Structured Query Language (SQL)

CO226: Database Systems

Structured Query Language (SQL)

Sampath Deegalla
dsdeegalla@pdn.ac.lk

24th July 2014

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL (Structured Query Language)

- SQL is a comprehensive database language.
- It supports both
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
- It includes features such as
  - views
  - security
  - authorization
  - transaction control
  - embedding SQL

SQL vs. Relational algebra/calculus

- SQL includes some features from relational algebra and it is greatly based on tuple relational calculus
- SQL is user friendly than both relational algebra and calculus
- In relational algebra, user should specify how to execute the query operations
- In SQL, user specifies what the result is to be

SQL vs. Relational algebra/calculus

- SQL includes some features from relational algebra and it is greatly based on tuple relational calculus
- SQL is user friendly than both relational algebra and calculus
- In relational algebra, user should specify how to execute the query operations
- In SQL, user specifies what the result is to be

SQL vs. Relational algebra/calculus

- SQL includes some features from relational algebra and it is greatly based on tuple relational calculus
- SQL is user friendly than both relational algebra and calculus
- In relational algebra, user should specify how to execute the query operations
- In SQL, user specifies what the result is to be

SQL vs. Relational algebra/calculus

- SQL includes some features from relational algebra and it is greatly based on tuple relational calculus
- SQL is user friendly than both relational algebra and calculus
- In relational algebra, user should specify how to execute the query operations
- In SQL, user specifies what the result is to be

SQL vs. Relational algebra/calculus

- SQL includes some features from relational algebra and it is greatly based on tuple relational calculus
- SQL is user friendly than both relational algebra and calculus
- In relational algebra, user should specify how to execute the query operations
- In SQL, user specifies what the result is to be

Terminology

SQL	Relational Model
table	relation
row	tuple
column	attribute

CREATE SCHEMA

- Specifies a new database schema by giving it a name
  - `CREATE SCHEMA COMPANY;`
  - In MySQL the following statments can be used to create a database schema
    - `CREATE SCHEMA COMPANY;`
    - `CREATE DATABASE COMPANY;`
    - CREATE SCHEMA is a synonym for CREATE DATABASE

CREATE TABLE

- Specifies a base relation (table) by giving it a name, specifying each of its attributes and initial constraints
- Each attribute is given a name and a data type to specify its domain of values
- A constraint `NOT NULL` may be specified on an attribute to disallow null values

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(10) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE DATE );
```

CREATE TABLE

- Specifies a base relation (table) by giving it a name, specifying each of its attributes and initial constraints
- Each attribute is given a name and a data type to specify its domain of values
- A constraint `NOT NULL` may be specified on an attribute to disallow null values

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(10) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE DATE );
```

CREATE TABLE

- Specifies a base relation (table) by giving it a name, specifying each of its attributes and initial constraints
- Each attribute is given a name and a data type to specify its domain of values
- A constraint NOT NULL may be specified on an attribute to disallow null values

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(10) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE DATE );
```

CREATE TABLE

- Specifies a base relation (table) by giving it a name, specifying each of its attributes and initial constraints
- Each attribute is given a name and a data type to specify its domain of values
- A constraint `NOT NULL` may be specified on an attribute to disallow null values

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(10) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE DATE );
```

CREATE TABLE

- Specifies a base relation (table) by giving it a name, specifying each of its attributes and initial constraints
- Each attribute is given a name and a data type to specify its domain of values
- A constraint `NOT NULL` may be specified on an attribute to disallow null values

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(10) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE DATE );
```

CREATE TABLE

- Key attributes can be specified via the `PRIMARY KEY`
- Use `UNIQUE` phrases to indicate alternate keys/unique values
- Use `FOREIGN KEY` to specify referential integrity

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(15) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9) NOT NULL,
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
);
```

CREATE TABLE

- Key attributes can be specified via the `PRIMARY KEY`
- Use `UNIQUE` phrases to indicate alternate keys/unique values
- Use `FOREIGN KEY` to specify referential integrity

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(15) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9) NOT NULL,
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
);
```

CREATE TABLE

- Key attributes can be specified via the `PRIMARY KEY`
- Use `UNIQUE` phrases to indicate alternate keys/unique values
- Use `FOREIGN KEY` to specify referential integrity

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(15) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9) NOT NULL,
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
);
```

CREATE TABLE

- Key attributes can be specified via the `PRIMARY KEY`
- Use `UNIQUE` phrases to indicate alternate keys/unique values
- Use `FOREIGN KEY` to specify referential integrity

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(15) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9) NOT NULL,
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
);
```

CREATE TABLE

- Key attributes can be specified via the `PRIMARY KEY`
- Use `UNIQUE` phrases to indicate alternate keys/unique values
- Use `FOREIGN KEY` to specify referential integrity

```
CREATE TABLE DEPARTMENT (
DNAME VARCHAR(15) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9) NOT NULL,
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
);
```

Attribute Data Types in SQL

- The basic data types include numeric, character string, bit string, boolean, date and time.
- Numeric data types

  INTEGER/INT and SMALLINT integer numbers of various sizes

  FLOAT/REAL and DOUBLE PRECISION floating-point (real) numbers of various precision

  DECIMAL(i,j)/DEC(i,j) or NUMERIC(i,j) formatted numbers: i - decimal digit, j-number of digit after decimal point

Attribute Data Types in SQL

- The basic data types include numeric, character string, bit string, boolean, date and time.
- Numeric data types

  INTEGER/INT and SMALLINT integer numbers of various sizes

  FLOAT/REAL and DOUBLE PRECISION floating-point (real) numbers of various precision

  DECIMAL(i,j)/DEC(i,j) or NUMERIC(i,j) formatted numbers: i - decimal digit, j-number of digit after decimal point

Attribute Data Types in SQL

- The basic data types include numeric, character string, bit string, boolean, date and time.
- Numeric data types

  INTEGER/INT and SMALLINT integer numbers of various sizes

  FLOAT/REAL and DOUBLE PRECISION floating-point (real) numbers of various precision

  DECIMAL(i,j)/DEC(i,j) or NUMERIC(i,j) formatted numbers: i - decimal digit, j-number of digit after decimal point

Attribute Data Types in SQL

- The basic data types include numeric, character string, bit string, boolean, date and time.
- Numeric data types

  INTEGER/INT and SMALLINT integer numbers of various sizes

  FLOAT/REAL and DOUBLE PRECISION floating-point (real) numbers of various precision

  DECIMAL(i,j)/DEC(i,j) or NUMERIC(i,j) formatted numbers: i - decimal digit, j-number of digit after decimal point

Attribute Data Types in SQL

- Character-string data types
  - CHAR(n)/CHARACTER(n) fixed length character-string: n-number of characters
  - VARCHAR(n)/CHAR VARING(n)/CHARACTER VARYING(n) varying length character-string: n-maximum number of characters
  - CHARACTER LARGE OBJECT/CLOB large text values, e.g., documents. In mysql TEXT data type is used

Attribute Data Types in SQL

- Character-string data types

  CHAR(n)/CHARACTER(n) fixed length character-string: n-number of characters

  VARCHAR(n)/CHAR VARING(n)/CHARACTER VARYING(n) varying length character-string: n-maximum number of characters

  CHARACTER LARGE OBJECT/CLOB large text values, e.g., documents. In mysql TEXT data type is used

Attribute Data Types in SQL

- Character-string data types

  CHAR(n)/CHARACTER(n) fixed length character-string: n-number of characters

  VARCHAR(n)/CHAR VARING(n)/CHARACTER VARYING(n) varying length character-string: n-maximum number of characters

  CHARACTER LARGE OBJECT/CLOB large text values, e.g., documents. In mysql TEXT data type is used

Attribute Data Types in SQL

- Character-string data types

  CHAR(n)/CHARACTER(n) fixed length character-string: n-number of characters

  VARCHAR(n)/CHAR VARING(n)/CHARACTER VARYING(n) varying length character-string: n-maximum number of characters

  CHARACTER LARGE OBJECT/CLOB large text values, e.g., documents. In mysql TEXT data type is used

Attribute Data Types in SQL

- Bit-string data types

BIT(n)	fixed length bit string
BIT VARYING(n)	varying length bit string
BINARY LARGE OBJECT/BLOB	large binary values, e.g., images

Attribute Data Types in SQL

- Bit-string data types
  - BIT(n) fixed length bit string
  - BIT VARYING(n) varying length bit string
  - BINARY LARGE OBJECT/BLOB large binary values, e.g., images

Attribute Data Types in SQL

- Bit-string data types

  BIT(n) fixed length bit string

  BIT VARYING(n) varying length bit string

  BINARY LARGE OBJECT/BLOB large binary values, e.g., images

Attribute Data Types in SQL

- Bit-string data types

  BIT(n) fixed length bit string
  BIT VARYING(n) varying length bit string
  BINARY LARGE OBJECT/BLOB large binary values, e.g., images

Additional Data Types

- BOOLEAN for TRUE/FALSE values
- DATE

DATE	Made up of year-month-day in the format yyyy-mm-dd
TIME	Made up of hour:minute:second in the format hh:mm:ss
TIME(i)	Made up of hour:minute:second plus i additional digits specifying fractions of a second. format is hh:mm:ss:ii...i
TIMESTAMP	Has both DATE and TIME components

Additional Data Types

- BOOLEAN for TRUE/FALSE values
- DATE

DATE	Made up of year-month-day in the format yyyy-mm-dd
TIME	Made up of hour:minute:second in the format hh:mm:ss
TIME(i)	Made up of hour:minute:second plus i additional digits specifying fractions of a second. format is hh:mm:ss:ii...i
TIMESTAMP	Has both DATE and TIME components

Additional Data Types

- BOOLEAN for TRUE/FALSE values
- DATE

DATE	Made up of year-month-day in the format yyyy-mm-dd
TIME	Made up of hour:minute:second in the format hh:mm:ss
TIME(i)	Made up of hour:minute:second plus i additional digits specifying fractions of a second. format is hh:mm:ss:ii...i
TIMESTAMP	Has both DATE and TIME components

Additional Data Types

- BOOLEAN for TRUE/FALSE values
- DATE

DATE	Made up of year-month-day in the format yyyy-mm-dd
TIME	Made up of hour:minute:second in the format hh:mm:ss
TIME(i)	Made up of hour:minute:second plus i additional digits specifying fractions of a second. format is hh:mm:ss:ii...i
TIMESTAMP	Has both DATE and TIME components

Additional Data Types

- BOOLEAN for TRUE/FALSE values
- DATE

DATE	Made up of year-month-day in the format yyyy-mm-dd
TIME	Made up of hour:minute:second in the format hh:mm:ss
TIME(i)	Made up of hour:minute:second plus i additional digits specifying fractions of a second. format is hh:mm:ss:ii...i
TIMESTAMP	Has both DATE and TIME components

Additional Data Types

- INTERVAL
  - INTERVAL Specifies a relative value rather than an absolute value. Can be DAY/TIME intervals or YEAR/MONTH intervals. Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value.
- Note: not available in MySQL

Additional Data Types

- INTERVAL

  INTERVAL Specifies a relative value rather than an absolute value. Can be DAY/TIME intervals or YEAR/MONTH intervals. Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value.

- Note: not available in MySQL

Specifying constraints in SQL

- `NOT NULL` to indicate null is not permitted for a perticular attribute
- `DEFAULT` to define a default value for an attribute
- `PRIMARY KEY` to specify one or more attributes that make up the primary key of a relation
- `UNIQUE` to specify alternate keys
- `FOREIGN KEY` to specify referential integrity constraints
- `CONSTRAINT` to give name to constraints

REFERENTIAL INTEGRITY OPTIONS

- We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)
- This must be specified with either ON DELETE or ON UPDATE
- RESTRICT reject update operation that will cause a violation
- SET NULL set to null value
- CASCADE propaget the update
- SET DEFAULT set to default value

REFERENTIAL INTEGRITY OPTIONS

- Examples

```
CREATE TABLE DEPT (
DNAME VARCHAR(15) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN)
ON DELETE SET DEFAULT ON UPDATE CASCADE
);
```

REFERENTIAL INTEGRITY OPTIONS

```
CREATE TABLE EMP (
ENAME VARCHAR(30) NOT NULL,
ESSN CHAR(9),
BDATE DATE,
DNO INTEGER DEFAULT 1,
SUPERSSN CHAR(9),
PRIMARY KEY (ESSN),
FOREIGN KEY (DNO) REFERENCES DEPT
ON DELETE SET DEFAULT ON UPDATE CASCADE,
FOREIGN KEY (SUPERSSN) REFERENCES EMP(SSN)
ON DELETE SET NULL ON UPDATE CASCADE
);
```

DROP TABLE

- Used to remove a relation (base table) and its definition
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example: `DROP TABLE DEPENDENT;`

ALTER TABLE

- Used to add an attribute to one of the base relations
- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the `NOT NULL` constraint is not allowed for such an attribute
- Example: `ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);`
- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the SELECT statement
- This is not the same as the SELECT operation of the relational algebra (we will discuss this later)
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a multi-set (sometimes called a bag) of tuples; it is not a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the `SELECT` statement
- This is not the same as the SELECT operation of the relational algebra (we will discuss this later)
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a multi-set (sometimes called a bag) of tuples; it is not a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the `SELECT` statement
- This is not the same as the SELECT operation of the relational algebra (we will discuss this later)
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a multi-set (sometimes called a bag) of tuples; it is not a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the `SELECT` statement
- This is not the same as the SELECT operation of the relational algebra (we will discuss this later)
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a multi-set (sometimes called a bag) of tuples; it is not a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the `SELECT` statement
- This is not the same as the SELECT operation of the relational algebra (we will discuss this later)
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a multi-set (sometimes called a bag) of tuples; it is not a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the SELECT statement
- This is not the same as the SELECT operation of the relational algebra (we will discuss this later)
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a multi-set (sometimes called a bag) of tuples; it is not a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

  - `<attribute list>` is a list of attribute names whose values are to be retrieved by the query
  - `<table list>` is a list of the relation names required to process the query
  - `<condition>` is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

  - `<attribute list>` is a list of attribute names whose values are to be retrieved by the query
  - `<table list>` is a list of the relation names required to process the query
  - `<condition>` is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

  - `<attribute list>` is a list of attribute names whose values are to be retrieved by the query
  - `<table list>` is a list of the relation names required to process the query
  - `<condition>` is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

  - `<attribute list>` is a list of attribute names whose values are to be retrieved by the query
  - `<table list>` is a list of the relation names required to process the query
  - `<condition>` is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

  - `<attribute list>` is a list of attribute names whose values are to be retrieved by the query
  - `<table list>` is a list of the relation names required to process the query
  - `<condition>` is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL

- Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

  - `<attribute list>` is a list of attribute names whose values are to be retrieved by the query
  - `<table list>` is a list of the relation names required to process the query
  - `<condition>` is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Simple SQL Queries

- Basic SQL queries correspond to using the SELECT, PROJECT, and JOIN operations of the relational algebra
- All subsequent examples use the COMPANY database
- Example of a simple query on one relation

Simple SQL Queries

- Query 0: Retrieve the birth date 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'
```

- SELECT-clause specifies the projection attributes and the WHERE- clause specifies the selection condition
- However, the result of the query may contain duplicate tuples

Simple SQL Queries

- Query 0: Retrieve the birth date 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'
```

- SELECT-clause specifies the projection attributes and the WHERE- clause specifies the selection condition
- However, the result of the query may contain duplicate tuples

Simple SQL Queries

- Query 0: Retrieve the birth date 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'
```

- SELECT-clause specifies the projection attributes and the WHERE- clause specifies the selection condition
- However, the result of the query may contain duplicate tuples

Simple SQL Queries

- Query 0: Retrieve the birth date 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'
```

- SELECT-clause specifies the projection attributes and the WHERE- clause specifies the selection condition
- However, the result of the query may contain duplicate tuples

Simple SQL Queries

- Query 0: Retrieve the birth date 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'
```

- SELECT-clause specifies the projection attributes and the WHERE- clause specifies the selection condition
- However, the result of the query may contain duplicate tuples

Simple SQL Queries

- 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
```

  - (DNAME='Research') is a selection condition
  - (DNUMBER=DNO) is a join condition

Simple SQL Queries

- 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
```

  - (DNAME='Research') is a selection condition
  - (DNUMBER=DNO) is a join condition

Simple SQL Queries

- 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
```

  - (DNAME='Research') is a selection condition
  - (DNUMBER=DNO) is a join condition

Simple SQL Queries

- 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
```

  - (DNAME='Research') is a selection condition
  - (DNUMBER=DNO) is a join condition

Simple SQL Queries

- 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
```

  - (DNAME='Research') is a selection condition
  - (DNUMBER=DNO) is a join condition

Simple SQL Queries

- 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 birth date.

Q2:

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

  - In Q2, there are two join conditions
  - 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

Simple SQL Queries

- 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 birth date.

Q2:

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

  - In Q2, there are two join conditions
  - 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

Simple SQL Queries

- 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 birth date.
  Q2:

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

  - In Q2, there are two join conditions
  - 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

Simple SQL Queries

- 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 birth date.
  Q2:

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

  - In Q2, there are two join conditions
  - 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

Simple SQL Queries

- 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 birth date.
  Q2:

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

  - In Q2, there are two join conditions
  - 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

Simple SQL Queries

- 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 birth date.
  Q2:

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

  - In Q2, there are two join conditions
  - 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

Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in different relations A query that refers to two or more attributes with the same name must qualify the attribute name with the relation name by prefixing the relation name to the attribute name
- Example:
  `EMPLOYEE.NAME, DEPARTMENT.NAME`

Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in different relations A query that refers to two or more attributes with the same name must qualify the attribute name with the relation name by prefixing the relation name to the attribute name
- Example:

```
EMPLOYEE.NAME, DEPARTMENT.NAME
```

Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in different relations A query that refers to two or more attributes with the same name must qualify the attribute name with the relation name by prefixing the relation name to the attribute name
- Example:
  `EMPLOYEE.NAME, DEPARTMENT.NAME`

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

  Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

ALIASES

- Some queries need to refer to the same relation twice
- In this case, aliases are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  Q8:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S (FROM EMPLOYEE E S)
WHERE E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation
  - We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors

UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:  SELECT SSN
FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:  SELECT SSN
FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:  SELECT SSN
FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:  SELECT SSN
FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:  SELECT SSN
FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:  SELECT SSN
FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

UNSPECIFIED WHERE-clause

- Example: Q10:

```
SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT
```

- It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

UNSPECIFIED WHERE-clause

- Example: Q10:

```
SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT
```

- It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

UNSPECIFIED WHERE-clause

- Example: Q10:
```
SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT
```
- It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

USE OF *

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

Examples:

Q1C:

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

Q1D:

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

USE OF *

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

Examples:

Q1C:

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

Q1D:

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

USE OF *

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

```
Q1C:
SELECT *
FROM EMPLOYEE
WHERE DNO=5
Q1D:
SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND DNO=DNUMBER
```

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes
  Examples:
  Q1C:

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

  Q1D:

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

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes
  Examples:
  Q1C:

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

  Q1D:

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

USE OF DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:  SELECT SALARY
FROM EMPLOYEE
Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE
```

USE OF DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:  SELECT SALARY
FROM EMPLOYEE
Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE
```

USE OF DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:  SELECT SALARY
FROM EMPLOYEE
Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE
```

USE OF DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:  SELECT SALARY
FROM EMPLOYEE
Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE
```

USE OF DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:  SELECT SALARY
FROM EMPLOYEE
Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE
```

USE OF DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:  SELECT SALARY
FROM EMPLOYEE
Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE
```

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in some versions of SQL there are set difference (EXCEPT/MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result
- The set operations apply only to union compatible relations ; the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in some versions of SQL there are set difference (EXCEPT/MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result
- The set operations apply only to union compatible relations ; the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in some versions of SQL there are set difference (EXCEPT/MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result
- The set operations apply only to union compatible relations ; the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in some versions of SQL there are set difference (EXCEPT/MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result
- The set operations apply only to union compatible relations ; the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in some versions of SQL there are set difference (EXCEPT/MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result
- The set operations apply only to union compatible relations ; the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS

- Query 4: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

```
Q4: (SELECT DISTINCT PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')
UNION
(SELECT DISTINCT PNUMBER
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith')
```

SET OPERATIONS

- Query 4: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

```
Q4: (SELECT DISTINCT PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')
UNION
(SELECT DISTINCT PNUMBER
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith')
```

SET OPERATIONS

- Query 4: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

Q4:
```
(SELECT DISTINCT PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')
UNION
(SELECT DISTINCT PNUMBER
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith')
```