - Commands are – grant, revoke

References

- Navathe
- Korth