Syracuse University
School of Information Studies

# Database Applications and Programming

## Agenda

- What is a database application?
- What are the common characteristics of a DB application, and how are they built?
- An understanding of the layers of an application (DB application)
- Exploration of the many application architectures of today
- What is SQL programming, and why is it useful?
- What role does it play in a database application?
- Learn the basic constructs of SQL programming
- Cover the four types of data logic: views, stored procedures, triggers, and user-defined functions

Syracuse University
School of Information Studies

Database Applications and Programming

# The End

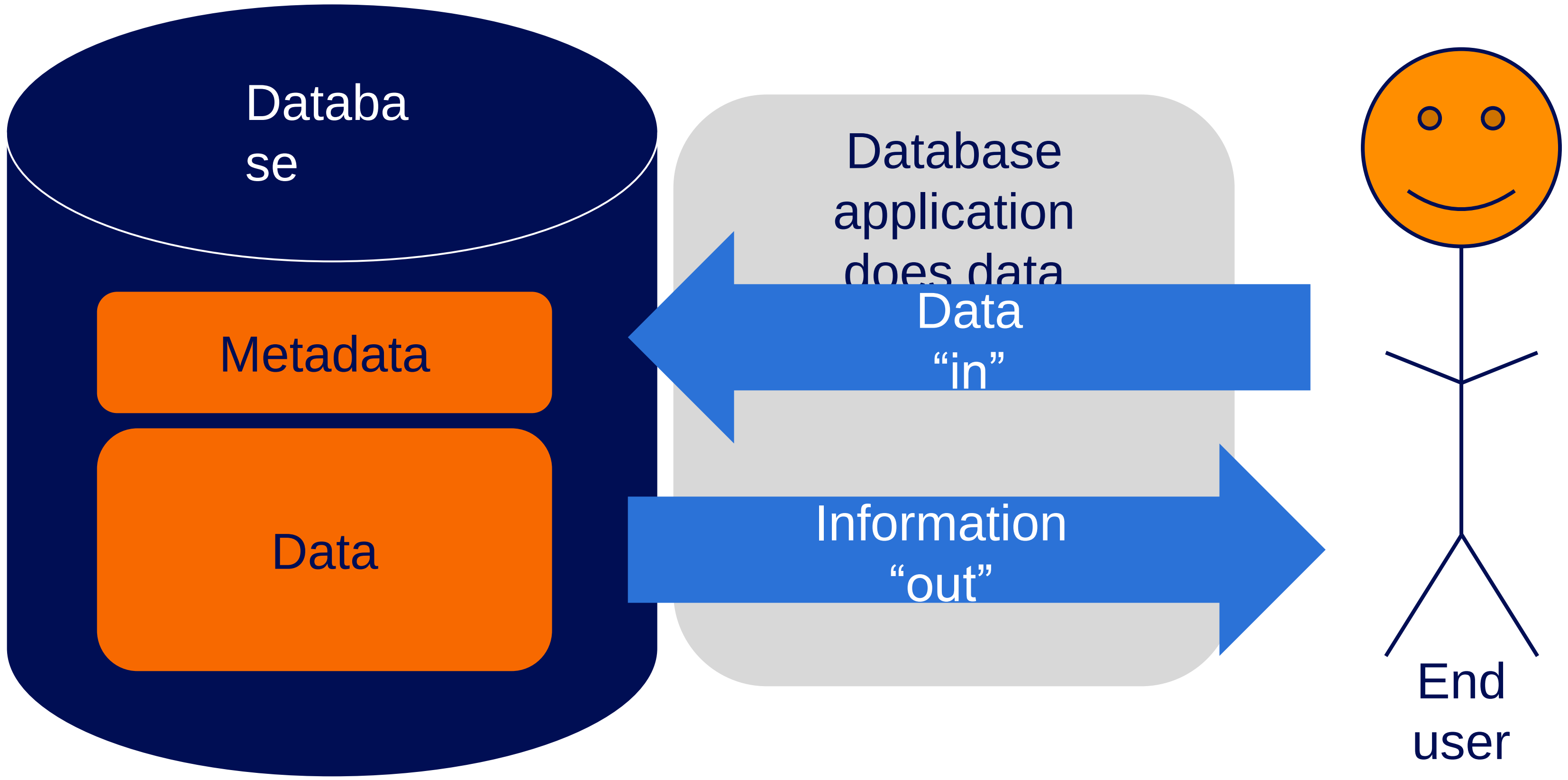
Syracuse University
School of Information Studies

# Database Applications

## What Is a Database Application?

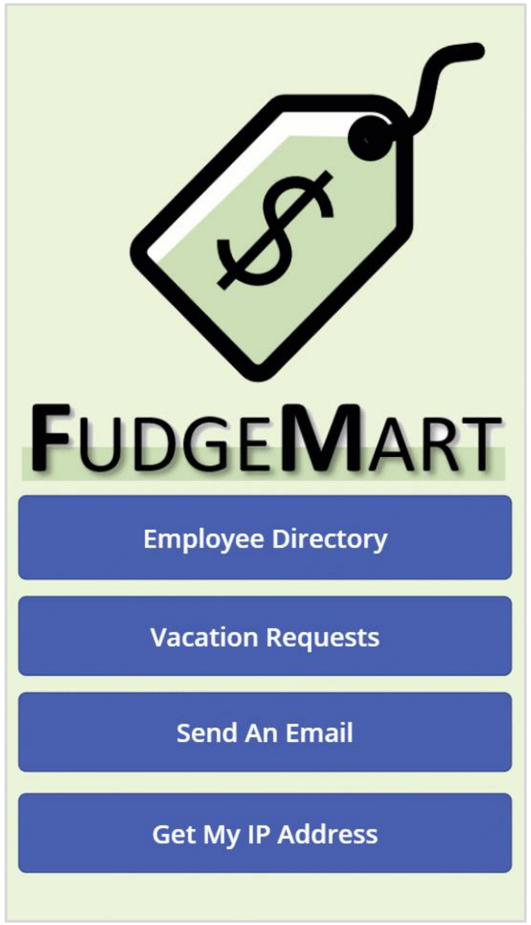
- Application that stores its data in a database management system
- Application is data-focused and its design is based on its data requirements
- Implementation is a data model-first approach
- Provides user interface for the CRUD operations
- Support for users and roles
- Screens and reports

# DB Application Facilitates Data Management



## Database Applications Have Many Names

- Database application
- Line of business (LOB) apps
- Forms over data apps
- Model-driven/model-first apps
- CRUD apps



## Database Application Development Software

- Software designed for the purpose of building database applications
- Not a DBMS, but might have one built-in
- No-code/low-code development platforms
- Examples
  - Oracle APEX, Microsoft PowerApps, Google AppSheet, Amazon Honeycode
- Examples with a DBMS built-in
  - Microsoft Access, Claris FileMaker Pro, Intuit QuickBase
- Also called rapid application development (RAD) platforms/tools

Syracuse University
School of Information Studies

Database Applications

## The End

Syracuse University
School of Information Studies

## Layers of a Database Application

# Layers of a Modern Data-Oriented Application

### Presentation

Code and layout responsible for the user interface

### Business logic

Transformational logic at the heart of what the application does

### Data logic

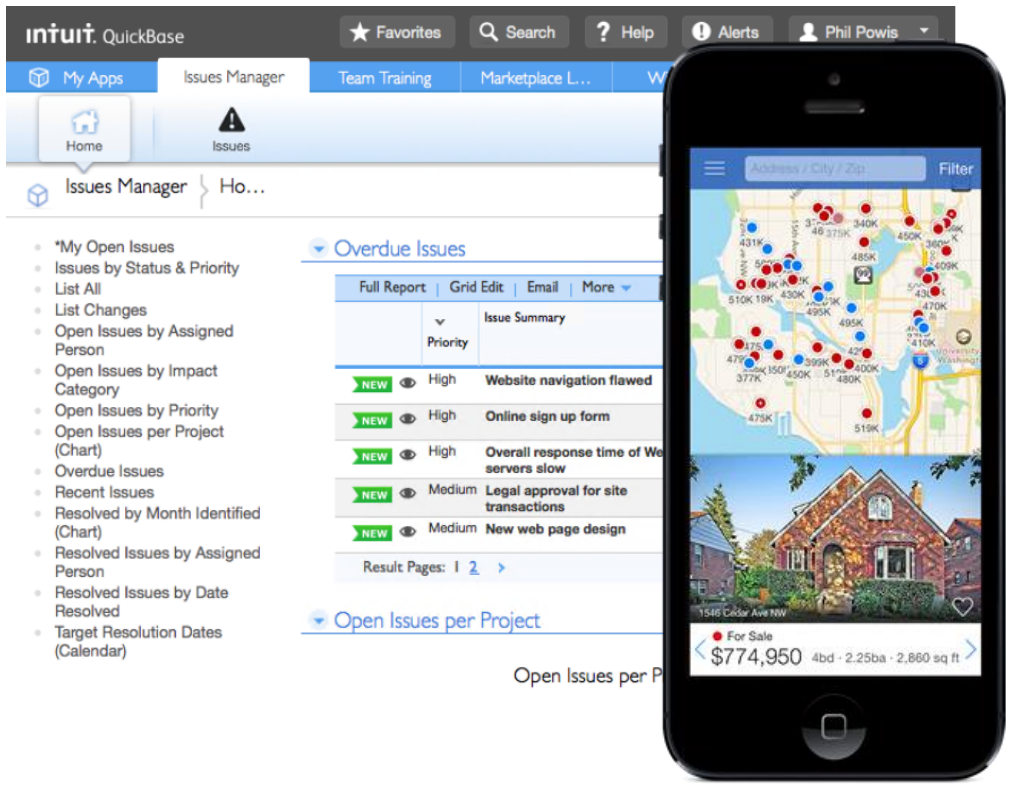
Data management (CRUD) operations

### Database

Data storage and retrieval

# Presentation Layer

- User interface concerns
  - Presentation of information
  - Application navigation/menus
  - Data collection/validation
- Web/native/mobile

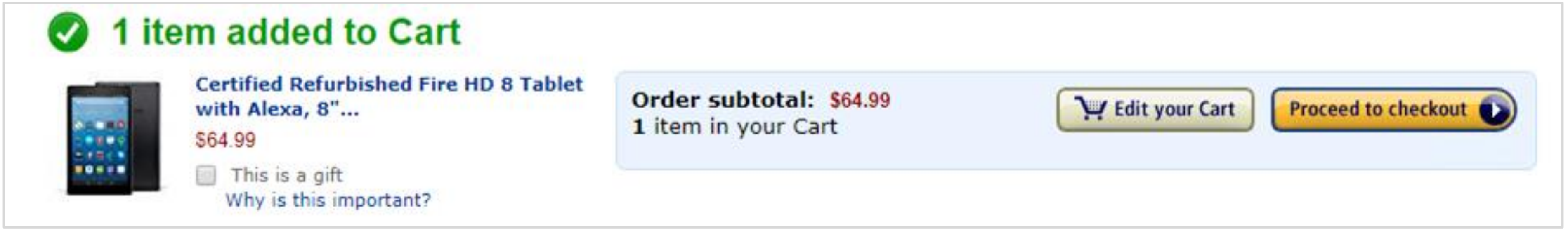


| | Priority | Issue Summary |
|---|---|---|
| NEW | High | Website navigation flawed |
| NEW | High | Online sign up form |
| NEW | High | Overall response time of We servers slow |
| NEW | Medium | Legal approval for site transactions |
| NEW | Medium | New web page design |

Result Pages: 1 2 >

## Business Logic Layer

- Main transformational logic of the application; part of the application's functionality
- Written in a programming language: Java, JavaScript, Python, C#, and so on

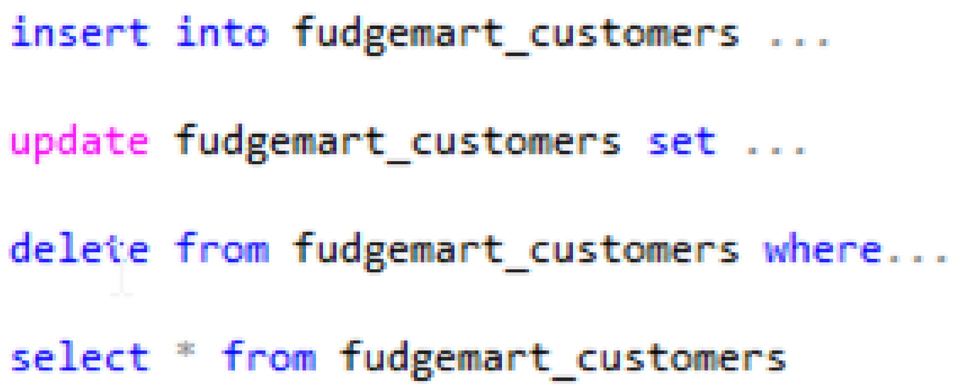

1 item added to Cart

Certified Refurbished Fire HD 8 Tablet with Alexa, 8"...
$64.99
This is a gift
Why is this important?

Order subtotal: $64.99
1 item in your Cart

# Data Logic Layer

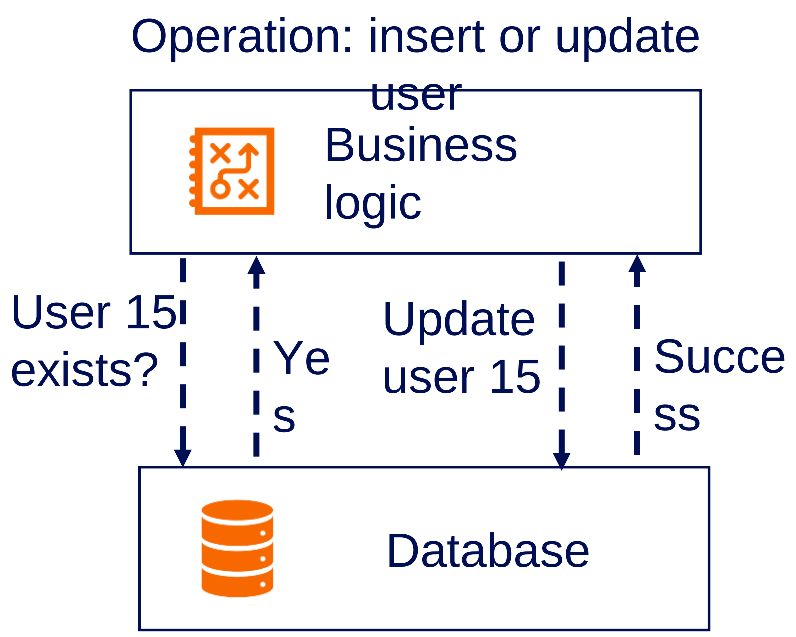
- Responsible for CRUD operations
- Typically, this is code that transforms operations into the DSL (domain-specific language) to communicate with the database
- Commonly SQL, but for other DBMS it could be something else



```
insert into fudgemart_customers ...

update fudgemart_customers set ...

delete from fudgemart_customers where...

select * from fudgemart_customers
```

# What Is Data Logic?

- Data logic can be defined loosely as any logical operation that requires one or more reads and/or writes from a database.
- It makes sense to put data logic in the DBMS if it supports it.
- Such operations are candidates for the data logic layer to minimize the network communication between layers.



## Database

- Persistent store of data and metadata
- As required, will handle concurrency and consistency issues
- Typically, a DBMS; not always relational

Syracuse University
School of Information Studies

Layers of a Database Application

# The End

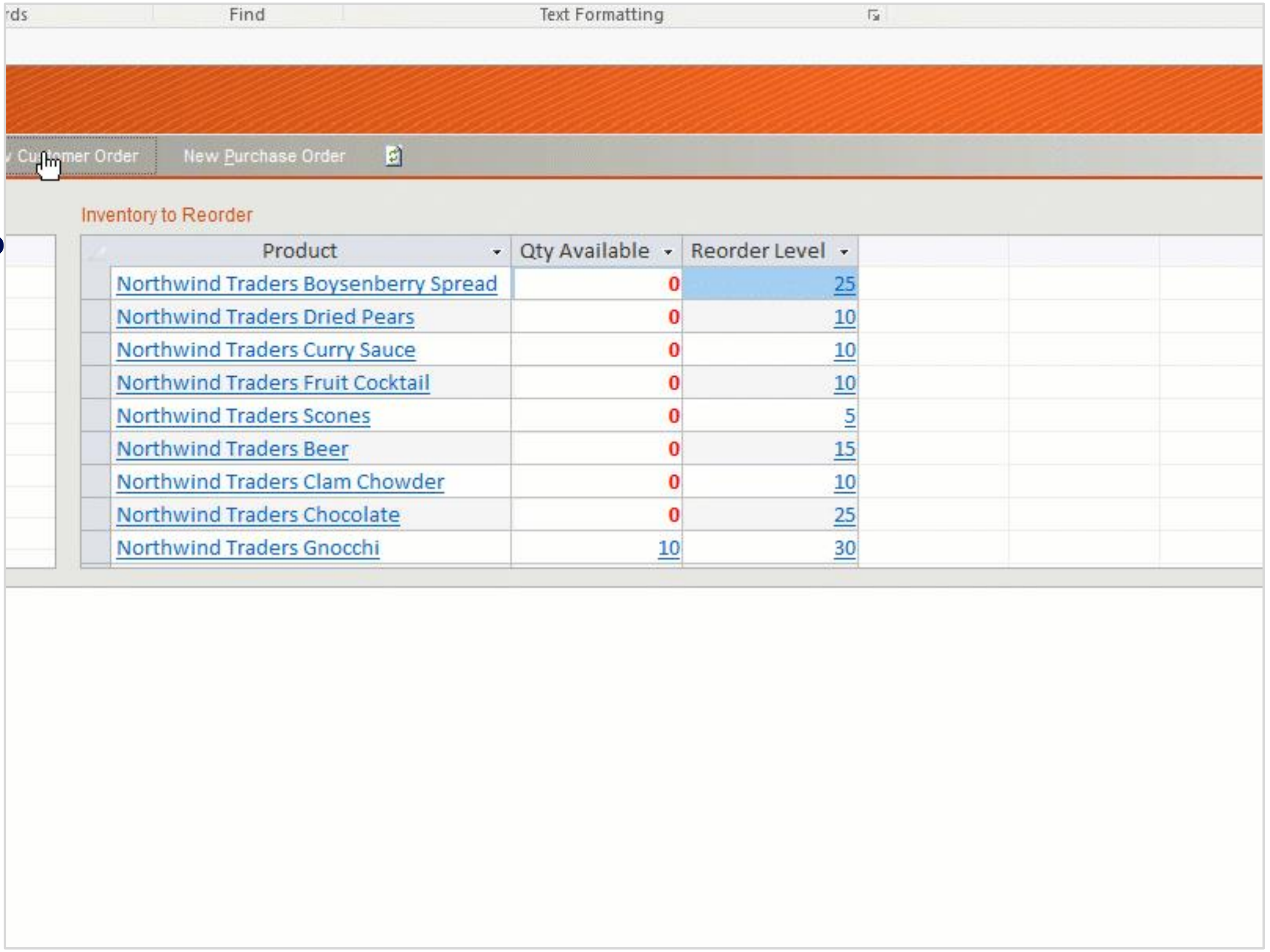
Syracuse University
School of Information Studies

## Demo

Layers of a Database Application

# Demo: Northwind Traders database application

- Layers?
  - Presentation?
  - Business logic?
  - Data logic?
  - Database?



Inventory to Reorder

| Product | Qty Available | Reorder Level |
|---|---|---|
| Northwind Traders Boysenberry Spread | 0 | 25 |
| Northwind Traders Dried Pears | 0 | 10 |
| Northwind Traders Curry Sauce | 0 | 10 |
| Northwind Traders Fruit Cocktail | 0 | 10 |
| Northwind Traders Scones | 0 | 5 |
| Northwind Traders Beer | 0 | 15 |
| Northwind Traders Clam Chowder | 0 | 10 |
| Northwind Traders Chocolate | 0 | 25 |
| Northwind Traders Gnocchi | 10 | 30 |

Syracuse University
School of Information Studies

## Demo: Layers of a Database Application

# The End

Syracuse University
School of Information Studies

# Development of Database Applications

## Model-First Approach

- The data model is the driver of the application
    - Conceptual model
    - Logical model
- Other functional requirements are captured based on user interactions with the data model
- The user stories approach, part of agile modeling method, helps to identify:
    - User roles
    - Required features
    - Necessary UI functionality

## What Is a User Story?

- A user story is a high-level functional requirement that focuses on the capability of a user within the system to be built.
- It should contain enough information so that a programmer can implement the story trivially with reasonable effort.
- There are two kinds of user stories, formal and informal. Informal user stories follow no predetermined format, whereas formal does.
- Each user story should be written by a key stakeholder, or at the very least with the assistance of the key stakeholders.
- They are part of the conceptual model.

## Informal/Formal User Stories

- Informal
  - Students should be able to purchase annual parking passes online.
- Formal
  - As a <role> I can <capability>, so that <receive benefit>.
  - As a student I should be able to purchase a parking pass online so that I can drive to school.
  - The benefit helps to prioritize the story and rationalize its value.

Syracuse University
School of Information Studies

Development of Database Applications

# The End

Syracuse University
School of Information Studies

# User Stories Example

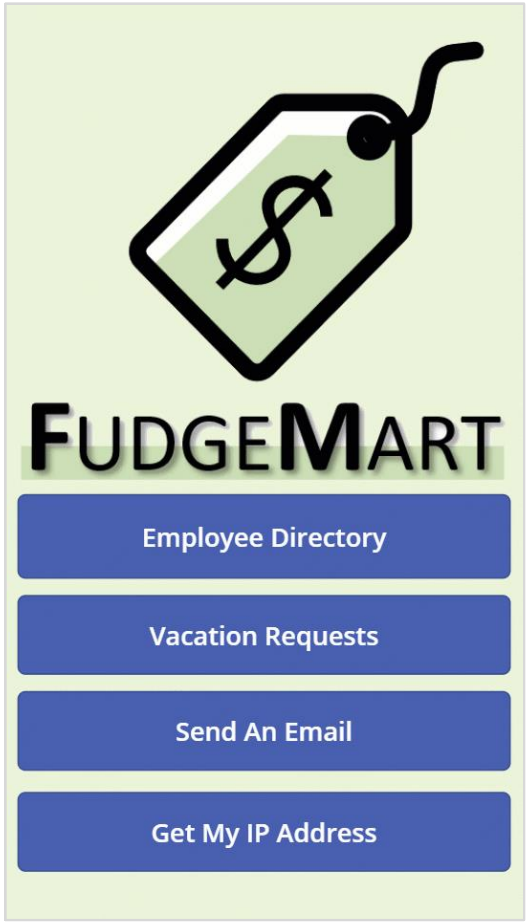
## Example User Stories

| Fudgemart vacation requests | |
|---|---|
| As an employee:<br>• I can make a vacation request so that I can use my earned vacation time. | As a manager:<br>• I can review and approve a vacation request so that my employee can take their earned vacation. |

# User Stories Become Application Features

As an employee:

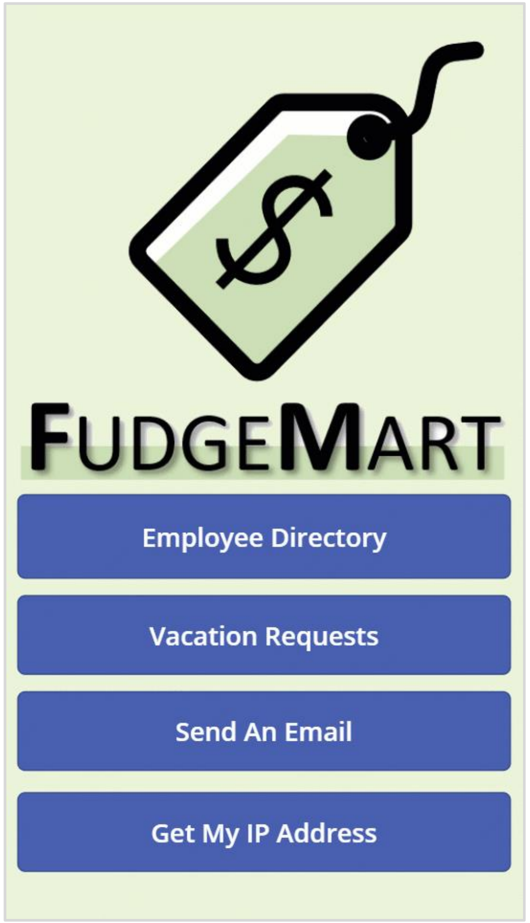
- I can make a vacation request so that I can use my earned vacation time.



## User Stories Become Application Features (cont.)

As a manager:

- I can review and approve a vacation request so that my employee can take their earned vacation.



Syracuse University
School of Information Studies

## User Stories Example

# The End

Syracuse University
School of Information Studies

## Why SQL Programming?

## SQL Programming?

- Code can be written in SQL
- This goes beyond simple INSERT, SELECT, etc.
- Common use cases and examples
  - Insert a value only if it does not exist already
  - When insert into table X, update value in table Y
  - Persist a shopping cart into an order
  - Look up a postal code based on a city and region
  - Sanitize input (e.g., obfuscate credit-sensitive data)
- This is called data logic

# Recall: Layers of a Modern Data-Oriented Application

**Presentation**

Code and layout responsible for the user interface

**Business logic**

Transformational logic at the heart of what the application does

**Data logic**

Data management (CRUD) operations

**Database**

Data storage and retrieval

Syracuse University
School of Information Studies

Why SQL Programming?

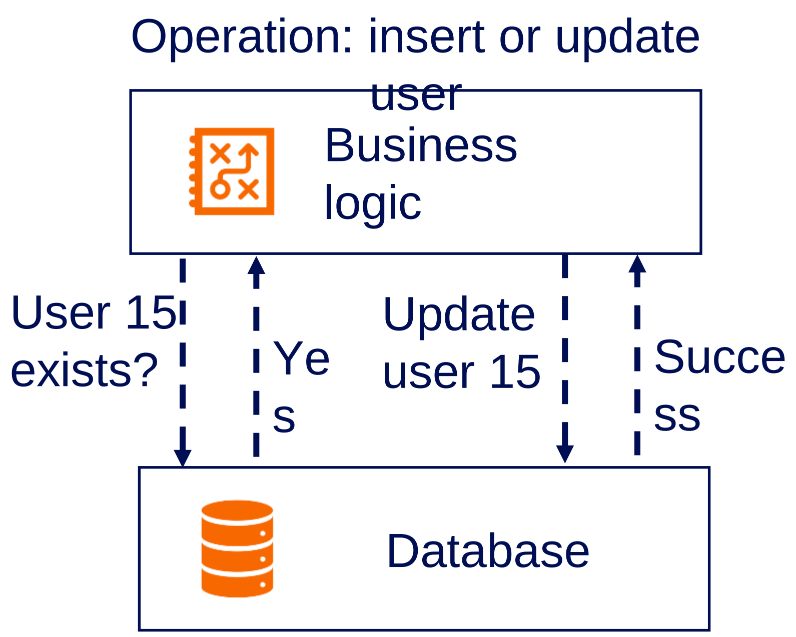
## The End

Syracuse University
School of Information Studies

## Understanding Data Logic

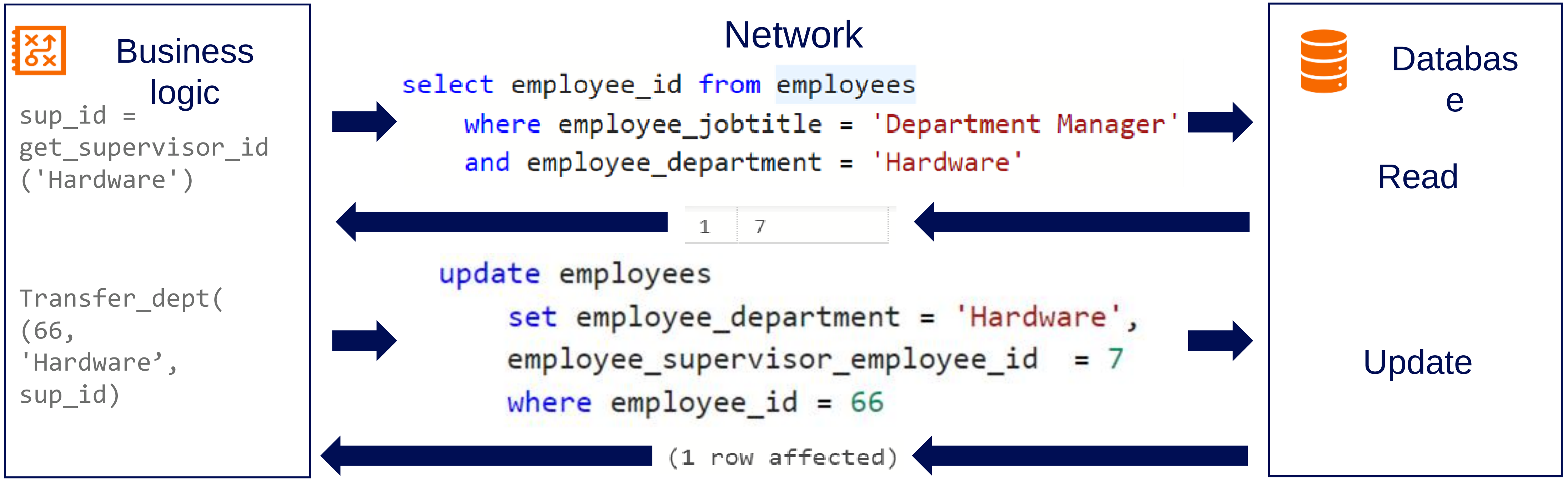
## Data Logic

- Data logic can be defined loosely as any logical operation that requires more than one read or write from the database.
- Such operations are candidates for the data logic layer to minimize the network communication between layers.
- Data logic is a special type of business logic, solely on data management. It does nothing else!



# Example as Business Logic

- Business logic: transfer employee to new department
- Example: transfer employee id=66 from ‘Toys’ to ‘Hardware’ department



```
sup_id =
get_supervisor_id
('Hardware')
```

```
select employee_id from employees
    where employee_jobtitle = 'Department Manager'
    and employee_department = 'Hardware'
```

```
1    7
```

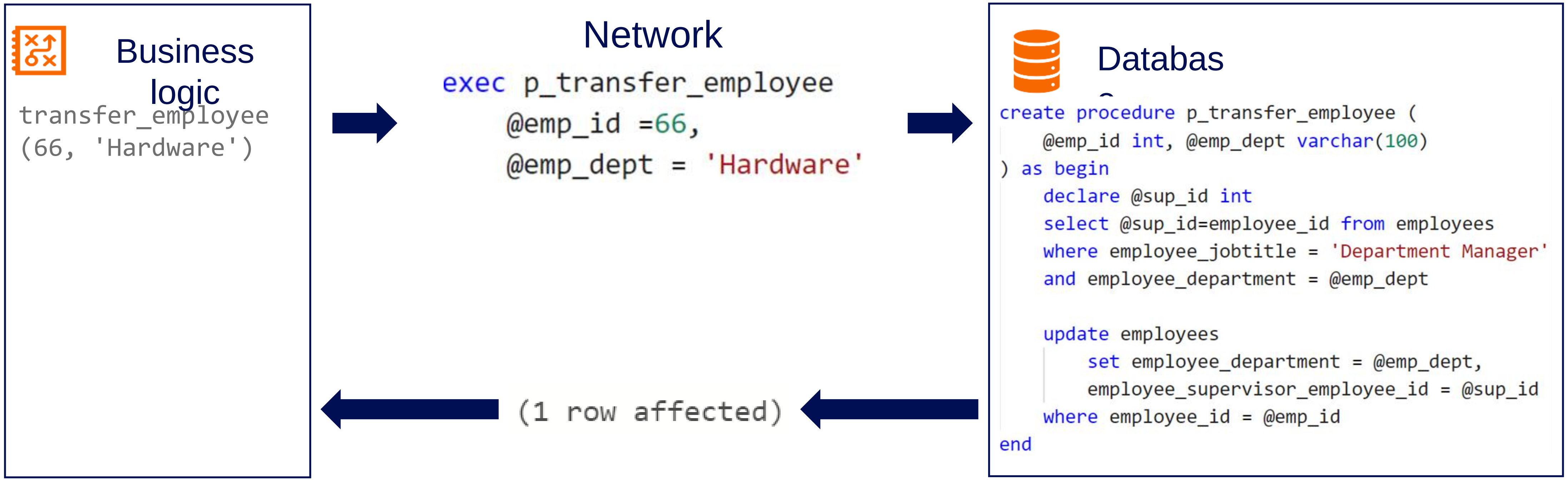
```
update employees
    set employee_department = 'Hardware',
    employee_supervisor_employee_id  = 7
    where employee_id = 66
```

```
(1 row affected)
```

```
Transfer_dept(
(66,
'Hardware’,
sup_id)
```

# Example as Data Logic

- Business logic: transfer employee to new department
- Example: transfer employee id=66 from ‘Toys’ to ‘Hardware’ department



```
transfer_employee
(66, 'Hardware')
```

```
exec p_transfer_employee
    @emp_id =66,
    @emp_dept = 'Hardware'
```

```
create procedure p_transfer_employee (
    @emp_id int, @emp_dept varchar(100)
) as begin
    declare @sup_id int
    select @sup_id=employee_id from employees
    where employee_jobtitle = 'Department Manager'
    and employee_department = @emp_dept

    update employees
        set employee_department = @emp_dept,
        employee_supervisor_employee_id = @sup_id
    where employee_id = @emp_id
end
```

```
(1 row affected)
```

# Recall Data Models: Where Is Data Logic?

Abstract

| | |
|---|---|
| Conceptual data model | • An abstract representation of the data requirements<br>• No implementation of the database itself; no implementation model selected |
| Logical data model | • Mapping of the conceptual model to an implementation model<br>• No implementation of the database; model selected |
| Internal data model | • The internalized implementation of the database applicat<br>• A DBMS and implementation model selected |
| External data model | • User's view of the database application<br>• DBMS and implementation model selected, internal model implemented |
| Physical data model | • How the internal/external model is stored by the DBMS and operating system<br>• All aspects of the database are an implementation |

Concret e

Data logic

Syracuse University
School of Information Studies

Understanding Data Logic

# The End

Syracuse University
School of Information Studies

## SQL Programming Basics

## Disclaimer: Every DBMS Is Different!

- Every DBMS has its own dialect of SQL for programming.
- These are usually called PL/SQL for “procedural language SQL.” They are based on SQL, with more traditional programming logic.
- Important disclaimer: The concepts we will learn are the same, but the specifics of each DBMS implementation are not.
- If you learn one DBMS’s PL, you’ve learned them all.
  - Microsoft SQL Server □ T-SQL
  - Oracle □ PL/SQL
  - PostgreSQL □ PL/pgSQL
  - IBM DB2 □ PL/SQL

Syracuse University
School of Information Studies

SQL Programming Basics

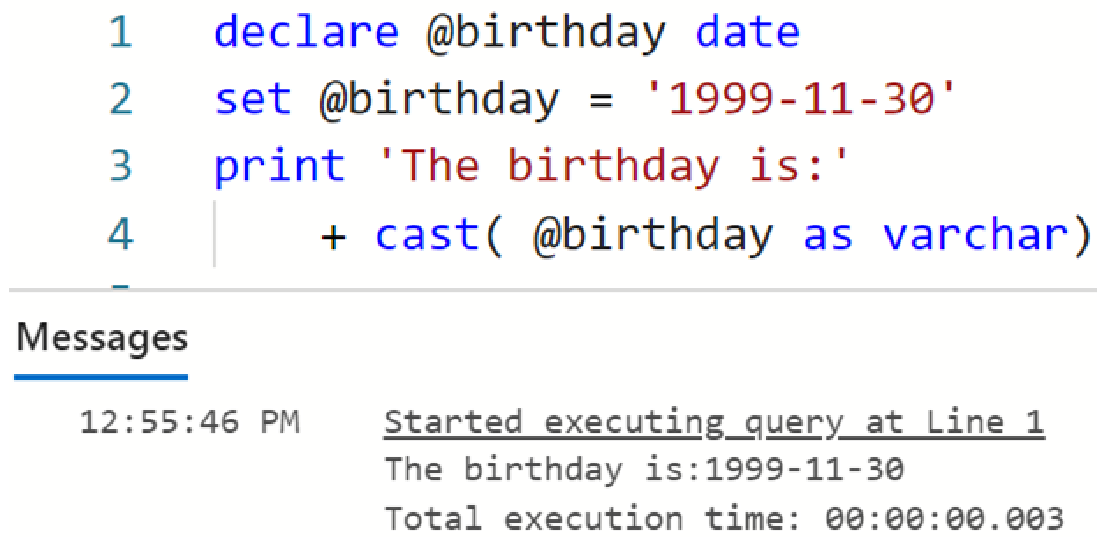
# The End

Syracuse University
School of Information Studies

# T-SQL Variables

## Variables: DECLARE and SET

- A variable is a named location in memory that stores a value.
- Each variable must begin with an @ symbol.
- Variables must be one of the SQL data types.
- The declare keyword creates the variable.
- The set keyword assigns a value to the variable.



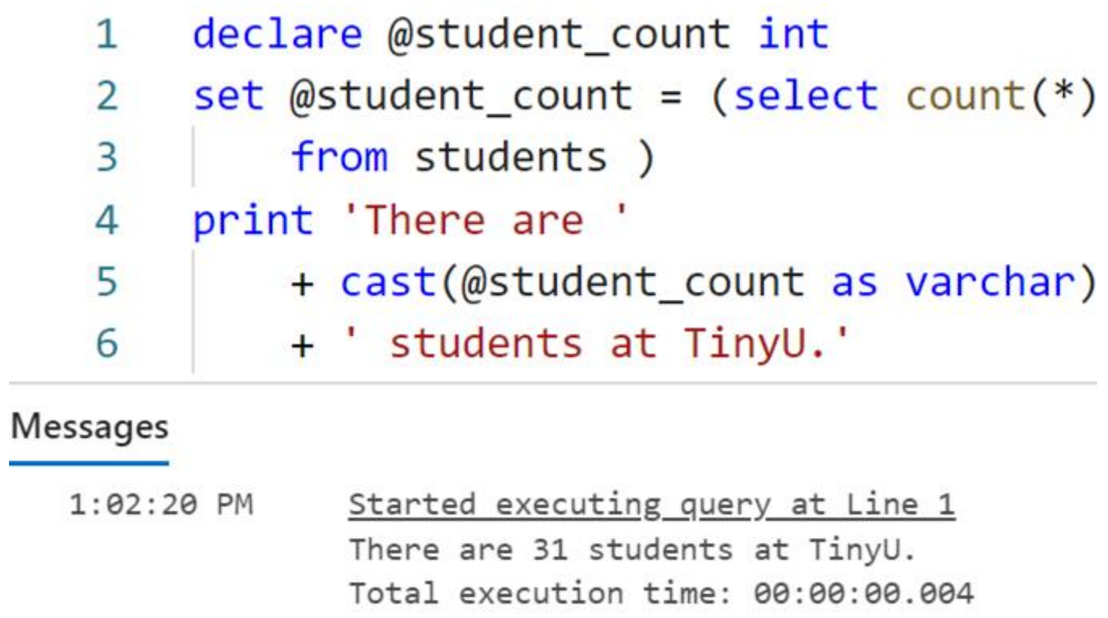
```
declare @birthday date
set @birthday = '1999-11-30'
print 'The birthday is:'
    + cast( @birthday as varchar)
```

Messages

```
12:55:46 PM    Started executing query at Line 1
               The birthday is:1999-11-30
               Total execution time: 00:00:00.003
```

## Assign SQL Output to a Variable

- You can assign the output of an SQL select statement to a variable.
- First, write the SQL select, then assign it to a variable.
- The SQL must return a single value (one row, one column).

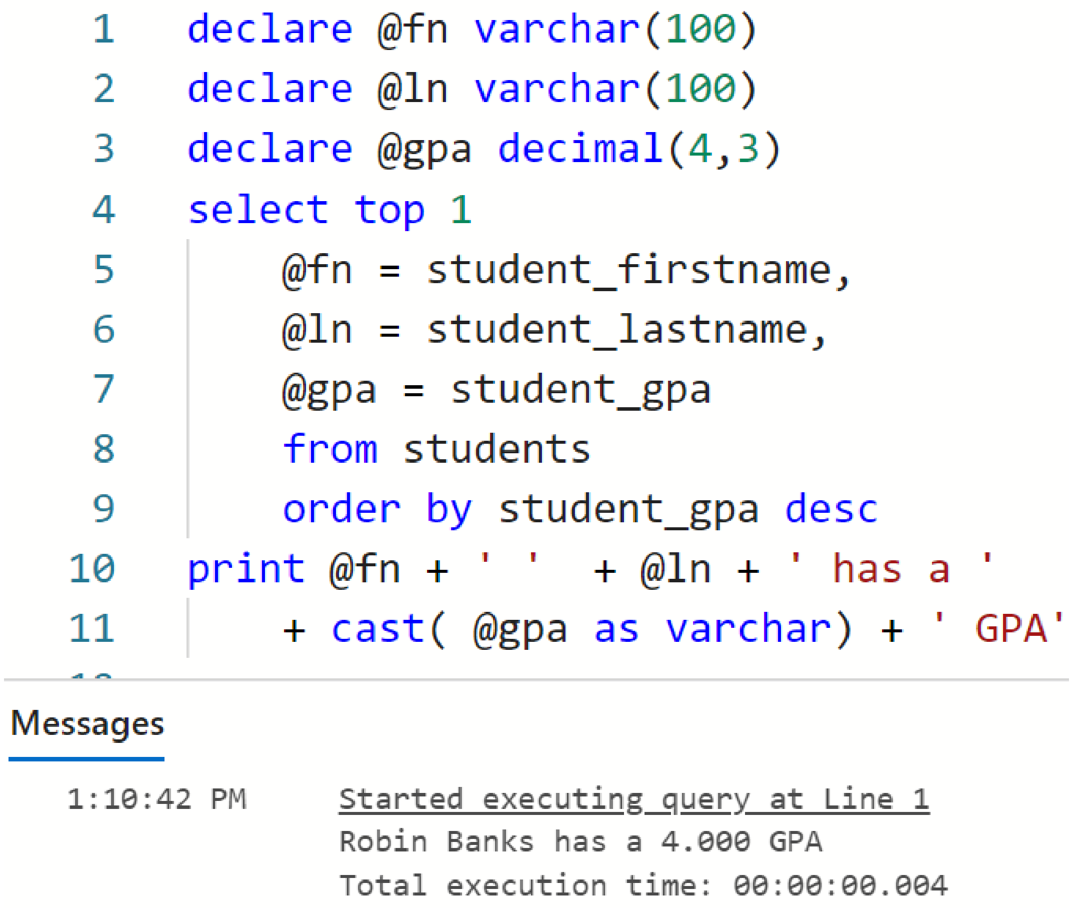


```
declare @student_count int
set @student_count = (select count(*)
    from students )
print 'There are '
    + cast(@student_count as varchar)
    + ' students at TinyU.'
```

```
1:02:20 PM      Started executing query at Line 1
                There are 31 students at TinyU.
                Total execution time: 00:00:00.004
```

# Assign Multiple Columns to Variables

- You can assign multiple columns of output to variables.
- The SQL must return a single row, or else the last row will be assigned to the variables.



```
declare @fn varchar(100)
declare @ln varchar(100)
declare @gpa decimal(4,3)
select top 1
    @fn = student_firstname,
    @ln = student_lastname,
    @gpa = student_gpa
    from students
    order by student_gpa desc
print @fn + ' '  + @ln + ' has a '
    + cast( @gpa as varchar) + ' GPA'
```

Messages

```
1:10:42 PM     Started executing query at Line 1
               Robin Banks has a 4.000 GPA
               Total execution time: 00:00:00.004
```

Syracuse University
School of Information Studies

T-SQL Variables

# The End

Syracuse University
School of Information Studies

## Demo

T-SQL Variables

## Demo: T-SQL Variables

- We will use the Azure Data Studio application.
- We will use the payroll database.
- Declare and assign a value t a variable.
- Use a variable in an SQL query.
- Assign several variables from an SQL query.

Syracuse University
School of Information Studies

Demo: T-SQL Variables

## The End

Syracuse University
School of Information Studies

## T-SQL Built-In Functions

## Built-In Functions

- Most PL/SQL implementations have a variety of built-in functions to allow us to manipulate text, dates, do math, etc.
- T-SQL function reference
    - What are the Microsoft SQL database functions?

## Common Text String Functions

- LEN(column)
  - Returns the number of characters in the column
- LEFT(column, count)
  - Returns the left-most count of characters from the column
- RIGHT(column, count)
  - Returns the right-most count of characters from the column
- CHARINDEX(find, column)
  - Returns the first index of find in the column; indexes start at 1

## Examples

```
declare @msg nvarchar(20)
set @msg = 'hello, world'

print len(@msg)  -- 12
print left(@msg, 5) -- 'hello'
print right(@msg, 5) -- 'world'
print charindex(',', @msg) -- 6
print left(@msg, charindex(',', @msg)-1) --hello
```

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| h | e | l | l | o | , |  | w | o | r | l | d |

## Common Date/Time Functions

- DATEPART(part, date)
  - Returns the part of the date you specify as an integer
- DATENAME(part, date)
  - Returns the part of the date you specify as a string (varchar)

```
print getdate() -- today's date and time
print datepart(month, '2018-07-20') -- 7
print datename(month, '2018-07-20') – 'July'
print datepart(year, '10:34:39 PM') -- 1900 1900-1-1 is epoch
print datepart(hour, '10:24:39 PM') -- 22
print datename(second, '10:24:39 PM') -- '39'
```

## More Useful String Functions

- ISNULL (check_expression, replacement_value)
  - When check_expression is null, use replacement_value
- STRING_AGG (expression, separator) [<order clause>]
  - Aggregate operator returns a single row of expression separated by separator
  - Useful for turning rows into single-column lists of data
- STRING_SPLIT (string, separator)
  - Returns a single-column table of values of string separated by separator
  - Useful for turning single column of data lists into rows
  - Requires CROSS APPLY join

Syracuse University
School of Information Studies

T-SQL Built-In Functions

## The End

Syracuse University
School of Information Studies

## Demo

Built-In Functions

# Demo: Built-In Functions

- We will use the Azure Data Studio application
- We will use the fudgemart_v3 database
- Use ISNULL to replace null with a flag
- STRING AGG
  - Emails by state
  - All orders by customer
- STRING_SPLIT
- STRING_SPLIT with CROSS APPLY for keyword extraction

Syracuse University
School of Information Studies

Demo: Built-In Functions

## The End

Syracuse University
School of Information Studies

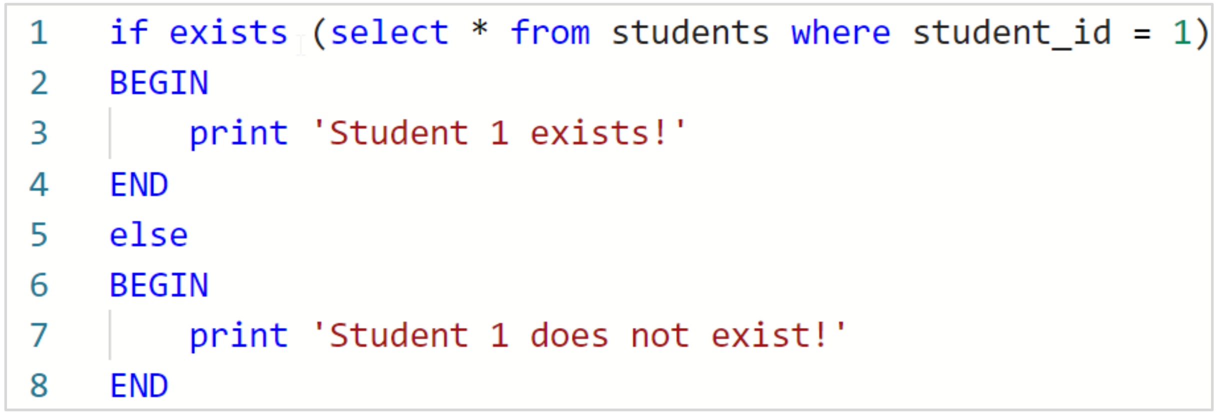
## Program Flow Control

## Branching

The T-SQL IF statement allows us to branch our script based on a true/false value.

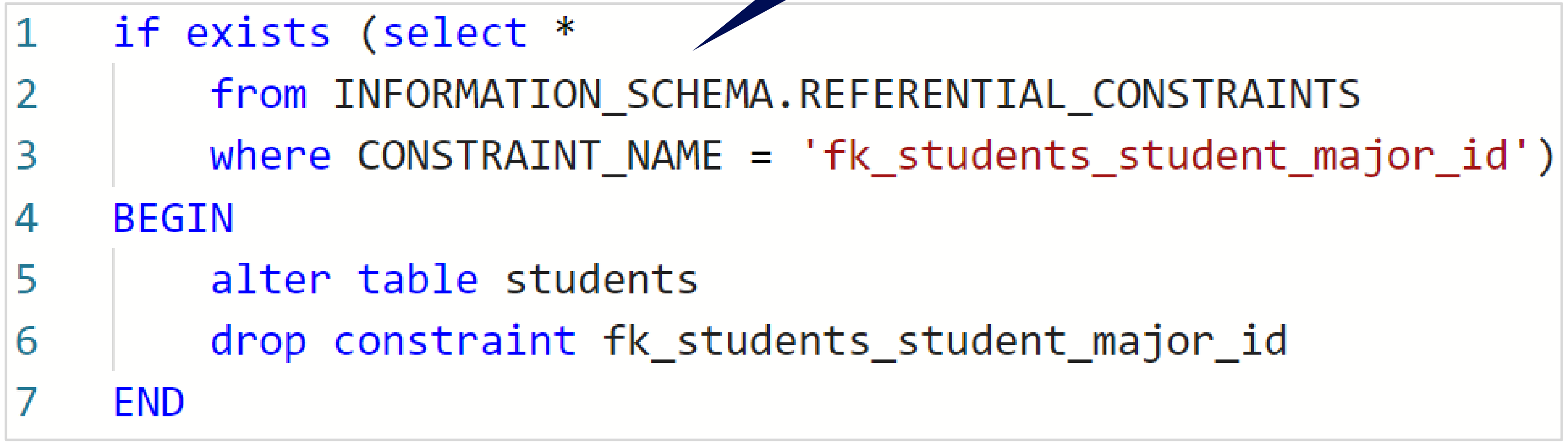
```
IF true-false-expression
BEGIN
    statements-when-true
END
ELSE
BEGIN
    statements-when-false
END
```

## Example: Exists



```sql
if exists (select * from students where student_id = 1)
BEGIN
    print 'Student 1 exists!'
END
else
BEGIN
    print 'Student 1 does not exist!'
END
```

Commonly used for schema inspection



```sql
if exists (select *
    from INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
    where CONSTRAINT_NAME = 'fk_students_student_major_id')
BEGIN
    alter table students
    drop constraint fk_students_student_major_id
END
```

Syracuse University
School of Information Studies

Program Flow Control

## The End

Syracuse University
School of Information Studies

## SQL Data Logic Types/Views

## Types of Data Logic in the RDBMS

Views

Stored procedures

Triggers

User-defined functions

Syracuse University
School of Information Studies

SQL Data Logic Types/Views

# The End

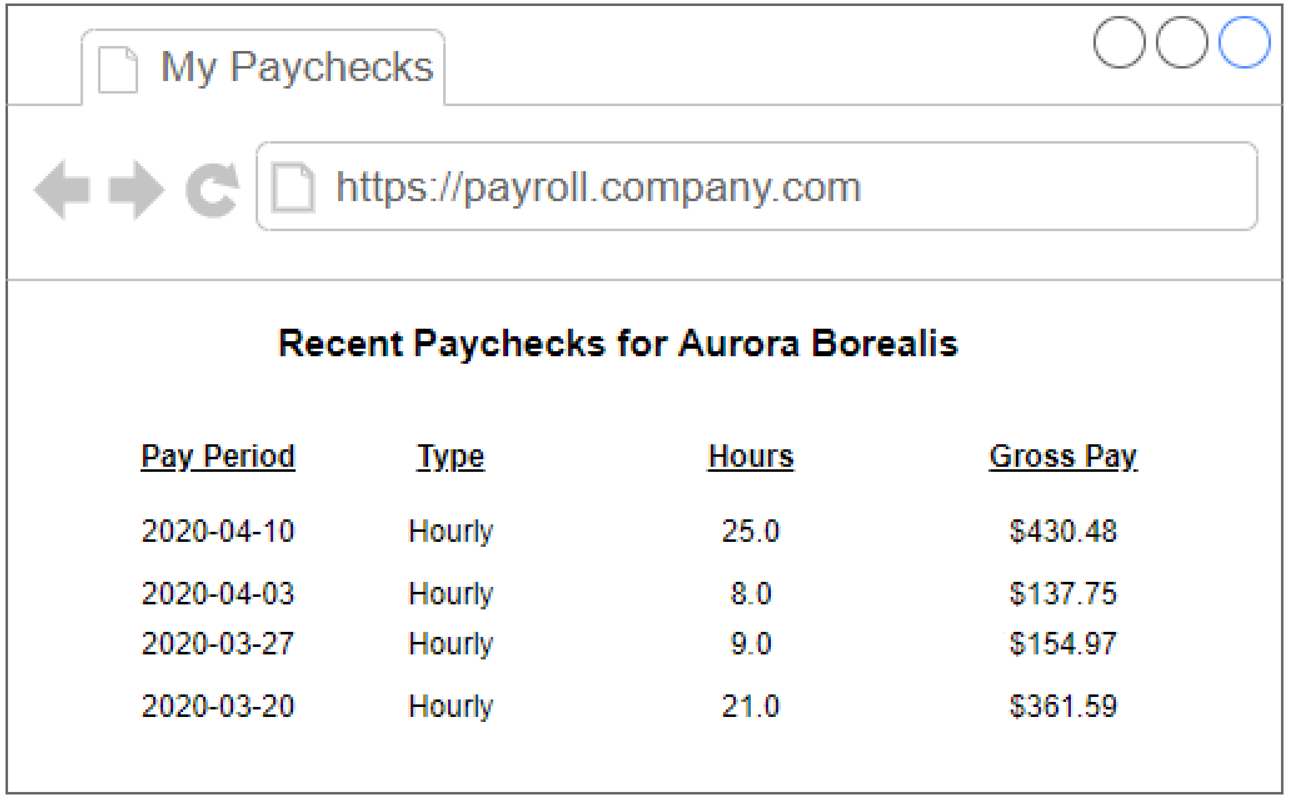
Syracuse University
School of Information Studies

# Demo

Views

# Demo: View Data Logic

- My paychecks!
- View to allow an employee to see their last four paychecks on the company portal



Recent Paychecks for Aurora Borealis

| Pay Period | Type | Hours | Gross Pay |
|---|---|---|---|
| 2020-04-10 | Hourly | 25.0 | $430.48 |
| 2020-04-03 | Hourly | 8.0 | $137.75 |
| 2020-03-27 | Hourly | 9.0 | $154.97 |
| 2020-03-20 | Hourly | 21.0 | $361.59 |

Syracuse University
School of Information Studies

Demo: Views

## The End

Syracuse University
School of Information Studies

# Stored Procedures

## Stored Procedures

- A stored procedure is a group of T-SQL statements that encapsulate data logic under a single command
- They help us build the external data model, creating data logic in our applications
- Examples
  - Write a customer record. If the customer exists, update; if not, insert.
  - Create a new customer bank account that requires us to insert into several tables.
  - When an order is added with a new shipping address, update the address in the customer table.

## Benefits of Stored Procedures

Keeps data logic “close to the data,” resulting in less network traffic

Isolated security—you can deny access to a table but allow access to a stored procedure that inserts into the table

Performance—the T-SQL in the stored procedure that is pre-compiled to it executes faster than the equivalent T-SQL code

Implement the external data model—abstracting relational complexity from the middle-tier programmer

## Syntax: Stored Procedures

A stored procedure has 0 or more inputs and 0 or 1 output.

```
CREATE PROCEDURE procedure_name (
     [@parameter_name AS type]
     [, …n]
) AS
BEGIN
     T-sql-statements
     [RETURN value]
END
```

Procedure inputs (parameters)

Procedure output

## Calling a Stored Procedure

- To execute or call a stored procedure

```
EXEC procedure_name arg1, arg2
```

- Return value can be set to a variable

```
EXEC @var = procedure_name arg1, arg2
```

- The arguments can be named for readability

```
EXEC @var = procedure_name @param1=arg1,
@param2=arg2
```

Syracuse University
School of Information Studies

Stored Procedures

# The End

Syracuse University
School of Information Studies

## Demo

Stored Procedures

# Demo: Stored Procedure Data Logic

- Switching departments
- Same example from before, but we will write it as a stored procedure
- Inputs
  - Employee ID
  - New department
- Find manager of new department
- Update employee with new department
- Execute the procedure!

Syracuse University
School of Information Studies

Demo: Stored Procedures

## The End

Syracuse University
School of Information Studies

## Triggers

## Triggers

- A trigger is a special type of stored procedure that will execute automatically based on a DML operation and event
- These are the DML operations, any combination of:
  - INSERT, UPDATE, or DELETE
- There are two events
  1. AFTER: trigger fires after the DML operation completes, after integrity constraints
  2. INSTEAD OF: trigger fires before the DML operation completes, before integrity constraints

## Trigger Use Cases

- Prevent automatic updates
- Audit changes to the database
- Ensure data integrity

## CREATE TRIGGER Statement

```
CREATE TRIGGER trigger_name
    ON table_name
     AFTER | INSTEAD OF
     {  [INSERT]  [,] [UPDATE ] [,] [DELETE] }
BEGIN
    sql statements
END
```

# Trigger Sequencing

```
UPDATE majors SET major_name = 'Comp Sci' WHERE major_code = 'CSC'
```

Data state before DML statement

majors **table:** 'Computer Science'

INSTEAD OF trigger event

inserted table: ‘Comp Sci’
deleted **table:** 'Computer Science’

Data integrity constraints checked

Check, Unique, Default, PK,FK constraints,

AFTER trigger event

inserted table: ‘Comp Sci’
deleted table: 'Computer Science’

Data state after DML statement

Majors table: ‘Comp Sci'

## Trigger Tables

- How does one access the internal data affected by a trigger?
- SQL Server provides two special tables to assist.
  - Inserted—table consisting of data to be:
    - Added or
    - Updated
  - Deleted—table consisting of data to be:
    - Removed or
    - Prior to being updated

Syracuse University
School of Information Studies

Triggers

# The End

Syracuse University
School of Information Studies

## Demo

Understanding the Trigger Tables

Demo: Understanding the Trigger Tables

- We will use the Azure Data Studio application
- We will use the payroll database
- A trigger to display the inserted and deleted tables
- Execute the p_transfer_employee procedure
- Follow changes

## Demo: AFTER Trigger

- We will use the Azure Data Studio application
- We will use the payroll database
- An AFTER trigger to automatically change the supervisor_employee_id to match the department
- Demonstrate it works on an update and a batch update

Syracuse University
School of Information Studies

Demo: Understanding the Trigger Tables

## The End

Syracuse University
School of Information Studies

# User-Defined Functions

## User-Defined Functions

- You can define your own functions in T-SQL.
- This allows us to encapsulate data logic into our databases and make our intentions clear.
- The function can be used in the same manner as a built-in function.
  - In a SELECT query
  - As part of a derived column
  - As part of a check constraint
  - In the FROM clause (table-valued function)

## Syntax: User-Defined Functions

A function has 0 or more inputs and exactly 1 output.

```
CREATE FUNCTION function_name (
    [@parameter_name AS type]
    [, …n]
) RETURNS type AS
BEGIN
    function_body
    RETURN value
END
```

Function inputs (parameters)

Function output

Syracuse University
School of Information Studies

User-Defined Functions

## The End

Syracuse University
School of Information Studies

## Demo

User-Defined Functions

## Demo: User-Defined Functions

- We will use the Azure Data Studio application
- We will use the payroll database
- Get manager for department
  - Given the department name as input
  - Returns the manager employee ID as output, or NULL if no match
  - Favor joins over data lookup functions
- Name last first function as a better example

Syracuse University
School of Information Studies

Demo: User-Defined Functions

## The End

Syracuse University
School of Information Studies

## Table-Valued Functions

## Table-Valued Functions

- Table-valued functions are nonscalar select statements wrapped in a function.
- Unlike views, table-valued functions take parameters as input.
- Unlike a view, a table-valued function can contain multiple
  T-SQL statements.

## CREATE FUNCTION (Table-Valued)

```
CREATE FUNCTION function_name (
     [@parameter_name AS type]
     [, …n]
) RETURNS TABLE AS
RETURN (
     select-statement
)

SELECT * from function_name(arg1);
```

## Views vs. Table-Valued Functions

| | Views | Table functions |
|---|---|---|
| Accepts parameters | No | Yes |
| Expanded out by query optimizer | Yes | Yes |
| Can be materialized in advance | Yes (indexed view) | No |
| Is updateable | Yes | Yes |
| Can contain multiple statements? | No | Yes |
| Can have triggers | Yes | No |
| Can use side-effecting operator? | Yes | No |

Syracuse University
School of Information Studies

## Table-Valued Functions

# The End

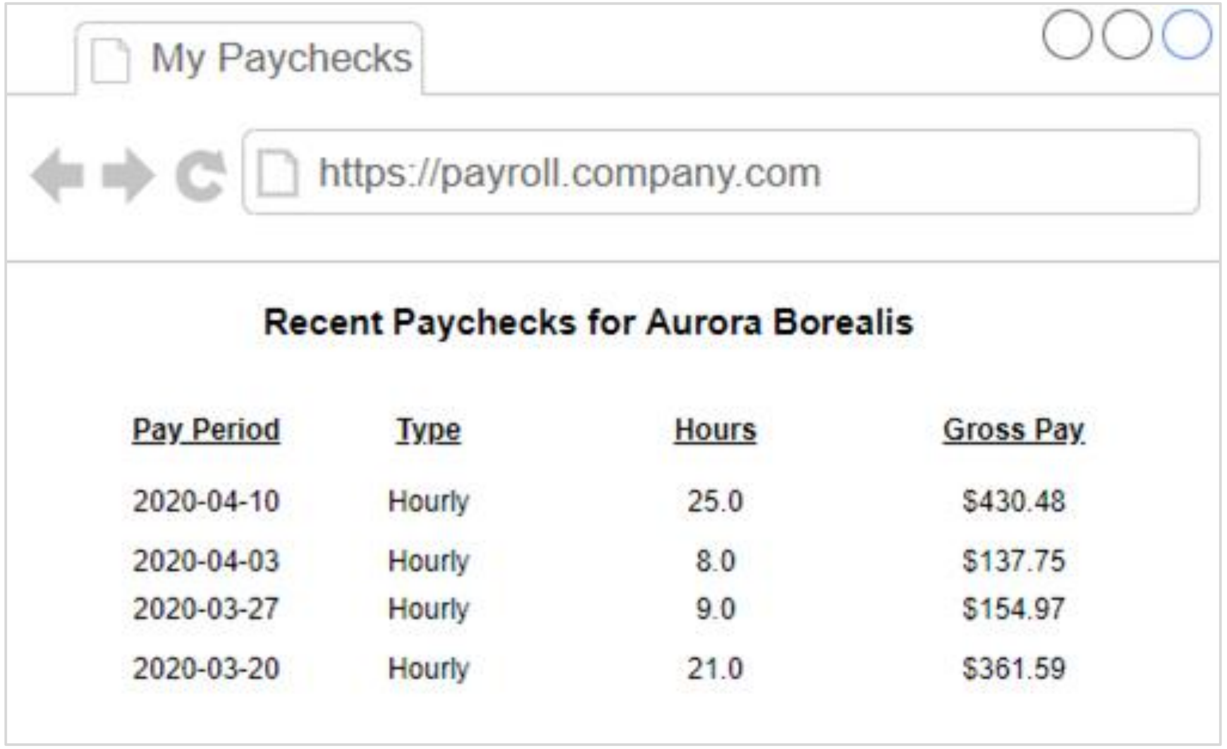
Syracuse University
School of Information Studies

## Demo

Table-Valued Functions

# Demo: Table-Valued Functions

- My paychecks!
- A table-valued function to allow an employee to see their last four paychecks on the company portal, just like the view but with a parameter



Recent Paychecks for Aurora Borealis

| Pay Period | Type | Hours | Gross Pay |
|---|---|---|---|
| 2020-04-10 | Hourly | 25.0 | $430.48 |
| 2020-04-03 | Hourly | 8.0 | $137.75 |
| 2020-03-27 | Hourly | 9.0 | $154.97 |
| 2020-03-20 | Hourly | 21.0 | $361.59 |

Syracuse University
School of Information Studies

Demo: Table-Valued Functions

## The End

Syracuse University
School of Information Studies

## Summary

## Summary

- Database applications are data-oriented applications and can be developed with no-code/low-code tools.
- Database applications are divided into layers, presentation, business logic, data logic, and database.
- Database applications come in a wide variety of architectures, from monolithic to microservices.
- User stories can be used to capture presentation and business logic concerns for database applications.
- SQL programming is useful for creating data logic.
- Data logic is business logic that strictly focuses on CRUD operations.
- There are four types of data logic that can be created in RDBMS: views, procedures, functions, and triggers

Syracuse University
School of Information Studies

Summary

## The End