COSC 304
Introduction to Database Systems

SQL DDL

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

SQL Overview

Structured Query Language or SQL is the standard query language for relational databases.

- It first became an official standard in 1986 as defined by the American National Standards Institute (ANSI).
- All major database vendors conform to the SQL standard with minor variations in syntax (different *dialects*).
- SQL consists of both a Data Definition Language (DDL) and a Data Manipulation Language (DML).

SQL is a ***declarative language*** (non-procedural). A SQL query specifies *what* to retrieve but not *how* to retrieve it.

- Basic SQL is not a complete programming language as it does not have control or iteration commands.
  - ⇨ Procedural extensions: PL/SQL (Oracle), T-SQL (SQL Server)

SQL History

- 1970 - Codd invents relational model and relational algebra
- 1974 - D. Chamberlin (also at IBM) defined Structured English Query Language (SEQUEL)
- 1976 - SEQUEL/2 defined and renamed SQL for legal reasons.
  - Origin of pronunciation 'See-Quel' but official pronunciation is 'S-Q-L'.
- Late 1970s - System R, Oracle, INGRES implement variations of SQL-like query languages.
- 1982 - standardization effort on SQL begins
- 1986 - became ANSI official standard
- 1987 - became ISO standard
- 1992 - SQL2 (SQL92) revision
- 1999 - SQL3 (supports recursion, object-relational)
- Updates: SQL:2003, SQL:2006, SQL:2008, SQL:2011,SQL:2016

SQL Basic Rules

Some basic rules for SQL statements:

- 1) There is a set of *reserved words* that cannot be used as names for database objects. (e.g. `SELECT`, `FROM`, `WHERE`)
- 2) SQL is *case-insensitive*.
  - Only exception is string constants. 'FRED' not the same as 'fred'.
- 3) SQL is *free-format* and white-space is ignored.
- 4) The semi-colon is often used as a statement terminator, although that is not always required.
- 5) Date and time constants have defined format:
  - Dates: 'YYYY-MM-DD' e.g. '1975-05-17'
  - Times: ‘hh:mm:ss[.f] ' e.g. '15:00:00'
  - Timestamp: ‘YYYY-MM-DD hh:mm:ss[.f] ' e.g. ‘1975-05-17 15:00:00'
- 6) Two single quotes '' are used to represent a single quote character in a character constant. e.g. 'Master''s'.

SQL DDL Overview

SQL contains a data definition language (DDL) that allows users to:

- add, modify, and drop tables
- create views
- define and enforce integrity constraints
- enforce security restrictions

SQL Identifiers

Identifiers are used to identify objects in the database such as tables, views, and columns.

- The identifier is the name of the database object.

An SQL identifier (name) must follow these rules:

- only contain upper or lower case characters, digits, and underscore ("_") character
- be no longer than 128 characters
  - DB vendors may impose stricter limits than this.
- must start with a letter (or underscore)
- cannot contain spaces
- Note: Quoted or ***delimited identifiers*** enclosed in double quotes allow support for spaces and other characters. E.g. `"select"`

Database Identifier Question

Question: Select ***one*** valid identifier.

A) 23test

B) 'fred'

C) test_!

D) field_

E) from

Delimited Database Identifiers

Question: **True or False:** `"from"` can be used as a valid identifier according to the SQL standard.

A) True

B) False

SQL Data Types

In the relational model, each attribute has an associated *domain* of values.

In SQL, each column (attribute) has a ***data type*** that limits the values that it may store. The standard SQL data types are similar to their programming language equivalents.

The database will perform (implicit) data type conversion when necessary.

Explicit data type conversion using functions such as `CAST` and `CONVERT`.

SQL Data Types (2)

Data Type	Description
BOOLEAN	TRUE or FALSE
CHAR	Fixed length string (padded with blanks) e.g. CHAR(10)
VARCHAR	Variable length string e.g. VARCHAR(50)
BIT	Bit string e.g. BIT(4) can store '0101'
NUMERIC or DECIMAL	Exact numeric data type e.g. NUMERIC(7,2) has a precision (max. digits) of 7 and scale of 2 (# of decimals) e.g. 12345.67
INTEGER	Integer data only
SMALLINT	Smaller space than INTEGER
FLOAT or REAL	Approximate numeric data types. Precision dependent on implementation.
DOUBLE PRECISION	
DATE	Stores YEAR, MONTH, DAY
TIME	Stores HOUR, MINUTE, SECOND
TIMESTAMP	Stores date and time data.
INTERVAL	Time interval.
CHARACTER LARGE OBJECT	Stores a character array (e.g. for a document)
BINARY LARGE OBJECT	Stores a binary array (e.g. for a picture, movie)

SQL User Defined Data Types

The **CREATE DOMAIN** command allows you to define your own types that are subsets of built-in types:

```
CREATE DOMAIN domainName AS dataType
    [DEFAULT defaultValue]
    [CHECK (condition)]
```

Example: Create user-defined domain for `Emp.title`:

```
CREATE DOMAIN titleType AS CHAR(2)
   DEFAULT 'EE'
   CHECK (VALUE IN (NULL,'EE','SA','PR','ME'));
```

SQL User Defined Data Types (2)

The **CHECK** clause can use a nested select statement to retrieve values from the database:

```
CREATE DOMAIN mgrType AS CHAR(5)
    DEFAULT NULL
    CHECK (VALUE IN (SELECT eno FROM emp
                     WHERE title = 'ME' OR title = 'SA'));
```

Domains can be removed from the system using **DROP**:

```
DROP DOMAIN domainName [RESTRICT | CASCADE]
```

- RESTRICT - if domain is currently used, drop fails.
- CASCADE - if domain is current used, domain dropped and fields using domain defaulted to base type.

- Example:

```
DROP DOMAIN mgrType;
```

SQL CREATE TABLE

The **CREATE TABLE** command is used to create a table in the database. A table consists of a table name, a set of fields with their names and data types, and specified constraints.

The general form is:

```
CREATE TABLE tableName (
    attr1Name attr1Type  [attr1_constraints],
    attr2Name attr2Type  [attr2_constraints],
    ...
    attrMName attrMType  [attrM_constraints],
    [primary and foreign key constraints]
);
```

SQL CREATE TABLE Example

The CREATE TABLE command for the Emp relation:

```
CREATE TABLE Emp (
    eno       CHAR(5),
    ename     VARCHAR(30) NOT NULL,
    bdate     DATE,
    title     CHAR(2),
    salary    DECIMAL(9,2),
    supereno  CHAR(5),
    dno       CHAR(5),
    PRIMARY KEY (eno),
    FOREIGN KEY (dno) REFERENCES Dept(dno)
          ON DELETE SET NULL ON UPDATE CASCADE
);
```

SQL Constraints

Constraints are specified in `CREATE` and `ALTER TABLE` statements.

Types of constraints:

- 1) **Required data** - To specify that a column must always have a data value (cannot be NULL) specify `NOT NULL` after the column definition.
  - e.g. `eno CHAR(5) NOT NULL`

- 2) **Domain constraints** - Used to verify that the value of a column is in a given domain using `CHECK`.
  - e.g.
```
title CHAR(2) CHECK (title IN
(NULL,'EE','SA','PR','ME'));
```
  - Forces the title to be either NULL or one of 4 defined values.
  - Can also be performed using user-defined types (domains).

SQL Constraints (2)

- 3) **Tuple constraints** - `CHECK` can also be used on an entire tuple instead of a single attribute:

```
CREATE TABLE student(
   num       CHAR(10)    NOT NULL,
   honors    CHAR(1),
   gpa       DECIMAL(3,2),
   CHECK ( (honors = 'Y' AND gpa > 3.50)
             OR honors = 'N')
);
```

- Note that `CHECK` clause can contain subqueries, but the `CHECK` is only performed when the relation itself is modified.
  - ⇨ Does not perform check when relation involved in subquery is modified.

SQL Constraints - Entity Integrity

Entity Integrity constraint - The primary key of a table must contain a unique, non-null value for each row. The primary key is specified using the `PRIMARY KEY` clause.

- e.g. `PRIMARY KEY (eno)` (for Emp relation)
- e.g. `PRIMARY KEY (eno,pno)` (for WorksOn relation)
- It is also possible to use `PRIMARY KEY` right after defining the attribute in the `CREATE TABLE` statement.

There can only be one primary key per relation, other candidate keys can be specified using `UNIQUE`:

- e.g. `UNIQUE (ename)`

SQL Constraints - Referential Integrity

Referential integrity constraint - Defines a foreign key that references the primary key of another table.

- If a foreign key contains a value that is not `NULL`, that value must be present in some tuple in the relation containing the referenced primary key.

Example: `WorksOn` contains two foreign keys:

- `WorksOn.eno` references `Emp.eno`
- `WorksOn.pno` references `Proj.pno`

Specify foreign keys using `FOREIGN KEY` syntax:

```
FOREIGN KEY (eno) REFERENCES Emp(eno)
```

SQL Referential Integrity Example

The `CREATE TABLE` command for the `WorksOn` relation:

```
CREATE TABLE WorksOn (
    eno   CHAR(5),
    pno   CHAR(5),
    resp  VARCHAR(20),
    hours SMALLINT,
    PRIMARY KEY (eno,pno),
    FOREIGN KEY (eno) REFERENCES Emp(eno),
    FOREIGN KEY (pno) REFERENCES Proj(pno)
);
```

SQL Referential Integrity and Updates

When you try to `INSERT` or `UPDATE` a **row in a relation containing a foreign key** (e.g. `WorksOn`) that operation is rejected if it violates referential integrity.

When you `UPDATE` or `DELETE` a **row in the primary key relation** (e.g. `Emp` or `Proj`), you have the option on what happens to the values in the foreign key relation (`WorksOn`):

- 1) `CASCADE` - Delete (update) values in foreign key relation when primary key relation has rows deleted (updated).
- 2) `SET NULL` - Set foreign key fields to NULL when corresponding primary key relation row is deleted.
- 3) `SET DEFAULT` - Set foreign key values to their default value (if defined).
- 4) `NO ACTION` - Reject the request on the parent table.

SQL Referential Integrity Example (2)

```
CREATE TABLE WorksOn (
    eno    CHAR(5),
    pno    CHAR(5),
    resp   VARCHAR(20),
    hours  SMALLINT,
    PRIMARY KEY (eno,pno),
    FOREIGN KEY (eno) REFERENCES Emp(eno)
                              ON DELETE NO ACTION
                              ON UPDATE CASCADE,
    FOREIGN KEY (pno) REFERENCES Proj(pno)
                              ON DELETE NO ACTION
                              ON UPDATE CASCADE
);
```

Enforcing Referential Integrity Question

Question: Select ***one*** true statement.

A) `SET NULL` can be used for the `WorksOn.eno` foreign key.

B) `ON UPDATE CASCADE` will modify all rows in the primary key table when a value is modified in the foreign key table.

C) `SET DEFAULT` cannot be used for the `WorksOn.eno` foreign key.

D) If a primary key row is deleted and it is referenced by a foreign key row, `NO ACTION` will generate an error to the user.

SQL CREATE TABLE Full Syntax

Full syntax of `CREATE TABLE` statement:

```
CREATE TABLE tableName (
  { attrName attrType  [NOT NULL] [UNIQUE] [PRIMARY KEY]
     [DEFAULT value] [CHECK (condition)] }
   [PRIMARY KEY (colList)]
   {[FOREIGN KEY (colList) REFERENCES tbl [(colList)],
     [ON UPDATE action]
     [ON DELETE action] }
   {[CHECK (condition)] }
);
```

used when matching to attributes that are not the primary key

Creating the Example Database

```
CREATE DOMAIN T_eno AS CHAR(5);
CREATE DOMAIN T_pno AS CHAR(5);
CREATE DOMAIN T_dno AS CHAR(5);

CREATE TABLE Emp(
   eno       T_eno,
   ename     VARCHAR(30) NOT NULL,
   bdate     DATE,
   title     CHAR(2),
   salary    DECIMAL(9,2),
   supereno  T_eno,
   dno       T_dno
   PRIMARY KEY (eno),
   FOREIGN KEY (dno) REFERENCES Dept(dno)
         ON DELETE SET NULL ON UPDATE CASCADE
);
```

Creating the Example Database (2)

```
CREATE TABLE WorksOn (
   eno   T_eno,
   pno   T_pno,
   resp  VARCHAR(20),
   hours SMALLINT,
   PRIMARY KEY (eno,pno),
   FOREIGN KEY (eno) REFERENCES Emp(eno)
               ON DELETE NO ACTION ON UPDATE CASCADE,
   FOREIGN KEY (pno) REFERENCES Proj(pno)
               ON DELETE NO ACTION ON UPDATE CASCADE
);
```

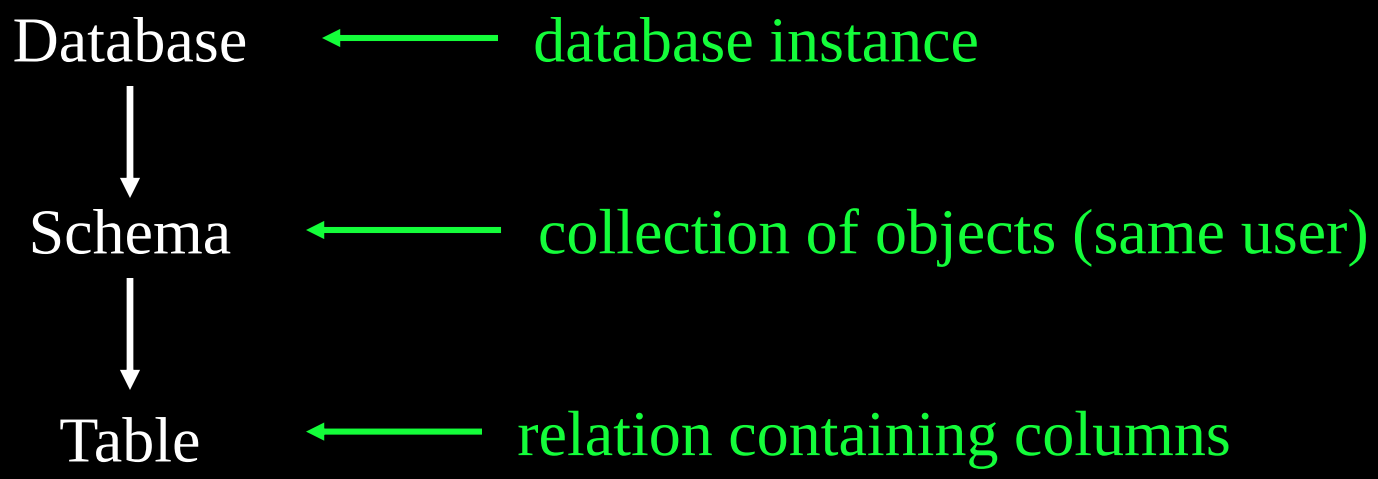
Question:

Write `CREATE TABLE` statements to build the `Proj` and `Dept` relations:

- `Dept(dno, dname, mgreno)`
- `Proj(pno, pname, budget, dno)`

Defining a Database

There is typically a hierarchy of database objects that you can create, alter, and destroy.



SQL does not standardize how to create a database. A database often contains one or more *catalogs*, each of which contains a set of *schemas*.

- To make things more complicated, many DBMSs do not implement everything and rename things. e.g. A database *IS* a schema for MySQL (there is no `CREATE SCHEMA` command).

Creating Schemas

A ***schema*** is a collection of database objects (tables, views, domains, etc.) usually associated with a single user.

Creating a schema: (User `Joe` creates the schema)

```
CREATE SCHEMA employeeSchema AUTHORIZATION Joe;
```

Dropping a schema:

```
DROP SCHEMA employeeSchema;
```

ALTER TABLE

The **ALTER TABLE** command can be used to change an existing table. This is useful when the table already contains data and you want to add or remove a column or constraint.

- DB vendors may support only parts of `ALTER TABLE` or may allow additional changes including changing the data type of a column.

General form:

```
ALTER TABLE tableName
    [ADD [COLUMN] colName dataType [NOT NULL] [UNIQUE]
       [DEFAULT value] [CHECK (condition)] ]
    [DROP [COLUMN] colName [RESTRICT | CASCADE]
    [ADD [CONSTRAINT [constraintName]] constraintDef]
    [DROP CONSTRAINT constraintName [RESTRICT | CASCADE]]
    [ALTER [COLUMN] SET DEFAULT defValue]
    [ALTER [COLUMN] DROP DEFAULT]
```

ALTER TABLE Examples

Add column `location` to `Dept` relation:

```
ALTER TABLE dept
  ADD location VARCHAR(50);
```

Add field `SSN` to `Emp` relation:

```
ALTER TABLE Emp
  ADD SSN CHAR(10);
```

Indicate that `SSN` is `UNIQUE` in `Emp`:

```
ALTER TABLE Emp
  ADD CONSTRAINT ssnConst UNIQUE(SSN);
```

DROP TABLE

The command **DROP TABLE** is used to delete the table definition and all data from the database:

```
DROP TABLE tableName [RESTRICT | CASCADE];
```

Example:

```
DROP TABLE Emp;
```

Question: What would be the effect of the command:

```
DROP TABLE Emp CASCADE;
```

Indexes

Indexes are used to speed up access to the rows of the tables based on the values of certain attributes.

- An index will often significantly improve the performance of a query, however they represent an overhead as they must be updated every time the table is updated.

The general syntax for creating and dropping indexes is:

```
CREATE [UNIQUE] INDEX indexName
   ON  tableName (colName [ASC|DESC] [,...])

DROP INDEX indexName;
```

- `UNIQUE` means that each value in the index is unique.
- `ASC`/`DESC` specifies the sorted order of index.

Indexes Example

Creating an index on `eno` and `pno` in `WorksOn` is useful as it will speed up joins with the `Emp` and `Proj` tables respectively.

- Index is not `UNIQUE` as `eno` (`pno`) can occur many times in `WorksOn`.

```
CREATE INDEX idxEno ON WorksOn (eno);

CREATE INDEX idxPno ON WorksOn (pno);
```

Most DBMSs will put an index on the primary key, but if they did not, this is what it would like for `WorksOn`:

```
CREATE UNIQUE INDEX idxPK ON WorksOn (eno,pno);
```

Database Updates

Database updates such as inserting rows, deleting rows, and updating rows are performed using their own statements.

Insert is performed using the `INSERT` command:

```
INSERT INTO tableName [(column list)]
VALUES (data value list)
```

Examples:

```
INSERT INTO emp VALUES ('E9','S. Smith',DATE '1975-03-05',
                        'SA',60000,'E8','D1');

INSERT INTO proj (pno, pname) VALUES ('P6','Programming');
```

Note: If column list is omitted, values must be specified in order they were created in the table. If any columns are omitted from the list, they are set to NULL.

INSERT Multiple Rows

INSERT statement extended by many databases to take multiple rows:

```
INSERT INTO tableName [(column list)]
VALUES (data value list) [, (values) ]+
```

Example:

```
INSERT INTO Emp (eno, ename) VALUES
   ('E10', 'Fred'), ('E11', 'Jane'), ('E12', 'Joe')
```

INSERT rows from SELECT

Insert multiple rows that are the result of a `SELECT` statement:

```
INSERT INTO tableName [(column list)]
  SELECT ...
```

Example: Add rows to a temporary table that contains only employees with `title ='EE'`.

```
INSERT INTO tmpTable
   SELECT eno, ename
   FROM emp
   WHERE title = 'EE'
```

UPDATE Statement

Updating existing rows is performed using UPDATE statement:

```
UPDATE tableName
SET col1 = val1 [,col2=val2...]
[WHERE condition]
```

Examples:

- 1) Increase all employee salaries by 10%.

```
UPDATE emp SET salary = salary*1.10;
```

- 2) Increase salaries of employees in department 'D1' by 8%.

```
UPDATE emp SET salary = salary*1.08
WHERE dno = 'D1';
```

DELETE Statement

Rows are deleted using the DELETE statement:

```
DELETE FROM tableName
[WHERE condition]
```

Examples:

- 1) Fire everyone in the company.

```
DELETE FROM workson;
DELETE FROM emp;
```

- 2) Fire everyone making over $35,000.

```
DELETE FROM emp
WHERE salary > 35000;
```

Practice Questions

Relational database schema:

```
emp (eno, ename, bdate, title, salary,
supereno, dno)
proj (pno, pname, budget, dno)
dept (dno, dname, mgreno)
workson (eno, pno, resp, hours)
```

1) Insert a department with number `'D5'`, name `'Useless'`, and no manager.
2) Insert a `workson` record with `eno='E1'` and `pno='P3'`.
3) Delete all records from `emp`.
4) Delete only the records in `workson` with more than 20 hours.
5) Update all employees to give them a 20% pay cut.
6) Update the projects for `dno='D3'` to increase their budget by 10%.

Conclusion

SQL contains a data definition language that allows you to `CREATE`, `ALTER`, and `DROP` database objects such as tables, triggers, indexes, schemas, and views.

Constraints are used to preserve the integrity of the database:

- `CHECK` can be used to validate attribute values.
- **Entity Integrity constraint** - The primary key of a table must contain a unique, non-null value for each row.
- **Referential integrity constraint** - Defines a foreign key that references a unique key of another table.

`INSERT`, `DELETE`, and `UPDATE` commands modify the data stored within the database.

Objectives

General:

- Recognize valid and invalid identifiers

Constraints:

- Define own domains using `CREATE DOMAIN`
- List 5 types of constraints and how to enforce them
  - required (not null) data, domain constraints, tuple constraints
  - entity integrity, referential integrity

Creating database objects:

- Describe hierarchy of database objects: database, schema, table
- Write `CREATE TABLE` statement given high-level description.
- List what `ALTER TABLE` can and cannot do.
- Be able to write `INSERT`, `DELETE`, and `UPDATE` commands.
- Create an index on fields of a table.