MODULE 2 DATABASE PROGRAMMING

INSERT, UPDATE, DELETE, Transactions, Constraints, and Referential Integrit

YESTERDAY...

What is a key?

What is a primary key?

What is a foreign key?

What is a join?

What is a union?

Databases: Only good for retrieving data?

ADDING INFORMATION

SYNTAX: **INSERT INTO** table_name (column1, column2, ..., column_n)
VALUES (value1, value2, ... value_n);

ADDING INFORMATION

SYNTAX: **INSERT INTO** table_name (column1, column2, ..., column_n)
VALUES (value1, value2, ... value_n);

SYNTAX: **INSERT INTO** table_name VALUES (value1, value2, ... value_n);

ADDING INFORMATION

SYNTAX: INSERT INTO table_name (column1, column2, ..., column_n)
select column1, column2, ..., column_n from table_two where condition;

UPDATING INFORMATION

SYNTAX: UPDATE table_name **SET** column = value
WHERE column = value;

DELETING INFORMATION

SYNTAX: DELETE FROM table_name WHERE column=value;

WHY DELETING IS BAD

- Can’t get data back
- What if you deleted the wrong record?
- Customer changes their mind

INSERTING AGAIN

REFERENTIAL INTEGRITY

Keys ensure that relationships between tables remain consistent.

PRIMARY KEY - allows FKs to establish a relationship, and enforces NOT NULL and UNIQUE,
FOREIGN KEY - enforces valid PK values, and limits deletion of the PK row if FK row exists

Constraints define the conditions with which a column must comply.

NOT NULL
UNIQUE
CHECK - specifies acceptable values that can be entered in the column
DEFAULT - provides a default value for the column
Identity Specification – Auto generate primary key

BANKING

Update MedvitzsAccount
set balance=balance-100

Update TomsAccount
set balance=balance+100

BANKING

Update MedvitzsAccount
set balance=balance-100

Update TomsAccount
set balance=balance+100

TRANSACTIONS

A **transaction** is a single unit of work. When it is successful, it should be "committed". If an error is encountered at any point it should be cancelled or rolled back.

TRANSACTIONS

BEGIN TRANSACTION
<sql statements>
[ROLLBACK || COMMIT] TRANSACTION

The ACID Test

Atomicity: Within a transaction, a series of database operations all occur or none occur.

Consistency: The completed transaction leaves things remaining in a consistent state at the end. Any rules in place before the transaction still pass after the transaction.

Isolation: Ensures that the concurrent execution of a transaction results as if the operations were executed serially.

Durability: Once a transaction has been committed it will remain so, even during a power loss, crash, or an error.

LET'S CODE!

WHAT QUESTIONS DO YOU HAVE?