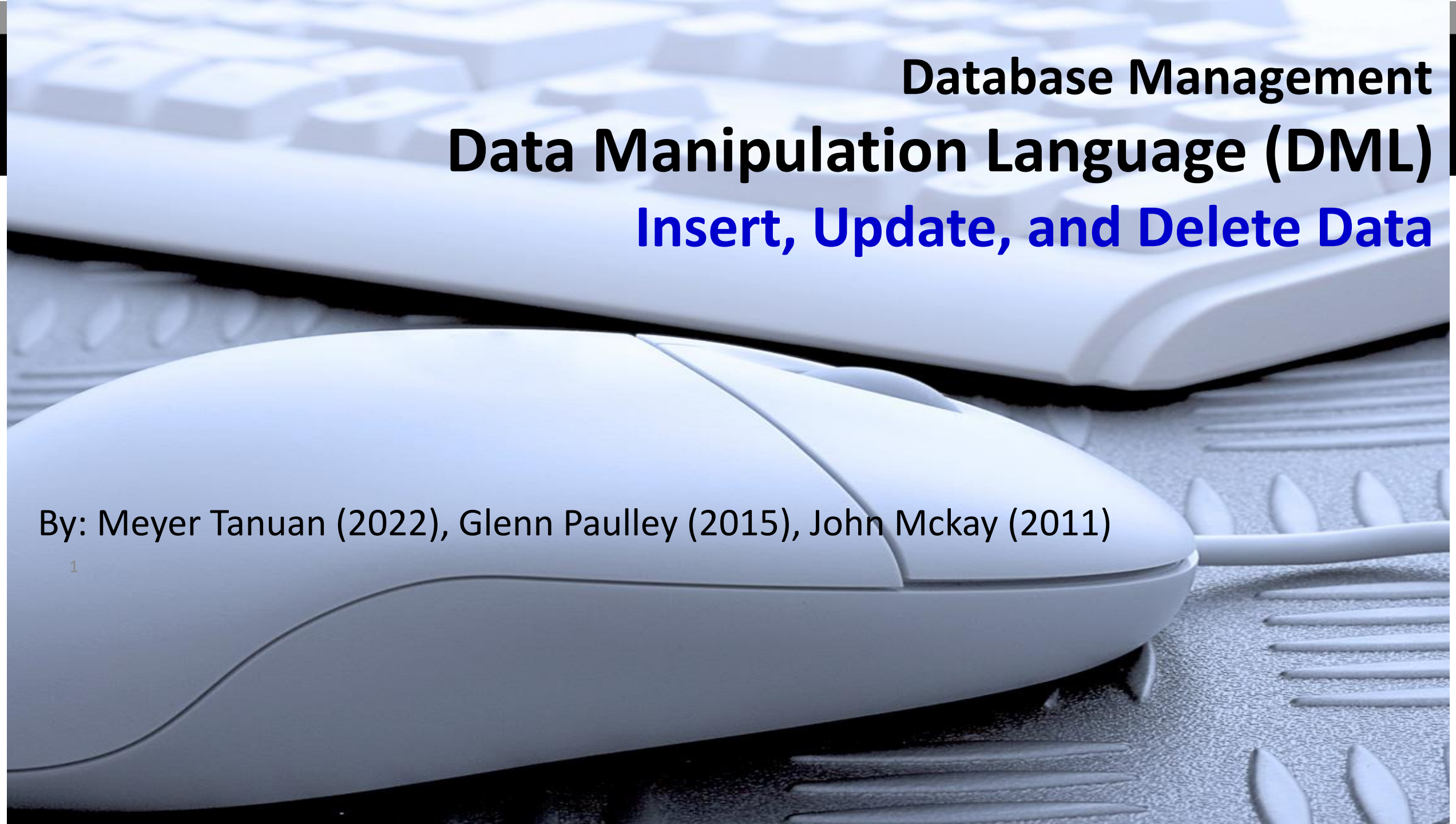

Database Management
Data Manipulation Language (DML)
Insert, Update, and Delete Data

By: Meyer Tanuan (2022), Glenn Paulley (2015), John Mckay (2011)

SQL Data Manipulation Language (DML)

- SELECT, INSERT, UPDATE, DELETE
- For DML class demo, we can duplicate the contents of a table (excluding constraints such as primary key) using the syntax:

```
SELECT *
    INTO Table2
FROM Table
```

- For example:

```
SELECT *
    INTO School2
FROM School;
```

INSERT

- Typical, SQL-standard compliant INSERT statement syntax:

```
INSERT INTO table
    (column1, column2, column3)
VALUES
    (value1, value2, value3)
```

- This syntax is supported by virtually every database system product

INSERT with ALL column values

- The safest way to INSERT a row is to list all columns first, then the values for each:

```
INSERT INTO School
    (code, name, frenchName)
VALUES
    ('LIB', 'Liberal Studies', 'Liberal Studies')
```

INSERT without a COLUMN list

- You can omit the column list if you provide data for **all** columns in their exact order in the table:

```
INSERT INTO School
VALUES
    ('LIB', 'Liberal Studies', 'Liberal Studies')
```

- If you miss a value, you'll get an error –"… supplied values does not match table definition"
- If you get the order wrong, but matching data types, it will succeed incorrectly

INSERT with PARTIAL set of column values

- If you attempt to INSERT only some of the column values, you need to specify the column list for the values being supplied:

```
INSERT INTO Person
   (number, lastName, firstName)
VALUES
   ('A123456', 'TANUAN', 'MEYER')
```

- This adds a row with NULL for the value of all other columns
  - street, city, provinceCode, …
- Can work only if column is either:
  - A NULLable column
  - An automatically-generated value such as IDENTITY

INSERT and Surrogate Key

- Our sample SIS database uses application-generated “surrogate” keys
- A surrogate key is an artificial identifier used to uniquely identify a business object
- Surrogate keys are typically used as the primary key for a table
- Often a good idea to ensure keys are different formats in order to easily differentiate the keys of different classes of business objects
- For example:
  - Clients, accounts, employees, products
  - Students, courses, accounts, payments, invoices, sections, instructors

INSERT and IDENTITY

- Surrogate keys are often automatically generated by the database management system
  - Most efficient way of generating identifiers that are guaranteed to be unique
  - In Microsoft SQL Server, an **IDENTITY** column specifies an automatically-generated unique value
    - Must be numeric; no built-in support for generating alphanumeric keys
- Usually one omits the surrogate key from INSERT statements
  - This behaviour can be overridden using the SET IDENTITY_INSERT statement

INSERT FROM SELECT

- You can also INSERT into a table values that come from a SELECT statement:

```
INSERT INTO Person (number, firstName, LastName
      , personalEmail, collegeEmail)
      SELECT number
            , firstName + '0'
            , lastName + '0'
            , personalEmail + '0'
            , collegeEmail + '0'
      FROM Person
      WHERE firstName LIKE 'P%'
```

- **Insert** statements are **atomic**; either they modify all of the intended rows correctly, or the entire statement is undone

WARNING!

For the next two SQL statements,

UPDATE and **DELETE**,

always remember to **include a WHERE clause**

Otherwise every row in the table will be modified or deleted

UPDATE statement

- UPDATE statement syntax:

```
UPDATE table
SET column1 = value1,
    column2 = value2
WHERE …
```

- Note that the UPDATE statement may SET the same columns as those referenced in the statement's WHERE clause
  - The set of rows to be updated is computed first, prior to any modifications being made
  - Update statements are **atomic**; either they modify all of the intended rows correctly, or the entire statement is undone

UPDATE examples

- You can update one or more columns in an UPDATE statement:

```
UPDATE Course
SET credits = 4, hours = 60
WHERE number = 'ACCT1025'
```

- You can restrict which rows to modify using the WHERE clause
- SQL Server cannot update more than one table with an UPDATE
- If you omit the WHERE clause, all rows in the table will be updated unless an error occurs.

```
UPDATE Person
SET personalEmail = NULL
```

Note: Some vendors have a "safety" setting to disallow missing WHERE clause and will not update any records.

UPDATE with Subquery

- You can use a subquery in an UPDATE statement:

```
UPDATE Employee
SET reportsTo =
  (SELECT number
   FROM Employee
   WHERE location = '3B117'
   AND schoolCode = 'EIT')
WHERE number = 5512736
```

DELETE

- DELETE statement syntax:

```
DELETE FROM table
WHERE …
```

- DELETE statements are atomic; either all intended rows are deleted without error, or the entire statement is undone

DELETE example

- You can delete one or more rows with a single DELETE statement:

```
DELETE FROM CourseOffering
WHERE courseNumber = 'PROG8080'
AND sessionCode = 'F08'
```

- If you omit the WHERE clause, all rows in the table will be deleted!

TRUNCATE TABLE

- Many database products and the ISO SQL Standard includes a statement, TRUNCATE, that specifically deletes every row in a table
  - Use TRUNCATE if deleting the entire table is desirable
  - Implementation of TRUNCATE is designed for efficiency
    - TRUNCATE is usually faster than DELETE
- Example:

```
TRUNCATE TABLE CourseOffering
```

- DELETE is logged and can be rolled back; TRUNCATE is not logged and cannot be rolled back

WARNING (again!)

For the previous two SQL statements,

UPDATE and **DELETE**,

always remember to **include a WHERE clause**

Otherwise every row in the table will be modified or deleted