Australian National University

SQL – Part 1

SQL and Data Definition Language

What is SQL?

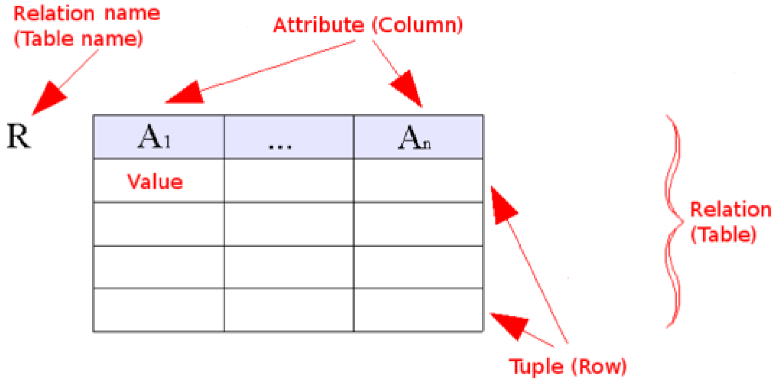
- **SQL** stands for **S**tructured **Q**uery **L**anguage
- SQL was initially developed at IBM (SEQUEL $\rightarrow$ SQL), as one of the first commercial languages for the relational data model.
  - 1986 – SQL was standardised by ANSI and ISO ($\rightsquigarrow$ SQL-86).
  - 1989 – SQL was revised ($\rightsquigarrow$ SQL-89).
  - 1992 – SQL was strengthened and much expanded ($\rightsquigarrow$ SQL-92).
  - 1999 – SQL was expanded and divided into a core specification plus optional specialised packages ($\rightsquigarrow$ SQL:1999).
  - 2003 – SQL was further expanded, e.g., XML support ($\rightsquigarrow$ SQL:2003).
  - 2011 — SQL was further expanded, e.g., improved support for temporal databases ($\rightsquigarrow$ SQL:2011).

What is SQL?

- SQL provides an interface to relational database systems, including:
  - Data Definition Language (DDL);
  - Data Manipulation Language (DML);
  - Data Control Language (DCL);
  - Transaction Control Language (TCL).

Relational Data Model and SQL

- Unlike the relational data model that is based on **sets**, SQL is based on **multisets**. It means that SQL allows a relation to have duplicate tuples.



Data Definition Language

StudentID	Name	CourseNo	Semester

Data Definition Language – Create Table

- The `CREATE TABLE` statement is used to create a new relation schema by specifying its name, its attributes and, *optionally*, its constraints.

```
CREATE TABLE table_name
       (attribute_name data_type [attribute constraints],
        ...,
        attribute_name data_type [attribute constraints],
       [table constraints]);
```

- For each attribute in a relation, we specify its name, its type and, *optionally,* a constraint specific to the attribute (i.e., attribute constraint).

```
attribute_name data_type [attribute_constraint]
```

Create Table – Example

```
CREATE TABLE STUDENT
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100));
```

StudentID	Name	DoB	Email

```
CREATE TABLE COURSE
       (No VARCHAR(20),
        Cname VARCHAR(50),
        Unit SMALLINT);
```

No	Cname	Unit

```
CREATE TABLE ENROL
       (StudentID INT,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50)),
```

StudentID	CourseNo	Semester	Status

Australian National University

Attribute Data Types

- **Numeric types**:
  - `INT` and `SMALLINT` provide domains of integer numbers of various sizes.
  - `FLOAT` or `REAL`, and `DOUBLE PRECISION` provide floating point numbers of various precision.
  - `NUMERIC(i,j)` or `DECIMAL(i,j)` provide fixed point numbers with parameters *precision i* and *scale j*:
    - **precision** for the total number of digits;
    - **scale** for the number of digits following the decimal point.
- **String types**:
  - `CHAR(n)` allows character strings of fixed length, where *n* is the number of characters.
  - `VARCHAR(n)` allows character strings of varying length, where *n* is the maximum number of characters.
  - `BIT(n)` allows bit strings of fixed length, where *n* is the number of bits.
  - `BIT VARYING(n)` allows bit strings of varying length, where *n* is the maximum number of bits.

Attribute Data Types

- **Date and time types**:
  - DATE provides date values (year, month, day).
  - TIME provides time values (hour, minute, second).
  - TIMESTAMP includes the DATE and TIME fields, plus a minimum of six positions for seconds and an optional WITH TIME ZONE qualifier.
  - INTERVAL specifies a relative value that can be used to increment or decrement a value of a date, time or timestamp.
- **Boolean type**: has the values of TRUE or FALSE.
- The CREATE DOMAIN statement is used to create a domain that is essentially a specific data type.

```
CREATE DOMAIN domain_name AS data_type
       [default expression][constraint,...,constraint];
```

Example: CREATE DOMAIN ssn_type AS CHAR(9);

Attribute Data Types – Example

```
CREATE TABLE Student
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100));
```

StudentID	Name	DoB	Email

```
CREATE TABLE Course
       (No VARCHAR(20),
        Cname VARCHAR(50),
        Unit SMALLINT);
```

No	Cname	Unit

```
CREATE TABLE Enrol
       (StudentID INT,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50));
```

StudentID	CourseNo	Semester	Status

Australian National University

Attribute Constraints

- The following constraints can be specified in SQL.

NOT NULL:	specify that NULL is not allowed for an attribute.
DEFAULT:	set a default value for an attribute.
CHECK:	limit the values taken from the domain of an attribute.
UNIQUE:	ensure that uniqueness of the values for an attribute or a set of attribute in a table.
PRIMARY KEY:	uniquely identify each tuple in a table.
FOREIGN KEY:	enforce referential integrity between two tables.
INDEX:	provides accelerated access to the rows of table.

Attribute Constraints – Not Null, Default and Check

```
CREATE TABLE COURSE
       (No VARCHAR(20) PRIMARY KEY,
        Cname VARCHAR(50) NOT NULL,
        Unit SMALLINT NOT NULL Default 6);

CREATE TABLE ENROL
       (StudentID INT NOT NULL CHECK (StudentID>0),
        CourseNo VARCHAR(20) NOT NULL,
        Semester VARCHAR(50) NOT NULL,
        Status VARCHAR(50),
        ...);
```

- If we don't want to have missing and unknown data, we can specify NOT NULL for attributes to forbid NULL values.
- Unit of any new tuple in COURSE is set to 6 if no explicit value is provided.
- CHECK() for StudentID excludes the student IDs such as 0 and -37.

Attribute Constraints – Unique and Primary Key

```
CREATE TABLE COURSE
       (No VARCHAR(20) PRIMARY KEY,
        Cname VARCHAR(50) UNIQUE,
        Unit SMALLINT NOT NULL Default 6);

CREATE TABLE ENROL
       (StudentID INT NOT NULL CHECK (StudentID>0),
        CourseNo VARCHAR(20) NOT NULL,
        Semester VARCHAR(50) NOT NULL,
        Status VARCHAR(50),
        PRIMARY KEY(StudentID, CourseNo, Semester),
        ...);
```

- If a primary key contains only one attribute, PRIMARY KEY can be defined as an attribute constraint (e.g., in COURSE); otherwise it is defined as a table constraint (e.g., in ENROL).
- PRIMARY KEY specifies a key while UNIQUE specifies additional keys.

Attribute Constraints – Foreign Key

```
CREATE TABLE STUDENT
        ( StudentID  INT PRIMARY KEY,
         Name VARCHAR(50),
         DoB Date,
         Email VARCHAR(100));

CREATE TABLE COURSE
        ( No  VARCHAR(20) PRIMARY KEY,
         Cname VARCHAR(50),
         Unit SMALLINT);

CREATE TABLE ENROL
        ( StudentID  INT,
         CourseNo  VARCHAR(20),
         Semester VARCHAR(50),
         Status VARCHAR(50));
```

- Every StudentID appearing in ENROL must exist in STUDENT.
- Every CourseNo appearing in ENROL must exist in COURSE.

Attribute Constraints – Foreign Key

```
CREATE TABLE STUDENT
        ( StudentID  INT PRIMARY KEY,
         Name VARCHAR(50),
         DoB Date,
         Email VARCHAR(100));

CREATE TABLE COURSE
        ( No  VARCHAR(20) PRIMARY KEY,
         Cname VARCHAR(50),
         Unit SMALLINT);

CREATE TABLE ENROL
        ( StudentID  INT,
         CourseNo  VARCHAR(20),
         Semester VARCHAR(50),
         Status VARCHAR(50),
         FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID),
         FOREIGN KEY(CourseNo) REFERENCES COURSE(No));
```

- StudentID in ENROL references StudentID in STUDENT.
- CourseNo in ENROL references No in COURSE.

Attribute Constraints – Foreign Key

```
CREATE TABLE ENROL
       ( StudentID INT,
         CourseNo VARCHAR(20),
         Semester VARCHAR(50),
         Status VARCHAR(50),
         FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID),
         FOREIGN KEY(CourseNo) REFERENCES COURSE(No));

CREATE TABLE STUDENT
       ( StudentID INT PRIMARY KEY,
         Name VARCHAR(50),
         DoB Date,
         Email VARCHAR(100));

CREATE TABLE COURSE
       ( No VARCHAR(20) PRIMARY KEY,
         Cname VARCHAR(50),
         Unit SMALLINT);
```

- Can we define ENROL before STUDENT and COURSE?

  Answer: No. ENROL has the foreign keys that reference STUDENT and COURSE.

Australian National University

Attribute Constraints – Index

- Indexes are used for fast retrieval based on columns other than the primary key.

```
   CREATE TABLE Customer
          (CustomerID INT NOT NULL,
           Name VARCHAR(50) NOT NULL,
           DOB DATE NOT NULL,
           Address VARCHAR(80),
           Phone INT CHECK (Phone>0),
           PRIMARY KEY(CustomerID));

 CREATE INDEX index1 ON  Customer (Name, DOB);

 CREATE UNIQUE INDEX index2 ON Customer (Phone);
```

Data Definition Language – Alter and Drop Table

- The `ALTER TABLE` statement is used to modify an existing relation schema, including:
  - changing the name of a table;
  - adding or dropping an attribute;
  - changing the definition of an attribute;
  - adding or dropping table constraints.

- The `DROP TABLE` statement is used to remove an existing relation schema from a database schema.

Data Definition Language – Alter and Drop Table

- Add a NOT NULL constraint:

```
ALTER TABLE Customer ALTER COLUMN Address SET NOT NULL;
```

- Add a UNIQUE constraint:

```
ALTER TABLE Customer ADD UNIQUE(Phone);
```

- Add a check() constraint:

```
ALTER TABLE Customer
ADD CONSTRAINT positive_id CHECK (CustomerID > 0);
```

- Add a Foreign Key constraint:

```
ALTER TABLE Enrol
ADD FOREIGN KEY(StudentID) REFERENCES Student(StudentID);
```

Data Definition Language – Alter and Drop Table

- Add an attribute EMAIL into the table CUSTOMER:

```
ALTER TABLE Customer ADD Email VARCHAR(100);
```

- Drop the attribute EMAIL in the table CUSTOMER:

```
ALTER TABLE Customer DROP COLUMN Email;
```

- Drop the table ENROL:

```
DROP TABLE Enrol;
```

- Drop the table CUSTOMER (if exists):

```
DROP TABLE IF EXISTS Customer;
```