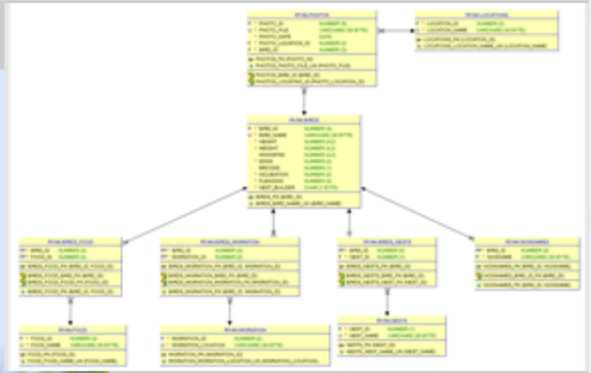
Database Design For Any IT Professional



Course Overview

Welcome to Database Design for Any IT Professional! This class is for anyone that needs to learn the basic principles of Relational Database Design, and has been designed for any experience level. This course is a building block for many IT-related careers and will help you contribute to your organization's success.

Hands-on approach

- Accommodates those with very little to no knowledge or experience
- Allows those with more knowledge and experience to dive deeper
- Encourages collaboration and ideas from students

The primary goal is to gain an overall understanding of the Relational Database Model, and how to effectively design and model a relational database using best-practice techniques. In this course, you will:

- understand the basic principles of the RDBMS
- learn how to effectively design a database through examples and interactive exercises
- use Referential Integrity and Normalization to design a solid database
- create an Entity Relationship Diagram for both your logical and physical models
- convert your logical database model to a relational model, then to a physical DB using SQL code

Tell Me About You

Polling Question (radio buttons)

Which of the following best describes your profession related to SQL?

- ○ Non-IT related
- ○ Database Designer/Modeler
- ○ End User, Functional User
- ○ Technical Manager
- ○ Manager, Organization Leader, Stakeholder
- ○ Developer, Programmer
- ○ Web Developer
- ○ Database Administrator
- ○ Business Analyst, Data Analyst
- ○ IT Security Professional
- ○ Business Intelligence (BI) developer, analyst
- ○ Other technical

Hour 1 Overview

Understanding the basics of the Relational Database

- Information, data, and databases in today's world
- Relational database key elements
    - Data, entities, attributes, relationships
    - Referential Integrity
    - Schemas, tables, columns, keys, constraints, data definition

Understanding the basics of Database Design

- Knowing your data
- Gathering requirements
- Asking the right questions
- Examining sample database models
- Establishing groups of data (entities) and a list of fields (attributes)

Getting to know the test organization and data

Brainstorming exercise

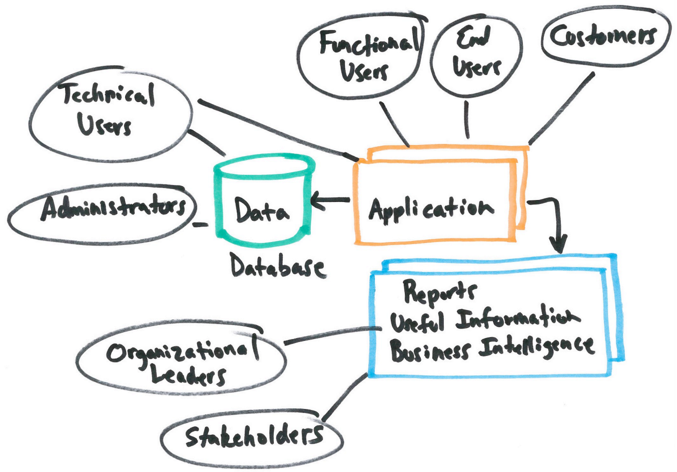
Exercise review

Break

Understanding the Basics of the Relational Database

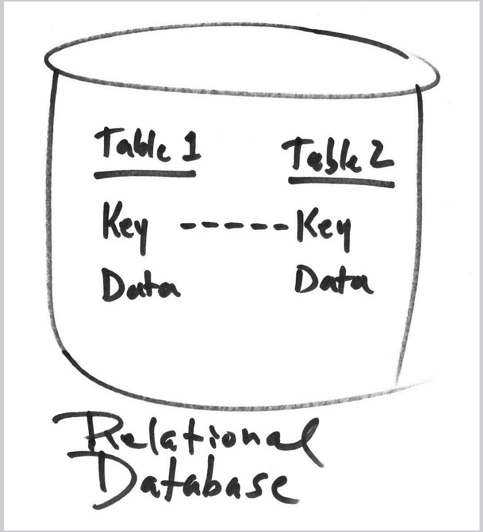
A database is a structure used to store information, or data. Organizations and individuals depend on data every day, and even every minute of each day.

- Mountains of information must become "Usable Data."
- Data is critical to the success of any organization.
    - End users, consumers, customers, customers' customers
- Users depend on data every day.
    - Organizational leaders
    - Managers
    - Technical users
    - Administrators
    - Functional and end users
    - Stakeholders
    - Customers



Understanding the Basics of the Relational Database

- RDBMS
  - Data sets divided into logical units called "tables."
  - Tables are related to one another directly or indirectly via one or more columns with like data.



Understanding the Basics of the Relational Database

- A table is comprised of rows and columns as shown below.

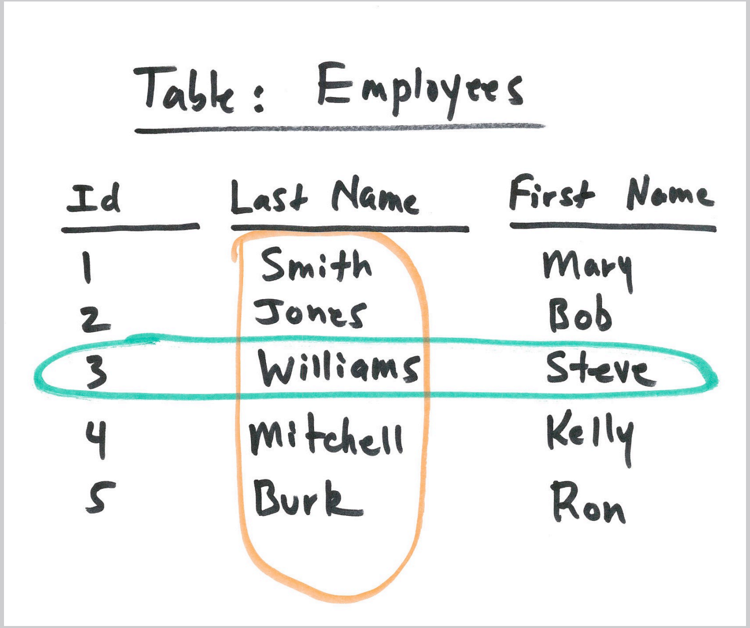
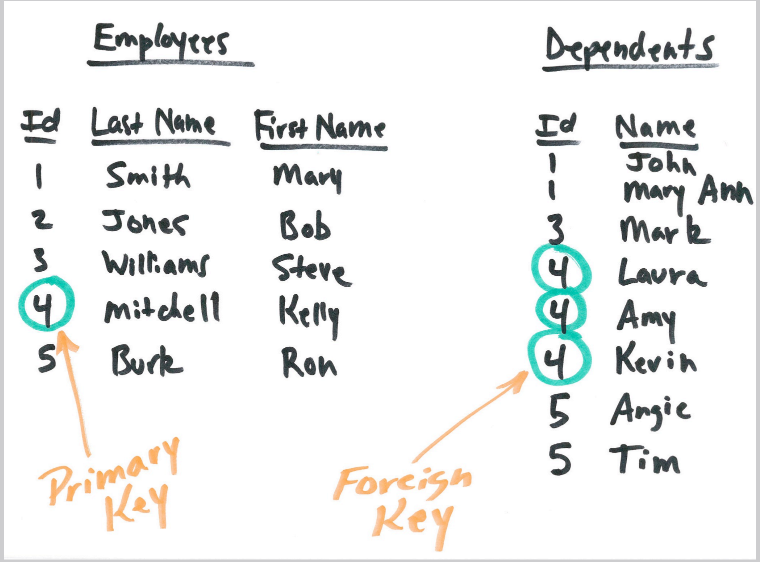


Table : Employees

Id	Last Name	First Name
1	Smith	Mary
2	Jones	Bob
3	Williams	Steve
4	Mitchell	Kelly
5	Burk	Ron

Understanding the Basics of the Relational Database

- The following figure shows how two or more tables can be related in a relational database.
- Referential Integrity is promoted through the use of primary and foreign keys.



Employers

Id	Last Name	First Name
1	Smith	Mary
2	Jones	Bob
3	Williams	Steve
4	mitchell	Kelly
5	Burk	Ron

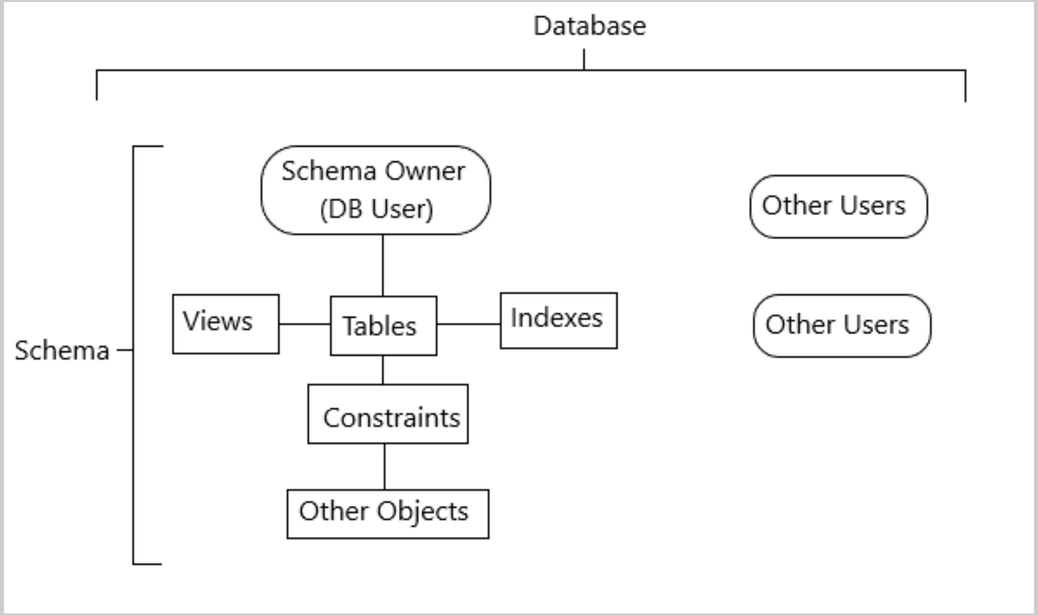
Dependents

Id	Name
1	John
1	mary Ann
3	Mark
4	Laura
4	Amy
4	Kevin
5	Angie
5	Tim

Primary Key
Foreign Key

The Database Schema

A Schema is simply a group of related objects in a database. A schema is owned by a single database user, and objects in the schema can be shared with other database users. Multiple schemas may exist in a database.

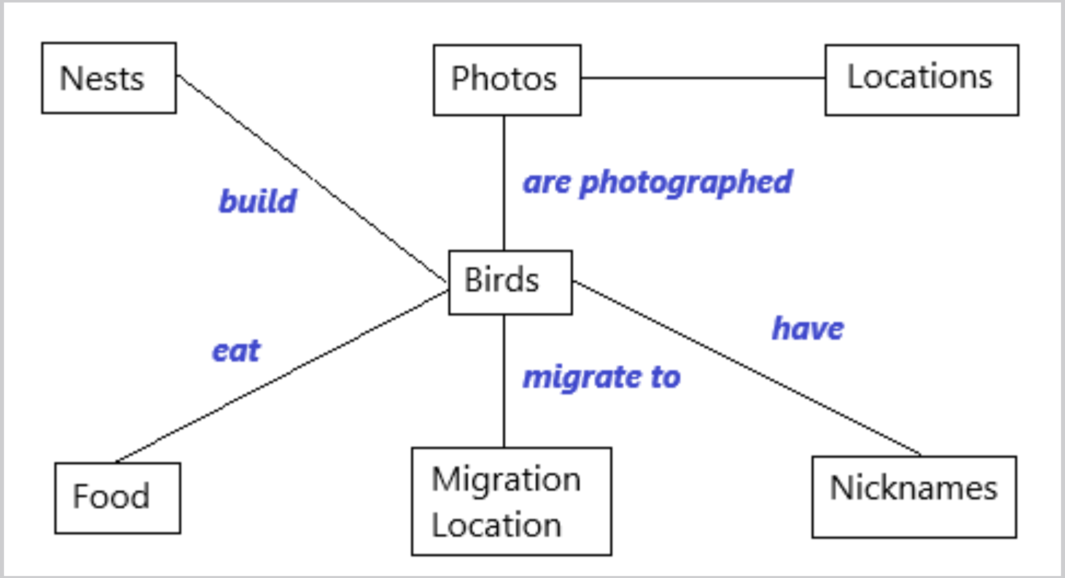


Relationships in a Relational Database

Tables within a relational database can have several different types of relationships:

one-to-one	one record in a table is related to only one record in another table each employee only has one pay record
one-to-many	one record in a table can be related to many records in another table each employee may have multiple dependents
many-to-many	many records in a table may be related to many records in another table many employees may have had many job titles, and many job titles may have many employees associated with them
recursive	a record in a table may be related to another record in the same table each employee is managed by another employee

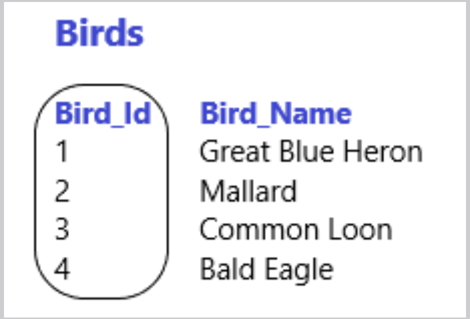
Relationships in a Relational Database



Relational Database Key Elements: Primary and Foreign Keys

Primary Key

- Tables almost always have a primary key.
- A primary key is a column in a table that ensures that every occurrence of an attribute, or column, is unique,
- A primary key identifies a parent record in a database to which child columns in other tables depend upon, or refer.



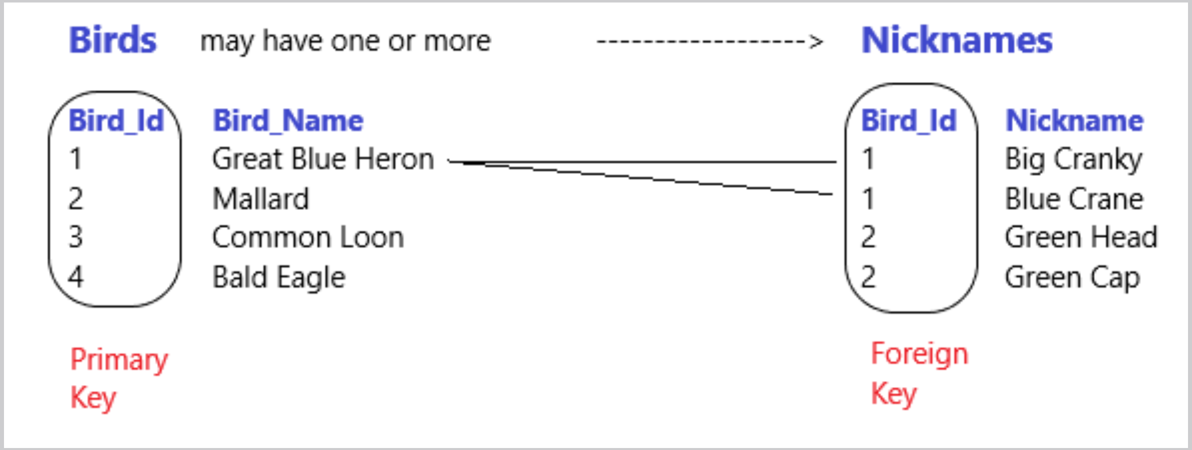
Birds

Bird_Id	Bird_Name
1	Great Blue Heron
2	Mallard
3	Common Loon
4	Bald Eagle

Relational Database Key Elements: Primary and Foreign Keys

Foreign Key

- Tables oftentimes have a foreign key.
- A foreign key is a column in a table that is considered a child record, and refers and depends upon a column in another table, that is a primary key.
- A foreign key does not have to be unique, like a primary key.



Bird_Id	Bird_Name
1	Great Blue Heron
2	Mallard
3	Common Loon
4	Bald Eagle

Bird_Id	Nickname
1	Big Cranky
1	Blue Crane
2	Green Head
2	Green Cap

Relational Database Key Elements: Referential Integrity

Does the Database Have Integrity???

- Data Integrity is the assurance of accurate, complete, clean, and relevant data in the database.
- Referential Integrity is the process of ensuring that data is consistent between related tables.
- Referential Integrity is a concept that deals with parent/child relationships in the database, and is implemented using:
  - Primary key constraints (parent records)
  - Foreign key constraints (child records)
- Referential Integrity establishes a dependency between related tables and is driven by the relationship you define.
- Relational database design is all about Referential Integrity.

Understanding the Basics of Database Design

To effectively design a database, you must:

- Understand the relational database model
- Understand referential integrity
- Know your data
- Know how your data will be used in your organization

To know your data, you must ask the right questions

Design methodology involves the following phases:

- Requirements analysis/gathering requirements
- Data modeling
- Design and normalization

Understanding the Basics of Database Design

Hallmarks of a Well-designed Relational Database

- Data storage needs have been met
- Data is readily available to the end user
- Data is protected through the use of built-in database constraints
- Data is protected through the use of built-in database security options
- Data is accurate and easy to manage
- Database performance is acceptable
- Database is scalable as the organization evolves or even transforms

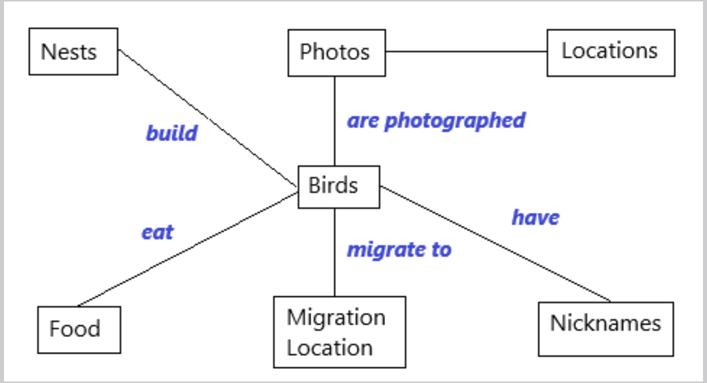
Redundant data is eliminated or minimized as much as possible!!!

Understanding the Basics of Database Design

Let's look at this example of data about "bird rescues" that will be eventually integrated with an existing "bird" database. Based on the following information we know about bird rescues, we will:

- Create a very basic data model
- Establish groups of data and lists of fields that will be our basis for modeling entities and attributes.

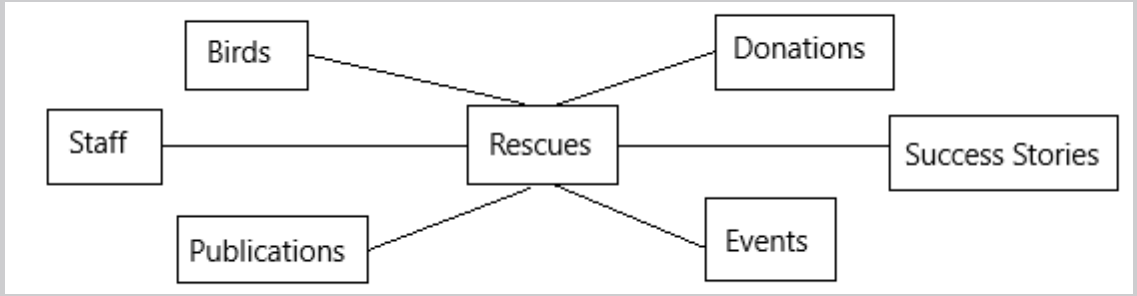
Bird rescues have a name, address, and contact information, as well as web and social media sites. there are various types of birds taken in, and each rescue may have multiple facilities. Rescues are managed by full-time employees as well as volunteers, and receive funding from various donors and fundraising events. A rescue may have publications, marketing material, and sponsors. They may also sell merchandise, have success stories, and interact with wildlife photographers.



Understanding the Basics of Database Design

Based on the previous information, we might begin to come up with a basic data model as well as groups of data and lists of fields (entities and attributes).

Rescues	Staff	Birds	Donations	Events	Publications	Successes
Name	Name	Id	Donor	Name	Name	Bird
Address	Type	Type	Contact Info	Date	Date	Date
Contact info	Salary	Name	Type of Donor	Location	Authors	Story
Website	Education	Issue	Amount	Sponsors		Staff
Social Media	Contact Info	Progress		Amount Raised		



Getting to Know the Test Organization and Data

Throughout the exercises within this course, you will be designing a database about Wildlife Photographers who take pictures of birds. This database should be design so that it can eventually be integrated with an existing “Bird” database. Here is some basic information about the data set to get you started:

Photographers have names, addresses, education, may have received awards, and may have various websites and/or social media sites. Each photographer has his/her own passion, approach, styles (action, portrait, close-up, artistic, underwater, still, etc.), preferred bird targets, and so forth. Photographers use various cameras and lenses and may produce media in a variety of formats using various editing software and may have different types of clients. Photographers may be published or contribute images to publications. Photographers may be a mentor or protégé, and may volunteer at a bird rescue or other non-profit organization. Photographers have various skill levels, such as beginner, novice, hobbyist, competent photographer, skilled, artist, and world-class. Photographers may have various products that they market and sell. Photographers may be self employed or work for an organization.

Equipment used by photographers includes cameras, lenses, and editing software. Cameras have a make and model, sensor type (full frame or crop sensor), megapixels, frames per second, ISO range, and cost. Lenses will have a make, lens type, aperature range, and cost.

Hour 1 Exercise

Based on the previous description of wildlife photographers and the test data set described on the previous slide, perform the following exercises to the best of your ability with the limited information:

1. What is your database about?
2. What is the purpose of your database?
3. Who do you anticipate the users of your database to be?
4. Who do you anticipate the customers to be?
5. List all of the basic entities and attributes of wildlife photographers you can think of. Eventually, this will be your bases for entities and attributes, that will eventually become your Entity Relationship Diagram (ERD).
6. Draw a very basic data model based on the lists above, that will eventually become your ERD.
7. Establish a basic list of fields (based on what you know now) for each entity.
8. List some rules about your data (what type of data is allowed, how is can be updated, what can be added to, how data can be deleted, any dependencies, anything you can think of). There are no right or wrong answers at this point, we are brain storming.

Hour 2 Overview

Understanding the logical database model (LDM)

- Entities, attributes, relationships
- Referential integrity, primary keys, foreign keys
- Entity relationship diagrams (ERD)
  - Logical ERD versus physical ERD
  - Creating a simple logical ERD

Eliminating redundant information, duplicate data

- Examples of duplicate data
- Normalization and the normal forms, examples with data
- Applying normalization to your logical ERD

Exercise

Exercise review

Break

Understanding the Logical Database Model

The two types of modeling that are discussed in this class are:

- Logical modeling
- Physical modeling

This hour focuses on logical modeling. Physical modeling is discussed in the final hour of this class.

Logical modeling involves:

- Gathering business requirements
- Converting requirements into a model that is easy to visualize
- Breaking down data logically into entities
- Creating a logical ERD to reflect entities, attributes, and relationships
- May or may not involve identifying keys (controversial)
- I prefer to identify keys early on as they become obvious
- A major goal is to eliminate redundant data

Understanding the Logical Database Model

We have already done some of the work to:

- Gather business requirements
- Group data
- Create lists of fields
- Create a basic model that shows basic relationships, even if they are not specific at this point.

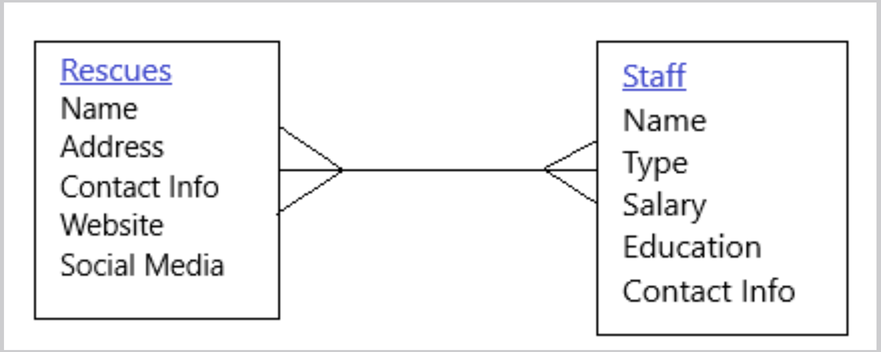
You will continue to make discoveries throughout the design process and must be flexible as your model evolves.

During this process, you need to break down your data into entities and attributes, as detailed and specifically as possible.

You will identify specific entity relationships and it should become obvious what your keys will be.

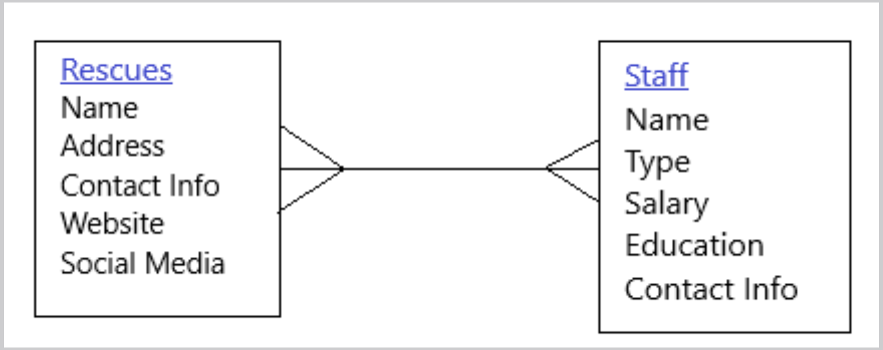
Understanding the Logical Database Model

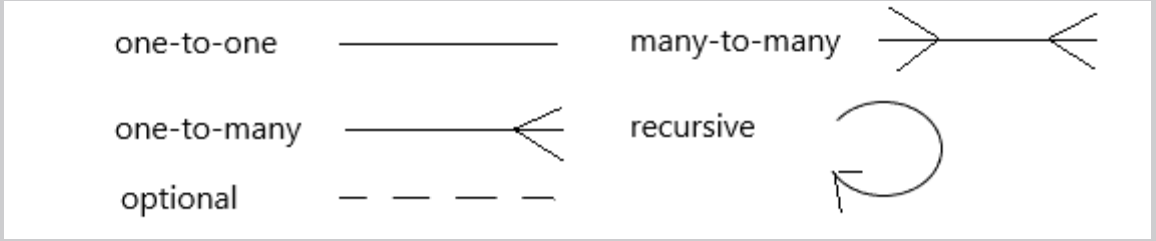
- Entity relationship diagrams (ERD)



Understanding the Logical Database Model

Here, we have taken a portion of the basic logical model from the first hour, and built upon it using the groups and lists we derived.

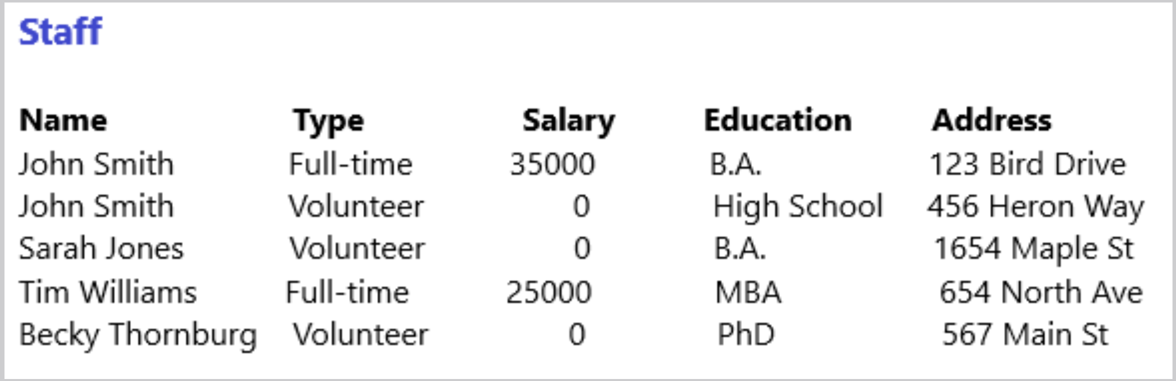




Eliminating Redundant Information, Duplicate Data

Shortly, we will think about eliminating redundant data.

For the previous example, here are some examples of potential redundancy:



Staff

Name	Type	Salary	Education	Address
John Smith	Full-time	35000	B.A.	123 Bird Drive
John Smith	Volunteer	0	High School	456 Heron Way
Sarah Jones	Volunteer	0	B.A.	1654 Maple St
Tim Williams	Full-time	25000	MBA	654 North Ave
Becky Thornburg	Volunteer	0	PhD	567 Main St

Eliminating Redundant Information, Duplicate Data

Normalization

Normalization is the process of reducing redundancies of data in a database, a critical technique during database design

The guidelines of normalization, or rules, are call the Normal Forms.

A normal form is a way of measuring the levels, or depth, to which a database has been normalized.

In this course, we explore the following normal forms:

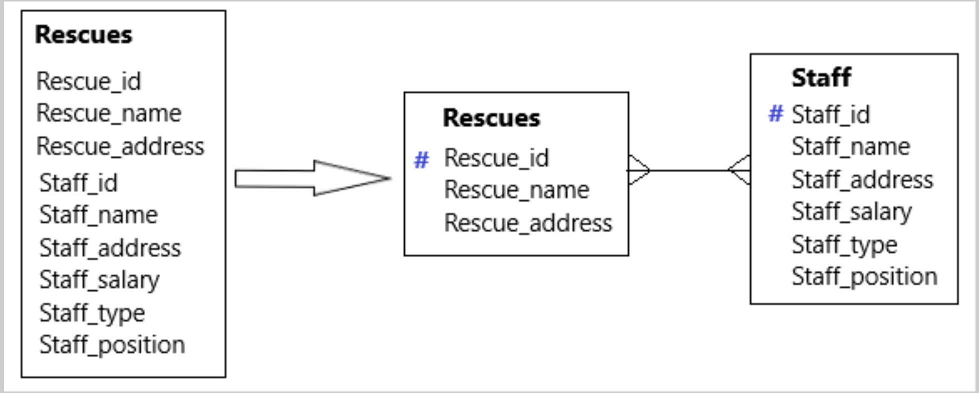
- First Normal Form (FNF)
- Second Normal Form (SNF)
- Third Normal Form (TNF)

Each subsequent normal form depends on normalization steps taken in the previous normal form. For example, in order to normalize a database in SNF, it must fist be in FNF.

Eliminating Redundant Information, Duplicate Data

First Normal Form (FNF) – The Key

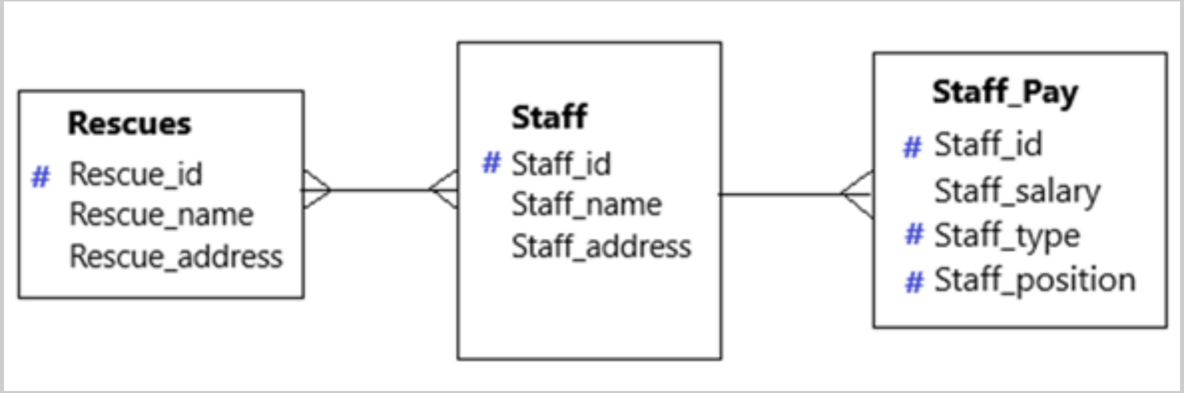
- The objective of the FNF is to divide the base data into logical units called entities.
- Each entity will have attributes about the data contained within.
- These entities and attributes will become tables and columns in the physical design.
- A primary key, when appropriate, should be assigned to each entity.
- To achieve FNF, data is broken into logical units of related information
- Assigning a primary key helps ensure no duplicate records within a group



Eliminating Redundant Information, Duplicate Data

Second Normal Form (SNF) – The Whole Key

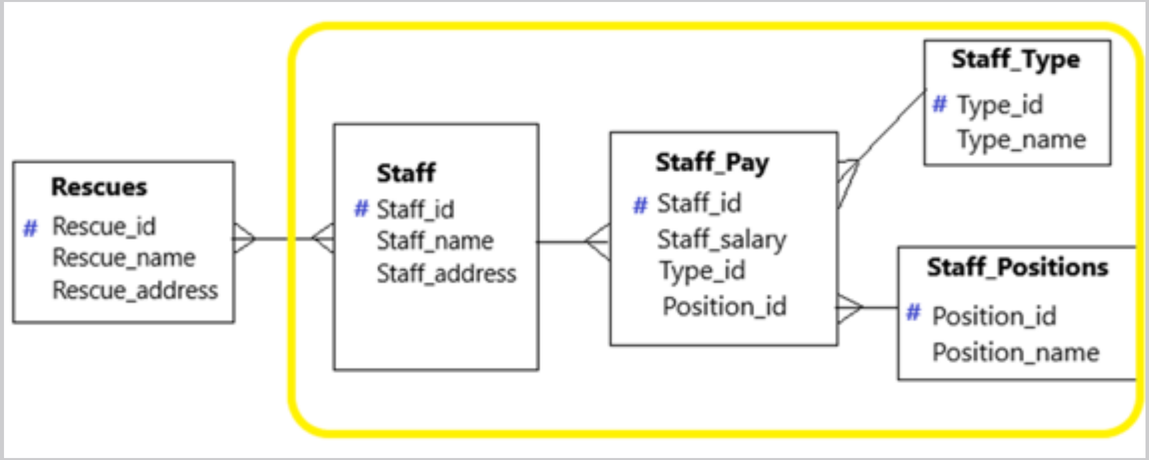
- The objective of the SNF is to take data that is only partly dependent on the primary key and enter that data into another entity.
- SNF is derived from FNF by further breaking down the original entity into two entities.
- Attributes that are only partly dependent on the primary key are moved to a new entity.



Eliminating Redundant Information, Duplicate Data

Third Normal Form (TNF) – Nothing But the Key

- The objective of the TNF is to remove data that is not directly dependent on the primary key.
- Attributes do NOT have attributes of their own.
- Any attribute appearing to have attributes of its own should be moved to another entity.
- Only attributes that are directly dependent on the key should remain in the entity.



Hour 2 Exercise

1. Take your logical model from the previous exercise and use it to create a logical ERD. Draw lines between your entities depicting all relationships (one-to-one, one-to-many, many-to-many). Identify any constraints you can imagine at this point (i.e. Primary Key, Foreign Key, Unique). Apply the rules of the First Normal Form to this model.

2. Take your ERD to the Second Normal Form (SNF) and modify any Primary Key and Foreign Key constraints as necessary, and modify any entity relationships as appropriate.

3. Finally, take the steps necessary to take your ERD to the Third Normal Form (TNF) and make any modifications necessary to any keys and constraints.

Hour 3 Overview

Finalizing your database design

- Checking your logical ERD for completeness
- Creating a physical ERD from your logical ERD and data requirements
    - Tables, and columns
    - Keys, constraints, data types, mandatory data
    - Views and indexes

Preparing your database for implementation

- Generating SQL code to create the physical database and populating with data
- Setting up your database environments (DEV, TEST, PROD)
- Creating the schema
- Change management

Exercise

Exercise review

Course summary and Q&A

Finalizing Your Database Design

After you have taken the steps to normalize your database to the preferred normal form (most often Third Normal Form):

- Study your entities and attributes
- Study your relationships
- Ensure you have identified Parent/Child (PK and FK) relationships
- Make any modifications necessary

Next, we will create a physical ERD from your logical ERD.

The physical ERD will have the following elements:

- Entities become tables
- Attributes become columns
- Data types are assigned to columns
- Mandatory data is identified (NULL, NOT NULL)
- Foreign keys are finalized if not already

Finalizing Your Database Design

Keys and Mandatory Columns

For the purpose of this class, we will use the following naming conventions for:

• Primary key	#
• Foreign key	F
• Mandatory columns	*

Data Types

When finalizing your physical ERD, data types need to be assigned to each and every column.

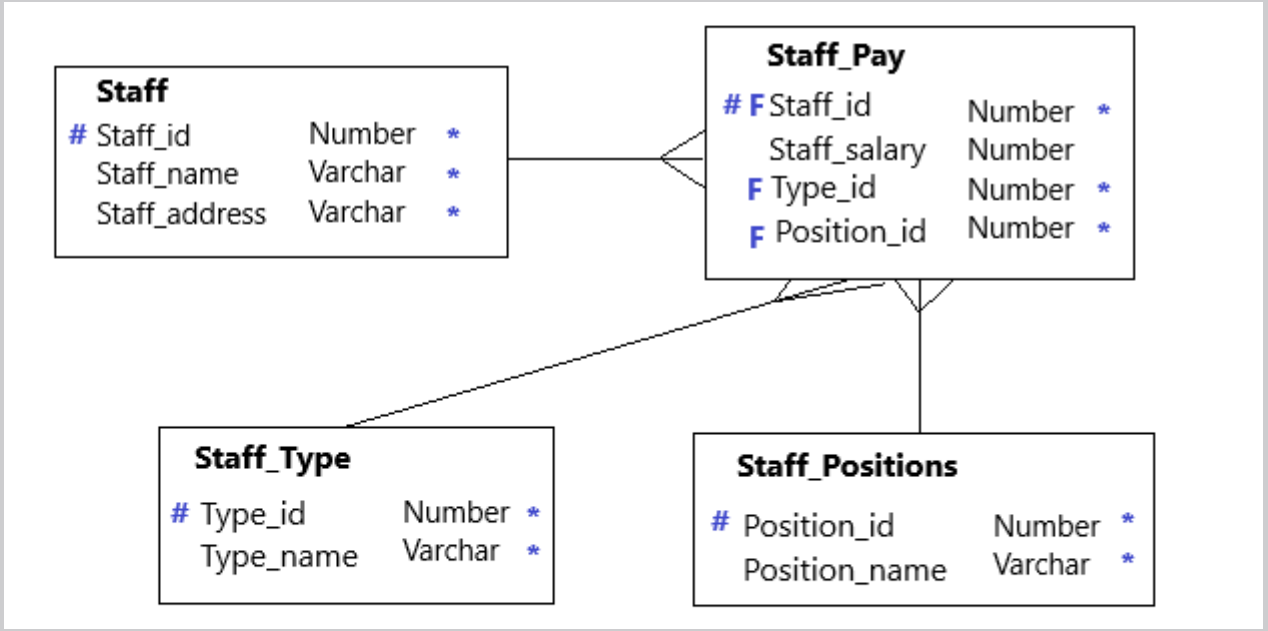
Data type names may vary between database software

For the purpose of this class, we will use the following basic data types:

• VARCHAR	character value
• NUMBER	numeric value
• DATE	date and time value

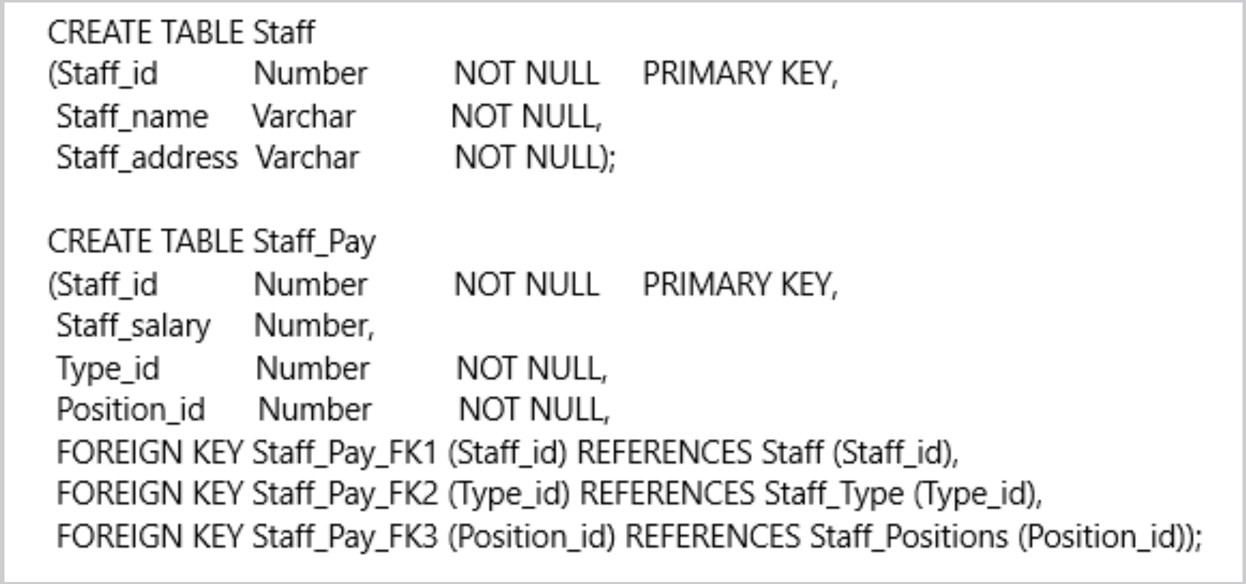
Finalizing Your Database Design

Physical ERD for a Portion of the Example Database in TNF



Preparing Your Database for Implementation

Example SQL Syntax to create tables based on your design:



```
CREATE TABLE Staff
(Staff_id        Number        NOT NULL    PRIMARY KEY,
 Staff_name      Varchar       NOT NULL,
 Staff_address   Varchar       NOT NULL);

CREATE TABLE Staff_Pay
(Staff_id        Number        NOT NULL    PRIMARY KEY,
 Staff_salary    Number,
 Type_id         Number        NOT NULL,
 Position_id     Number        NOT NULL,
 FOREIGN KEY Staff_Pay_FK1 (Staff_id) REFERENCES Staff (Staff_id),
 FOREIGN KEY Staff_Pay_FK2 (Type_id) REFERENCES Staff_Type (Type_id),
 FOREIGN KEY Staff_Pay_FK3 (Position_id) REFERENCES Staff_Positions (Position_id));
```

Preparing Your Database for Implementation

Things to think about as you prepare you physical design for eventual implementation:

- Setting up your database environments (DEV, TEST, PROD)
- Creating the schema
- Generating SQL code to create the physical database and populating with data
- Change management
- Resource management
- Using built-in database features to control data integrity
- Using built-in database features to control access to data and security
- Backup and recovery plan
- Performance tuning

Hour 3 Exercise

1. Use your logical ERD from the previous exercise to design a physical ERD with the following elements:
   - Entities
   - Attributes
   - Relationships
   - Datatypes
   - Primary and Foreign Keys

2. Can you think of any useful views that you might eventually design?

3. If time permits, write the SQL code (DDL) to create at least one of your tables.

Summary and Q&A

- In this class, you have learned some of the basic and most important concepts of Relational Database Design.
- When designing a database , there are not necessarily right and wrong answers.
- Even experts disagree on best practices at times and design methodologies.
- You learn about Referential Integrity and how to reflect that in your design.
- You learned about logical and physical database modeling.
- You built a basic database model.
- You used Normalization to complete your logical ERD
- You created a physical ERD from your logical design

Q&A

Thank You!

Polling Question (check boxes)

Which of the following subjects interest you for future courses?

- ❑ Advanced SQL Queries Workshop
- ❑ Oracle SQL*Plus
- ❑ Oracle SQL Developer
- ❑ Intro to Oracle PL/SQL Programming
- ❑ Introduction to Unix
- ❑ Writing Unix Shell Scripts
- ❑ Oracle Database Administration
- ❑ Project Management
- ❑ Organizational Leadership