# Team 16 Presentation

Presented By: Trevor Richardson, Justin Galiger, Nihal Adhikary, Dhruv Makwana, and Sean Pichay

### Timeline 1: Functional Dependencies & Logical Design

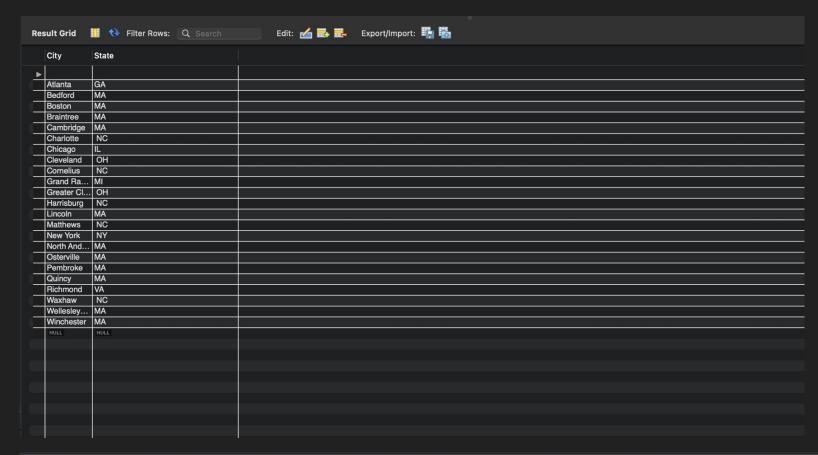
- Normalization
  - 1NF
    - The table is already in 1NF, no changes needed
  - o 2NF
    - The table is already in 2NF, no changes needed
  - 3NF: The table is not in 3NF
    - The violation of 3NF was that city determines state, which is a transitive dependency.
    - Conversion
      - EMPLOYEEDATA (<u>Linkedin URL</u>, First Name, Last Name, Position, Company, Specialties, Industry, City)
      - Location(<u>City</u>, <u>State</u>)
  - o BCNF
    - The table is already in BCNF, no changes needed
  - For both tables, the highest normalization form was 3NF, as both of the tables were already finished in 1NF and 2NF, and upon completion of 3NF BCNF was also completed.

# Timeline 2: Table Creation & Information Importation

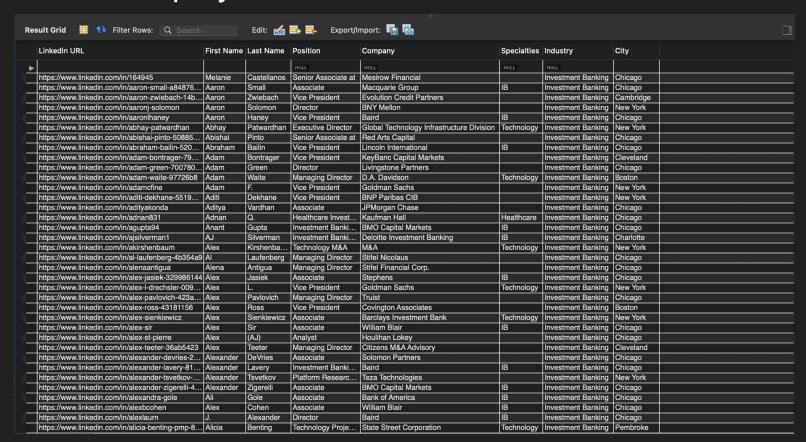
- Table: Location
- Columns: pk-City, State
- Data Import: Columns imported matched columns from company csv file

- Table: EmployeeData
- **Columns:** pk-Linkedin URL-uk, First Name, Last Name, Position, Company, Specialties, Industry, fk-City
- Data Import: Columns imported matched columns from company csv file

#### Table: Location



#### Table: EmployeeData

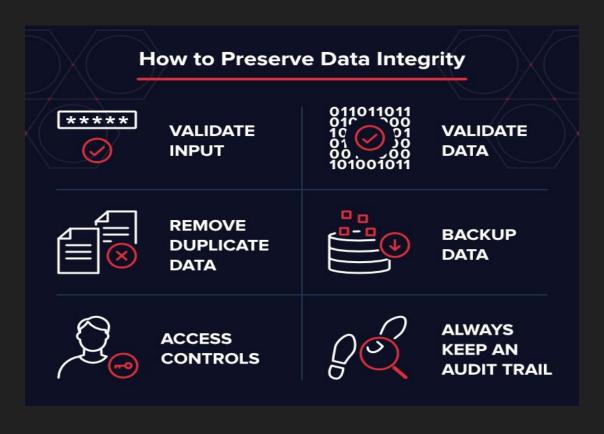


# Timeline 3: Data Integrity, Disaster Recovery, & Security

Data is a very important tool for any operation. In order to efficiently and effectively use data, we need to manage and organize it properly. We need to ensure data integrity. There are several forms of integrity, namely entity, referential, and domain.



# Basics of Data Integrity:



#### Data Backup:

- 1. Identify the failure
- 2. Stop all activity to prevent any more damage or loss
- 3. Find the cause
- 4. Restore the database from a backup
- 5. Use an incremental backup to ensure data is up to date
- 6. Find a solution to the failure
- 7. Test to ensure that the database functions properly and that data is correct
- 8. Communicate with the company to inform them of said failure and what was done to eliminate it and recover data.
- Improve the recovery plan

#### Security:

We will mostly use row-based security. Depending on the company and what data they consider to be sensitive, the security model should determine which specific user rows have what permissions. Each row or column should have different permissions. These permissions decided whether a particular user row can update and view data or even update the procedure. Some other useful security measures are:

- strong passwords
- two-factor authentication



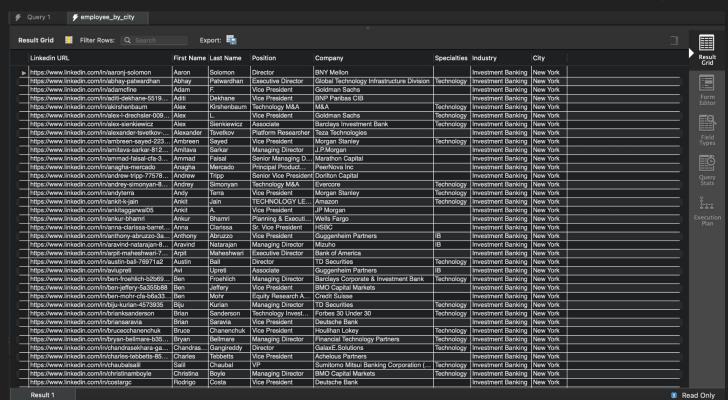
#### Timeline 4: Functions, Procedures, Views

- Stored Procedure: (Employee Data By City) This stored procedure takes in a VARCHAR parameter (City Name) which then locates all the associated data that matches that city name and returns all of the employee information from employees that live in the specified city.
- Stored Function: (Number Of Employees In City) This stored function takes in a VARCHAR parameter (City Name) which then locates all the associated data that matches that city name and then counts the number of rows that are returned and returns that number in the form of an INT. (eg. Enter 'New York', function returns 222 as that is the number of employees that are located in that city.)

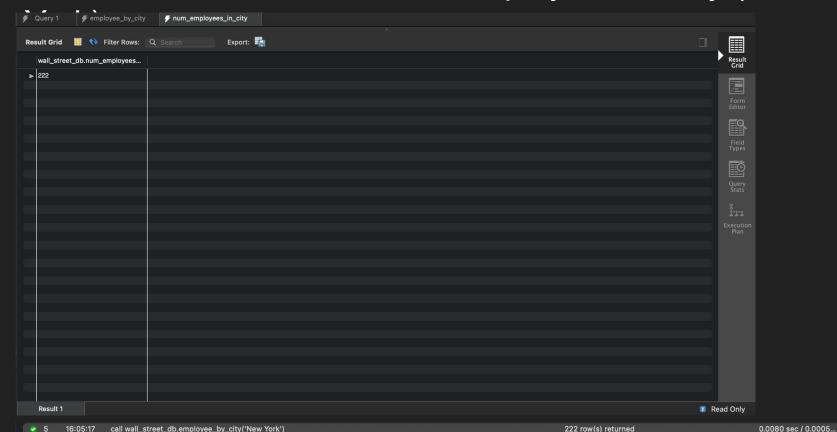
#### Views:

- city\_state (City, State)
- company\_data (Company, Industry, City)
- employee\_data (Linkedin URL, First Name, Last Name, City)

# Stored Procedure: Employee Data By City (New York)



# Stored Function: Number Of Employees In City (New



#### Updated Conceptual Design:

Entity: Location

Entity: **EmployeeData** 

Relationship: Location contains EmployeeData

Cardinality: One to many. Each Employee can only be associated with one City while each city can have multiple employees associated with it.

Participation:

Entity 1: (**Location**) has total participation, meaning each location must be associated with at least one employee (assuming that "City" attribute in "EmployeeData" entity is required and cannot be null).

Entity 2: (**EmployeeData**) has partial participation, meaning that an employee may or may not be associated with a location (assuming that "City" attribute in "EmployeeData" entity is optional and can be null).

#### Updated Logical Design:

Tables: Location & EmployeeData

Columns: Location(pk\_City, State)

EmployeeData(pk\_LinkedIn\_URL, FirstName, LastName, Position, Company, Specialties, Industry, City (foreign key; references pk\_City of Location table))

Indexes: Clustered(pk\_City), Non-Clustered(City, State), Clustered(pk\_LinkedIn\_URL), Non-Clustered(City)

Views: company\_data, employee\_data, city\_state

Stored Procedure: Employee Data by City

Stored Function: Number of Employees in City

#### Future Feature Implementation:

- 1. Cloud Based Hosting: Remote access, migrating MySQL database.
- 2. Data Warehousing: Easy to analyze data, consolidate employee data.
- 3. Business Intelligence: Tools to turn data into useful decision.
- 4. Predictive Analysis: Forecast future, employee turnover and satisfaction.
- 5. Mobile Access: Employee personal info, engagement, and company resource.