CS 422 - DBMS

Lesson 4 - Chapters 6, 7

Data Manipulation Language (DML) Statements

- **SELECT –** to query data in the database
- **INSERT –** to insert data into a table
- **UPDATE –** to update data in a table
- **DELETE –** to delete data from a table

Database Updates - INSERT

INSERT INTO TableName [(columnList)]

VALUES (dataValueList)

- *columnList* is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.
- Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.

INSERT

- **dataValueList must match columnList as follows:**
  - number of items in each list must be same;
  - must be direct correspondence in position of items in two lists;
  - data type of each item in dataValueList must be compatible with data type of corresponding column.

Example 6.35 INSERT...VALUES

Insert a new row into Staff table supplying data for all columns.

```
INSERT INTO Staff
VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M',
  '1957-05-25', 8300, 'B003');
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Example 6.36 INSERT using Defaults

Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO Staff (staffNo, fName, lName, position, salary, branchNo)
    VALUES ('SG44', 'Anne', 'Jones', 'Assistant', 8100, 'B003');
```

or

```
INSERT INTO Staff
    VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL, NULL, 8100,
                                                        'B003');
```

INSERT … SELECT

- **Second form of INSERT allows multiple rows to be copied from one or more tables to another:**

```
INSERT INTO TableName [ (columnList) ]
   SELECT …
```

Example 6.37 INSERT... SELECT

Assume there is a table **StaffPropCount** that contains names of staff and number of properties they manage:

StaffPropCount(staffNo, fName, lName, propCnt)

- Populate *StaffPropCount* using *Staff* and *PropertyForRent* tables.

```
INSERT INTO StaffPropCount
  SELECT s.staffNo, fName, lName,
            COUNT(propertyNo) as propCount
  FROM Staff s LEFT JOIN PropertyForRent p
            ON s.staffNo = p.staffNo
  GROUP BY s.staffNo, fName, lName
```

staffNo	fName	lName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

Database Updates - UPDATE

UPDATE TableName
SET columnName1 = dataValue1 [, columnName2 = dataValue2...]
[WHERE searchCondition]

- *TableName* can be name of a base table or an updatable view.
- SET clause specifies names of one or more columns that are to be updated.
- WHERE clause is optional:
  - if omitted, named columns are updated for all rows in table;
  - if specified, only those rows that satisfy *searchCondition* are updated.
  - New *dataValue(s)* must be compatible with data type for corresponding column.

Example 6.38/39 UPDATE All Rows

Give all staff a 3% pay increase.

```
UPDATE Staff
SET salary = salary*1.03;
```

Give all Managers a 5% pay increase.

```
UPDATE Staff
SET salary = salary*1.05
WHERE position = 'Manager';
```

Example 6.40 UPDATE Multiple Columns

Promote David Ford (staffNo='SG14') to Manager and change his salary to $18,000.

```
UPDATE Staff
SET position = 'Manager', salary = 18000
WHERE staffNo = 'SG14';
```

Database Updates - DELETE

DELETE FROM TableName

[WHERE searchCondition]

- *TableName* can be name of a base table or an updatable view.
- *searchCondition* is optional; if omitted, all rows are deleted from table. If *search_condition* is specified, only those rows that satisfy condition are deleted.
- This does not delete table.

Example 6.41/42 DELETE Specific Rows

- **Delete all viewings that relate to property PG4.**

```
DELETE FROM Viewing
    WHERE propertyNo = 'PG4';
```

- **Delete all records from the Viewing table.**

```
DELETE FROM Viewing;
```

Research about TRUNCATE TABLE command and how it is different from DELETE TABLE

Data Definition Language (DDL)

- Data Definition Language (DDL) is the part of SQL that allows one to set up a database's schema, that is, to define attributes, domains for those attributes, define tables, define relationships among those tables and so on.
- DDL is used to create objects, to alter objects and to drop objects.

Start With An Attribute

- One of the smallest units one defines is an attribute.
- An attribute is given a name (identifier) and assigned a type.

SQL Identifier

- A SQL identifier is the name assigned to a table, column, view, etc.
  - The identifier is a string of characters from some set.
  - The string is at most 128 characters long.
  - The standard set of characters consists of capital letters (A,B,..), small letters (a,b,…), digits (0,1,…) and the underscore character (_).
  - An identifier starts with a letter.
  - An identifier contains no spaces (allowed by Access but not by standard SQL).

SQL Data Types

Exact Numeric Data Types

DATA TYPE	FROM	TO
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 -1
numeric	-10^38 +1	10^38 -1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

SQL Data Types

Approximate Numeric Data Types

DATA TYPE	FROM	TO
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38

Date and Time Data Types

DATA TYPE	FROM	TO
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079
date	Stores a date like June 30, 1991	
time	Stores a time of day like 12:30 P.M.	

SQL Data Types

Character, Unicode Strings and Binary Data Types

Data type	Description	Max size
CHAR(N)	Fixed width character string	8,000 characters
VARCHAR(N)	Variable width character string	8,000 characters
VARCHAR(MAX)	Variable width character string	1,073,741,824 characters
NCHAR	Fixed width Unicode string	4,000 characters
NVARCHAR	Variable width Unicode string	4,000 characters
NVARCHAR(MAX)	Variable width Unicode string	536,870,912 characters
BINARY(N)	Fixed width binary string	8,000 bytes
VARBINARY	Variable width binary string	8,000 bytes
VARBINARY(MAX)	Variable width binary string	2GB

CHAR and VARCHAR

- CHAR and VARCHAR are character strings – need to include length while defining an attribute
  - staffNo CHAR(6)
    - When a variable or a table column is assigned with a string that is shorter than its nominal size, it is padded with trailing spaces to fill the specified field length.
  - fName VARCHAR(30)
    - When a character value whose length is less than the nominal size is assigned to the column or variable, SQL Server does not add trailing spaces to it, but records it as is.

CHAR and VARCHAR contd..

- **char** gives better performance
- Use **char** when the sizes of the column data entries are consistent.
  - CHAR(2) for state abbreviation.
  - CHAR(8) for product codes like '004-3228'
- Use **varchar** when the sizes of the column data entries vary considerably.
- Use **varchar(max)** when the sizes of the column data entries vary considerably, and the string length might exceed 8,000 bytes.

Main DDL statements

- SQL DDL allows database objects such as schemas, domains, tables and views to be created and destroyed.

- The main SQL DDL statements are:

CREATE SCHEMA	**DROP SCHEMA**
CREATE/ALTER TABLE	**DROP TABLE**
CREATE VIEW	**DROP VIEW**

Creating a Table (CREATE TABLE)

CREATE TABLE [dbName.][schemaName.]TableName

Define fields with business rules:

{(colName dataType [**NOT NULL**] [**UNIQUE**]
[**DEFAULT** defaultOption]
[**CHECK** searchCondition] [,...]}

Identify primary key ⟹ [**PRIMARY KEY** (listOfColumns),]

Identify candidate keys ⟹ {[**UNIQUE** (listOfColumns)] [,...]}

Identify foreign keys, relationships and actions to maintain referential integrity:

{[**FOREIGN KEY** (listOfFKColumns)
REFERENCES ParentTableName [(listOfCKColumns)],
[**ON UPDATE** referentialAction]
[**ON DELETE** referentialAction]] [,...]}

Business rules ⟹ {[**CHECK** (searchCondition)] [,...] })

CREATE TABLE Example

```
CREATE TABLE PropertyForRent1(
      propertyNo char(4) NOT NULL,
      city varchar(30) NOT NULL,
      pType varchar(10) NOT NULL DEFAULT 'Flat',
      rent float NOT NULL DEFAULT 600,
      staffNo varchar(10),
      branchNo varchar(10) NOT NULL,
   PRIMARY KEY (propertyNo),
   FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON DELETE SET NULL ON UPDATE CASCADE,
   FOREIGN KEY (branchNo) REFERENCES Branch(branchNo) ON DELETE NO ACTION ON UPDATE CASCADE);
```

System rejects any attempt to insert a null

Default value

Primary key will not accept NULL

Specifies FKs along with the referential action

CREATE TABLE Examples

```
CREATE TABLE employees
(   empID INT PRIMARY KEY,
    lastName VARCHAR(50) not null,
    firstName VARCHAR(50) not null,
    salary MONEY
);
```

```
CREATE TABLE employees
(   empID INT,
    lastName VARCHAR(50) not null,
    firstName VARCHAR(50) not null,
    salary MONEY,
    CONSTRAINT emp_pk PRIMARY KEY (empID)
);
```

Same

```
CREATE TABLE employees
(   lastName VARCHAR(50) not null,
    firstName VARCHAR(50) not null,
    salary MONEY,
    PRIMARY KEY (lastName, firstName)
);
```

Composite Key

Identity column

- Auto-increment field
  - Allows a unique number to be generated automatically when a new record is inserted into a table.
  - Often this is the primary key field

Seed

Increment

```
CREATE TABLE Persons
(   personID INT IDENTITY (1,1) PRIMARY KEY,
    lastName VARCHAR(50) NOT NULL,
    firstName VARCHAR(50),
    noOfDependents INT
);
```

Integrity Enhancement Features

- Facilities provided by the SQL standard for integrity control.
- Integrity control consists of constraints that we wish to impose in order to protect the database from becoming inconsistent.
- Five types of integrity constraints:
  - **Required data**
  - **Domain constraints**
  - **Entity integrity**
  - **Referential integrity**
  - **Enterprise constraints**

Integrity Enhancement Feature: Required data

Some columns must contain a valid value, they are not allowed to contain nulls.

position VARCHAR(10) NOT NULL

NULL is the default

- If nothing is said, NULL is implied.

lName VARCHAR(30)

lName VARCHAR(30) NULL

would be the same and say that a staff might not have a last name.

staffNo CHAR(5) NOT NULL

says that a staff must have a non null staffNo.

Integrity Enhancement Feature: Domain Constraints

- Every column has a domain (set of legal values)
- The type of the column puts some restrictions on the values the column is allowed to have.
- Further restrictions can be placed on the data by introducing a **domain constraint**.
- 2 ways to specify domains in the CREATE and ALTER table statements.
  - **CHECK clause** – allows a constraint to be defined on a column or the entire table.
  - **CREATE DOMAIN statement**

Integrity Enhancement Feature: Domain Constraints using CHECK clause

- CHECK(condition) verifies that the value obeys some condition

```
jobPos  VARCHAR(15) CHECK ( jobPos  IN  ('manager','tester'))
```

- allows values of 'manager', 'tester' or NULL for the *jobPos* attribute.

```
wage  DECIMAL(5, 2) CHECK ( wage > 5.25 AND wage < 500.0)
```

```
sex  CHAR  NOT NULL CHECK (sex IN ('M', 'F'))
```

Integrity Enhancement Feature: Domain Constraints using CHECK clause

- To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    Lastname varchar(50) NOT NULL,
    FirstName varchar(50),
    Age int, City varchar(50),
    CONSTRAINT chk_person CHECK (Age >=18 and
                                 City = 'Fairfield'));
```

Integrity Enhancement Feature: Entity Integrity

Primary Key

- Primary key of a table must contain a unique, non-null value for each row (known as **Entity Integrity**).
- ISO standard supports PRIMARY KEY clause in CREATE and ALTER TABLE statements:

```
PRIMARY KEY ( staffNo )

PRIMARY KEY ( clientNo, propertyNo )
```

- Can only have one PRIMARY KEY clause per table.

Integrity Enhancement Feature: Entity Integrity

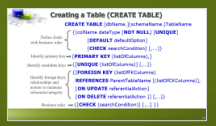
Candidate Keys

- Recall that a candidate key is an attribute or set of attributes having the same features as the primary key (uniqueness).
- To specify that a set of attributes must have this uniqueness property, use the keyword UNIQUE.

PRIMARY KEY (date, time, room)

UNIQUE (date, time, tutee)

UNIQUE (date, time, tutor)



Integrity Enhancement Feature: Referential Integrity

- FK is column or set of columns that links each row in child table containing the FK to the row of the parent table containing the matching CK value.
- Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table or FK should be wholly null.
- ISO standard supports definition of FKs with FOREIGN KEY clause in CREATE and ALTER TABLE:

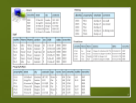
```
FOREIGN KEY (personID)  REFERENCES  RealPerson
FOREIGN KEY (branchNo)  REFERENCES Branch
```

Integrity Enhancement Feature: Referential Integrity contd..

- Any INSERT/UPDATE attempting to create FK value in child table without matching CK value in parent is rejected.
- A foreign key takes its value from another table (from parent in the relationship). What happens if the parent value is updated or deleted? How will referential integrity be maintained?
- Action taken for any update/delete of a CK value in parent table that has some matching rows in child is dependent on the **referential action** specified using the ON UPDATE and ON DELETE sub clauses of the FOREIGN KEY clause.

Four Choices For Maintaining Referential Integrity – ON DELETE

- **CASCADE**: Delete row from parent and delete matching rows in child, and so on in cascading manner.
- **SET NULL**: Delete row from parent and set FK column(s) in child to NULL. This option is valid only if the foreign key columns do not have the NOT NULL qualifier specified.
- **SET DEFAULT**: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.
- **NO ACTION**: Reject delete from parent. This is the default setting if the ON DELETE rule is omitted.



Four Choices For Maintaining Referential Integrity – ON UPDATE

- **CASCADE**: Make same update to child as made to parent.
- **SET NULL**: Set the child value to NULL when the parent is updated. Only valid if FK columns do not have NOT NULL qualifier.
- **SET DEFAULT**: Set the child value to its DEFAULT when the parent is updated. Only valid if DEFAULT specified for FK columns.
- **NO ACTION**: Prevent parent from being updated if children are affected. Default.

ON DELETE / ON UPDATE

- Specify an update rule such that if an owner number is updated in the PrivateOwner table, the corresponding column(s) in the PropertyForRent table are set to the new value:

FOREIGN KEY (ownerNo) **REFERENCES**
PrivateOwner **ON UPDATE CASCADE**

- Specify a deletion rule such that if a staff record is deleted from the Staff table, the values of the corresponding staffNo column in the PropertyForRent table are set to NULL:

FOREIGN KEY (staffNo) **REFERENCES**
Staff **ON DELETE SET NULL**

Integrity Enhancement Feature: General (Enterprise) Constraints

- An **enterprise (general) constraint** (*business rule*) is an additional condition placed on the database that does not fall into one of the previous categories of domain constraint, entity integrity and referential integrity.
- Could use **CHECK** and **UNIQUE** clauses in CREATE and ALTER TABLE.
- Need to use **Triggers** to add a complex constraint.
  - An example would be allowing a staff to handle only 100 properties at any time.

ALTER TABLE

- The ALTER TABLE keyword is used to change the schema (not the instance) of a table.
- For example, one can
  - Add a new column to a table
  - Drop a column from a table
  - Add a new table constraint
  - Drop a table constraint
  - Set a default for a column
  - Drop a default for a column
  - Change column datatype

Basic Format of ALTER TABLE

ALTER TABLE TableName
{**ADD** [**COLUMN**] colName dataType [**NOT NULL**] [**UNIQUE**]
[**DEFAULT** defaultOption] [**CHECK** searchCondition] [,...]}
[**DROP** [**COLUMN**] colName [**RESTRICT|CASCADE**]]
[**ADD** [**CONSTRAINT** [ConstrName]] tblConstrDef]
[**DROP CONSTRAINT** ConstrName [**RESTRICT|CASCADE**]]
[**ALTER [COLUMN] SET DEFAULT** DefaultOption]
[**ALTER [COLUMN] DROP DEFAULT**]

RESTRICT: The DROP operation is rejected if the column is referenced by another DB object. It is the default setting.

CASCADE: The DROP operation proceeds and automatically drops the column from any DB objects it is referenced by.

ALTER TABLE Examples

Add new column to Client table, *preferred number of rooms*.

```
ALTER TABLE Client
        ADD prefNoRooms int;
```

Change Staff table by setting default for salary column to 10000.

```
ALTER TABLE Staff
        ADD CONSTRAINT sal_constraint
                DEFAULT 10000 FOR salary;
```

ALTER/MODIFY Column

- To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name
        ALTER COLUMN column_name datatype;
```

- Modify the datatype of column *salary* to float of Staff table.

```
ALTER TABLE Staff
        ALTER COLUMN salary float;
```

DROP COLUMN Example

- We want to delete the column named "DateOfBirth" in the "Persons" table.

- We use the following SQL statement:

```
ALTER TABLE Persons
  DROP COLUMN DateOfBirth;
```

DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE]

For example,

DROP TABLE PropertyForRent;

- Removes named table and all rows within it.
- With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.
- With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).